Centro de Estudios de Postgrado

Máster en Profesorado de Educación Secundaria Obligatoria y Bachillerato, Formación Profesional y Enseñanza de Idiomas

UNIVERSIDAD DE JAÉN
Centro de Estudios de Postgrado

Trabajo Fin de Máster

MANIPULACIÓN DE DOCUMENTOS XML: TECNOLOGÍAS PARA EL ANÁLISIS Y PROCESAMIENTOS DE DOCUMENTOS XML; EXTRACCIÓN DE INFORMACIÓN CONTENIDA EN DOCUMENTOS XML

Alumno/a: **Liébana Rosas, Manuel**

Tutor/a: Prof. D. Pedro González García
Prof. ª D. ª Carmen Martínez Cruz

Dpto.: Informática

Junio, 2016

Índice

Índice de Figuras

Índice de Tablas

1. Resumen

1.1. Resumen y palabras clave

Este Trabajo Fin de Máster pertenece al Máster en Profesorado de Educación Secundaria Obligatoria, Bachillerato, Formación Profesional y Enseñanza de Idiomas, 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 divide en dos partes distintas, la primera 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.

El trabajo también incluye una segunda parte en la que se realiza una proyección didáctica, esto es, una contextualización en un centro de secundaria y el desarrollo de una unidad didáctica sobre los contenidos que se han desarrollado en el estudio epistemológico.

PALABRAS CLAVE: documento XML, DTD, DOM, procesar, analizar, información, estructura, unidad didáctica

1.2. Abstract y Keywords

This Final Master Project belongs to the Master in Teaching of Secondary Education, A levels (sixth form), Professional Training (alternative sixth form) and Language Teaching, it has as its main topic the "Manipulating XML documents: technologies for the analysis and processing of XML documents; extracting information contained in XML documents."

The project is divided into two different parts, the first one is an epistemological study that contains background information, a state of art, a concept development, future trends and a conclusion. This first part will be more theoretical about the concepts of XML documents, analysis, processing and technologies used.

The project also includes a second part where there is a teaching unit, that means, a contextualization in a secondary school and the development of a teaching unit about the contents that have been developed in the epistemological study.

KEYWORDS: XML document, DTD, DOM, process, analyze, information, structure, teaching unit

2. 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.

3. Estudio epistemológico

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, tecnologías relacionadas (DOM, SAX, StAX, VTD), una vista al futuro y, por último, unas conclusiones sobre lo desarrollado.

3.1. 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. [1]

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. [1]

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. [2]

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" [3].

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. [2]

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 [4]. Algunos de los lenguajes basados en XML son:

- RSS
- ePUB
- ODF
- SOAP
- SVG
- XHTML

3.2. 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo>
   <libro id="lb101">
      <autor>Gómez Jurado, Juan</autor>
      <titulo>Cicatriz</titulo>
      <genero>Thriller</genero>
      <precio>14.95</precio>
      <fecha_publicacion>15-12-2016</fecha_publicacion>
   </libro>
</catalogo>
```

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. [1]

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 [1]:

- **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.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo>
   <libro id="lb101">
      <autor>Gómez Jurado, Juan</autor>
      <titulo>Cicatriz</titulo>
    <![CDATA[ <color> Amarillo </color> ]]>
   </libro>
</catalogo>
```

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. [5]

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. [1]

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE libro [
    <!ELEMENT libro (titulo)>
    <!ELEMENT titulo (#PCDATA)>
]>
<libro>
    <titulo>Cicatriz</titulo>
</libro>
```

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE libro SYSTEM "libro.dtd">
<libro>
    <titulo>Cicatriz</titulo>
</libro>
```

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 [6], ya definidos para lenguajes determinados, por ejemplo, en la Figura 6, 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

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.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

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.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="libro">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="genero" type="xs:string"/>
      <xs:element name="precio" type="xs:float"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<libro
    xmlns="http://www.w3schools.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:schemaLocation="libro.xsd">

    <titulo>Cicatriz</titulo>
    <autor>Gómez Jurado, Juan</autor>
    <genero>Thriller</genero>
    <precio>14.95</precio>
</libro>
```

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.

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.

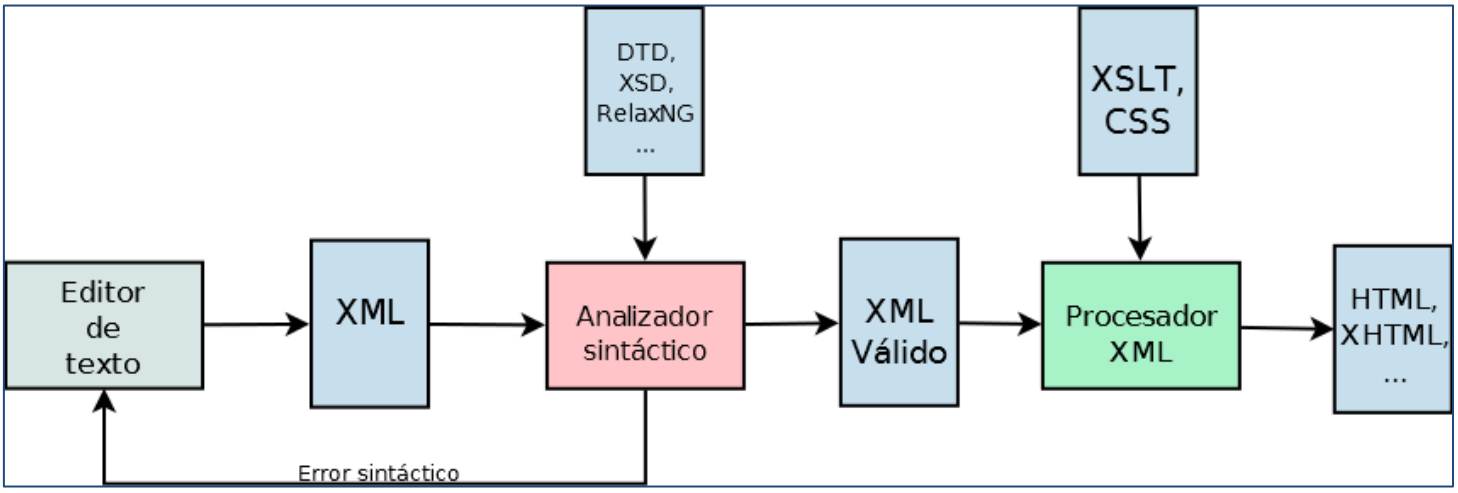

Figura 9. Esquema aplicación XML

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. [5]

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.

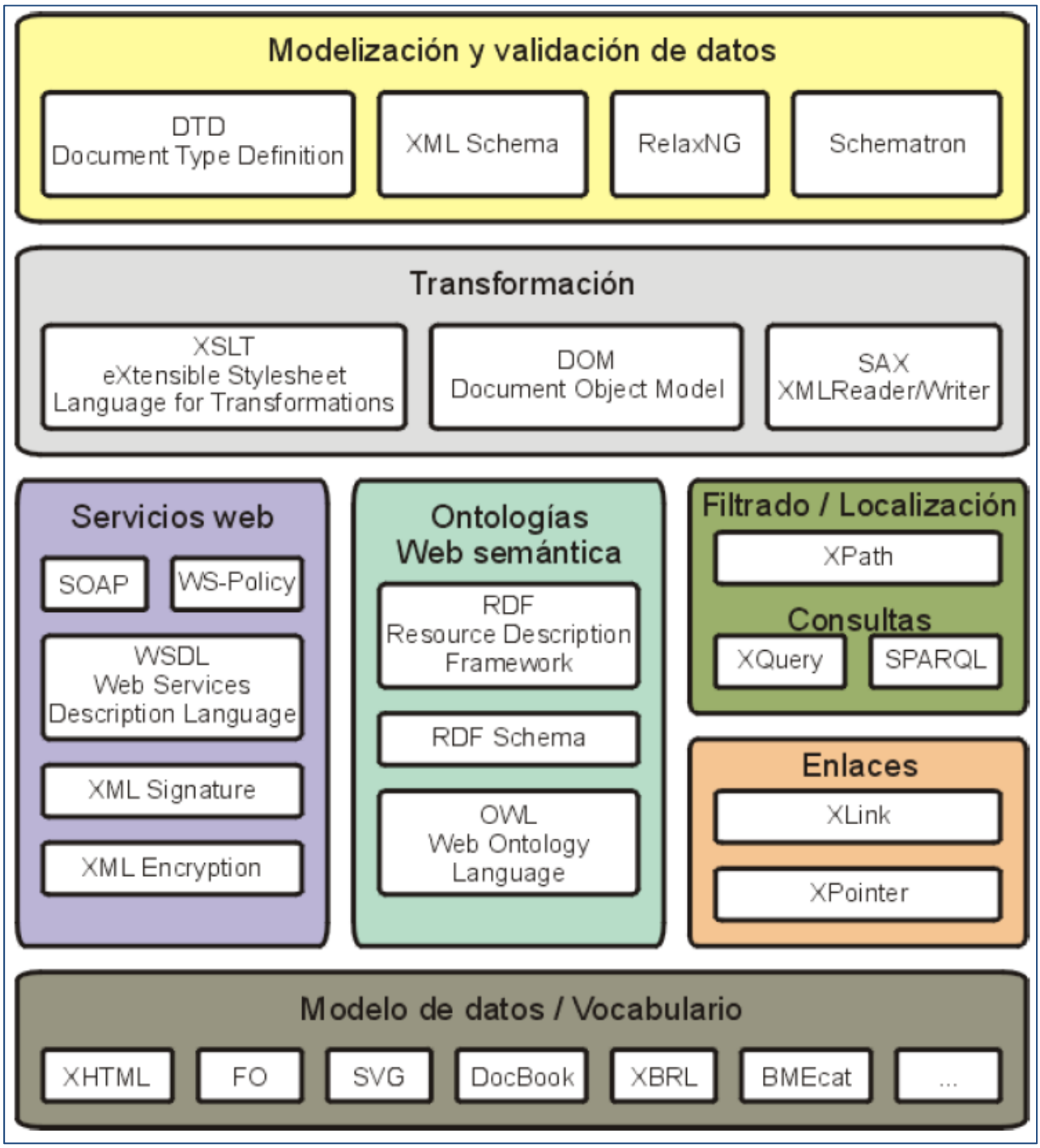

Figura 10. Normalizaciones recomendadas para XML. Fuente: [25]

3.3. Desarrollo de los conceptos

3.3.1. 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. [7] 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." [8]

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 hijo texto donde se indica qué valor contiene cada elemento.

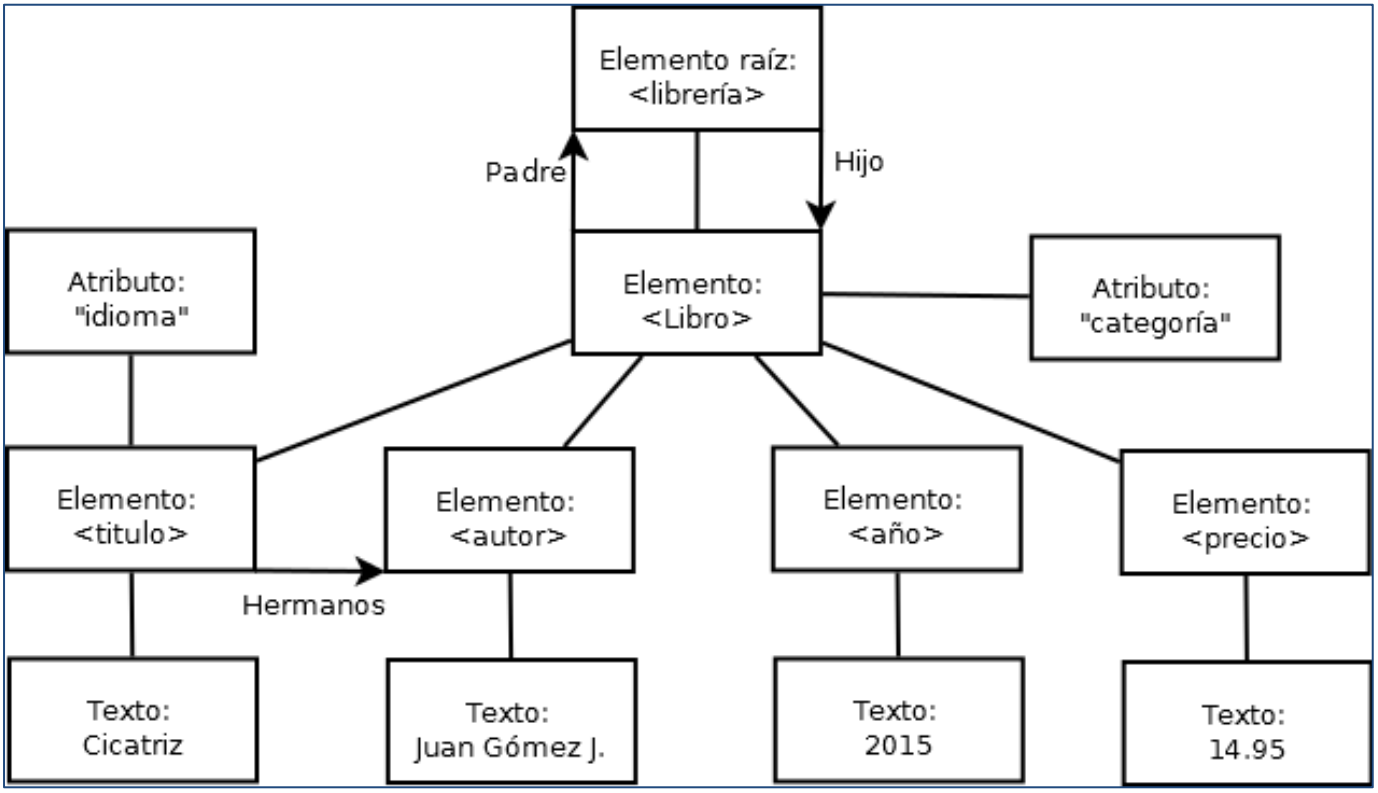

Figura 11. Árbol XML DOM

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.

W3C define 3 niveles de soporte de DOM [7]:

- 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

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) [7]. 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 [5]. Las 3 interfaces más importantes son: *ContentHandler, DTDHandler y ErrorHandler.*

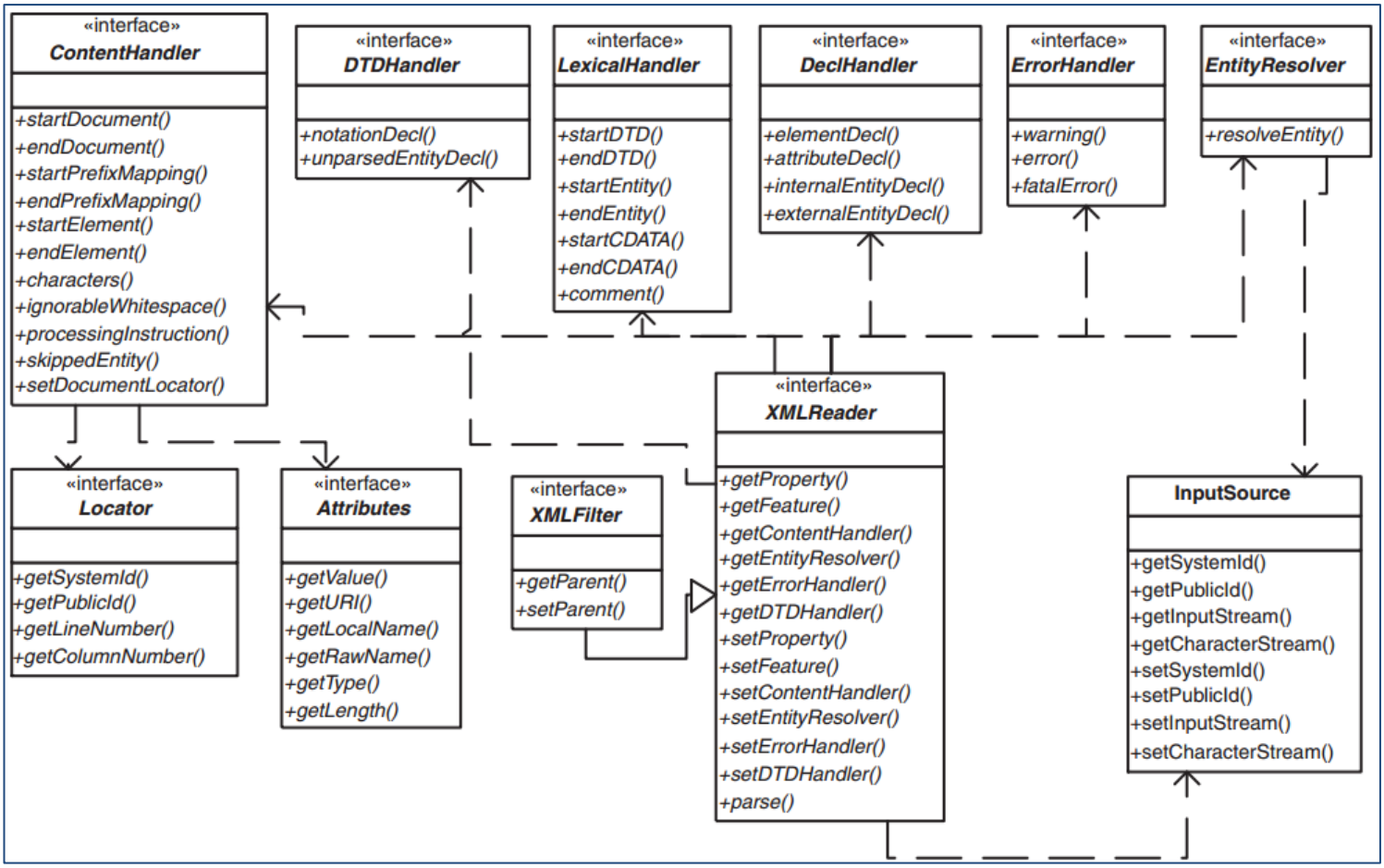

Figura 12. SAX UML. Fuente: [5]

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. [7] 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 [9].

Tabla 3. Diferencias entre DOM, SAX y StAX. Fuente [9]

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.

VTD

En los epígrafes anteriores se han estudiado las tecnologías de DOM y SAX, y se ha visto que ambas tienen sus ventajas e inconvenientes y éstos suelen ser complementarios. Es por eso, que hay casos en los que ninguna de las 2 tecnologías responde todo lo bien que sería deseable, DOM suele otorgar más potencia y permite editar la estructura fácilmente, pero en cuanto hay un documento más grande ya no se puede usar. Y SAX y StAX, aunque sean más veloces y requieran menos memoria, su acceso ordenado al documento hace que muchas veces sea necesario recorrer varias veces el fichero original, además SAX no trabaja bien con XPath ni con XSLT. Por eso, se necesita una alternativa que contenga 3 propiedades:

- Capacidad de acceso aleatorio
- Alto rendimiento
- Bajo uso de memoria

Para ello, se crea VTD-XML (*Virtual Token Descriptor*), otra tecnología para el procesamiento de XML con mejoras sobre DOM y SAX. VTD se basa en una tokenización no extractiva, es decir, VTD retiene en memoria el mensaje XML sin codificar, intacto, y representa los tokens codificados en binario en lo que se llama un *Virtual Token Descriptor*. En la Figura 13, se puede ver que este registro VTD es un entero de 64 bits que codifica el *starting offset* de un token (30 bits), su longitud (20 bits), para algunos tipos de tokens en la longitud se indica un prefijo de longitud y un Qname, la profundidad de anidado del token (8 bits) y, por último, el tipo de token (4 bits). [10]

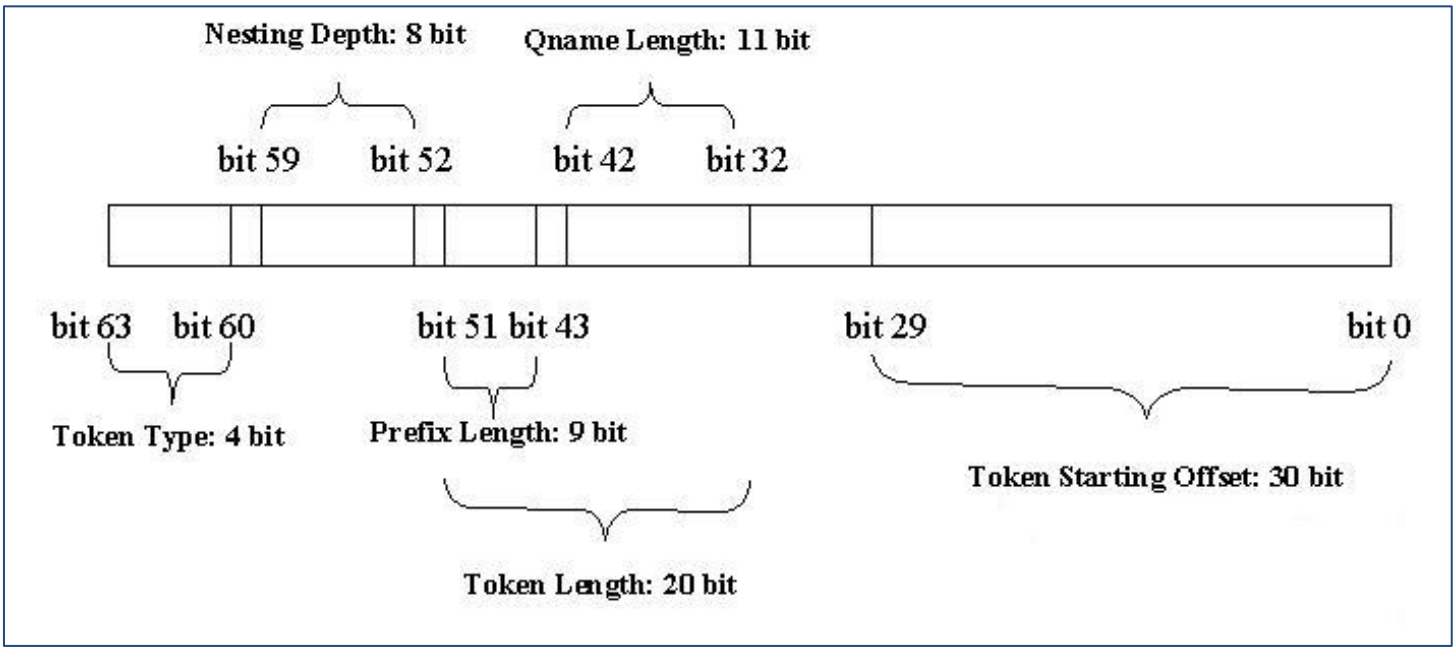

Figura 13. Representación de un VTD. Fuente: [11]

De esta forma, el documento XML se mantiene intacto y sin codificar, y al ser todos registros de enteros de 64 bits se manejan eficientemente como arrays. Para crear una vista jerárquica del documento y poder acceder aleatoriamente a los elementos se hace uso de *Location Caches* (LC). Se organizan en tablas, cada tabla por un nivel de profundidad y cada entrada de la tabla de LC es un entero de 64 bits, los primeros 32 bits identifican el registro VTD que se corresponde al elemento y los 32 siguientes muestran el primer elemento hijo en el siguiente nivel de profundidad en la LC.

Además, en una de las versiones posteriores se añadió otra característica que mejoró considerablemente el rendimiento llamada el buffer de reutilización. Este sistema permite que si la aplicación está continuamente procesando documentos que le van llegando pueda reutilizar los buffers de memoria que se usaron la primera vez, se crean una sola vez y se usan muchas veces. Esto elimina el coste del proceso de creación de objetos y eliminación de basura, que en DOM y SAX supone entre un 50-80%.

En la mayoría de los casos, el almacenamiento de texto es innecesario y hace perder mucha memoria y potencia, por eso, VTD-XML realiza muchas operaciones de texto con registros de VTD. Y al trabajar con enteros de la misma longitud los trata como arrays con las ventajas que eso conlleva en memoria y rapidez.

En la Figura 14, se puede ver una gráfica con la comparación de rendimiento en MB por segundo de las distintas tecnologías de análisis y procesamiento de

documentos XML [11]. Los documentos son de distintos tamaños pero, en general, son grandes y se ve como la tecnología con mejor rendimiento con diferencia es VTD-XML, tanto con reutilización de buffers como sin ella, después un escalón por debajo se encuentran SAX, Piccolo y la tecnología Pull (StAX) y, por último, la más lenta sin apenas llegar a 10MB/s está DOM, que como se comentó anteriormente no era demasiado veloz.

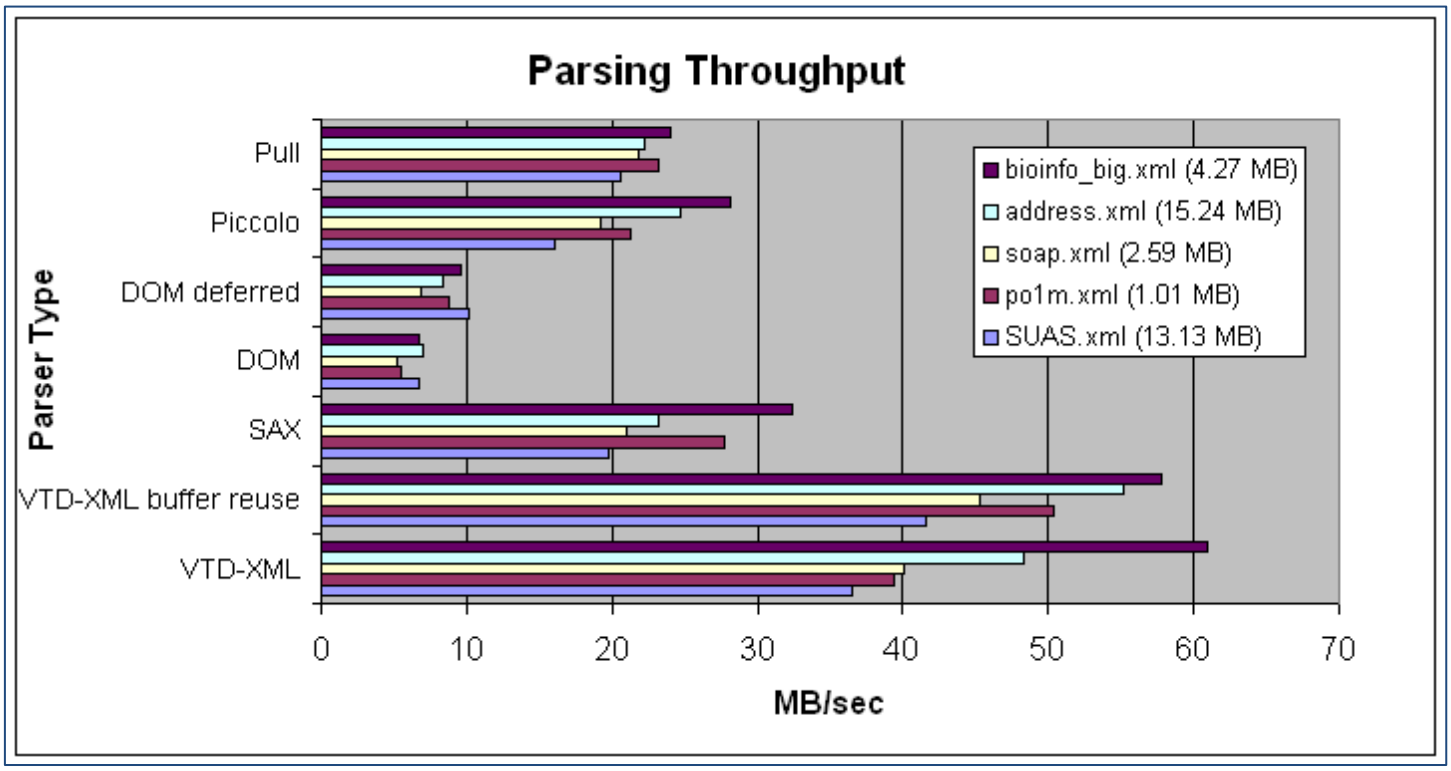

Figura 14. Benchmarking procesadores XML. Fuente: [12]

Para las tecnologías de XML se realizan diversos estudios de benchmarking, otro de estos estudios es el que se puede encontrar en [4]. En donde se obtiene en conclusión que para el manejo de XML, dom4j y DOM son buenas opciones, y aunque es menos flexible en transformaciones OJXQI da buen rendimiento en modificaciones estándar. Pero el array de enteros de VTD es el que da mejores resultados, tanto en uso de memoria como en velocidad en casi todos los tests, comparada con otras APIs basadas en memoria, aunque su uso es más complejo.

Para las API basadas en intercambio de información StAX ha demostrado ser mejor que SAX y XOM, estas API no mantienen la estructura del documento en memoria por lo que no tienen ventajas sobre las anteriores si se desea manipular el orden de los elementos o hacer transformaciones.

Una vez, vistos los resultados del benchmarking parece ser que la solución más veloz es el método VTD-XML, pero es más importante saber qué método es el ideal en cada momento. Aunque VTD sea más rápida, si se quiere estar creando y modificando la estructura de un documento XML es mejor usar DOM, ya que, almacena un árbol en memoria todo el tiempo. Por ello, es muy importante saber el rendimiento de cada uno, pero más importante aún es conocer las ventajas e inconvenientes de unos sobre otros para usar el adecuado en cada situación.

3.3.2. 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 [12]. 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. [13]

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Figura 15. Ejemplo documento XML. Fuente: [13]

/biblioteca//autor	<autor>Milan Kundera</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
//autor	<autor>Milan Kundera</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
//autor//libro	No devuelve nada porque **<libro>** no es descendiente de **<autor>**.
//@año	año="1973" año="1973" año="1969"
//autor/**text()**	Milan Kundera Mario Vargas Llosa Mario Vargas Llosa
//libro/**text()**	No devuelve nada porque **<libro>** no contiene texto.
//libro//**text()**	La vida está en otra parte Milan Kundera Pantaleón y las visitadoras Mario Vargas Llosa Conversación en la catedral Mario Vargas Llosa

Figura 16. Ejemplo XPath. Fuente: [13]

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. Fuente [14]

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>

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

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

</bookstore>
```

Figura 17. Documento XML. Fuente: [15]

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.

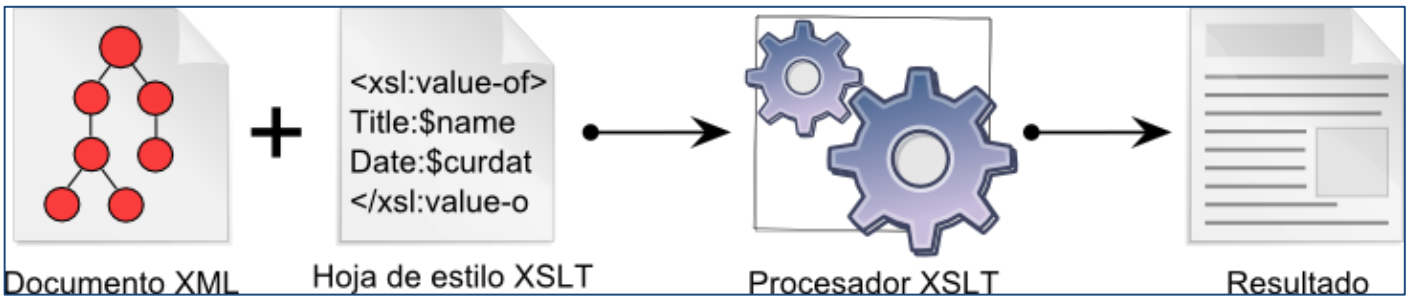

Figura 18. Funcionamiento de XSLT. Fuente: [25]

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.

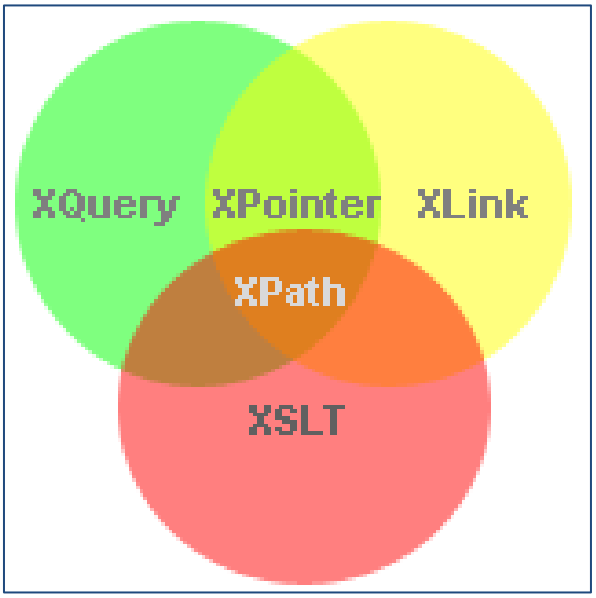

Figura 19. Relación entre tecnologías. Fuente: [15]

3.4. Tendencias futuras

En lo que respecta al futuro, XML no es una tecnología que evolucione demasiado, lo hace poco a poco y, sobre todo, se producen grandes innovaciones en las tecnologías alrededor de ella, algunas como las que se han visto XPath, XQuery, etc. van incorporando cambios y nuevas versiones más a menudo. Las versiones actuales son: XPath 3.1, XQuery 3.1, HTML 5.1, etc.

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. [15]

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:

- Carácter: Se corresponde con un entero entre 0 y 0x10FFFF, es decir, un carácter representado en UNICODE
- Lista: Un tipo estructurado, una lista ordenada que puede contener 0 y varios miembros de cualquier tipo.

- Mapa: Un tipo estructurado, que asocia 0 o varias claves a un valor, tanto la clave como el valor puede ser de cualquier tipo.

De esta manera, un string no es un dato primitivo, sería una lista de caracteres.

Y la construcción de un modelo de datos de más alto nivel es: *element item*. Este *element item* es una lista con 3 miembros:

- Nombre del item
- Un mapa de atributos: puede ser un mapa vacío, cuyas claves son otros nombres de item y los valores son string.
- Una lista de contenido: una lista con 0 o más miembros y cada miembro es un carácter u otro *element item*.

En la Figura 20, se puede ver un documento en XML y en la Figura 21 se puede ver ese mismo documento pero en MicroXML, de esta manera se pueden ver algunas de las diferencias en la sintaxis entre estas dos tecnologías.

```xml
<?xml version="1.0" encoding="utf-8"?>
<a:feed xmlns:a="http://www.w3.org/2005/Atom" xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en"
      xml:base="http://copia.ogbuji.net">
  <a:id>http://copia.ogbuji.net/atom1.0</a:id>
  <a:title>Copia</a:title>
  <a:updated>2005-07-15T12:00:00Z</a:updated>
  <a:author>
    <a:name>Uche Ogbuji</a:name>
    <a:uri>http://uche.ogbuji.net</a:uri>
  </a:author>
  <a:link href="/blog" />
  <a:link rel="self" href="/blog/atom1.0" />
  <a:entry>
    <a:id>http://copia.ogbuji.net/blog/2005-09-16/xhtml</a:id>
    <a:title>XHTML tutorial pubbed</a:title>
    <a:link href="http://copia.posterous.com/xhtml-tutorial-pubbed"/>
    <a:category term="xml"/>
    <a:category term="css"/>
    <a:category term="xhtml"/>
    <a:updated>2005-07-15T12:00:00Z</a:updated>
    <a:content type="xhtml">
      <div>
        <p>
          <a href="http://www.ibm.com/developerworks/edu/x-dw-x-xhtml-i.htm">
            "XHTML, step-by-step"
          </a>
        </p>
        <blockquote>
          <p>Start working with Extensible Hypertext Markup Language. In this tutorial,
          author Uche Ogbuji shows you how to use XHTML in practical Web sites.</p>
        </blockquote>
        <p>In this tutorial</p>
        <ul>
          <li>Tutorial introduction</li>
          <li>Anatomy of an XHTML Web page</li>
          <li>Understand the ground rules</li>
          <li>Replace common HTML idioms</li>
          <li>Some practical considerations</li>
          <li>Wrap up</li>
        </ul>
      </div>
    </a:content>
  </a:entry>
</a:feed>
```

Figura 20. Documento en XML. Fuente: [25]

```xml
<feed lang="en" base="http://copia.ogbuji.net">
  <id>http://copia.ogbuji.net/atom1.0</id>
  <title>Copia</title>
  <updated>2005-07-15T12:00:00Z</updated>
  <author>
    <name>Uche Ogbuji</name>
    <uri>http://uche.ogbuji.net</uri>
  </author>
  <link href="/blog" />
  <link rel="self" href="/blog/atom1.0" />
  <entry>
    <id>http://copia.ogbuji.net/blog/2005-09-16/xhtml</id>
    <title>XHTML tutorial pubbed</title>
    <link href="http://copia.posterous.com/xhtml-tutorial-pubbed"/>
    <category term="xml"/>
    <category term="css"/>
    <category term="xhtml"/>
    <updated>2005-07-15T12:00:00Z</updated>
    <content type="xhtml">
      <div>
        <p>
          <a href="http://www.ibm.com/developerworks/edu/x-dw-x-xhtml-i.htm">
            "XHTML, step-by-step"
          </a>
        </p>
        <blockquote>
          <p>Start working with Extensible Hypertext Markup Language. In this tutorial,
          author Uche Ogbuji shows you how to use XHTML in practical Web sites.</p>
        </blockquote>
        <p>In this tutorial</p>
        <ul>
          <li>Tutorial introduction</li>
          <li>Anatomy of an XHTML Web page</li>
          <li>Understand the ground rules</li>
          <li>Replace common HTML idioms</li>
          <li>Some practical considerations</li>
          <li>Wrap up</li>
        </ul>
      </div>
    </content>

  </entry>
</feed>
```

Figura 21. Documento en MicroXML. Fuente: [25]

XML lleva ya mucho tiempo utilizándose y teniendo un gran impacto en la sociedad y no solo en el mundo de la Web. Esto se ve bien reflejado en el mundo de la prensa y las publicaciones, donde en muchos sitios XML se está incorporando totalmente en sus flujos de trabajo. La Association of American University Presses (AAUP) realizó un *webinar* en donde distintas universidades explicaban cómo son sus

flujos de trabajo, cuáles usaban XML y cómo lo hacían y qué planes tenían para el futuro. [17]

Sobre esto, se hace un caso de estudio en [1], en donde se analiza cuáles son los flujos de trabajo actuales y cómo son los flujos de trabajo con XML, donde se estudian nuevas tecnologías y proyectos que existen como Text Encoding Initiative (TEI), MathML, the Darwin Information Typing Architecture (DITA) y DocBook.

Esto demuestra que XML está muy presente en la actualidad y que está cambiando la forma en que las personas trabajan, no solo en el mundo de la Web, sino en todos los aspectos, tal como se observa con el ejemplo de la prensa. Esto lleva a pensar que XML y sus tecnologías complementarias tienen un gran futuro por delante y que seguirán surgiendo nuevas herramientas que crearán nuevas funciones, además, de corregir algunos problemas que tiene XML como el ya comentado exceso de complejidad en determinados casos.

Por último, el mundo tecnológico se está preparando para un gran avance como es el Internet de las cosas, donde XML también tendrá un importante papel. El internet de las cosas está basado en procesadores pequeños, baratos y de poca potencia insertados en distintos dispositivos. Estos dispositivos no tienen potencia suficiente para interfaces de usuario o servidores por lo que se manejan distribuidamente sobre la red. Existen técnicas basadas en XML para crear interfaces de usuario remotas para el Internet de las cosas. [18]

La tecnología para estas interfaces son los XForms, que originalmente se diseñaron para mejorar el manejo de formularios en la web. Tienen dos partes, la primera es el modelo, donde se especifica los detalles de los datos a coleccionar, de donde viene, su estructura y restricciones, y permite combinar datos de distintas fuentes y enviarlos a distintos sitios. La segunda parte de los XForms es la interfaz de usuario, donde se visualizan los valores, y se especifica el control, modificación y envío de los datos de forma independiente al dispositivo.

Los XForms no es una tecnología nueva, existe desde hace varios años y ya se usa para el control de distintos dispositivos. Pero, con la aparición del Internet de las cosas se tendrán que manejar y controlar infinidad de dispositivos y muchos de ellos se harán con XForms, por lo que, esta tecnología tendrá un nuevo auge, y es posible, que ante tanta demanda XML evolucione por esta rama de la tecnología, para proporcionar mejores servicios.

3.5. Conclusiones

En este estudio epistemológico, se han desarrollado una serie de conceptos sobre el tratamiento de documentos XML. En el estudio de antecedentes, se explicó cuándo y cómo surgió XML, en el estado del arte se ha descrito el estado actual de la tecnología y se ha explicado su funcionamiento, la importancia de un documento bien formado y válido, y distintas formas de definir la estructura de un documento XML, como DTD y XML Schema.

En el apartado de desarrollo de conceptos se entra en detalle en las formas para analizar y procesar documentos, se distingue entre métodos basados en memoria como DOM o VTD, y métodos basados en eventos como SAX StAX. Se ven las ventajas e inconvenientes y se especifica cuándo es mejor el uso uno u otro método. Esto hace ver, que XML tiene muchas utilidades distintas como la transmisión de información, o editar y transformar datos.

Respecto a la transformación de datos, se estudia XSLT, el cual permite a XML conversiones a infinidad de formatos distintos como HTML, XHTML, PDF, etc. Por último en este apartado, se indica que XML también almacena información, y esta información tiene que ser consultada, para ello, se estudia XPath y XQuery. Son lenguajes de consultas sobre los que se obtienen los datos que contienen los documentos XML.

Para terminar, se analiza cuáles son algunas mejoras que tiene que introducir XML y alguna de las tecnologías que pueden conseguir corregir esos defectos. Para ello, se describe cómo es MicroXML y las ventajas que tiene en determinados escenarios sobre XML. Además, se indica que XML está cambiando el flujo de trabajo que existen actualmente en el mundo de la prensa y las publicaciones, donde ahora en algunas empresas se incluye XML.

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.

4. Proyección didáctica

En esta parte de proyección didáctica, se va a realizar una contextualización de un centro de enseñanza secundaria y se va a desarrollar una unidad didáctica. Esta unidad didáctica estará enfocada a enseñar algunos de los conocimientos vistos en la parte epistemológica, es decir, documentos XML y su procesamiento.

4.1. Introducción

La unidad didáctica se impartirá en el centro I.E.S. Virgen del Carmen de Jaén, ya que, este centro tiene formación profesional de ciclo superior que es donde está ubicada la unidad didáctica "Procesamiento de XML".

Primero se hará una contextualización del centro donde se comentará cómo es el centro, el tipo de alumnado del centro y algunas características del entorno, después, se realizará la justificación de la necesidad de enseñar esta unidad. Por último, se desarrollará la unidad didáctica con la normativa a utilizar, los elementos curriculares, objetivos, criterios de evaluación, temporalización, actividades a realizar, metodología, criterios de calificación, criterios de recuperación, temas transversales, atención a la diversidad.

4.1.1. Contextualización

La proyección didáctica se realiza en el centro público I.E.S. Virgen del Carmen, situado en un barrio céntrico de Jaén, el alumnado de este centro pertenece a familias con un nivel socio-económico y cultural medio.

En general, se trata de alumnos cuyas familias se ocupan, fundamentalmente, en el sector servicios, y que se podrían clasificar dentro de un estrato medio de la población en cuanto a sus niveles de ingresos. Pertenecen, normalmente, a familias estructuradas donde la mayoría de los miembros cohabitan en el núcleo familiar, disponiendo de los espacios básicos para poder realizar sus actividades de estudio y de ocio.

Un grupo de alumnos, menor que el anterior, tienen familias dedicadas al sector primario, pero, en general, las características familiares son similares a las expuestas anteriormente.

Un tercer grupo de alumnos son aquellos que, procedentes de otros países, se integran en el instituto con un nivel académico inferior a la media de los estudios que están vigentes en el actual sistema educativo y en el centro, y sobre el cual hay que realizar acciones especiales para su integración tanto personal como escolar.

El centro dispone de E.S.O., bachillerato y formación profesional y en el área de informática dispone de los ciclos de "Desarrollo de Aplicaciones Multiplataforma (DAM)", "Administración de Sistemas Informáticos en Red (ASIR)" y "Sistemas Microinformáticos y en Red (SMR)". La unidad didáctica a desarrollar se encuadra en el módulo de "Lenguajes de marcas y sistemas de gestión de información (LMSGI)" perteneciente al primer curso del ciclo superior de DAM.

Aunque ya se ha indicado el perfil socio-económico de los alumnos y sus familias, se va a especificar el tipo de alumno que se encuentra en los ciclos de grado superior, existen 3 tipos de alumnos:

1. Alumnos que vienen del grado medio. Este grupo de alumnos suele dar muy buenos resultados en la parte más práctica pero tienen más deficiencias con la teoría y conceptos abstractos.
2. Alumnos que vienen del bachiller. Este grupo suele tener un nivel alto, algunos de estos alumnos pueden venir de la universidad en la que han tenido dificultades y no han conseguido acabar.
3. Por último, alumnos muy mayores que quieren "reciclarse" o las personas que lo hacen para conseguir puntos para las oposiciones. Este grupo tiene un nivel muy bueno, bastante por encima del resto de los alumnos.

El centro dispone de la plataforma Moodle, en la que se depositarán todos los contenidos de la unidad didáctica que necesiten los alumnos. También, servirá como medio de comunicación entre alumnos y profesor, y se usará para la entrega de prácticas y resolución de dudas.

4.1.2. Justificación

Como se ha mencionado, se va a encuadrar la unidad didáctica de "Procesamiento de XML" en el módulo de "Lenguajes de marcas y sistemas de gestión de información". Se ha visto en el estudio epistemológico, que XML es una herramienta muy útil y es necesario conocerla en la actualidad debido a su gran versatilidad, es por eso que, en un módulo sobre lenguajes de marcas es recomendable incluir XML, ya que, es junto a HTML los mayores lenguajes de marcado de la actualidad.

En los siguientes apartados, se verá en detalle cómo se hará esta unidad didáctica, se trabajará en cómo procesar un documento XML, tecnologías como DTD, XML Schema, XPath, y XSLT; además, al ser una unidad didáctica intermedia los alumnos ya conocerán la teoría sobre muchos de los conceptos, pero tendrán que poner en práctica lo aprendido.

Se trabajará también con Swing y Java como tema transversal, porque es uno de los lenguajes de programación más usados y también es conveniente que los alumnos aprendan esos conocimientos.

4.2. Unidad didáctica: "Procesamiento de XML"

4.2.1. Normativa

La normativa que regula tanto el título DAM como el módulo LMSGI es:

- Lo dispuesto en el Real Decreto 450/2010, de 16 de abril, por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma (que sustituye a la regulación del título de Técnico Superior en Desarrollo de Aplicaciones Informáticas, contenida en el Real Decreto 1661/1994, de 22 de julio). [19]
- ORDEN de 16 de junio de 2011, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma. BOJA 142 de 21 de Julio de 2011. [20]

4.2.2. Elementos curriculares

Título, etapa, módulo, curso, trimestre, temporalización

Título: "Procesamiento de XML"

Etapa: Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Multiplataforma

Módulo: Lenguajes de Marcas y Sistemas de Gestión de Información

Curso: 1º

Trimestre: 2º trimestre

Temporalización: Se puede ver en la Tabla 5 cómo se va a desarrollar la unidad didáctica en el tiempo. En total se hará en 14 horas.

Tabla 5. Temporalización Unidad didáctica

Semana	Martes	Miércoles
11-15 de abril 2016	2 horas	2 horas
18-22 de abril 2016	2 horas	2 horas
25-29 de abril 2016	2 horas	2 horas
2-6 de mayo 2016		2 horas

Unidad Didáctica	PROCESAMIENTO DE XML		
Real Decreto	Real Decreto 450/2010, de 16 de abril, por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma (que sustituye a la regulación del título de Técnico Superior en Desarrollo de Aplicaciones Informáticas, contenida en el Real Decreto 1661/1994, de 22 de julio). [19]	**Orden**	ORDEN de 16 de junio de 2011, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma. BOJA 142 de 21 de Julio de 2011. [20]
Módulo profesional		Lenguajes de Marcas y Sistemas de Gestión de Información	
Competencia general	La competencia general de este título consiste en desarrollar, implantar, documentar y mantener aplicaciones informáticas multiplataforma, utilizando tecnologías y entornos de desarrollo específicos, garantizando el acceso a los datos de forma segura y cumpliendo los criterios de «usabilidad» y calidad exigidas en los estándares establecidos. [19]		
Resultados de aprendizaje	5. Realiza conversiones sobre documentos XML utilizando técnicas y herramientas de procesamiento.		
Duración	**Objetivos generales**	**Competencias profesionales, personales y sociales**	
14 horas	h) y w)	e), h), t) y w)	

Instrumentos de Información / Evaluación			
Práctica	**Exposición**	**Observación directa**	**Asistencia**
65%	20%	10%	5%

Contenidos	Criterios de Evaluación	Indicadores
1. Conversión de documentos XML, necesidad y ámbitos de aplicación.	a) Se ha identificado la necesidad de la conversión de documentos XML. b) Se han establecido ámbitos de aplicación.	Identifica la necesidad de la conversión de documentos XML Establece ámbitos de aplicación
2. Técnicas de transformación de documentos XML. Tecnologías.	c) Se han analizado las tecnologías implicadas y su modo de funcionamiento.	Analiza las tecnologías implicadas y su modo de funcionamiento.
3. Descripción de la estructura y de la sintaxis.	d) Se ha descrito la sintaxis específica utilizada en la conversión y adaptación de documentos XML.	Describe la sintaxis específica utilizada en la conversión y adaptación de documentos XML.
4. Especificaciones de conversión. Utilización de plantillas.	e) Se han creado especificaciones de conversión	Crea especificaciones de conversión
5. Utilización de herramientas de procesamiento.	f) Se han identificado y caracterizado herramientas específicas relacionadas con la conversión de documentos XML.	Identifica y caracteriza herramientas específicas relacionadas con la conversión de documentos XML.
6. Conversión de formatos de salida.	g) Se han realizado conversiones con distintos formatos de salida.	Realiza conversiones con distintos formatos de salida.
7. Elaboración de documentación.	h) Se han documentado y depurado las especificaciones	Documenta y depura las especificaciones
Orientaciones Pedagógicas		
Este módulo profesional contiene la formación necesaria para desempeñar la función de explotación de sistemas informáticos. La gestión y explotación		

<table>
<tr><td colspan="2">de sistemas de información incluye aspectos como [20]:
• La utilización de lenguajes de marcado en el tratamiento y transmisión de la información.
• La caracterización de la información transmitida y almacenada.
• La adaptación de la información a las tecnologías utilizadas en su presentación, transmisión y almacenamiento.
• El almacenamiento y recuperación de la información.</td></tr>
<tr><td>Temas transversales</td><td>• Programación en Java
• Día del Libro</td></tr>
</table>

Competencias profesionales, personales y sociales

La formación del módulo contribuye a alcanzar las competencias profesionales, personales y sociales de este título que se relacionan a continuación [20]:

e) Desarrollar aplicaciones multiplataforma con acceso a datos utilizando lenguajes, librerías y herramientas adecuados a las especificaciones.

h) Desarrollar interfaces gráficos de usuario interactivos y con la usabilidad adecuada, empleando componentes visuales estándar o implementando componentes visuales específicos.

t) Establecer vías eficaces de relación profesional y comunicación con sus superiores, compañeros y subordinados, respetando la autonomía y competencias de las distintas personas.

w) Mantener el espíritu de innovación y actualización en el ámbito de su trabajo para adaptarse a los cambios tecnológicos y organizativos de su entorno profesional.

Líneas de actuación

Las líneas de actuación en el proceso de enseñanza aprendizaje que permiten alcanzar los objetivos del módulo profesional versarán sobre [20]:

- La caracterización y transmisión de la información utilizando lenguajes de marcado.
- La utilización de técnicas de transformación y adaptación de la información.
- El almacenamiento de la información.

Objetivos generales

La formación del módulo contribuye a alcanzar los objetivos generales de este ciclo formativo que se relacionan a continuación [20]:

h) Emplear herramientas de desarrollo, lenguajes y componentes visuales, siguiendo las especificaciones y verificando interactividad y usabilidad, para desarrollar interfaces gráficos de usuario en aplicaciones multiplataforma.

w) Identificar los cambios tecnológicos y organizativos en su actividad, analizando sus implicaciones en el ámbito de trabajo, para mantener el espíritu de innovación.

Objetivos didácticos

- Identificar la necesidad de la conversión de documentos XML
- Establecer ámbitos de aplicación
- Analizar tecnologías implicadas y su modo de funcionamiento
- Describir la sintaxis específica utilizada en la conversión y adaptación de documentos XML
- Crear especificaciones de conversión
- Identificar y caracterizar herramientas específicas relacionadas con la conversión de documentos XML
- Realizar conversiones con distintos formatos de salida
- Documentar y depurar las especificaciones.

Metodología

El espacio no se especifica porque es el mismo para todas las sesiones, se realiza en el aula de informática. Al ser FP todas las sesiones son de 2 horas.

Sesión	Actividad de Enseñanza-Aprendizaje	Duración (Minutos)	Tipo de Agrupamiento	Recursos materiales
1	**Actividad de presentación-motivación** Presentación de la unidad didáctica. Diálogo en clase sobre transformación y procesamiento de documentos XML	20	Grupo clase	Proyector

	Actividad de conocimientos previos Diálogo general sobre los contenidos de la UD para evaluar las ideas previas	15	Grupo clase	Proyector
	Actividad de desarrollo de contenidos Explicación de los contenidos 1 y 2	50	Grupo clase	Proyector, PC
	Actividad práctica Explicación de la realización de la práctica que evaluará la UD	35	Grupo clase	Proyector, PC
2	**Actividad de desarrollo de contenidos/ práctica** Explicación de los contenidos de la materia transversal Swing y Java para la realización de la práctica	120	Grupo clase	Proyector, PC
3	**Actividad de desarrollo de contenidos** Explicación del contenido 3	45	Grupo clase	Proyector
	Actividad práctica Realización de la práctica incluyendo los contenidos 1,2 y 3 previamente explicados	75	Grupos 4-5 alumnos	Proyector, PC
4	**Actividad de desarrollo de contenidos** Explicación del contenido 4 y 5	60	Grupo clase	Proyector
	Actividad práctica Realización de la práctica incluyendo los contenidos 4 y 5 previamente explicados	60	Grupos 4-5 alumnos	Proyector, PC
5	**Actividad de desarrollo de contenidos** Explicación del contenido 6 y 7	60	Grupo clase	Proyector

	Actividad práctica Realización de la práctica incluyendo los contenidos 6 y 7 previamente explicados	60	Grupos 4-5 alumnos	Proyector, PC
6	**Actividad práctica** Realización de la práctica sobre todos los contenidos de la UD	120	Grupos 4-5 alumnos	Proyector, PC
7	**Actividad de evaluación** Exposición del grupo de trabajo de la práctica realizada	120	Grupos 4-5 alumnos	Proyector, PC

Ahora, se van a explicar cada una de las sesiones con más detalle:

Sesión 1: (12 de abril 2016)

- Actividad de presentación-motivación: Se trata de una actividad de presentación de la unidad didáctica. Se realiza un diálogo en clase donde el profesor explica las ideas principales de la unidad didáctica a desarrollar, que en este caso es sobre transformación y procesamiento de documentos XML. Esta no es la primera unidad del módulo de LMSGI, por lo que, los alumnos ya tienen ciertos conocimientos de XML.
- Actividad de conocimientos previos: Se hace un diálogo general con toda la clase sobre los contenidos de la unidad didáctica para evaluar las ideas previas. El profesor va formulando preguntas a toda la clase para obtener la información que saben los alumnos sobre el tema, de esta forma, descubre el nivel general de conocimientos sobre la materia nueva.
- Actividad de desarrollo de contenidos: El profesor explica los contenidos de:
    - Conversión de documentos XML, necesidad y ámbitos de aplicación.
    - Técnicas de transformación de documentos XML. Tecnologías.

  Para ello, hace uso de transparencias en el proyector, dichas transparencias también las tienen accesibles los alumnos mediante la plataforma Moodle, en donde se guarda toda la información de la unidad didáctica para que los alumnos puedan consultarla siempre que quieran.

- Actividad práctica: Posteriormente vendrá detallado, pero los alumnos realizarán una práctica por grupos de 4-5 alumnos que servirá para evaluar parte de la unidad didáctica. En esta actividad el profesor explicará de qué trata esta práctica, qué es lo que tienen que hacer y los plazos que tendrán para completarla.

Sesión 2: (13 de abril 2016)

- Actividad de desarrollo de contenidos/ práctica: Como tema transversal está la programación en Java, por lo que, la práctica que tienen que realizar los alumnos se hará en ese lenguaje. Lo que tienen que hacer los alumnos es un procesador de XML, es decir, un programa Java con interfaz de usuario hecha en Swing, donde se permita validar documentos XML por DTD y XML Schema, hacer consultas en XPath y transformaciones mediante XSLT. Para ello, harán uso de la API Saxon9 la cual permite realizar todos estos procesos sobre documentos XML, los alumnos deberán integrar esta API y hacer uso de sus funciones

  Esta sesión se utilizará entera para la explicación de programación con Swing, ya que, los alumnos sí saben Java pero aún no han visto Swing. Por lo que, en esta sesión el profesor irá haciendo con su ordenador y el proyector, las bases de la aplicación a desarrollar y, a la vez, enseña a los alumnos las principales características de Swing. Los alumnos irán creando la aplicación al mismo tiempo en sus ordenadores siguiendo las indicaciones que les da el profesor.

Sesión 3: (19 de abril 2016)

- Actividad de desarrollo de contenidos: En esta actividad el profesor realiza la explicación sobre “descripción de la estructura y de la sintaxis en la conversión de los documentos XML”. Para ello, hace uso del proyector donde usa unas transparencias sobre el tema.
- Actividad práctica: En la siguiente parte de la sesión se hace la práctica explicada el día anterior. En este caso, se comienza realizando la parte correspondiente a validar mediante DTD y XSD, donde los alumnos tienen que cargar un documento XML y el fichero de validación y crear las funciones que validen esos documentos mediante la API de Saxon9. El profesor estará ayudando y resolviendo las dudas que se presenten durante la actividad. La parte de validar mediante DTD y XSD quedará de forma similar a la Figura 22 y la Figura 23 respectivamente.

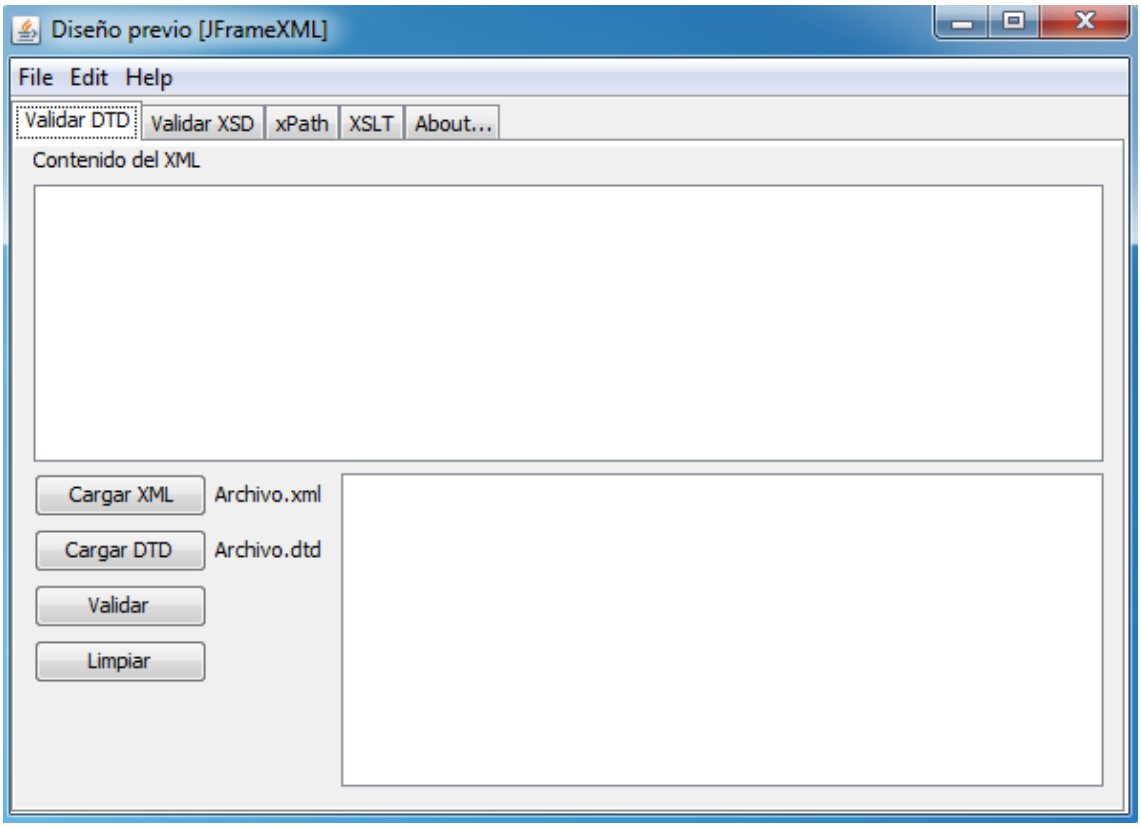

Figura 22. Práctica Validar DTD

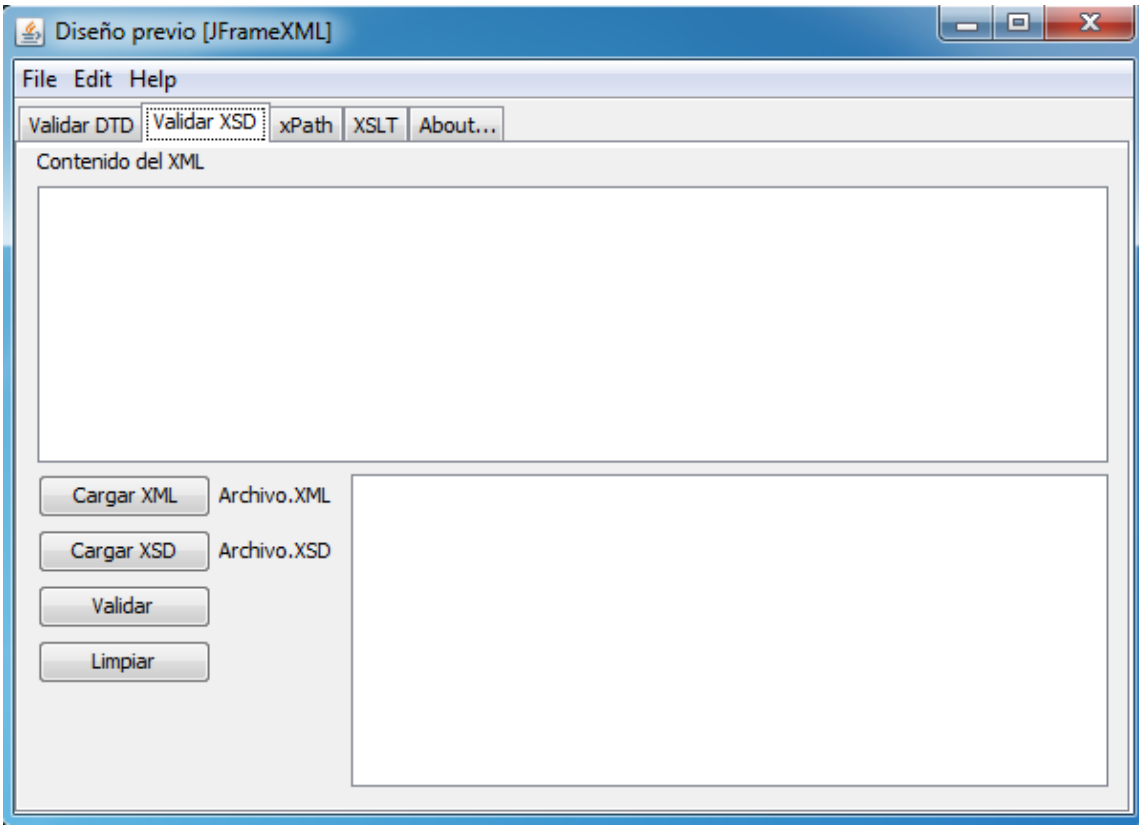

Figura 23. Práctica Validar XSD

Sesión 4: (20 de abril 2016)

- Actividad de desarrollo de contenidos: En esta actividad el profesor realiza la explicación del contenido 4 y 5:
    - Especificaciones de conversión. Utilización de plantillas.
    - Utilización de herramientas de procesamiento.

  Para ello, hace uso del proyector y las transparencias sobre la materia, que todos los alumnos tienen disponibles

- Actividad práctica: En esta actividad los alumnos siguen realizando su práctica, en esta sesión tienen que añadir las consultas de XPath, por eso, tienen que cargar un documento XML e introducir la consulta que quieran hacer, la salida se mostrará por la propia aplicación. El profesor estará ayudando y resolviendo las dudas que se presenten durante la actividad. La práctica realizada en esta sesión será similar a la Figura 24

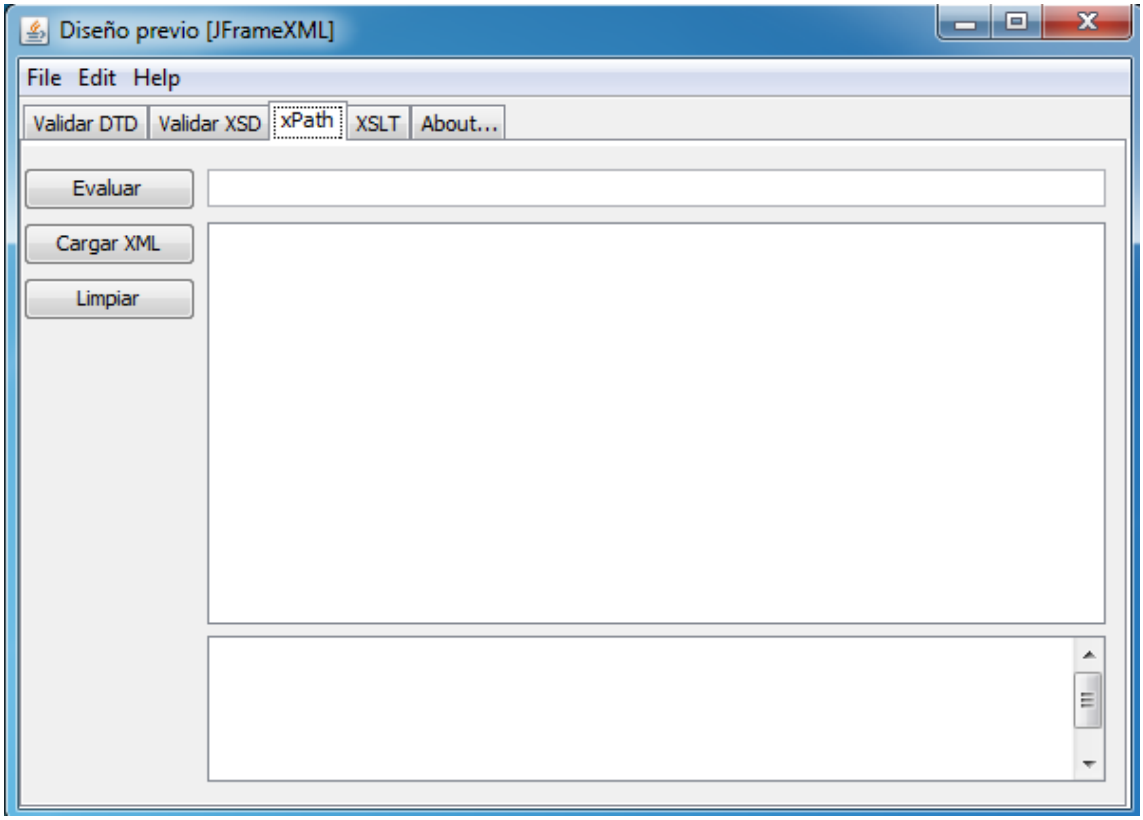

Figura 24. Práctica consulta XPath

Sesión 5: (26 de abril 2016)

- Actividad de desarrollo de contenidos: En esta actividad el docente realiza la explicación de los últimos contenidos teóricos de la unidad didáctica:

- Conversión de formatos de salida.
- Elaboración de documentación.

- Actividad práctica: Una vez explicados todos los contenidos sobre la transformación de XML, los alumnos realizan esta actividad práctica. En la práctica que están desarrollando tienen que hacer la parte de transformación con XLST, para ello, cargan el documento XML y el fichero XLST y tienen que programar la función que realice la conversión al formato indicado.

  El profesor estará ayudando y resolviendo las dudas que se presenten durante la actividad. La parte de transformación con XLST se verá de forma semejante a la Figura 25.

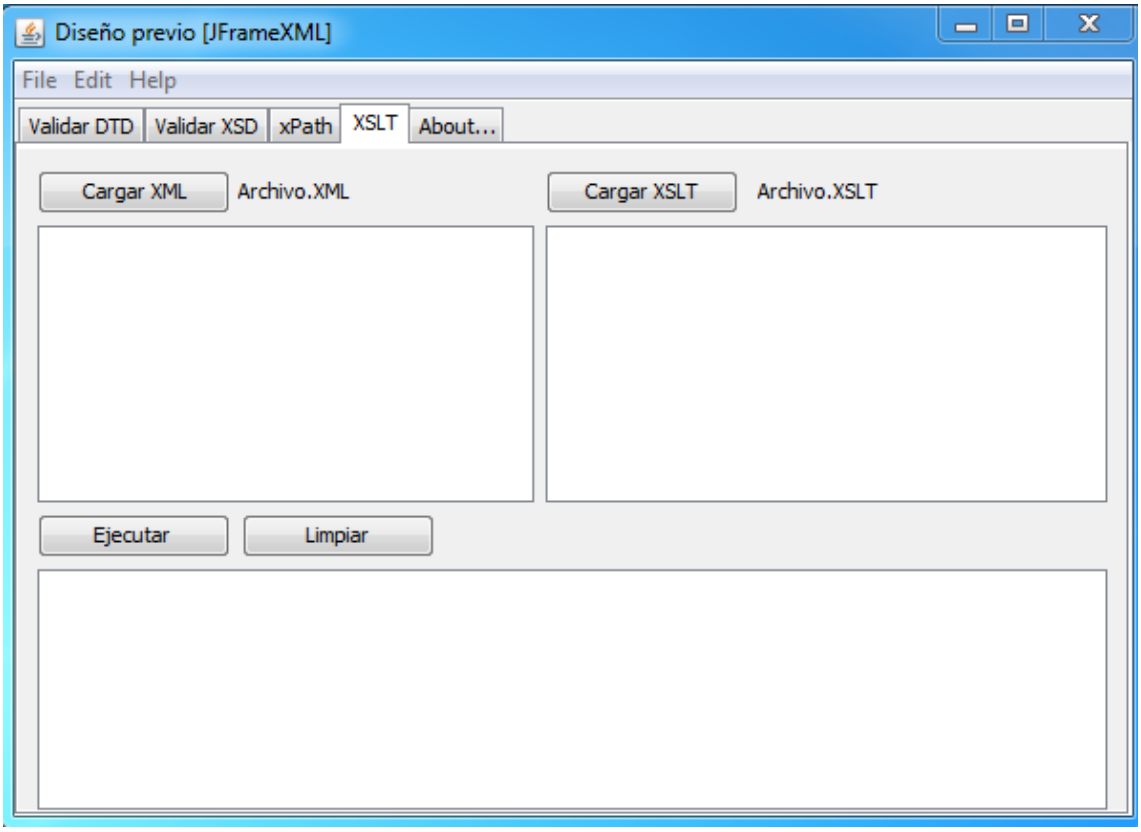

Figura 25. Práctica Transformación XSLT

Sesión 6: (27 de abril 2016)

- Actividad práctica: Esta sesión se dedica completamente a esta actividad práctica, en la que los alumnos tienen 2 horas para terminar las partes que no haya acabado de la práctica. Además, deben realizar la documentación de todo

el programa. El profesor estará ayudando y resolviendo las dudas que se presenten durante la actividad.

Sesión 7: (4 de mayo 2016)

- Actividad de evaluación: En esta actividad de la última sesión, los alumnos de cada grupo de trabajo deben hacer una presentación y una exposición oral a toda la clase sobre la práctica que han realizado durante toda la unidad didáctica. Además, contestarán individualmente a las preguntas que les realice el profesor sobre la práctica

Criterios de calificación

La calificación total de la unidad didáctica se divide en la media ponderada entre:

- Práctica (65%)

  Esta parte califica la práctica realizada durante toda la unidad didáctica.

  - La práctica realizada entre grupos de 4-5 alumnos se calificará entre 0 y 10 y todos los integrantes del grupo tendrán la misma nota en esta parte.
  - Las prácticas no entregadas o entregadas fuera de plazo se calificarán con 0.

- Exposición oral de la práctica (20%)

  Este apartado da nota a la presentación y exposición oral que hacen los alumnos de los grupos sobre la práctica que han hecho

  - Se evaluará tanto la presentación como la exposición oral, se puntuará entre 0 y 10. Los alumnos de cada grupo tendrán una nota distinta, dependiendo de la exposición de cada uno y las respuestas a las preguntas del profesor.
  - La falta injustificada a la exposición supondrá un 0.
  - Los alumnos con faltas justificadas acordarán con el profesor un día alternativo para realizar la exposición.

- Observación directa (10%)

Esa parte califica el comportamiento, participación y trabajo en clase del alumno.

  - Se mide entre 0 y 10 y los alumnos empiezan todos con un 5.
  - Cada punto positivo significa 1 punto más en esta parte y cada punto negativo 1 menos.
  - Si existe alguna falta de respeto o parte disciplinario el alumno tiene un 0 directamente en esta parte.

- Asistencia (5%)

  En este apartado se puntúa la asistencia

  - Los alumnos que hayan acudido como mínimo al 80% de las horas de clase dispondrán del punto de asistencia, los que no lo hayan hecho tendrán un 0 en esta parte.

- Para que se haga la media el alumno debe tener mínimo un 4 tanto en la parte de práctica como en la de exposición oral.
- Y para que la nota de observación directa y asistencia cuenten el alumno tiene que tener mínimo un 4 sumando la parte de práctica y exposición oral.

Criterios de recuperación

Si el alumno ha suspendido la evaluación podrá recuperarla siguiendo los siguientes criterios:

- Prueba Práctica (60%)
  - El alumno debe realizar una práctica similar a la realizada durante la unidad didáctica pero eliminando las funciones de validar DTD y XSD, para compensar el hacer el programa individualmente.
  - Se puntúa de 0 a 10 y para hacer media el alumno tiene que obtener mínimo un 4.
  - Si no se entrega la práctica o se entrega fuera de plazo se califica con 0.
- Prueba Escrita (40%)
  - El alumno hace un examen escrito sobre los conceptos que se explicaron en la unidad didáctica

- Se puntúa de 0 a 10 y para hacer media el alumno tiene que obtener mínimo un 4.

La puntuación máxima que puede obtener el alumno en recuperación será de 9

Temas transversales

En la unidad didáctica se tratan dos temas transversales:

- **Programación en Java:** Para que los alumnos aprendan el lenguaje Java, éste se introduce como materia transversal. De esta manera, se trabaja con lenguajes de marcas, XML y a la vez se enseña a programar en Java. Esto se realiza en la parte práctica donde los alumnos tienen que crear un procesador de XML en Java y Swing.
- **Día del libro:** La unidad didáctica coincide con el día del libro, por ello, se introduce como tema transversal. Se fomentará la lectura entre el alumnado, para esto, durante la unidad didáctica se utilizarán documentos XML que contengan información sobre libros. Se usará un "libros.xml" en el que se estructurará la información de los libros añadiendo el autor, nombre, editorial, precio, etc.

Atención a la diversidad

La unidad didáctica pertenece al ciclo superior de FP, esto significa que no se pueden hacer adaptaciones significativas a los alumnos, es decir, no se puede cambiar el currículo.

En este caso, existe un alumno con problemas auditivos, su audición es reducida pero no inexistente. Para ello se han tomado algunas medidas:

- Se le sitúa al principio de la clase para que no tenga problemas al escuchar las explicaciones del profesor.
- En las explicaciones se habla con un tono elevado

- El trabajo en grupo hace que los demás compañeros le sirvan de apoyo a la hora de realizar las actividades.
- En los exámenes o cuando el alumno lo requiera se le repetirán las indicaciones hasta que el alumno las haya oído.

Con estas medidas se resuelve el problema, ya que, el alumno escucha todo lo que ocurre en clase. No se ha modificado el currículo en ningún momento, por lo que, son adaptaciones no significativas, que son las únicas posibles en FP.

Además de esta adaptación, el profesor estará pendiente para que todos los alumnos lleven un ritmo normal en clase y comprobar si existe algún otro alumno que requiera alguna adaptación.

5. Bibliografía

[1] J. Fawcett, L. R.E. Quin, and D. Ayers, *Beginning XML*, 5th ed. Indianapolis, USA: Wrox, 2012.

[2] R. Eckstein and M. Casabianca, *XML Pocket Reference*, 2nd ed. Sebastopol, USA: O'Reilly, 2001.

[3] W3C. (2006, Agosto) Extensible Markup Language (XML) 1.0 (Fourth Edition). [Online]. https://www.w3.org/TR/2006/REC-xml-20060816/

[4] B. Oliveira, V. Santos, and O. Belo, "Processing XML with Java – A Performance Benchmark," *International Journal of New Computer Architectures and their Applications*, vol. III, no. 1, pp. 72-85, 2013.

[5] A. Skonnard and M. Gudgin, *Essential XML Quick Reference*. USA: Addison Wesley, 2001.

[6] E. Rusty Harold, *XML 1.1 Bible*, 3rd ed. Indianapolis, USA: Wiley, 2004.

[7] B. Benz and J. R. Durant, *XML Programming Bible*, 1st ed. Indianapolis, USA: Wiley, 2003.

[8] W3Schools. XML DOM Tutorial. [Online]. http://www.w3schools.com/xml/dom_intro.asp

[9] Oracle. The Java Tutorials. [Online]. https://docs.oracle.com/javase/tutorial/jaxp/stax/why.html

[10] Sourceforge. (2004, Agosto) VTD-XML: The Future of XML Processing. [Online]. http://vtd-xml.sourceforge.net/technical/2.html

[11] XimpleWare. XML Parsing Performance Benchmark of VTD-XML 1.5. [Online]. http://www.ximpleware.com/benchmark1.html

[12] W3C. (2015, Septiembre) XML Path Language (XPath). [Online]. https://www.w3.org/TR/xpath/

[13] M. Bartolomé Sintes. (2015, Mayo) XPath: XML Path language. [Online]. http://www.mclibre.org/consultar/xml/lecciones/xml_xpath.html

[14] W3Schools. XQuery Example. [Online]. http://www.w3schools.com/xsl/xquery_example.asp

[15] S. Kattau. (2012, Julio) MicroXML: The future of XML? [Online]. http://sdtimes.com/microxml-the-future-of-xml/3/

[16] MicroXML Community Group. (2012) MicroXML. [Online]. https://dvcs.w3.org/hg/microxml/raw-file/tip/spec/microxml.html

[17] Association of American University Presses. (2016, Febrero) The future of XML. [Online]. https://aaup.adobeconnect.com/p24m8vd3ab3/?launcher=false&fcsContent=true&pbMode=normal

[18] S. Pemberton, "XML Interfaces to the Internet of Things with XForms," in *XML LONDON 2015*, Londres, 2015, pp. 163-168.

[19] Gobierno de España. (2010, Mayo) BOE num. 123. [Online]. https://www.boe.es/boe/dias/2010/05/20/pdfs/BOE-A-2010-8067.pdf

[20] Junta de Andalucía. (2011, Julio) BOJA num.142. [Online]. http://www.juntadeandalucia.es/boja/2011/142/d20.pdf

[21] M. Bartolomé Sintes. (2016, Mayo) XML: Lenguaje de Marcas Extensible. [Online]. http://www.mclibre.org/consultar/xml/lecciones/xml_quees.html

[22] M. Bartolomé Sintes. (2015, Mayo) XSLT: Transformaciones XSL. [Online]. http://www.mclibre.org/consultar/xml/lecciones/xml_xslt.html

[23] U. Ogbuji. (2012, Mayo) Introducing MicroXML, Part 1: Explore the basic principles of MicroXML. [Online]. http://www.ibm.com/developerworks/library/x-microxml1/x-microxml1-pdf.pdf