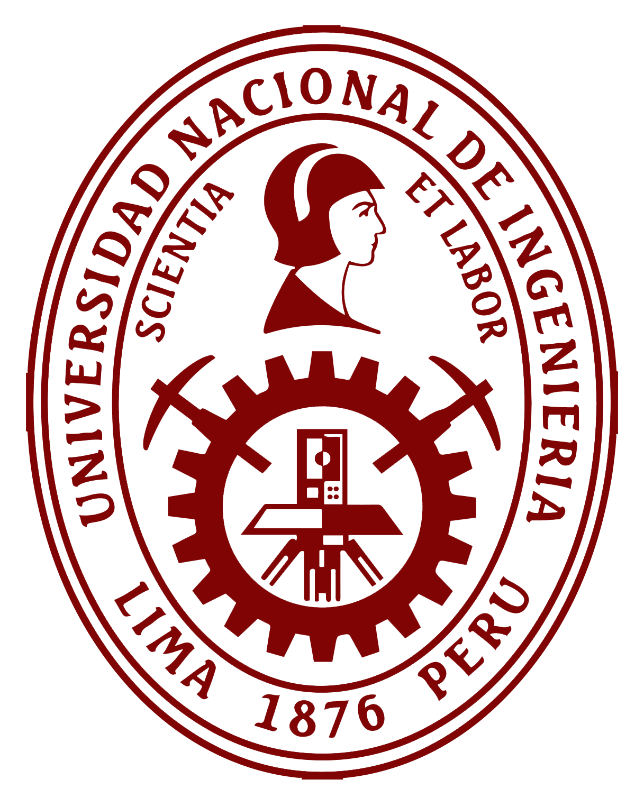
**UNIVERSIDAD NACIONAL DE INGENIERIA**

**FACULTAD DE INGENIERIA DE SISTEMAS**



SISTEMAS UNI

**TEMA :** MANEJO DE DOCUMENTOS XML

**DOCENTE :** GUSTAVO CORONEL CASTILLO

**INTEGRANTES :**

* RIVERA CAMPOS, JUAN
* RIVEROS TIPAYA, MARCOS

2020

Este trabajo de investigación está dedicado primordialmente a Dios que nos ha dado la vida y la fortaleza necesaria para terminar este proyecto, hacia nuestros padres a quienes le debemos todo lo que tenemos, a nuestro profesor quien nos brinda las herramientas necesarias para posteriormente desenvolvernos en nuestra vida profesional y todas las personas que confiaron en nosotros.

Índice

[1. Resumen 4](#_Toc32920407)

[1.1. Resumen y palabras clave 4](#_Toc32920408)

[2. Introducción 5](#_Toc32920409)

[3. Desarrollo del Tema 7](#_Toc32920410)

[3.1. Antecedentes 7](#_Toc32920411)

[3.2. Estado del arte 10](#_Toc32920412)

[3.3. Desarrollo de los conceptos 19](#_Toc32920413)

[3.3.1. Tecnologías para el análisis y procesamiento de documentos XML 19](#_Toc32920414)

[3.3.2. Acceso a la información en XML 27](#_Toc32920415)

**4. Caso Desarrollado …………………………………………………………………………………………….32**

**5. Conclusiones………………………………………………………………………………………………………35**

**6. Recomendaciones……………………………………………………………………………………………….36**

# Resumen

## Resumen y palabras clave

Este Trabajo tiene como tema principal “Manipulación de documentos XML: tecnologías para el análisis y procesamientos de documentos XML; extracción de información contenida en documentos XML.”

El trabajo se basa en un estudio epistemológico que contiene los antecedentes, un estado del arte, un desarrollo de conceptos, tendencias futuras y una conclusión. Esta primera parte será más teórica sobre los conceptos de documentos XML, su análisis, procesamiento y las tecnologías que se usan.

**PALABRAS CLAVE:** documento XML, DTD, DOM, procesar, analizar,

información, estructura,

# Introducción

Los documentos XML permiten infinidad de posibilidades, a partir de ellos se pueden construir otras tecnologías, realizar conversiones, intercambiar información, estructurar datos, crear bases de datos. Con un simple fichero de texto se pueden realizar una gran cantidad de herramientas, se podría decir que XML causó una revolución en la tecnología y especialmente en la Web.

XML da la posibilidad de definir unas normas o reglas que todos los documentos deben seguir, gracias a ello, se consigue que todos los ficheros tengan una estructura similar y que cualquier fichero se pueda analizar y procesar de idéntica manera. Esta estructura se consigue de distintas formas pero lo más normal es hacerlo mediante un DTD o un esquema.

Una vez que se tiene un documento estructurado es necesario analizarlo y procesarlo para manipular la información que contenga. Para ello, existen diversas tecnologías, algunas basadas en memoria como DOM, donde se representa el documento en forma de árbol, y otras basadas en eventos como SAX, donde según se va leyendo el fichero se realiza el análisis.

XML no es el único lenguaje de marcas, existen otros como HTML, pero XML permite la transformación a otros formatos de archivos, por eso, a partir de XML se puede conseguir un XHTML, ePUB, etc. Otorga una gran versatilidad, a partir de un documento genérico se puede convertir a otros formatos.

Otra de las características de XML es que permite almacenar información, esto favorece el uso de bases de datos documentales, donde en lugar de las tablas que existen en las bases de datos relacionales, se usan documentos en donde se almacena toda la información. Además, han ido apareciendo distintos lenguajes que permiten el acceso a los datos en estos documentos.

Visto todo lo que significa la tecnología XML, se entiende que es fundamental que esta información pertenezca al curriculum de un módulo sobre lenguajes de marcas. Esta tecnología va encuadrada en el módulo de “lenguajes de marcas y sistemas gestores de información” que está dentro del ciclo superior de “desarrollo de aplicaciones multiplataforma”.

HTML y XML son los dos lenguajes de marcas con más importancia ahora mismo, por lo que, es totalmente necesario que los alumnos conozcan estas herramientas. Con este trabajo, serán capaces de saber qué es un XML, cómo analizarlo y procesarlo y cómo tener el acceso a la información contenida en él.

Este es un conocimiento que, además, sirve de base para otros módulos posteriores que deben estudiarse como “acceso a datos”, donde se estudiarán bases de datos documentales como puede ser mongoDB o baseX. Por eso, este tema y esta unidad didáctica son fundamentales en el currículo de los alumnos.

# Desarrollo del Tema

En este epígrafe, se va a hacer un estudio teórico sobre XML y las tecnologías para su análisis y procesamiento. Comenzará con los antecedentes, estado del arte, descripción de XML y por ultimo las tecnologías relacionadas (DOM, SAX, StAX, VTD).

## Antecedentes

Un ordenador se considera que representa la información, principalmente, de dos maneras: en modo binario y en forma de texto. Aunque en un ordenador se represente todo de forma binaria, incluido el texto, se interpreta que aquello que no es texto es binario (imagen, sonido, audio, etc.).

La forma de codificar los datos a binario es variable, cada software lo puede hacer de una manera distinta, esto implica que para acceder a la información y decodificar ese archivo el programa del ordenador debe conocer como ha sido codificado. Eso sucede usando el mismo software, u otro, que sea capaz de traducir la información codificada.

Con el texto ocurre distinto porque éste se codifica siguiendo un estándar, por ejemplo, el código ASCII, por el cual un carácter se representa con un número binario determinado. De esta manera, el fichero de texto se puede decodificar en cualquier software.

El formato binario tiene las ventajas de que es más eficiente, genera ficheros más pequeños que si tuviera que representarse en texto, se accede más rápido a los datos, ya que, usa un lenguaje más cercano al ordenador y tiene un acceso directo a los datos, y no de forma secuencial, como ocurre con los de texto. Mientras, que los archivos de texto son más fáciles de transportar y exportar a otro software distinto, no habrá problemas para la decodificación y son más fáciles de manipular.

Una vez vistas las ventajas de ambos formatos surge el problema del intercambio de información entre aplicaciones y sistemas. Porque un archivo se realiza en un software determinado, con una extensión determinada y el sistema que quiera trabajar con él deberá saber decodificar ese fichero.

Además, en la actualidad este problema se ve incrementado por la gran cantidad de diferentes sistemas operativos, tanto de ordenadores como de dispositivos electrónicos. Con el tiempo, han aparecido estándares para determinados formatos como PDF, JPG, MP3, etc., pero sigue habiendo aplicaciones que usan sus propios formatos, como DOC de Word o MOV para Apple.

Con los ficheros de texto no existe el problema que se acaba de mencionar, ya que, cualquier dispositivo es capaz de entender texto plano, es decir, solo texto sin ningún tipo de información adicional o formato. Sin embargo, estos ficheros son menos seguros que los binarios porque su contenido está a la vista de cualquiera, y además, en un texto no se puede representar una fotografía u otros tipos de archivos.

Visto este escenario, se intentaba que dentro del mismo fichero de texto hubiera información sobre cómo representar el contenido del fichero. Así, surgen los lenguajes de marcas, donde un documento de texto incluye en él anotaciones o etiquetas, que contienen la manera de interpretar la información dentro del fichero. Algunos lenguajes de marcas son: LaTeX, RTF, HTML o XML.

El XML o *eXtensible Markup Language* (Lenguaje de Marcas Extensible) no es un lenguaje de marcas, sino que es un metalenguaje, esto quiere decir, que XML permite diseñar otros lenguajes de marcas. Fue desarrollado en 1996, por un grupo de trabajo llamado “*SGML Editorial Review Board*” que pertenecía a *World Wide Web Consortium* (W3C). La W3C es la encargada de la organización y crecimiento de la web.

Primero se creó HTML, el lenguaje con el que están diseñadas las páginas web. Debido al crecimiento de Internet, W3C decidió que lo mejor era crear unas reglas, de manera, que cualquiera pudiera crear lenguajes de marcas a su medida, pero manteniendo la misma estructura y sintaxis, y así, fuera posible tratarlos con la misma herramienta. Estas reglas y tecnologías es XML, que ya no está solo enfocado a Internet, sino, al intercambio de información estructurada en distintas plataformas.

Se definió un documento XML como: “Un objeto de datos es un documento XML si está bien formado como viene definido en su especificación, además será válido si sigue unas restricciones” .

SGML (*Standard Generalized Markup Language*) es el estándar mundial en lenguajes de marcas, HTML es una aplicación de él y XML es un subconjunto de SGML. Mientras que HTML se encarga de describir el formato, XML indica cómo debe ser la estructura y la semántica. HTML se creó antes y por eso no cumple las normas indicadas por XML.

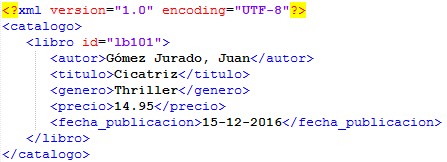
Debido a la simplicidad de su estructura, XML es muy usado para la representación de datos en diversas aplicaciones, y gracias, a su portabilidad se usa para el intercambio seguro de datos entre sistemas de naturalezas totalmente heterogéneas, lo que facilita la comunicación y la transmisión. Además, el hecho de ser una plataforma independiente lo hace muy atractivo para la mayoría de aplicaciones

Algunos de los lenguajes basados en XML son:

* RSS
* ePUB
* ODF
* SOAP
* SVG
* XHTML

## Estado del arte

XML es una tecnología sencilla que se complementa con otras tecnologías que le añaden muchas más posibilidades. Un ejemplo de un documento XML puede ser el que se observa en la Figura 1.



**Figura**

**1**

**. Ejemplo de documento XML**

**Estructura y componentes de un documento XML**

Los documentos XML tienen una estructura determinada:

* **Prólogo**: Con esto empieza un documento XML, en esta parte se declara qué tipo de documento XML es (versión XML, codificación de caracteres), se asocia el DTD, se declara el DOCTYPE, se indican otros documentos que pueden ser complementarios, como un XSLT y se indican las instrucciones de procesamiento.
* **Elemento raíz:** Todos los componentes del contenido del documento deben pertenecer a este elemento raíz. Se abre después del prólogo y se cierra al final.

Esta sería la estructura de un documento XML, la primera parte se usa para declarar las características del documento y, posteriormente, un elemento raíz donde se enganchan los demás componentes del documento. Algunos de estos componentes son:

* **Elementos:** Es un conjunto de datos que viene indicado por una etiqueta de apertura y otra de cierre. Representa un componente lógico en el documento, un elemento puede contener otros elementos y, también, puede ser un elemento vacío. En la Tabla 1, se ven las partes de un elemento con un ejemplo.

**Tabla 1. Partes de un elemento**

|  |  |
| --- | --- |
| <titulo>**Cicatriz**</titulo> | Elemento |
| <titulo> | Etiqueta de apertura |
| </titulo> | Etiqueta de cierre |
| **Cicatriz** | Contenido del elemento |

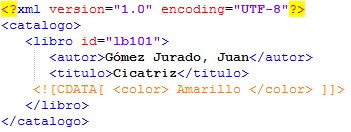
* **Etiquetas y atributos:** Las etiquetas son identificadores que empiezan por ‘<’ y terminan por ‘>’ y delimitan a los elementos. Además, las etiquetas puede contener atributos, que es una manera de añadir una propiedad al elemento. Un ejemplo de un atributo puede ser:

<libro id=**"lb101"**>

* **Comentarios:** Es una etiqueta que empieza por ‘<!--‘ y termina por ‘-->’ e indica que el texto introducido en medio no debe ser procesado como parte del contenido. Pueden ir en cualquier parte excepto dentro de una etiqueta o dentro de otro comentario. Un ejemplo de comentario es:

<!—Esto es un comentario -->

* **Secciones CDATA:** Sirven para introducir trozos de texto sin que sea procesado como código en XML, es decir, se puede poner texto con caracteres como ‘<’ sin que se identifique como una etiqueta. Normalmente, se usa para colocar código de otros lenguajes. Un ejemplo de la sección CDATA es el que se muestra en la Figura 2.



**Figura**

**2**

**. Documento XML con sección CDATA**

* **Entidad:** Representa un carácter, se usa para poder utilizar caracteres especiales. XML trae definidas por defecto 5 entidades que se pueden ver en la Tabla 2.

**Tabla 2. Entidades definidas en XML**

|  |  |
| --- | --- |
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &apos; | ' |
| &quot; | " |

**Documento bien formado / Documento válido**

Hay que tener en cuenta dos conceptos distintos cuando se habla de documentos XML, uno es que un documento esté bien formado y el otro que sea un documento válido. Para ello, vamos a ver qué significa cada concepto.

Un documento XML se dice que está bien formado cuando el documento cumple las reglas de sintaxis de XML. Estas normas se deben cumplir obligatoriamente, de hecho, si no se cumplen, en un analizador o parser se mostrarán como errores. Estas normas son:

* Debe haber un solo elemento raíz
* Cada elemento debe tener una etiqueta de apertura y una etiqueta de cierre o ser una etiqueta de elemento vacío.
* Se deber cerrar primero los elementos que se han abierto últimos
* Los atributos deben aparecer en las etiquetas de apertura.
* Los valores de los atributos deben tener un valor asociado y deben ir entre comillas.
* Los nombres de los elementos y atributos deben empezar por una letra, dos puntos ‘:’, o guión bajo ‘\_’, después pueden ir seguidos de letras, números, guiones o puntos.
* Los nombres de los elementos y atributos no deben contener espacios.
* Dos atributos de un mismo elemento no pueden tener el mismo nombre.
* No se pueden usar los caracteres ‘<’, ‘>’, ‘&’ excepto en las secciones CDATA

Si no cumple estas normas no se considera un documento XML y un procesador de XML lo rechazará.

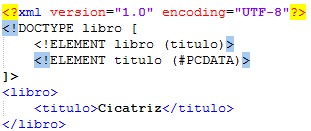
Un documento se dice que es válido cuando está bien formado y, además, cumple con la gramática definida para el esquema de ese documento. Generalmente, la gramática viene dada por el DTD *(Document Type Definition*), pero hay otros métodos para definirlas que se verán a continuación.

**Validación de datos**

Las formas más comunes para validar documentos XML es mediante DTD y XML Schema, pero también hay otras tecnologías como RelaxNG o Schematron. Este documento para validar son unas reglas gramaticales que deben cumplir los documentos XML, de esta manera, se pueden definir unas normas que harán de plantilla para que todos los documentos XML las cumplan y estén hechos de la misma forma. La ventaja que da este sistema es que asegura que todos los documentos tendrán la misma estructura siempre que sean válidos.

DTD es el más usado porque ya existía con SGML, pero tiene sus críticos, ya que, no cumple las reglas de XML. Un DTD se puede definir en el propio documento XML o puede ser en un fichero externo, que se indicará en el prólogo del documento XML. Definirlo en el propio documento tiene la desventaja de que esa gramática solo sirve para ese documento, no sirve como plantilla para otros, por eso, no se suele definir así. La única ventaja que tiene la definición en el propio documento es que la validación y el XML están juntos, no hay que depender de un fichero externo.

En la Figura 3, se ve un ejemplo de cómo sería la definición de un DTD en el propio documento, se define en el prólogo usando la etiqueta DOCTYPE y se indica cuáles serán las reglas del documento XML.



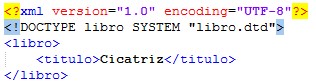
**Figura**

**3**

**. Ejemplo de un DTD interno**

Por otro lado, tener un DTD en un fichero aparte permite que sirva de gramática a muchos documentos, lo único que hay que hacer es indicar la ruta al DTD.

Tal como se ve en la Figura 4, donde se indica que el DTD está en el fichero libro.dtd, y el analizador se encargará de comprobar que el fichero sea válido con ese DTD.

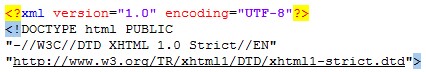


**Figura**

**4**

**. DTD definido por el usuario**

El ejemplo visto en la Figura 4 es si el DTD está definido por el propio usuario, pero también hay DTDs en Internet, ya definidos para lenguajes determinados, por ejemplo, en la Figura 5, se ve como se definiría un DTD para validar una página hecha en XHTML 1.0. En este caso, en la sección DOCTYPE, se indica la URL en Internet del DTD sobre el que se quiera validar.



**Figura**

**5**

**. DTD de Internet**

En un documento DTD se definen los elementos que se pueden usar en el documento XML, los atributos que pueden ir en los elementos, hasta se pueden indicar posibles valores para esos atributos, y las entidades del documento XML.

Tal como se ha indicado anteriormente, el otro método más usado aparte del DTD es la validación por XML Schema. Esta tecnología se diseñó para que, a diferencia de DTD, siguiera las restricciones de XML y permitiera más posibilidades a la hora de validar [6]. Algunas de las ventajas de XML Schema frente a DTD son:

* Su sintaxis está basada en XML por lo que se puede analizar y manipular como cualquier otro documento XML.
* Soporta tipos de datos (int, float, boolean, …), mientras que, DTD trata todo como cadenas de caracteres. Esto facilita usarlo como una base de datos o validar que el tipo de dato sea correcto, y por tanto, otorga más seguridad, ya que, se crean validaciones más restrictivas.
* Soporta la integración con el espacio de nombres, mientras que, DTD solo asocia un documento con ese DTD.

Pero XML Schema también tiene algunos inconvenientes como:

* Es más difícil de entender que un DTD
* Tiene más incompatibilidades con algunas aplicaciones que DTD
* No es posible la definición de entidades
* Algunas tecnologías como DOM o SAX tienen algunas características para DTD pero no para XML Schema.

Un XML Schema es un documento XML pero con extensión xsd, es decir, tiene la misma apariencia que un XML pero con la restricción de que el elemento raíz se debe llamar schema. En esa primera etiqueta, se define el espacio de nombres usando alguna de las 3 posibilidades que se ven en la Figura 6.



**Figura**

**6**

**. Definición espacio de nombres Schema**

Una vez definido el espacio de nombres, las etiquetas del Schema usarán el prefijo elegido, normalmente se usa ‘xs’. Un ejemplo de un Schema es el que está en la Figura 7, al principio se define el espacio ‘xs’ y las etiquetas pertenecientes al esquema empiezan de esa forma.



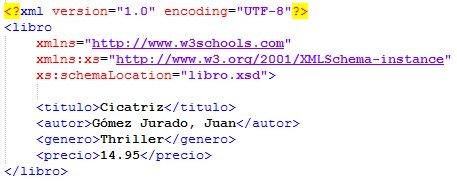
**Figura**

**7**

**. Ejemplo de**

**Schema**

Al igual que con los DTD, en XML Schema se tiene que asociar un documento XML con su Schema, tal como, se observa en la Figura 8. Al contrario que en DTD, ahora no hay una etiqueta DOCTYPE, por lo que, hay que definir un espacio de nombres para el documento XML y otro para el Schema, además, de indicar dónde está el esquema. Tal como se ve en la Figura 8, se usa el espacio por defecto para el documento, el espacio ‘xs’ para el esquema y se indica que el Schema está en “esquema.xsd”. Una vez definido esto, se continúa con el documento en XML.

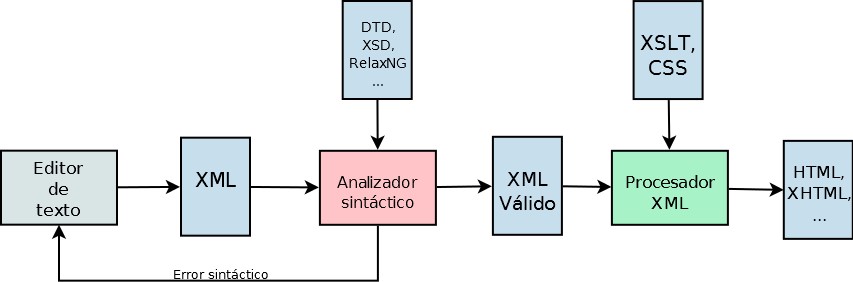


**Figura**

**8**

**. Ejemplo documento XML con XML Schema**

Para terminar con los documentos XML, en la Figura 9 se ve un diagrama de cómo es el funcionamiento general de una aplicación XML. Lo primero, es crear el documento XML, que se puede hacer con un simple editor de texto, una vez que se tiene el documento XML se le pasa al analizador sintáctico junto con el DTD o Schema.



**Figura**

**9**

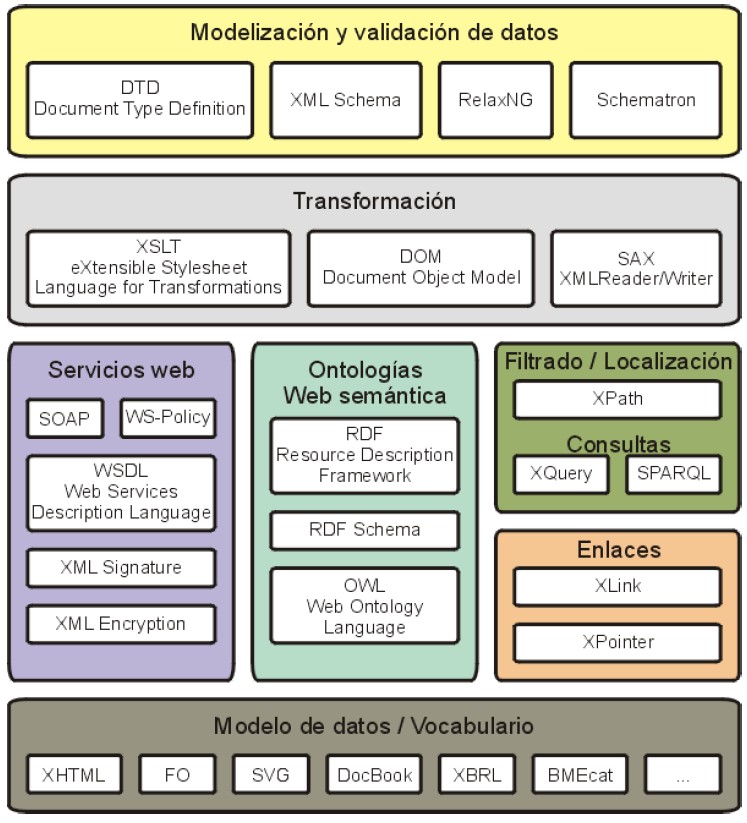
**. Esquema aplicación XML**

El analizador comprobará que el documento esté bien formado y que sea válido, si ocurre algún error lo informará y habrá que corregir el documento XML, si no hay errores, se obtiene un XML válido que se le pasará al procesador XML. En el procesador, se añade junto al documento XML un fichero de transformación como XSLT o un CSS para conseguir el documento final, que puede ser de infinidad de tipos como HTML, XHTML, una base de datos, etc., en función de las reglas que hayamos definido en el documento XML.

XML está relacionado con numerosas tecnologías, tanto para validar como transformar o como modelo de datos, algunas de las normalizaciones recomendadas por W3c y otras organizaciones son las que se pueden ver en la Figura 10.

A la hora de analizar y procesar un XML existen varios tipos de representación y de tecnologías. Algunas de estas representaciones están basadas en memoria como VTD o DOM, que representa un documento XML en forma de árbol, y otras en eventos, como SAX o StAX. Además, en el procesamiento se permite la posibilidad de transformar un documento XML en otro tipo de documento, para ello se usan hojas de transformación como XSLT o CSS.

XML también permite el acceso a la información, para ellos, hay otras herramientas como son XPath, XQuery, etc. Incluso, XML se está introduciendo en el mundo de las bases de datos, está surgiendo bases de datos documentales, que no están formadas por tablas sino por documentos, una de estas bases de datos es mongoDB.



**Figura**

**10**

**. Normalizaciones recomendadas para XML**

## 3.2 Desarrollo de los conceptos

### Tecnologías para el análisis y procesamiento de documentos XML

En este epígrafe, se va a analizar algunas tecnologías y métodos para el análisis y procesamiento de los distintos documentos XML, además, se realizará una comparación entre ellas, para ver, cuáles deben ser usadas en determinadas situaciones.

#### DOM

La primera tecnología para el análisis y procesamiento de XML que se va a estudiar es Document Object Model (DOM, modelo de objetos del documento), define diversas interfaces para analizar y manipular documentos de distinto tipo, como XML, HTML o XHTML. La definición de W3C es: “El DOM de W3C es una plataforma e interfaz de lenguaje neutral que permite a programas y scripts acceder y actualizar dinámicamente el contenido, estructura y estilo de un documento.”

DOM usa una estructura en forma de árbol tal como se puede observar en la Figura 11, donde hay un elemento raíz, que en este ejemplo es la librería, que tiene un hijo elemento libro. Este elemento libro tiene el atributo categoría y 4 hijos: título, autor, año y precio; cada uno de estos elementos son hermanos entre ellos y tienen un Además, DOM permite acceder a su estructura mediante un lenguaje de programación C++, Java, JavaScript, etc. Lo que permite que sea mucho más fácil el acceder y manipular cada nodo del árbol del documento.



**Figura**

**11**

**. Árbol XML DOM**

W3C define 3 niveles de soporte de DOM :

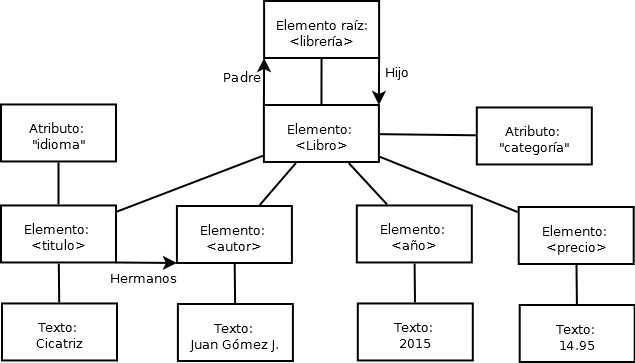
* Nivel 1 (1998): Formado por dos partes: DOM HTML y el núcleo DOM, el cual incluye la funcionalidad para la representación de documentos XML.
* Nivel 2(2000): Permite el acceso a las distintas partes del documento XML como CSS, o eventos, pero no permite acceder a los DTD’s.
* Nivel 3(2004): Incluye las características del nivel 2 y, además, permite validar DTD y XML Schema, o el uso de XPath.

El DOM de XML es un estándar para saber cómo conseguir, cambiar, añadir o eliminar elementos XML. Para ello, proporciona diversos objetos, propiedades y métodos para todos los elementos XML.

Las propiedades XML DOM más típicas son (donde x es un objeto nodo):

* x.nodeName – El nombre de x
* x.nodeValue – El valor de x
* x.parentNode – El nodo padre de x
* x.childNodes – Los nodos hijos de x
* x.attributes – Los nodos atributos de x

20



Los métodos XML más usuales son (donde x es un objeto nodo):

* x.getElementsByTagName(nombre) – Obtiene todos los elementos con una determinada etiqueta
* x.appendChild(nodo) – Inserta el hijo nodo a x
* x.removeChild(nodo) – Elimina el hijo nodo de x

En otras palabras, DOM lee el documento XML, lo procesa y genera un árbol de nodos en el que cada uno representa un elemento del documento. Los tipos de nodos más usuales son:

* Document: Representa todo el documento XML, por eso, solo hay uno por documento. El resto de elementos son hijos de este nodo, cuyo único hijo será el elemento raíz.
* Element: Cada elemento XML es un nodo element. Además, es el único tipo de nodo que puede tener atributos.
* Attribute: Cada atributo XML se representa con un nodo attribute, contienen información sobre un nodo element pero no se consideran un hijo.
* CharacterData: Es el texto del documento XML, puede ser del tipo CDATASection, Text o Comment.

Una vez analizado el documento XML y construido el árbol de nodos se puede añadir o eliminar elementos. Por último, DOM permite serializar el árbol en un fichero o un string para guardarlo. Aunque luego se mencionarán qué tecnologías es mejor utilizar en función de qué situación, hay que indicar que DOM tiene el inconveniente de que siempre tiene el árbol en memoria RAM. Esto implica que si se está trabajando con documentos excesivamente grandes puede ser un problema y ocasionar un desbordamiento de memoria.

#### SAX

Además de DOM, existen otras tecnologías que permiten el análisis y procesamiento de los distintos documentos XML, otra de ella es SAX (Simple API for XML) . Originalmente, surgió como una API para el lenguaje Java, pero ya es un estándar que se puede usar con otros lenguajes de programación.

SAX es un analizador basado en eventos, es decir, define un conjunto de interfaces cuyos métodos son controladores de eventos. A diferencia de DOM, va recorriendo el documento poco a poco y reacciona en función de los eventos que reciba al analizar el documento, como puede ser el inicio de una etiqueta de apertura.

Es por esto que SAX no genera estructuras internas, como DOM hacía con el árbol, ya que, es una interfaz de “*streaming*”, de transmisión de información; conforme va leyendo el documento XML, identifica los tokens, éstos se procesan en el mismo orden en el que aparecen, el analizador indica al programa la naturaleza del token, el programa posee un manejador de eventos al que se invocará mientras se identifican los tokens, hasta llegar a la etiqueta de cierre del elemento raíz.

En la Figura 12 se muestran las interfaces, clases y las relaciones entre ellas de las que dispone SAX . Las 3 interfaces más importantes son: *ContentHandler, DTDHandler y ErrorHandler.*

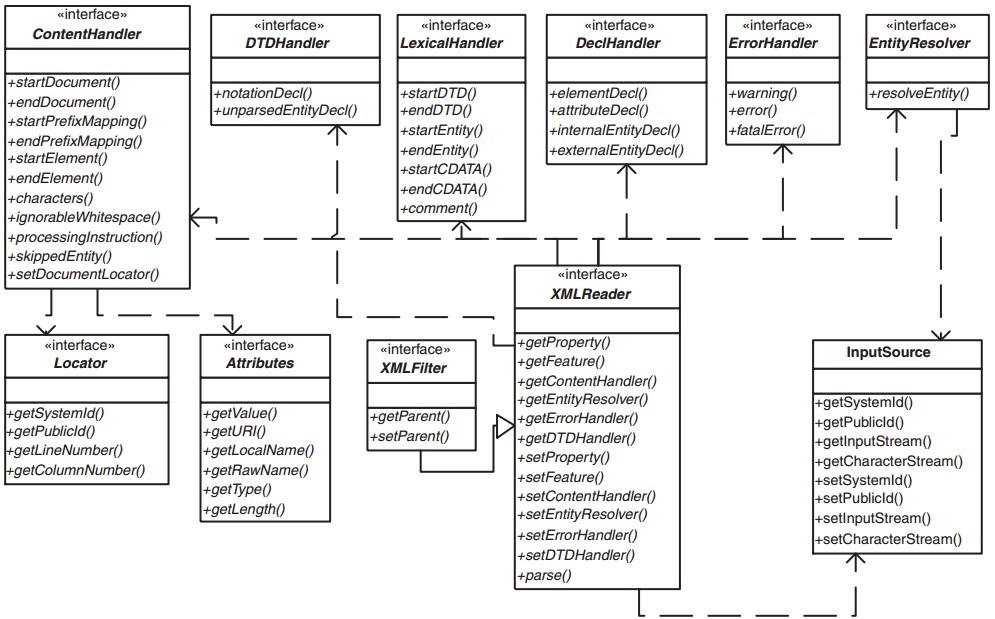


**Figura**

**12**

**. SAX UML**

**.**



Una vez que se ha definido SAX hay que saber cuándo hay que usar esta tecnología sobre DOM, saber qué ventajas y desventajas tiene una sobre otra.

SAX es más rápido y versátil, pero menos potente y requiere más programación que DOM. DOM es más lento y menos versátil, pero una vez que tiene construido el árbol no hay que realizar nada más. SAX requiere de mucha menos memoria ya que analiza el documento en serie, al contrario que DOM, que analiza el documento completo y almacena todo el árbol en memoria. Además, SAX no permite accesos aleatorios, DOM una vez que tiene el árbol se puede acceder rápidamente a cualquier nodo y es más sencillo a la hora de modificar la estructura del documento.

Ya que se han analizado las ventajas y desventajas de cada uno de los métodos hay que indicar en qué situaciones es mejor el uso de una u otra tecnología. DOM almacena el árbol en memoria, así que, habría que tener cuidado si se analiza un documento XML excesivamente grande, y en ese caso, mejor usar SAX. SAX es más recomendable cuando se quiera acceder solo a partes del documento sin ser necesario guardarlo completamente. DOM permite más posibilidades a la hora de crear un documento XML y de modificar su estructura.

Por esto, suele ser más recomendable el uso de DOM o tecnologías basadas en memoria cuando se necesita modificar la estructura de un documento y SAX, o tecnologías basadas en eventos, para transmisión de información.

#### StAX

Hasta ahora, se han visto dos tipos de API para XML, una basado en DOM donde el documento entero se lee y se almacena en un árbol en memoria, y otra basada en eventos (SAX) donde la aplicación recibe eventos conforme se va leyendo el documento. StAX es una tecnología distinta que se asemeja a SAX con algunas diferencias.

SAX tiene un estilo “*Push*”, mientras que, StAX es “*Pull*”, en SAX se analiza un documento XML y cuando se encuentra un token se crea el evento y “empuja” la información del XML al objeto de la aplicación y se notifica al manejador cuando se encuentra un elemento en el XML, es decir, la información fluye del XML a la aplicación cliente.

En StAX hay un cursor que marca un punto dentro del documento XML, la aplicación cliente mueve el cursor por los elementos del documento y “tira” de ellos obteniendo la información que requiera la aplicación. La diferencia es que aquí es la aplicación externa y no el analizador el que indica qué y cuándo hay que coger los datos del documento XML.

Las ventajas y desventajas que dispone StAX con respecto a DOM, son las mismas que tenía SAX, pero si posee algunas nuevas ventajas sobre SAX como son:

* Con el análisis “*pull*” es el cliente quien controla los métodos y el hilo del programa para obtener la información del documento, mientras que, con el estilo “*push*” es el analizador quien controla el envío de información.
* Las librerías y el código que usa StAX es más simple que SAX.
* Los clientes de StAX pueden leer varios documentos a la vez con un mismo hilo.
* StAX puede filtrar documentos XML para no analizar elementos innecesarios para el cliente y soporta vistas para datos que no sean XML.
* A diferencia de SAX que solo puede leer XML, StAX puede tanto leer como escribir XML.

En la Tabla 3, se encuentran las principales diferencias que existen entre las 3 tecnologías vistas hasta ahora .

**Tabla 3. Diferencias entre DOM, SAX y StAX.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Característica** | **StAX** | **SAX** | **DOM** |
| **Tipo API** | *Pull, streaming* | *Push, streaming* | Árbol en memoria |
| **Facilidad de uso** | Alta | Media | Alta |
| **Dispone de XPath** | No | No | Sí |
| **Eficiencia de CPU y**  **Memoria** | Buena | Buena | Variable |
| **Lectura en serie** | Sí | Sí | No |
| **Lee XML** | Sí | Sí | Sí |
| **Escribe XML** | Sí | No | Sí |
| **Crea, lee, actualiza y elimina** | No | No | Sí |

Una vez vistas las diferencias entre las tecnologías, hay que indicar que, si lo que se desea es recorrer y modificar la estructura de un documento, que no sea excesivamente grande, lo ideal es DOM, ya que almacena todo el árbol en memoria y cambiar su jerarquía es sencillo. Mientras que, si se desea acceder solo a ciertas partes del documento o se quiere usar para el envío de información lo ideal es usar StAX. Ya que tiene mejor rendimiento en esos casos respecto a DOM y tiene también ventajas sobre SAX.

Con vistas a una unidad didáctica, lo más recomendable sería que los alumnos aprendieran, por lo menos, una tecnología de cada tipo. Es decir, una que sea basada en memoria como es DOM y otra basada en eventos como es StAX. De esta forma, se estudiarían y se comprobarían de manera práctica las ventajas e inconvenientes de una tecnología respecto a otra.

Generalmente, se habla de dos modelos de programación con XML: *Streaming* o transmisión de información y el modelo DOM.

El modelo DOM analiza y representa todo el documento mediante un árbol en memoria, esto otorga gran flexibilidad a los desarrolladores porque se puede navegar y actuar sobre el árbol de manera muy potente, pero tiene el inconveniente del gran gasto de memoria, especialmente, si son documentos grandes.

El modelo de *streaming* (SAX, StAX) se refiere cuando se transmite conjuntos de información y se analizan en serie, normalmente, en tiempo real y de recursos dinámicos donde no se conoce el contenido anteriormente. Esto implica que solo se puede ver el estado y la localización de un conjunto de datos en un momento determinado y no todo el documento. Estos modelos son muy útiles cuando hay limitaciones de memoria, por ejemplo, en smartphones o aplicaciones que procesan varias peticiones simultáneamente.

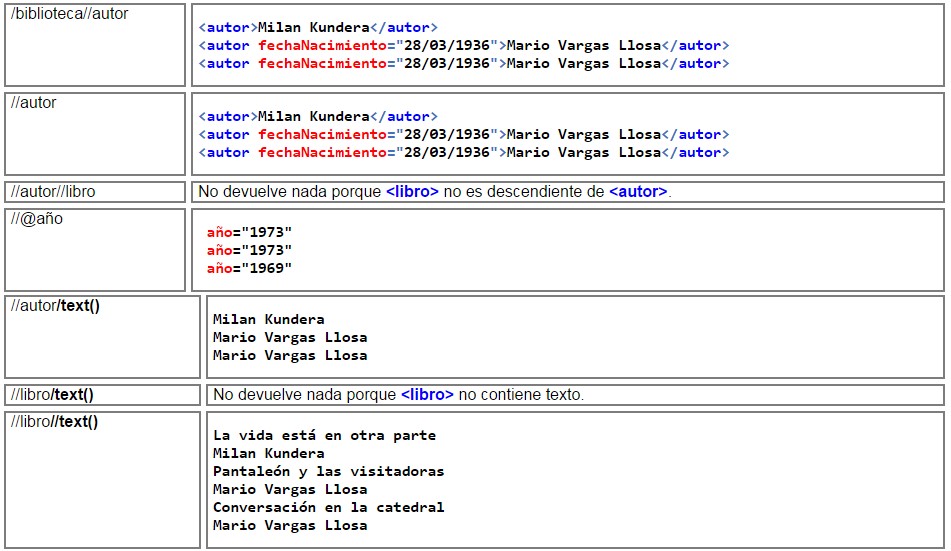
### Acceso a la información en XML

Hasta ahora, se ha visto cómo analizar y procesar documentos XML pero estos ficheros contienen información a la que es necesario acceder. Esto también es muy importante, ya que, hasta existen bases de datos NoSQL que en lugar de tablas usan documentos como XML, JSON o BSON, por lo que, una de las funciones más importantes que tienen los documentos XML es almacenar información. Algunos de los lenguajes para la consulta de información en XML son:

#### XPath

XPath es un lenguaje de consulta que crea expresiones para recorrer y procesar un documento XML, diseñado para ser usado con XLST y XPointer. El propósito principal de XPath es direccionar o acceder a las partes de un XML, además, proporciona métodos para la manipulación de datos como texto, números o booleanos. Usa una sintaxis que no es XML para facilitar su uso con URIs y valores de los atributos de XML.

XPath opera sobre la estructura lógica de un documento XML, es decir, modela XML como un árbol de nodos y navega a través de su jerarquía . Dado el documento XML de la Figura 15, se puede ver un ejemplo de uso de XPath en la Figura 16, donde en la parte izquierda se ven las distintas entradas en lenguaje XPath y a la derecha las salidas que corresponden con lo introducido.

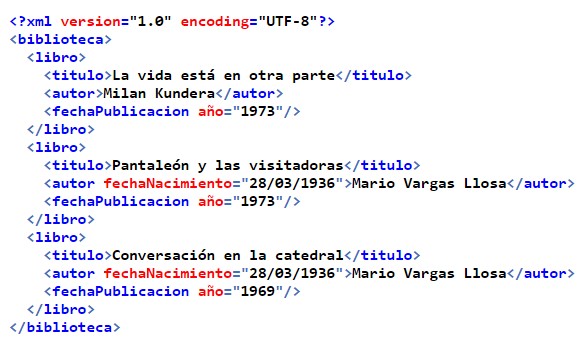


**Figura**

**16**

**. Ejemplo XPath**

30



**Figura**

**15**

**. Ejemplo documento XML**

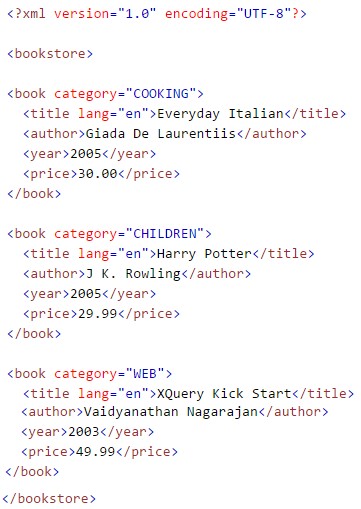
#### XQuery

Se puede decir que XQuery es a XML lo que SQL a las tablas de las bases de datos, está diseñado para hacer consultas de cualquier dato en XML, tanto documentos como bases de datos o cualquier tipo en lo que pueda estar XML. Semánticamente es similar a SQL aunque incluye algunas características para programación, proporciona medios para extraer y manipular información de cualquier fuente representada con XML.

Trabajando sobre el documento XML de la Figura 17, en la Tabla 4 se puede ver un ejemplo de XQuery en la primera columna se muestra la entrada de XQuery y en la siguiente columna aparece cual sería la salida del documento.

**Tabla 4. Ejemplo XQuery.**

|  |  |
| --- | --- |
| **Entrada XQuery** | **Salida** |
| **doc("books.xml")/bookstore/ book/title** | <title lang="en">Everyday Italian</title>  <title lang="en">Harry Potter</title>  <title lang="en">XQuery Kick Start</title> |
| **doc("books.xml")/bookstore/ book[price<30]** | <book category="CHILDREN">  <title lang="en">Harry Potter</title>  <author>J K. Rowling</author>  <year>2005</year>  <price>29.99</price>  </book> |



**Figura**

**17**

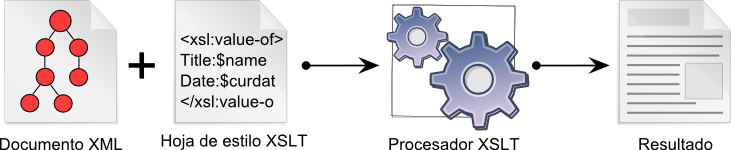
**. Documento XML**

#### XSLT

XLST (*Extensible Stylesheet Language Transformations*) como su nombre indica es una lenguaje para transformar documentos XML en otros documentos XML o con otros formatos como puede ser HTML o PDF. El documento original no se altera, se crea otro nuevo con el formato distinto, además hace uso de XPath para navegar a través del documento XML.

El funcionamiento de XSLT es el que se puede ver en la Figura 18, un documento XML junto con una hoja de estilo o transformación XSLT son las entradas a un procesador XSLT, y éste, obtiene otro documento en el formato indicado en la hoja de transformación.

Tal como se ha visto, estas 3 tecnologías están muy relacionadas entre ellas, así puede verse en la Figura 19, que representa la forma en que están conectadas.



**Figura**

**18**

**. Funcionamiento de XSLT**

# Caso Desarrollado

Usaremos el siguiente código como caso a desarrollar:

01.<?xml version="1.0"?>

02. <empresa>

03. <empleado id="1">

04. <nombre>José Ernesto</nombre>

05. <username>jose</username>

06. <password>321423</password>

07. </empleado>

08. <empleado id="2">

09. <nombre>Daniel Pérez</nombre>

10. <username>dperez</username>

11. <password>433543</password>

12. </empleado>

13. </empresa>

Usaremos las librerías de dom del repositorio de Maven

01. package com.pcalumnos;

02.

03. import java.io.File;

04. import javax.xml.parsers.DocumentBuilder;

05. import javax.xml.parsers.DocumentBuilderFactory;

06. import org.w3c.dom.Document;

07. import org.w3c.dom.Element;

08. import org.w3c.dom.Node;

09. import org.w3c.dom.NodeList;

10.

11. /\*\*

12. \*

13. \* @author alumno

14. \*/

15. public class CrearXml {

16.

17. public static void main(String args[]) {

18. try {

19. File archivo = new File("/ruta/datos.xml");

20. DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

21. DocumentBuilder documentBuilder = dbf.newDocumentBuilder();

22. Document document = documentBuilder.parse(archivo);

23.

24. Document.getDocumentElement().normalise();

25. System.out.println("Elemento raiz: "+ DocumentElement().NodeName());

26. listaEmpleados=document.getElementsByTagName("empleado");

27.

28. for (int temp = 0; temp < listaEmpleados.getLength(); temp++) {

29. Node nodo = listaEmpleados.item(temp);

30. System.out.println("Elemento:" + nodo.getNodeName());

31. if (nodo.getNodeType() == Node.ELEMENT\_NODE) {

32. Element element = (Element) nodo;

33. System.out.println("id: " + element.getAttribute("id"));

34. System.out.println("Nombre: " + element.getElementsByTagName("nombre").item(0).getTextContent());

35. System.out.println("username: " + element.getElementsByTagName("username").item(0).getTextContent());

36. System.out.println("password: " + element.getElementsByTagName("321423").item(0).getTextContent());

37. }

38. }

39. } catch (Exception e) {

40. e.printStackTrace();

41. }

42. }

43.

Como podemos ver DocumentBuilder nos permite parsear el archivo XML.

La función document.getDocumentElement().normalize();  le da al texto un formato visual más agradable pero no es indispensable.

La función de NodeList listaEmpleados = document.getElementsByTagName(«empleado»); es básicamente la de obtener la lista de nodos empleado que vienen en nuestro archivo XML.  Es necesario conocer la estructura de los nodos XML que vienen en el archivo para extraer información especíca.  Después de leer cada nodo empleado es posible obtener los atributos del mismo usando element.getAttribute(«id») o si es necesario el contenido de texto de un nodo particular como en  element.getElementsByTagName(«nombre»).item(0).getTextContent().

# Conclusiones

Una vez finalizada la parte epistemológica, se ha visto que XML es una tecnología muy completa y que es fundamental hoy en día. Surgió hace ya tiempo, ha evolucionado y han surgido numerosas herramientas a sus alrededor, forma parte de la Web en el intercambio de información, en la representación de esta información y en el almacenamiento de los datos y no solo en la web. En otros aspectos, como la prensa o los libros electrónicos también está muy integrado XML, en donde ya es el presente y el futuro.

Se puede decir que XML ha sido vital para el avance de la informática tal como se conoce en la actualidad, y en vista de los próximos avances, como puede ser el Internet de las cosas seguirá teniendo un papel determinante.

Por todos estos motivos, debe tener un papel importante en el curriculum de los alumnos. En módulos como Lenguajes de marcas y sistemas gestores de información se debe enseñar XML por todas las posibilidades que ofrece, para estructurar información, enviar datos usando servicios web o almacenar información. Además, es muy interesante su aprendizaje debido a su gran versatilidad, ya que, se puede transformar la información a otros muchos formatos. Es por todo esto, que la tecnología XML es fundamental hoy en día en la informática.

# Recomendaciones

Una de las mayores ventajas de XML es que tiene infinidad de utilidades, pero esto implica que sea en ocasiones demasiado compleja o pesada para lo que se desea hacer. Este es uno de los mayores problemas que tiene XML, es por este motivo que surgió JSON, otra herramienta para el intercambio de información. JSON tiene algunas ventajas sobre XML como que tiene una sintaxis más simple o es más rápido, y aunque no tiene tantas funciones como XML se usa más en determinados casos.

Por esta razón, uno de los objetivos a superar por XML es reducir su complejidad y mejorar su velocidad. Uno de los avances en este aspecto es una tecnología que surgió hace relativamente poco, en 2012, como MicroXML. En lugar de reemplazar a XML la intención es que MicroXML lo complemente, ya que, es un subconjunto o una simplificación de XML.

Todo documento MicroXML es un documento bien formado en XML, pero MicroXML intenta ser más simple tanto en la sintaxis como en el modelo de datos. [16] Tiene algunas ventajas sobre XML como:  MicroXML no restringe sobre cómo los analizadores se tienen que recuperar de los errores.  Las características que más problemas daban en cuestión de seguridad en XML han sido eliminadas: se elimina la declaración de tipos de documentos, incluyendo declaraciones de entidades, un documento MicroXML contiene todo lo necesario para el correcto funcionamiento, no necesita acceder a ningún recurso externo.

Se ha indicado también que MicroXML simplifica el modelo, para ello el modelo abstracto de datos usa 3 tipos de datos primitivos. Se corresponde con un entero entre 0 y 0x10FFFF, es decir, un carácter representado en UNICODE  Lista: Un tipo estructurado, unlista ordenada que puede contener 0 y varios miembros de cualquier tipo.



