8.4

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE CONSULTA Y VISUALIZACION, BASADO EN XSLT, DE DOCUMENTOS XML GENERADOS POR EL EDITOR COLABORATIVO ELXI.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

PABLO MIGUEL OROZCO SUAREZ

ASESOR: DR. MANUEL ROMERO SALCEDO

MEXICO, D. F. 2002

INDICE

AGRADECIMIENTOS

Quiero agradecer a mis padres por el ejemplo y el apoyo que siempre me han dado. Aprovecho para decir que este logro también les pertenece a ellos.

También agradezco a la Universidad Nacional Autónoma de México por ofrecerme la mejor formación profesional que he podido tener. Quiero extender este agradecimiento a todos y cada uno de los profesores de la Facultad de Ingeniería que contribuyeron a este logro con su enseñanza.

Sólo me resta agradecer al Dr. Manuel Romero Salcedo por su dirección. Sin su participación la realización de este trabajo no hubiera podido llevarse a cabo.

Pablo M. Orozco Suárez

CAPÍTULO 1

INTRODUCCIÓN

1.1 ANTECEDENTES

En los últimos años, el mundo ha sido testigo de una vertiginosa evolución tecnológica que se ha manifestado en diversos ámbitos. En materia de cómputo, por ejemplo, estamos llenos de realidades que hace sólo algunos años eran imposibles de imaginar. Conceptos como realidad virtual, Internet, comercio electrónico, inteligencia artificial, entre muchos otros, han dejado de ser tópicos futuristas y se han convertido en parte de nuestra vida diaria.

Una de las consecuencias más significativas que ha traído consigo esta evolución es la "globalización", el mundo de hoy es un mundo electrónicamente globalizado a causa de una red mundial llamada WWW[1] (también conocida como la Web o Internet).

La Web, desde un inicio, se ha distinguido por el enorme número de usuarios que, día con día, deciden incorporarse a ésta. Como es de suponerse, al igual que el enorme número de usuarios la cantidad de información que fluye por este medio también ha ido en aumento considerablemente. Debido a esto, la necesidad de innovar tecnologías, capaces de lograr un manejo de datos más eficiente, para lograr un intercambio de información más efectivo, va en aumento cada día. Esto se debe, principalmente, a todos los formatos diferentes en que la información se puede presentar en la actualidad (desde una hoja de papel hasta una presentación multimedia). Además, los diferentes medios o canales por los que esta información debe ser transmitida, a causa de la enorme demanda que tiene, cada vez son más variados.

Para resolver este problema, en los últimos años, ha surgido una tecnología que propone estructurar toda esa información. Basada en una sintaxis para árboles de datos, que es muy adecuada para representar datos procedentes de fuentes heterogéneas, como es el caso de toda la información contenida en la Web, XML[2] (*Lenguaje de Marcado eXtensible*), se ha convertido en el estándar para el intercambio de información entre aplicaciones Web.

1.2 DEFINICIÓN DEL PROBLEMA

Actualmente, diversos centros de investigación de todo el mundo se encuentran trabajando en el desarrollo, e incluso en la aplicación, de tecnologías basadas en

[1] Del término en inglés "*World Wide Web*".
[2] Del término en inglés "*eXtensible Markup Language*".

XML. Por ejemplo, en el Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS) de la UNAM se desarrolló un sistema groupware[3], llamado ELXI (*Editor coLaborativo de documentos XML en Internet*), que asiste a grupos de autores en la elaboración de documentos XML de manera colaborativa.

El presente trabajo de tesis, dirigido por el Dr. Manuel Romero Salcedo, al igual que ELXI, forma parte del Proyecto de Investigación "*Ambiente Multipropósito de Edición Colaborativa en Internet e Intranet*" del CONACYT con número de referencia 400316-5-J32043-A.

El sistema de visualización y consulta desarrollado en la presente tesis, pretende, tomando la versión 1.0 de ELXI como punto de partida, donde el tipo de documentos que se maneja son artículos de investigación, dar solución a la problemática siguiente:

Un medio para organizar y presentar, de una forma coherente al usuario, la totalidad de documentos generados por ELXI, proporcionando además a estos documentos, de manera individual, los estilos de formato que puedan requerir para su publicación.

Considerando que la parte fundamental de la problemática es brindar un estilo adecuado a los documentos XML, que se tienen previamente generados por ELXI, y atendiendo a las siguientes consideraciones:

- XML es un lenguaje para compartir información y no para dar formato a ésta.
- Los artículos de investigación son documentos que requieren propiedades muy específicas de estilo para su correcta presentación.

Se tomó la decisión de basar el desarrollo del sistema en la tecnología XSLT[4] (*Lenguaje de Transformación Extensible basado en hojas de Estilo*). Como veremos más adelante, XSLT es un lenguaje de programación que resulta ideal para transformar la estructura lógica y física de los documentos XML, permitiéndonos incluso conversiones de formato XML a otros formatos, que son ideales para la publicación electrónica, como HTML[5] o PDF[6]. Si tomamos en cuenta que la parte fundamental de la problemática es brindar un estilo adecuado a los documentos XML, que se tienen previamente generados por ELXI, es una consecuencia lógica pensar en la solución que el propio XML sugiere.

[3] Sistemas de software que soportan y apoyan el trabajo en grupo.
[4] Del término en inglés "*eXtensible Stylesheet Language Transformation*".
[5] Del término en inglés "*HyperText Markup Language*".
[6] Del término en inglés "*Portable Document Format*".

1.3 OBJETIVO

Generar un sistema de consulta y visualización, basado en XSLT, para los diversos documentos XML generados por ELXI. Este sistema deberá ser capaz, por un lado, de convertirse en un medio de acceso y búsqueda eficiente de los diferentes artículos de investigación, y por el otro, constituir la vía para proporcionarles a éstos los distintos estilos de formato que requieren para su publicación.

1.4 ORGANIZACIÓN DEL DOCUMENTO

El presente documento está organizado de la siguiente manera. Posterior a la Introducción (capítulo 1), en el capítulo 2, se detallan los antecedentes de la tecnología XML. Después de este capítulo, en el capítulo 3, se revisan algunos de los conceptos más importantes de dicha tecnología. El desarrollo teórico del trabajo, finaliza con el capítulo 4, donde se presenta un panorama general de la tecnología XSLT. Con base en los conceptos anteriores, en el capítulo 5, se exponen los puntos más significativos del desarrollo práctico del trabajo. Finalmente, las conclusiones y perspectivas de la presente tesis son dadas a conocer en el capítulo 6.

CAPÍTULO 2

ANTECEDENTES DE XML

2.1 INTRODUCCIÓN A LOS LENGUAJES DE MARCAS

XML, sin alguna duda, es el producto de la evolución tecnológica en el procesamiento de texto[7]. Actualmente, el procesamiento de texto ha evolucionado tanto, que cuenta con procesadores o editores de texto[8] de la talla de *Microsoft Word* o *Adobe Pagemaker*, los cuales han estado basados, desde sus inicios, en el método de procesamiento de texto por reproducción[9]. Estos editores de texto, cuentan en la actualidad con una interfaz de usuario muy sofisticada, la cual está diseñada para realizar la reproducción en pantalla de manera idéntica a la presentación final del documento (ya impresa en papel). Este tipo de procesadores de texto son conocidos como editores WYSYWYG[10] (Lo que ves es lo que obtienes). Es posteriormente a estos editores (todavía en uso), que aparece un nuevo tipo de procesamiento de texto, el cual es conocido como marcado de formato o lenguaje de marcas.

Para entender mejor como funciona un lenguaje de marcas, pensemos en un documento cualquiera, como uno de los que estamos acostumbrados a leer en nuestra vida diaria. Este documento, invariablemente, debe tener algún tipo de estructura. Aún los párrafos más simples que integren nuestro documento, contendrán signos de puntuación o espacios en blanco que definirán su estructura. Por ejemplo, sí vemos una frase de un renglón, al principio de un texto y arriba, la interpretamos como el título. La estructura de un texto la deducimos de forma implícita a partir de la distribución explícita del texto en la página. De esta manera nuestro cerebro asocia la apariencia de un texto con su estructura.

En ocasiones, es necesario hacer dichas diferencias un poco más explícitas, agregando información contextual al documento (por ejemplo, el subrayado de una palabra para denotar su importancia o el uso de la palabra *Capítulo* para especificar precisamente el título de un capítulo). Estas marcas, aparte de simplificar el entendimiento del texto escrito, ayudan a evitar ambigüedades.

Ahora pensemos en la versión electrónica de ese mismo documento. En general, las computadoras no pueden extraer información estructural de dichos documentos, por lo tanto, la versión impresa debería ser tan sólo un documento

[7] El procesamiento de texto es una disciplina de la ciencia de la informática dedicada a la creación de sistemas de computadoras capaces de automatizar partes de la creación de un documento y del proceso de edición.
[8] Software que realiza el procesamiento de texto.
[9] Se denomina reproducción, al formato de archivo que contiene la combinación de datos reales del documento y la descripción del formato deseado.
[10] Del término en inglés *"What You See Is What You Get"*.

sin la información adicional, de cambio de fuente o tamaño de letra, a la que estamos tan acostumbrados.

En cierta forma, las versiones impresas de los documentos electrónicos serían equivalentes a las que se producirían con una máquina de escribir. Para resolver este problema, los procesadores de texto agregan marcas al contenido de un documento, donde especifican las características de su versión en papel. Por ejemplo: <negritas> "Prueba" </negritas>, es equivalente a **"Prueba"**, al imprimirse en papel.

En este posible lenguaje de marcas las etiquetas <negritas> y </negritas> describen el tipo de formato. Para este caso, se comenzaría con la orden dada por la etiqueta <negritas>, que indica que a partir de esa marca todo el texto que continua debe imprimirse con letra más obscura, mientras que la orden de desactivar estará dada por la etiqueta </negritas>, que indica el fin de la cadena de texto que será impresa con letra más oscura. En este sentido, la palabra *marca* define todo aquello que es parte de un documento electrónico, pero que no es texto imprimible. Por lo tanto, un lenguaje de marcas está compuesto de reglas que describen como deben interpretarse las marcas, bajo que condiciones están permitidas y que significa cada una de ellas. Los elementos lógicos son, por ejemplo, una cita, el título, una línea de un diálogo o cualquier otra estructura que merezca ser identificada del resto del texto. Las operaciones tipográficas son instrucciones de formato que se aplican a los elementos lógicos, por ejemplo, las citas suelen ser impresas en itálicas, las líneas de diálogo son precedidas por un guión.

Las marcas cumplen dos objetivos primordiales:

- Separar los elementos lógicos de un documento.
- Especificar las operaciones tipográficas que deben ejecutarse sobre dichos elementos lógicos.

Existen dos tipos básicos de lenguajes de marcas, de *Formato* y de *Estructura*. Los lenguajes de formato contienen marcas que describen las operaciones tipográficas a ser aplicadas en cada uno de los elementos del documento. Por ejemplo, el título de un texto debe ser impreso en una sola línea, a la cabeza del documento y con una fuente seis puntos más grande que el resto del texto. Así, los lectores podrán inferir que se trata de un título. En los lenguajes estructurales o de estructura, las marcas describen únicamente la estructura lógica de un documento.

De esta manera, un título puede ser anotado como: <Titulo> Don Quijote <FinDeTitulo>. Si bien, este ejemplo dice que el título es "Don Quijote"', no nos dice como debe aparecer. Las marcas estructurales especifican el tipo de elemento y no su apariencia. La mayor parte de los lenguajes de marcas son una combinación de ambos tipos. Por ejemplo, *Microsoft Word* incluye marcas con las

que se puede definir la estructura del documento y marcas que describen como debe ser impreso este documento, es decir, información de formato y de estructura.

Las marcas estructurales tienen varias ventajas sobre las de formato. En primer lugar, estas últimas conllevan menos información sobre los elementos de un documento. Dos elementos estructuralmente diferentes pueden estar anotados en su formato de la misma manera, a pesar de que representen conceptos distintos.

Por ejemplo, si representamos las palabras clave (comunes en los documentos tipo artículo de investigación) junto con las palabras en lenguaje extranjero en negritas, no hay forma en que la computadora pueda identificar la diferencia entre ambas. Además, si fuera necesario cambiar todas las palabras clave, el texto debe ser analizado para encontrar primero, todas aquellas palabras que fueron anotadas como negritas y posteriormente, por cada una, habría que decidir si se trata de una palabra clave o no. En cambio, sí se utilizan marcas estructurales, puesto que éstas no definen apariencia, dos elementos lógicamente diferentes serán anotados de forma diferente, sin importar si en la versión impresa ambos utilizan las mismas operaciones tipográficas o no.

Si bien, para documentos simples, esta diferencia entre formato y estructura puede ser irrelevante, no lo es así para documentos complejos como libros o periódicos. En estos casos, por cada elemento lógico del documento pueden ser necesarias una o más operaciones tipográficas. Por ejemplo, un título requiere que se especifique el tipo y tamaño de sus fuentes y un salto de línea de cierto tamaño.

2.2 SGML

Hace varios años, las bases de datos lograron manejar de manera independiente la información con la que trabajaban y los formatos de entrada o salida con que presentaban esta información, dicho de otra manera, lograron el concepto de la Gestión de Información Abierta[11]. Este fue un paso que consolidó un enorme avance para el mundo de las bases de datos, ya que gracias a este concepto, ahora es posible manejar casi cualquier base de datos de manera independiente al programa o a la computadora que se utiliza.

Sin embargo, los documentos a diferencia de las bases de datos siempre han presentado un formato de salida específico, por lo que hablar de independencia de datos dentro de los documentos es hablar de uno de los problemas que actualmente enfrenta la tecnología.

[11] Del término en inglés "*Opened Information Management o OIM*".

Como un primer intento de solución a este problema, en los años 60, IBM trabajó en el desarrollo de un lenguaje de marcas que permitiera el manejo de un mismo documento desde diferentes procesadores de texto, este lenguaje fue conocido como GML (*Lenguaje de Marcas Generalizado*)[12]. El problema, básicamente consistía en que cada procesador de texto utilizaba sus propias marcas para describir los elementos lógicos de los documentos, por lo que estos eran incompatibles con otros procesadores de texto. Conocer las marcas que utiliza cada procesador de texto permitiría diseñar filtros para pasar la información de un lenguaje de marcas a otro sin perder el formato. Por lo que la forma que IBM creó para resolver este problema, consistía en trabajar con marcas que pudieran editarse desde cualquier procesador, haciendo posible el traspaso de un documento de un editor a otro sin pérdida de información. Posteriormente, GML pasó a manos de ISO, que lo convirtió en un estándar oficial en los 80, denominándose finalmente SGML (*Lenguaje de Marcado Estandarizado y Generalizado*)[13]. SGML, está actualmente como una norma oficial internacional[14].

Este estándar, de carácter general, se aplica desde entonces para diseñar otros lenguajes de marcas, cuyos ejemplos más conocidos son HTML y ahora XML. SGML, más que un lenguaje de marcas, es un estándar internacional para la descripción de otros lenguajes de marcas. Dicho de otra manera, SGML es un metalenguaje, esto es, un lenguaje capaz de describir formalmente otro lenguaje.

Existen tres características de SGML que le distinguen de otros lenguajes de marcas (características que obviamente comparte con XML):

- Su énfasis sobre marcado estructurado, más que de formato y siempre basado en normas.
- La clasificación y edición de documentos de acuerdo a su tipo de documento.
- Su concepto de independencia de datos, que permite el manejo de una versión genérica para cada documento.

2.2.1 MARCADO ESTRUCTURADO BASADO EN NORMAS

Un lenguaje de marcas, orientado a la estructura de los documentos, usa marcas que simplemente proporcionan nombres a partes de un documento. Por contraste, un lenguaje de marcas orientado al formato de los documentos, define que formato debe llevarse a cabo y en que partes de dicho documento. Con marcado estructurado, en lugar de marcado de formato, el mismo documento puede procesarse fácilmente y de diversas formas. Por ejemplo, un programa podría extraer nombres de personas o lugares, ya sea de uno o varios documentos, para

[12] Del término en inglés "*Generalized Markup Language*".
[13] Del término en inglés "*Standard Generalized Markup Language*".
[14] ISO 8879: 1986, Procesamiento de la información -Sistemas de texto y oficina- Lenguaje normalizado general para documentos.

después crear un índice o una base de datos; mientras otro programa, podría poner estos nombres con algún tipo de letra distintiva. Esto sería muy sencillo, ya que cada nombre llevaría etiquetas (marcas), que describirían que efectivamente se trata del nombre de una persona o lugar, pero si en vez de marcas que describieran la estructura del documento usáramos marcas que describan su formato (pensemos en que los nombres están en letra de tipo negrita), entonces la computadora llenaría la base de datos no sólo con nombres de personas o lugares, sino con cualquier palabra que en el documento apareciera con letra de tipo negrita.

2.2.2 TIPOS DE DOCUMENTOS

SGML introduce la clasificación de documentos de acuerdo a su tipo. El tipo de un documento se define formalmente por las partes que constituyen su estructura. La definición de un informe, por ejemplo, podría ser la que esté formada por: un título, un autor, un resumen y una secuencia de uno o más párrafos. Partiendo de esta definición, cualquier documento que carezca de un título o cualquier otro de los elementos que se definieron en el ejemplo, no sería formalmente un informe. Algo muy importante para poder trabajar con varios tipos de documentos es, primero, definir con claridad que será valido para cada tipo de documento y que no. Esta definición nos ayudará a evitar confusiones cuando trabajemos con documentos que posean una estructura física similar. En la terminología usada por SGML, cada documento distinto pertenece a algún tipo de documento.

La definición formal que describe cada tipo de documento se denomina DTD (*Definición de Tipo de Documento*)[15]. Entendamos una DTD como el esquema para realizar cada tipo de documento.

2.2.3 INDEPENDENCIA DE DATOS

Uno de los objetivos básicos de SGML, como ya mencionamos, era asegurar que los documentos iban a poder ser transportables de un entorno de hardware y software a otro sin ninguna pérdida de información. La necesidad de alcanzar este objetivo, que permite una gran independencia en el manejo de los datos, es muy fácil de comprender. Pensemos en dos herramientas cualesquiera, si éstas no son capaces de hablar el mismo idioma no podrán trabajar juntas.

Este concepto de independencia de datos, dicho de otra manera, significa administrar la información de tal forma que se mantenga abierta para su utilización por cualquier programa y no sólo por el que lo creo (incluso programas o aplicaciones que no existían en el momento en que se creo la información). En SGML, los datos son almacenados dentro de la computadora en versiones genéricas del documento. Estas versiones no son registradas como parte de la

[15] Del término en inglés *"Document Type Definition"*.

aplicación que las usa, por lo que mantienen su característica de versiones genéricas. En consecuencia, cualquier otra aplicación que lo requiera podrá utilizar esa misma versión del documento sin ningún problema.

2.3 HTML

> *Publicar información, para su distribución global, requiere de un mismo lenguaje que pueda ser entendido, potencialmente, por todas las computadoras. Este lenguaje de publicación, para el caso de la Web, es HTML [Raggett et al., 1999].*

En 1989, un investigador llamado Tim Berners Lee propuso compartir documentos de texto, de un lugar remoto a otro, utilizando algún tipo de enlace entre estos documentos. Posteriormente, tomando el estándar SGML como base, Tim Berners Lee desarrolló un lenguaje de hipertexto llamado HTML. Este proyecto, basado en el concepto de crear un lenguaje de marcas propio, describía documentos de texto y vínculos entre estos, los cuales permitirían navegar entre un documento y otro siempre de manera independiente a la computadora.

Posteriormente, HTML sería denominado el sistema de hipertexto de la Web, convirtiéndose en el sistema de información de hipertexto más popular y diversificado de todos. Si lo comparáramos con SGML, HTML se creó y se adoptó en un periodo de tiempo muy corto, por lo que es muy probable que el gran éxito que experimentó, se haya debido principalmente a la facilidad de uso que tenía. Con HTML casi cualquier persona, aún sin ningún conocimiento en el área de programación, es capaz de diseñar y desarrollar sus propias páginas Web[16].

Como es de suponerse, HTML ha heredado algunas de las principales virtudes de su antecesor SGML. Por ejemplo, en HTML los autores pueden crear documentos con cualquier procesador o editor de textos. Además, estos documentos son compatibles con casi todos los tipos de computadoras y sistemas existentes. Cabe mencionar, que el tipo de marcado con que se describen los documentos HTML, está orientado a la estructura de los mismos y no ha su formato.

Sin embargo, también existen algunos puntos de divergencia entre SGML y su subconjunto HTML. Uno de los más importantes está dado en los tipos de documentos. Mientras SGML es un lenguaje de marcas extensible, que permite la definición y clasificación de documentos en un número indeterminado de DTD HTML sólo dispone de una, obligando a cualquier documento a ser diseñado no de acuerdo a su propio tipo de documento, sino de acuerdo al predefinido por la DTD de HTML.

[16] Nombre que se les da a los documentos que puede ser accesados dentro de la Web.

2.4 CÓMO HTML DA ORIGEN A XML

Después de algunos años, en que la Web crecía y se hacía cada día más popular, las pocas posibilidades de expansión que ofrecía manejar una sola DTD (debido al limitado número de etiquetas que ésta permitía), comenzaban a representar un problema contra toda posibilidad de crecimiento de la Web. En consecuencia, la molestia de la gente ante este hecho fue en aumento, dando como resultado que las empresas creadoras de navegadores[17] vieran la posibilidad de incrementar sus ventas sí comercializaban navegadores con extensiones de HTML. Aunque estas extensiones estaban fuera de toda normatividad, representaban una pronta solución al problema de la creación de documentos debido al mayor número de etiquetas que ofrecían.

Sin embargo, el usar las nuevas etiquetas traía consigo dos problemas fundamentales. En primer lugar, estas nuevas etiquetas estaban orientadas a dar formato a los documentos y no al manejo de su estructura, como se decidió que deberían ser las etiquetas de HTML en un inicio. Por lo que las enormes posibilidades que en cuanto a manejo de datos ofrecía HTML estaban mermadas. El segundo problema, la normalización de documentos. Una de las características más importantes de HTML, pasaba a segundo término. El que cada navegador fuera libre de diseñar sus propias etiquetas, limitó a que el formato de los documentos e incluso en algunas ocasiones también el contenido, dependiera exclusivamente del navegador con que se estuviera trabajando.

Como no es difícil de imaginar, HTML se convirtió en un lenguaje de marcas que no seguía los estándares de normatividad de SGML. De esta manera, llegado el punto donde HTML dejó de servir para su función inicial, el W3C (*Consorcio del World Wide Web*)[18] tomó la decisión de desarrollar un nuevo subconjunto de SGML, conocido actualmente como XML, y del que se publicaron las primeras especificaciones de su versión 1.0 en 1998.

Posteriormente, con la aparición de XML, se decidió hacer una reformulación de HTML dentro de la propia especificación XML. Conocida como XHTML, esta nueva especificación se encuentra en su versión 1.0 [Pemberton et al., 2000].

2.5 DISCUSIÓN

Existen, desde mi punto de vista, dos aspectos básicos que son causa de la declive de HTML. Primeramente, desde su concepción, HTML fue desarrollado pensando en utilizar una serie única de elementos, o dicho de otra manera, HTML solamente fue creado para trabajar con sólo un tipo de documento. Sabiendo que

[17] Término usado para nombrar a los programas de software que permiten visualizar el contenido de páginas Web.

[18] Consorcio formado por grandes empresas, tales como Netscape, Microsoft, IBM, entre muchas otras. Este grupo es el encargado de determinar los estándares y tecnologías que se integrarán a la Web.

ningún tipo de documento serviría para todos los propósitos, se esperaba que este punto tan frágil del HTML terminara por romperse en algún momento. Indudablemente, desde su inicio el objetivo de HTML fue lograr un lenguaje que se distinguiera por su sencillez más que por su solidez y, aunque esta sencillez fue la clave de su éxito a corto plazo, también lo fue de su declive a largo plazo.

Aunque los días de HTML estaban contados, existe un segundo punto que aceleró su decadencia, la falta de especificación. Cuando el crecimiento exponencial de la Web estalla, en parte gracias a la sencillez de diseño que permitía HTML, no existía una especificación para el diseño de páginas Web. En consecuencia, cuando HTML dispuso de una DTD formal varios años después, ya existían miles de páginas con código HTML erróneo. Aunque este hecho parece no haber causado demasiados problemas, sí nos demuestra que la falta de especificación no fue un hecho que dio inicio con la guerra comercial entre las compañías creadoras de navegadores. La falta de especificación y la falta de interés por demandar el uso de ésta, son hechos a los que se les resto importancia desde el mismo inicio de la Web.

Ahora, más allá de crear polémica sobre las posibles causas que pudieron llevar a la extinción al HTML, es importante canalizar todas las posibles críticas sobre este hecho hacia lograr un sólido desarrollo de XML.

Hasta este punto, hemos estudiado los antecedentes que dieron origen a XML, el lenguaje de marcas que, sin duda, representa el próximo paso en la evolución de la Web. Respaldado, como acabamos de ver, por las empresas más importantes de software en el mundo, la trayectoria de décadas de su antecesor SGML y la basta experiencia que ha heredado de los éxitos y fracasos cosechados por HTML.

Llegado a fin, el presente capítulo pretende consolidar un marco teórico con los antecedentes y algunos de los conceptos más simples, necesarios, para profundizar en el tema de XML. Conocer y entender la familia de tecnologías que hoy constituyen a XML, requiere, preferentemente, el conocimiento previo de dichos conceptos.

CAPÍTULO 3

CONCEPTOS BÁSICOS DE XML

Durante este capítulo, estudiaremos algunos de los conceptos y tecnologías derivadas, más importantes, de XML. Cabe mencionar, que este capítulo podría extenderse tanto como se quisiera, por lo que es importante decir, que el objetivo del mismo será únicamente proporcionar los conocimientos generales para comprender, mediante una visión general del tema, la arquitectura global de XML. Hecho que resulta importante, desde el mismo objetivo del presente trabajo, en el cual se propone el desarrollo de un sistema cuya base estará constituida por XML.

Antes de comenzar formalmente con el capítulo y con el único objetivo de ayudar a comprender de manera más sencilla los conceptos que estudiaremos a continuación, veremos previamente algunas definiciones de XML.

> *El lenguaje de marcado extensible, abreviado como XML, es una forma restringida de SGML que describe una clase de objetos de datos llamados documentos XML [Bosak et al., 2000].*
>
> *XML no es una tecnología, es una familia de tecnologías [Bos 1999].*

3.1 ESTRUCURA DE LOS DOCUMENTOS XML

Todo documento XML posee una estructura lógica y una física. Lógicamente, un documento XML está conformado por *elementos*. Cada elemento representa, entonces, cada componente lógica que integra este documento. Por ejemplo, la mayor parte de los documentos, como libros o revistas, se pueden dividir en componentes (capítulos, artículos, etc.). Estos componentes, que a su vez se derivan en otros componentes (títulos, párrafos, etc), se pueden considerar elementos físicos del documento. La información adicional, que a veces incluyen los elementos de un documento, es conocida como *atributos*. La tarea de los atributos es describir propiedades de los elementos. Si lleváramos este concepto a las bases de datos, para su explicación, pensaríamos en un registro completo como un elemento y cualquiera de sus campos sería uno de los atributos. Por ejemplo, para un registro de cliente, la información completa sería el elemento, y sus atributos lo serían el domicilio, el nombre, o cualquier otro campo que forme parte de este registro.

Físicamente, el documento está compuesto de unidades llamadas *entidades*. Una entidad, por ejemplo, puede estar formada, por un solo carácter o por todos los caracteres que forman parte de un libro. La manera como funcionan estas entidades en XML, consiste, primeramente, en asignar un nombre a cada una,

para posteriormente, insertar una *referencia de entidad*[19] en la parte del documento que se requiera. Por ejemplo, una entidad podría llamarse <capitulo1>, y representar el capítulo completo de un libro. Ahora, cada vez que necesitemos que aparezca este capítulo completo, nos referiremos a la entidad sólo como <capitulo1>.

Una característica de las entidades es que no necesariamente tienen que estar dentro del mismo documento, pueden estar en otros documentos, incluso cuando estos pertenezcan físicamente a otra computadora. Este tipo de entidades son conocidas como entidades externas. Un documento XML puede dividirse en muchos archivos distintos, incluso residentes en la Web, que serán llamados, cada una de ellos, en la terminología XML, entidad. Estas entidades, en conjunto, que constituyen el documento XML, son denominadas la estructura física del documento.

Algo importante de mencionar es que las entidades no siempre son guardadas como archivos. Por ejemplo, las entidades pueden estar guardadas dentro de bases de datos, o ser creadas en tiempo real por un programa de computadora. Una entidad puede contener parte de un archivo o varios archivos completos.

3.2 DOCUMENTOS VÁLIDOS Y DOCUMENTOS BIEN FORMADOS

Como en cualquier otro lenguaje, de programación o escrito, en XML existen formas de notación, que hacen referencia a la forma de crear un documento, que pueden ser o no ser correctas. Para XML existen dos conceptos de lo que puede ser un documento correcto. La primera, y más simple, es conocida como *documento bien formado.*

Entonces, un documento XML es bien formado si:

- *Cada una de las entidades analizadas, a las que se referencia directa o indirectamente en el documento, está bien formada.*
- *Contiene al menos un elemento.*
- *Tiene exactamente un elemento raíz*[20]*, del cual ninguna parte aparece en el contenido de ningún otro elemento. Para el resto de los elementos, si la etiqueta de comienzo está contenida de algún otro elemento, la etiqueta de fin está contenida dentro del mismo elemento. Es decir, los elementos delimitados por etiquetas de comienzo y fin, deben estar anidados adecuadamente [Bosak et al., 2000].*

Como es lógico suponer, un documento no puede ser bien formado sí las entidades que lo conforman no lo son. Además, para que un documento pueda ser

[19] Nombre que se le da a la etiqueta que contiene el nombre de una entidad de texto. Se representa la entidad mediante esta referencia de entidad y no mediante la entidad completa con la finalidad de simplificar.
[20] Nombre del elemento padre que contiene el resto de los elementos en un documento XML.

entendido, y después interpretado, será necesario que contenga físicamente algún tipo de estructura o lo que es lo mismo, al menos un elemento. Por otro lado, la correcta anidación de elementos permitirá que los documentos puedan extenderse, cada que se requiera, de manera independiente al manejo que se le esté dando o se le quiera dar a su contenido.

Para que un documento pueda ser clasificado como válido, primero, será indispensable que se trate de un documento bien formado. Una vez satisfecho este punto diremos que se trata de un documento válido sí éste está declarado conforme a una DTD y, además, se ajusta positivamente a ésta. De esta manera, los documentos que carezcan de una DTD no serán documentos válidos, debido a que no son validados por su DTD; aunque, estrictamente tampoco serán inválidos porque estos tampoco infringen su DTD. Esto le permitirá a un documento que no es válido, pero que tampoco es inválido, ser un documento bien formado.

Es muy importante reflexionar bien antes de decidir si se crea un documento sólo bien formado o válido. Por ejemplo, sí se trata de un documento poco extenso y de un tipo muy particular, bastará probablemente con que éste esté bien formado. Pero sí pensamos que será uno de varios documentos, que se integrará algún sistema de información o que será un documento sumamente extenso, será necesario escribir una DTD y validarlo periódicamente. De esta manera, el documento dará garantías de compatibilidad, facilitando de gran manera la obtención de información.

3.3 GRUPO DE CARACTERES UTILIZADO EN XML

Los documentos están formados por caracteres, por lo que sí queremos crear un documento estándar lo más importante será conocer que grupo de caracteres va a integrarlo. Entre más amplio y más común sea este código de caracteres las posibilidades que nos ofrecerá serán mayores. El grupo de caracteres más conocido, y que por ende domina el mercado, es ASCII. ASCII contiene ciento veintiocho caracteres entre letras, símbolos y signos de puntuación.

Uno de los objetivos primordiales de XML era lograr la creación de documentos en casi cualquier procesador de textos. Es por eso que el grupo de caracteres que utiliza XML es Unicode. Unicode incluye miles de caracteres que son comunes en lenguas de todo el mundo, incluyendo las del código ASCII. Los primeros ciento veintiocho caracteres que integran Unicode, llamados UTF-8, son compatibles con ASCII. Lo que significa, que a nivel de bits, los primeros ciento veintiocho caracteres de Unicode son iguales a los ciento veintiocho caracteres ASCII Gracias a esto, como la mayor parte de los procesadores de texto trabajan con ASCII, Unicode permite la creación de documentos XML en casi todos los procesadores de texto.

3.4 DTD

Una definición de tipo de documento está constituida por una serie de definiciones que determinan que elementos del documento son legales y en que lugar lo son, dicho de otra manera, en XML la DTD es una definición formal de los tipos de elementos y atributos permitidos en un documento de algún tipo específico. Las DTD son herramientas de estandarización muy poderosas. Por ejemplo, mientras una DTD puede estar hecha para la creación de formularios, lo que implicaría que sería muy estricta ya que sólo permitiría a cierto tipo de elemento estar en un determinado lugar; otra DTD, diseñada como guía para la publicación de artículos, tal vez, sólo exija que el contenido del documento incluya un *título de artículo* por cada artículo integrado al cuerpo del documento. Las DTD nos permiten realizar un procesamiento de textos mucho más robusto, ya que el software podrá estar especializado en el procesamiento de un solo tipo de documento y no de todos como se hace actualmente.

Dos aspectos importantes que deben tenerse siempre en cuenta, cuando se trabaja con documentos XML, son:

- Algunos documentos XML carecen de declaración de tipo documento. Esto no implica que no se ajusten a algún tipo de documento. Simplemente, significa que se ajustan a una DTD que no ha sido formalmente definida.
- En XML no existen DTD predefinidas, por lo que es importante no olvidar que es labor del diseñador especificar su propia DTD para cada tipo de documento.

En la especificación de XML se describe la forma en que deben ser definidas las DTD [Idem]. Esta definición puede darse de manera interna (cuando va incluidas junto al código XML) o de manera externa (si se encuentra en un archivo propio).

Que cada usuario pueda crear su propia DTD es una gran ventaja, ya que proporciona total libertad para la estructuración de cada documento. Pero por otro lado, también puede suponer un grave inconveniente, ya que es muy fácil que dentro de documentos de un mismo sector (arquitectura, edición, educación, etc.) existan diversas DTD. Lo que haría del manejo de documentos XML una tarea muy complicada debido a las diferentes estructuras en que dichos documentos, aún siendo del mismo tipo, se podrían presentar. Por este motivo, en la actualidad se está definiendo una DTD estándar para cada sector distinto que así lo requiera. De tal forma que existan DTD avaladas por asociaciones, formadas por empresas y organismos, que garanticen que cualquier usuario que adopte alguna de estas DTD para trabajar, lo haga con las mismas etiquetas y normas que los demás.

CREACIÓN DE UNA DTD

Cuando pensamos en crear una DTD propia, podemos pensar como sí se tratará de crear nuestro propio lenguaje. La DTD puede decir concretamente algo como: "El resumen de un libro debe estar formado por los siguientes elementos". Sin embargo, no hay que olvidar que las declaraciones de tipo de documento son opcionales, por lo que los documentos XML pueden no ajustarse a ninguna DTD y no, necesariamente, dejar de seguir siendo documentos XML.

Ahora, veamos un ejemplo de cómo crear una DTD para un documento XML:

```
<!DOCTYPE cliente [
<!ELEMENT cliente (nombre, calle, ciudad,  pais, numero)
<!ELEMENT nombre  (#PCDATA)>
<!ELEMENT calle  (#PCDATA)>
<!ELEMENT ciudad  (#PCDATA)>
<!ELEMENT pais  (#PCDATA)>
<!ELEMENT numero  (#PCDATA)>
]>

<cliente>
<nombre> Pablo Orozco </nombre>
<calle>Insurgentes 308</calle>
<ciudad> DF </ciudad>
<pais> México </pais>
<numero> 01 </numero>
</cliente>
```

En este ejemplo podemos ver un documento XML que contiene una declaración de tipo de documento y, posteriormente, declaraciones del tipo de elemento.

La declaración de tipo de documento es:

```
<!DOCTYPE cliente [
<!ELEMENT cliente (nombre, calle, ciudad,  pais, numero)
<!ELEMENT nombre  (#PCDATA)>
<!ELEMENT calle  (#PCDATA)>
<!ELEMENT ciudad  (#PCDATA)>
<!ELEMENT pais  (#PCDATA)>
<!ELEMENT numero  (#PCDATA)>
]>
```

Las declaraciones del tipo de elemento son:

```
<!ELEMENT cliente (nombre, calle, ciudad, pais, numero)
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT numero (#PCDATA)>
```

Cabe mencionar que en las declaraciones de tipo de elemento también es posible, a su vez, declarar otros elementos o atributos, más adelante veremos esto con más detalle. En el ejemplo anterior, todas las declaraciones que definen la etiqueta cliente residen dentro del mismo documento. Sin embargo, está definición pudo hacerse, parcial o completamente, en algún otro lugar. Ahora, veamos el mismo ejemplo pero con dichas declaraciones externas al documento:

```
<! DOCTYPE cliente SYSTEM "http://www.dondesea.com/cliente.dtd">
<cliente>
<nombre> Pablo Orozco </nombre>
<calle>Insurgentes 308</calle>
<ciudad> DF </ciudad>
<pais> México </pais>
<numero> 01 </numero>
</cliente>
```

La palabra SYSYEM indica que se busque de manera externa al documento, en la referencia citada, las declaraciones de tipo documento. Otra manera de hacer referencia a una DTD externa, cuando esta se encuentra en el mismo directorio pero en un archivo independiente al documento, es:

```
<! DOCTYPE cliente SYSTEM "cliente.dtd">
<cliente>
<nombre> Pablo Orozco </nombre>
<calle>Insurgentes 308</calle>
<ciudad> DF </ciudad>
<pais> México </pais>
<numero> 01 </numero>
</cliente>
```

Si, ahora abrimos con un editor de texto el archivo *cliente.dtd* veremos algo como:

```
<! ELEMENT cliente (nombre, calle, ciudad, pais, numero)
<! ELEMENT nombre (#PCDATA)>
<! ELEMENT calle (#PCDATA)>
<! ELEMENT ciudad (#PCDATA)>
<! ELEMENT pais (#PCDATA)>
<! ELEMENT numero (#PCDATA)>
```

Como pudimos observar, todas las declaraciones de tipo documento empiezan con <!DOCTYPE y después llevan el nombre de un elemento que se define en la DTD. Si alguna de las declaraciones DTD está almacenada externamente, la tercera parte de la declaración de tipo de documento debe ser SYSTEM. La última parte debe ser un URI (*Identificador Universal de Recursos*)[21] que indique donde están las declaraciones externas.

DECLARACIÓN DEL TIPO DE ELEMENTO

Ahora, veamos la declaración del tipo de elemento, para posteriormente explicarla. En nuestro ejemplo anterior teníamos:

<! ELEMENT cliente (nombre, calle, ciudad, pais, numero)>

Las declaraciones del tipo de elemento deben empezar como <!ELEMENT, y van seguidas por el nombre del elemento que se está declarando. Finalmente, debe haber una *especificación del contenido*. La especificación de contenido que se muestra, en este caso, nos dice: debe haber un elemento *nombre*, seguido de un elemento *calle*, seguido de un elemento *ciudad*, seguido de un elemento *pais*, seguido de un elemento *numero*. Los nombres de los tipos de elemento son *nombres XML*[22] (para este ejemplo los nombres son: cliente, nombre, calle, ciudad, pais y numero). A cada tipo de elemento se le permite sólo un tipo de contenido determinado. A continuación, veremos los cuatro tipos de especificación de contenido para los tipo de elemento.

Contenido EMPTY, puede no tener contenido. Suele usarse para los atributos:

<! ELEMENT nombredelelemento EMPTY>

Contenido ANY, puede tener cualquier contenido:

<! ELEMENT nombredelelemento ANY>

Contenido ELEMENT, sólo puede tener subelementos:

<! ELEMENT nombredelelemento (subelemento-uno, subelemento-dos)>

[21] Nombre que se le da a una referencia que nos indica una dirección. Está dirección puede estar dentro o fuera de la computadora en la que se está trabajando. Del término en inglés "*Unifom Resource Identifiers*".
[22] Las restricciones para el uso de caracteres permitidos para los nombres XML, se describen en la especificación XML en la sección *Nombres y Name tokens*.

Contenido MIXED, puede tener caracteres de tipo de datos o una mezcla de caracteres y subelementos[23]:

<! ELEMENT nombredelelemento (#PCDATA)>
<! ELEMENT nombredelelemento (#PCDATA | subelemento)>

DECLARACIÓN DE ATRIBUTOS

A diferencia de los elementos, los atributos de un elemento no pueden contener otros elementos o atributos. Otra diferencia importante es que el orden de los atributos puede ser cualquiera, mientras los elementos deben ordenarse como se indicó en su especificación. Los nombres de los atributos suelen representar propiedades de un elemento, mientras que los subelementos representan más bien partes de este elemento. Dos elementos diferentes pueden tener atributos con el mismo nombre porque, finalmente, serán atributos diferentes.

Los atributos se declaran para un determinado tipo de elemento. A menudo esta declaración aparecerá enseguida de la declaración del tipo de elemento. Ejemplo:

<!ELEMENT cliente (#PCDATA)>
<!ATTLIST cliente email CDATA #REQUIRED>

La declaración de atributos comienza con <!ATTLIST, es seguida del nombre del elemento, después del nombre del atributo, su tipo y, finalmente un indicador que dirá si el dato debe ser ingresado de manera obligatoria, por omisión, etc. Para este caso tenemos que el único atributo se llama email y es válido para todos los elementos cliente. Su valor deberá estar dado en datos de tipo carácter, cualesquiera, y será obligatorio. La declaración de atributos puede hacerse en forma de lista, como se muestra:

<!ATTLIST cliente email CDATA #REQUIRED
fax CDATA #REQUIRED
telefono CDATA #REQUIRED>

Otros ejemplos de declaraciones para listas de atributos:

<!ATTLIST cliente edad (20ª30|30ª40|40omás) #REQUIRED>

Para este ejemplo tenemos como atributo la edad del cliente. Este dato tendrá, forzosamente, que estar dado por uno de los tres valores que se sugieren.

<!ATTLIST cliente edad (20ª30 | 30ª40 | 40omás) 20ª30>

[23] Elementos que están contenidos en otro elemento o elemento padre. Para nuestro ejemplo los elementos nombre, calle, ciudad, país y numero; son subelementos o elementos hijo del elemento cliente o elemento padre.

Para este caso se está sugiriendo un valor por omisión. En caso de que no se introduzca ninguno de los valores sugeridos el atributo edad tomará el valor de *20@30.*

<!ATTLIST cliente edad NUMBER #REQUIRED>

Ahora el atributo edad tendrá que tomar, obligatoriamente, un valor numérico cualquiera.

<!ATTLIST cliente edad NUMBER #IMPLIED>

El atributo edad podrá omitirse si así se desea. En caso de introducirse algún valor tendrá que ser numérico.

Las modificaciones para cada declaración dependerán, exclusivamente, de las necesidades de cada diseñador. Ahora, para finalizar, veamos como quedaría una DTD XML, con declaraciones de tipo de elemento y declaraciones de atributos:

```
<?xml version=" 1.0 "?>
<! DOCTYPE cliente [
<! ELEMENT historial (nombre+, dirección?, reconocimientos*)
<! ELEMENT nombre (#PCDATA)>
<!ATTLIST nombre email CDATA #IMPLIED
telefono NUMBER #REQUIRED>
<! ELEMENT direccion (#PCDATA)>
<!ATTLIST direccion calle CDATA #REQUIRED
delegacion CDATA #REQUIRED>
<! ELEMENT reconocimientos (#PCDATA)>
]>
```

XML permite indicar con que frecuencia queremos repetir los elementos que forman nuestro documento. La tabla con los indicadores de frecuencia es la que se muestra a continuación:

- ? Opcional de 1 a 0 veces
- * Opcional de 0 a más veces
- + Opcional de 1 a más veces

3.5 XPATH

Todo el procesamiento realizado con un documento XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma distinta.

El tratamiento del documento XML comienza por la localización del mismo a lo largo del conjunto de documentos existentes en el mundo. Para llevar a cabo esta localización de forma unívoca, se utilizan los URI, de los cuales los URL[24] son sin duda los más conocidos.

Una vez localizado el documento XML, la forma de seleccionar información dentro de él es mediante el uso de XPath[25] (*Lenguaje de rutas de XML*). Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un archivo XML.

> *XPath, en sí, es un lenguaje sofisticado y complejo, cuyo principal objetivo es direccionar partes de un documento XML [Clark et al., 1999].*

Además, como casi todas las especificaciones derivadas de XML, XPath es aún muy reciente, por lo que no es fácil encontrar herramientas que incorporen todas sus funcionalidades.

XPath es a su vez la base sobre la que se han especificado nuevas herramientas para el tratamiento de documentos XML. Herramientas tales como XLink, Xpointer[26] y XQuery[27], que también están en estado de desarrollo, pero que sin duda cambiarán el modo en que actualmente concebimos la navegación por la Web. Así, XPath sirve para decir cómo debe procesar una hoja de estilo el contenido de un documento XML, pero también para poder poner enlaces o cargar en un navegador partes determinadas de un documento XML, en vez de todo.

3.5.1 CONCEPTOS BÁSICOS

Una instrucción en lenguaje XPath se denomina como expresión. Donde dichas expresiones pueden incluir una gran variedad de operaciones sobre distintos tipos de operandos. En nuestro caso nos vamos a enfocar a dos tipos de operandos: *llamadas a funciones y rutas.*

Una ruta es la más importante de los tipos de expresiones que se pueden especificar en notación XPath. La sintaxis de una ruta es similar a la usada para describir las rutas de los directorios de archivos en Unix o Linux. Sin embargo, sólo la sintaxis es similar. El significado de las expresiones es totalmente diferente.

Por ejemplo, la siguiente ruta en Unix hace referencia a un único directorio *dir* : */usr/home/dir*. Sin embargo, la siguiente expresión hace referencia a todos los elementos *párrafo* y no a uno solo como podría pensarse: */libro/capitulo/parrafo*.

---

[24] Del término en inglés "*Uniform Resource Locators*".
[25] Del término en inglés "*XML Path Language*".
[26] Las definiciones de XLink y XPointer se tratan más adelante, dentro de sus respectivos capítulos.
[27] Del término en inglés "*XML Query Language*". XQuery permitirá traer e interpretar toda la información almacenada en documentos XML. [Chamberlin 2002].

Hay que tener en cuenta que una expresión en XPath no devuelve los elementos que cumplen con el patrón que representa dicha expresión, sino que devuelve una referencia a dichos elementos; es decir, una expresión XPath nos devuelve una lista de apuntadores a los elementos que cumplen con el patrón. Dicha lista puede estar vacía o contener uno o más nodos. Por lo tanto, podemos decir que XPath es un lenguaje que sirve para seleccionar muchos nodos a la vez, lo cual puede resultar muy útil.

3.5.2 ATRIBUTOS

Si sólo quisiéramos seleccionar un nodo que cumple con ciertas características podríamos, por ejemplo, utilizar los atributos de éste para hacer referencia a dicho nodo. Estos atributos deberán especificarse dentro de una ruta utilizando corchetes. Veamos un ejemplo:

```
/libro/capitulo[@num="1"]/parrafo
```

Mediante la anterior ruta estamos indicando que se escojan todos los elementos párrafo de todos los elementos capitulo que tengan un atributo llamado num al cual se le haya asignado el valor "1" (en XML todos los atributos tienen valores de tipo cadena).

3.6 HOJAS DE ESTILO PARA XML

Las hojas de estilo permiten asignar un formato adecuado a los documentos. Debido a que las especificaciones de estilo en que queremos se muestre dicho documento, usualmente, se presentan en una entidad aparte, la hoja de estilo, el software que no esté interesado en el estilo de los documentos puede ignorarlo de manera muy sencilla.

Si queremos trabajar con estilos en los documentos XML podemos optar por dos soluciones:

- Las CSS[28] (*Hojas de Estilo en Cascada*) que se utilizan también con HTML.
- Las XSL[29] (*Lenguaje Extensible basado en hojas de Estilo*).

Las CSS son conocidas por todos los diseñadores de HTML. Si no se conocen las posibilidades de trabajar con especificaciones CSS, deben entenderse como una descripción del formato en el que se desea que aparezcan las etiquetas definidas en un documento. Por ejemplo, se puede definir una hoja de estilo para un archivo XML con el siguiente código:

[28] Del término en inglés *"Cascading Style Sheets"*.
[29] Del término en inglés *"eXtensible Stylesheet Language"*.

```
nombre {display:block; font-family:Arial; font-size:small; width:30em}
ciudad { display:block; font-family:Times; font-size:small; width:30em }
numero { display:block; font-family:Courier; font-size:small; width:30em }
```

Con lo anterior, lo que le estamos indicando es que queremos presentar el texto incluido entre <nombre> y </nombre> con un tipo de letra *Arial*[30], entre <ciudad> y </ciudad> con un tipo de letra *Times* y, entre <numero> y </numero> con un tipo de letra *Courier*. Sin embargo, las CSS sólo son eficaces para describir formatos y presentaciones, y no sirven para decidir qué tipos de datos deben ser mostrados y cuáles no deben salir en la pantalla. Dicho de otra forma, las CSS se utilizarán para documentos XML sólo en los casos en los que todo su contenido deba mostrarse sin ninguna restricción.

Como las CSS resultan insuficientes, el W3C ha trabajado en una alternativa complementaria llamada XSL. XSL contiene muchas características de CSS, pero integrará también las principales nociones del lenguaje de hojas de estilo DSSSL[31] (*Lenguaje de Especificación y Semántica de Estilo de Documentos*). XSL es extensible, igual que XML, por lo que es útil para trabajar con todos los tipos de documentos y no sólo con los de HTML. Además, XSL no sólo permite especificar cómo queremos presentar los datos de un documento, sino que también sirve para filtrar los datos de acuerdo a ciertas condiciones que el programador puede decidir.

En la actualidad, las cosas han cambiado mucho para la especificación XSL. En un inicio, esta especificación estaba dividida en dos partes, contenidas dentro de la misma especificación (XSL):

> *Una parte encargada de la transformación de la estructura del documento, en la cual los elementos de éste son seleccionados, agrupados y reordenados; y otra parte o segundo proceso que permitiría proporcionar a los elementos resultantes un estilo de salida específico [Rusty 2000].*

Sin embargo, posteriormente, debido a la independencia que existía entre estos procesos se decidió que ambos deberían tener su propio nombre y definición. Por lo tanto, XSL dejo de estar formado por una sola especificación y dio lugar a dos:

- *XSLT, un lenguaje que permite transformar un documento XML.*
- *XSLFO*[32]*, un lenguaje que permite asignar características de formato a elementos de los documentos XML [Crane 2000].*

---

[30] *Arial, Times y Courier*; son tres de los tipos de letra más comúnmente usados por los procesadores de texto.
[31] Estándar de ISO basado en SGML que regula las normas de presentación de documentos de marcas para la Web. Del término en Inglés *"Document Style Semantics and Specification Language"*.
[32] Del término en inglés *"XSL Formatting Objects"*.

3.6.1 XSLT

Podemos pensar en XSLT más como un lenguaje de programación que como una simple hoja de estilo. Donde dicho lenguaje complementa una trilogía de especificaciones creadas para lograr la transformación de los documentos XML: *CSS, XSLT y XSLFO.*

XSLT es un lenguaje que se usa para convertir documentos XML en otros documentos; por ejemplo, puede convertir un documento XML con una DTD específica en otro que obedezca otra DTD totalmente distinta, un documento XML bien formado en otro que sigue una DTD, o, lo más habitual, convertir un documento XML en otro documento con un formato distinto, tal como HTML.

Los programas XSLT están escritos en XML y, generalmente, se necesita un programa en conjunto con un documento XML para procesarlos. Más adelante, en el siguiente capítulo, estudiaremos estos conceptos con más detalle.

> *Una transformación en el lenguaje XSLT es expresada como un documento XML bien formado [Kay 1999].*

Algo importante de mencionar, es que el estilo de programación de XSLT es muy distinto a la mayor parte de los lenguajes de programación (tales como C, Java, etc). Pareciéndose más a lenguajes orientados al paradigma de programación funcional. En la práctica, esto significa que la programación de XSLT está basada en reglas, dicho de otra manera: *cuando ocurre algo en la entrada, se hace algo en la salida.*

El siguiente capítulo, el capítulo 4, está dedicado por completo a explicar los detalles y conceptos más importantes de la programación con XSLT, razón por la cual no se ahondara más sobre el tema en el presente capítulo.

3.6.2 XSLFO

XSLFO es una implementación específica de un determinado programa XSLT. Dicho de otra manera, si utilizamos XSLFO con un documento XML el resultado es otro documento con la información de éste último asignada a algún estilo de visualización impresa. XSLFO no preserva nada de la semántica de la información original, solamente describe como debe mostrarse el documento en pantalla o en papel.

Esta es quizá la principal diferencia entre un XSLFO y XSLT, ambos transforman un documento XML en otro documento, pero mediante XSLFO sólo podemos describir cómo se van a visualizar los elementos del documento.

PROPIEDADES

XSLFO permite especificar detalladamente el estilo que producirá un determinado documento. Propiedades como el tamaño de la página, el color de texto, entre muchos otros, pueden ser definidas mediante el uso de XSLFO.

No es casual que muchas de estas propiedades sean iguales a las usadas en las CSS de HTML, de hecho, la intención es hacer que no haya que cambiar los nombres de estas últimas para aprovechar la experiencia adquirida durante los últimos años.

El siguiente código, fragmento de un documento XSLFO, ejemplifica la forma de indicar el tipo de letra que deseamos que tenga el elemento titulo de un documento XML al ser visualizado:

```
...
<fo:flow>...

  <fo:block font-size="20pt" font-family="serif">
   <xsl:apply-templates select="titulo"/>
  </fo:block>...

</fo:flow>
```

El objeto fo:flow describe la secuencia del contenido que será visualizado, para este caso sólo se tiene un elemento fo:block, con letra tamaño 20pt y tipo serif.

Como se puede ver en el ejemplo, existe una gran similitud entre CSS y XSLFO, de hecho, las principales diferencias entre ambas sólo son:

- XSLFO utiliza una sintaxis XML.
- XSLFO posee un vocabulario mucho más extenso para definir propiedades de estilo, en consecuencia es más poderoso.

Algo importante de mencionar es que ambas especificaciones se limitan a la transformación de datos y no a la de documentos como es el caso de XSLT.

3.7 LENGUAJES DE ENLACE DE XML

La cuestión de los hipervínculos[33] es tan importante para los documentos XML, que el W3C no se ha conformado con crear una sola norma y ha desarrollado dos:

[33] Un hipervínculo es un enlace cuya intención primordial es mostrarse a usuarios humanos [Maler Junio2001].

- XLink (*Lenguaje de Ligado de XML*)[34]
- XPointer

3.7.1 XLINK

A los diseñadores de HTML, el hecho de desarrollar un lenguaje exclusivamente para enlaces entre documentos, les puede parecer una complicación sin sentido, ya que están acostumbrados a las pocas posibilidades que brinda HTML (donde el ligado de documentos se realiza mediante el uso de una única etiqueta).

Sin embargo, si pensamos en las muchas variaciones que pueden tener los vínculos, como tener muchos finales, estar almacenados en bases de datos o documentos independientes, se comprenderá mucho más fácilmente las ventajas de la solución adoptada para XML.

> *En Xlink, un enlace es una relación explícita entre recursos o porciones de recursos [Maler et al., 06/2001].*

En XML existen dos tipos básicos de hipervínculos:

- Enlaces simples.
- Enlaces extendidos.

ENLACES SIMPLES

Este tipo de enlace, también llamado enlace sencillo, no es más complejo que el usado por HTML. Los enlaces simples están formados por dos puntos extremos únicamente, el origen y el destino (de igual manera que los enlaces de HTML).

Un ejemplo de enlace simple:

```
<ENLACESIMPLE xml: links="simple" href="http://www.dondesea.com/doc.xml">
Texto del enlace
</ENLACESIMPLE >
```

Este mismo ejemplo en HTML:

```
<A HREF="http://www.dondesea.com/doc.xml">Texto del enlace</A>
```

El enlace se diseña mediante el atributo xml:links, que indica que se trata de un enlace Xlink. Posteriormente, links="simple" indica el tipo de enlace que es (para

[34] Del término en inglés *"XML Linking Language"*, ya que anteriormente era conocido como XLL.

este caso se trata de un enlace simple). Finalmente, href nos dice cual es la dirección a la que se está haciendo referencia.

Como podemos ver, la principal diferencia entre el enlace simple de XLink y el de HTML es el nombre que se le da al enlace. Mientras en HTML lo denominamos A, en XLink el usuario puede llamar a sus elementos de enlace como quiera. Esto representa algunas ventajas, por ejemplo, cada enlace podrá tener diferentes atributos, comportamientos, etc. Otra diferencia importante entre los enlaces HTML y XML, es que en XML los puntos extremos de los enlaces pueden ser tanto origen como destino, o incluso ambas.

ENLACES EXTENDIDOS

La principal diferencia que existe entre los enlaces simples y los enlaces extendidos, es que en los segundos debemos decidir como especificar la dirección de más de un enlace de destino. A continuación veremos un ejemplo:

```
<ejemplo xml:link="entended">
<locator href="doc1.xml" role="Documento 1"/>
<locator href="doc2.xml" role="Documento 2"/>
<locator href="doc3.xml" role="Documento 3"/>
<P>Comentarios
</ejemplo>
```

Del ejemplo anterior podemos comentar lo siguiente. Tenemos tres localizadores que proporcionan a cada dirección un origen. Estos localizadores podrán tener otros atributos como show, actuate y behavior, cuya semántica será la misma que la de los enlaces simples. Los localizadores de los enlaces extendidos son muy similares a los enlaces simples. De hecho, podemos definir un enlace simple como la combinación de un enlace y un solo localizador.

Una aplicación para los enlaces extendidos, podría ser la representación de cada localizador en un menú que permitirá acceder a cada uno de los recursos referenciados, donde cada localizador sería capaz de presentar sus propios atributos. Incluso, cada localizador podría presentar su contenido en una pequeña ventana al presionar un mismo botón. Cada localizador podrá tener un papel independiente o uno común, según las necesidades del diseñador.

Otra aplicación interesante para el caso de los enlaces extendidos son los llamados *Grupos de enlaces.* Esta aplicación nos permite procesar un grupo de documentos vinculados entre sí. Por ejemplo, si se tiene un documento principal al cual se le han hecho ya una serie de críticas, podremos hacer que el navegador muestre toda esta información en la misma ventana o crear una pequeña tabla con las referencias señaladas.

Será mediante los Grupos de enlace que el usuario indicará al navegador la lista de todos los documentos que están relacionados. Podemos pensar en los Grupos de enlace como pequeñas bases de datos de hipervínculos. Un ejemplo de un Grupo de enlaces:

```
<documentos-relacionados xml:link="gruop">
<doc xml:link="document" href="documento1.html">
<doc xml:link="document" href="documento2.html">
<doc xml:link="document" href="documento3.html">
</documentos-relacionados>
```

3.7.2 XPOINTER

> *Podemos entender los Xpointer como apuntadores[35] a documentos XML, ya que nos permiten acceder, incluso, a las estructuras internas de estos documentos [Maler et al., 09/2001].*

Con Xpointer, podemos acceder, no a un documento completo sino a un solo elemento de éste haciendo referencia a su posición o a su nombre.

Ahora, hagamos una analogía con HTML, para explicar más claramente lo que es XPointer. En la Web encontramos direcciones como ésta:

http://www.dondesea.com/banking#about

La primera parte de esta dirección esta formada por el protocolo (para este caso http[36]), que describe el mecanismo que el navegador deberá utilizar para interpretar el resto de la dirección. Después del protocolo, tenemos el *hostname*[37] (www.dondesea.com), que indica en que parte de la Web está ubicada la computadora a la que nos estamos conectando y, posterior a éste, la ruta o localización de los datos, ya dentro de la computadora. El identificador #about, hace referencia a un solo elemento HTML en particular, y no a todo el documento.

Los XPointers son muy parecidos al elemento #about de HTML, sólo que los primeros son mucho más flexibles. Los XPointers serán como una extensión de la dirección que permitirán al usuario señalar hacia el contenido del documento y no sólo hacia el documento.

Con XPointer será posible hacer referencia sólo a la parte del documento que nos interesa (un párrafo, una tabla, una imagen, etc.), y no a todo el documento. Por ejemplo, podemos crear un documento sólo con fragmentos de otros documentos,

---

[35] Término usado en computación para denominar el señalamiento desde cualquier lugar físico a una dirección específica, usualmente a una localidad de memoria.
[36] Del término en inglés: *"Hypertext Transfer Protocol"*.
[37] Identificador que nos permite distinguir la dirección de una computadora del resto de las direcciones.

y este documento se actualizará de manera dinámica cada que los comentarios se actualicen en sus respectivos documentos. Podríamos decir que los XPointers nos permitirán crear "documentos vivos". Sin XPointer sólo podríamos hacer referencia a documentos completos, y sería imposible lograr que hubiera una actualización dinámica de nuestro documento.

La forma más simple en el manejo de XPointers, por ejemplo, nos permite hacer referencia a un elemento XML mediante su ID[38] de manera independiente a su localización dentro del documento. Por ejemplo:

```
<?xml versión="1.0"?>
<!DOCTYPE congresos SYSTEM "congresos.dtd">

<congresos>
<fecha ID="Enero.2000">
<duracion> 15-16 de Enero</duracion>
</fecha>
<fecha ID="Febrero.2000">
<duracion> 17-19 de Febrero</duracion>
</fecha>
</congresos>
```

Ahora, si nos interesara conocer la fecha en que en Enero será realizado el congreso X sólo tendríamos que hacer lo siguiente:

http://www.dondesea.com/congresos.xml#id(Enero.2000)

El Xpointer (para este caso id(Enero.2000)) sólo tiene la función de señalar algo. Por lo que es posible que se presente la ventana del navegador con la parte señalada únicamente remarcada. Las acciones posteriores a la acción de XPointer, de exclusivamente señalar, dependerán del diseñador y del resto del contexto en el que sea utilizado Xpointer.

Con XPointer también es posible hacer referencia, de manera implícita, al elemento raíz del documento. Aunque de entrada esto parece no tener mucho sentido, en términos de localización más complejos, esto puede resultar muy útil debido a que podemos tomar la raíz (o cualquier otro ID) del documento como punto de partida. Si ya tenemos la referencia del elemento raíz, ahora podemos preguntar por su segundo elemento hijo, o por cualquier otro elemento que queramos.

[38] Nombre o identificador que se le asigna a un elemento XML.

3.8 DISCUSIÓN

A lo largo de este capítulo, se ha presentado una visión general de la tecnología XML. XML, por un lado, constituye un formato de texto, mientras que por el otro, representa una tecnología que pretende extenderse de manera indefinida.

Debido a este hecho, resultaría imposible para el presente trabajo abarcar todo el conjunto de especificaciones que conforman dicha tecnología. Razón por la cual, el objetivo de este capítulo ha sido presentar, mediante un panorama general, lo que es XML, ahondando solamente en los puntos de mayor interés para este trabajo.

Una vez estudiada la tecnología XML, se pretende complementar el marco teórico de la tesis, con un estudio más amplio sobre la especificación XSLT (algunos de los conceptos más generales de ésta, ya han sido revisados durante este capítulo).

La importancia de estudiar la especificación XSLT, más a fondo, resalta desde el objetivo mismo de la tesis, donde se propone basar en dicha especificación el desarrollo del sistema de consulta y visualización. Razón por la cual, el siguiente capítulo está dedicado, en su totalidad, al estudio de XSLT.

CAPÍTULO 4

XSLT: LENGUAJE DE TRANSFORMACIÓN EXTENSIBLE BASADO EN HOJAS DE ESTILO

El 18 de agosto de 1998 la W3C presenta el primer boceto de lo que sería la especificación XSL. Basada en el estándar DSSSL y definido mediante una sintaxis XML, esta especificación tenía como principal objetivo proporcionar un formato a los documentos XML de una manera estandarizada. Sin embargo, como se comentó anteriormente, poco tiempo después de su presentación, la definición original de XSL desapareció dando lugar a dos nuevas especificaciones:

- XSLFO: un lenguaje que permite asignar características de formato a elementos de los documentos XML.
- XSLT: un lenguaje que permite transformar un documento XML en otro documento XML.

El 16 de noviembre de 1999 fue publicada lo que sería la primera versión de la recomendación[39] XSLT en su versión 1.0. Es importante mencionar que este capítulo trata exclusivamente sobre esta especificación y no sobre ambas.

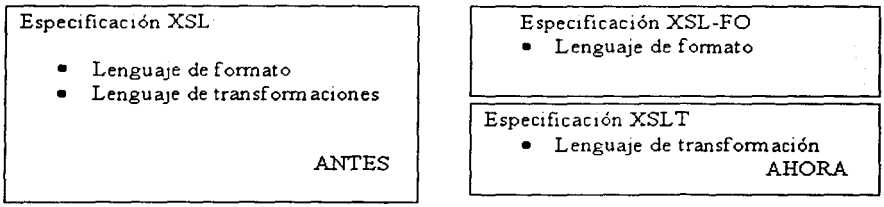

Figura 4.1: "*Evolución de XSL*"

En la Figura 4.1, podemos observar como en un inicio la especificación XSL representaba una sola especificación que contemplaba la realización de dos procesos: un primer proceso encargado de proporcionar formato a los documentos, y un segundo proceso, capaz de transformar la estructura de dichos documentos. Posteriormente, se realizo la reformulación de cada uno de dichos procesos dentro de especificaciones independientes, dando como origen XSLFO y XSLT.

[39] Para el consorcio W3C una recomendación es el equivalente a un estándar.

4.1 DESCRIPCIÓN GENERAL DE XSLT

XSLT es un lenguaje que fue diseñado originalmente para transformar un documento XML en otro documento XML. Sin embargo, la capacidad de XSLT va más allá de sólo poder realizar estas transformaciones, por lo que una definición general para XSLT podría ser:

> *XSLT es un lenguaje para transformar la estructura de los documentos XML [Kay et al., 2000].*

Siendo XML un lenguaje que proporciona tan sólo un medio estándar para intercambiar información de manera estructurada entre programas de computadoras, podríamos preguntarnos que pasa cuando esta información ha llegado a su etapa final, o dicho de otra manera, cuando esta información requiere ser presentada en un formato distinto al que se utilizó para enviarla a través de la Web. Si quisiéramos utilizar nuestra información para presentársela a un lector, por ejemplo, difícilmente podríamos pensar que el hacerlo en un formato XML sería una buena idea. Por lo tanto, una vez concluido el intercambio de datos requerimos transformar nuestros documentos XML en algo diferente.

XSLT representa la solución a este problema, ya que nos permite transformar los documentos XML en documentos con formatos de salida que son mucho más legibles o comprensibles para un lector: como documentos HTML o PDF. Sin embargo, es importante mencionar que la tarea de XSLT no termina aquí. La transferencia de datos entre aplicaciones distintas con frecuencia requerirá de pasar de un primer modelo de datos a un segundo modelo totalmente distinto. Por ejemplo, pensemos en información residente dentro de una base de datos, donde dicha información es utilizada para generar, de manera dinámica, el código HTML de un página Web[40].

El garantizar que todo el mundo esté usando XML no significa que la necesidad de convertir datos va a desaparecer, por el contrario, siempre existirán múltiples formatos de datos en uso y en consecuencia una gran necesidad de poder intercambiar estos datos de un formato a otro. Por lo que XSLT representa la herramienta ideal para llevar adelante esta ambiciosa tarea.

Como nos lo muestra la figura 4.2, la función primordial de XSLT consiste en constituirse como la herramienta que nos permita pasar de XML a cualquier otro formato de datos.

[40] Un ejemplo de dicha aplicación puede encontrarse en: *http://www.linuxfocus.org/*

Actualmente existen dos versiones de XSLT, la versión 1.0, que se encuentra en calidad de recomendación, que es la que utilizan la mayoría de los procesadores[41]; y la versión 1.1, que se encuentra como borrador de trabajo[42].

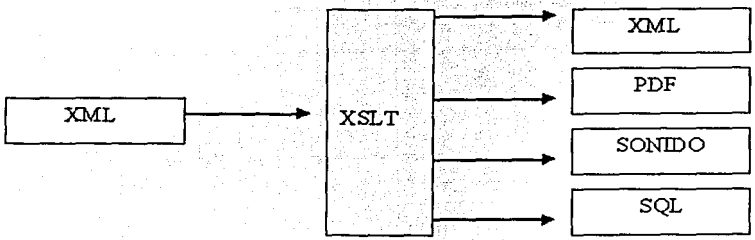

Figura 4.2: "*La tarea de XSLT*"

4.2 HERRAMIENTAS XSLT

La transformación de un documento XML requiere de tres parámetros fundamentalmente. Primero, el documento XML que requerimos transformar y la hoja de estilo donde especificamos las particularidades de la transformación y, finalmente, un procesador XSLT que será el encargado de recibir, como datos de entrada, el documento XML junto con una hoja de estilo XSLT, para posteriormente devolver, a la salida, un nuevo documento, normalmente en formato XML o HTML. Aunque los procesadores XSLT fueron diseñados prioritariamente para generar texto en estos formatos, existen algunos procesadores capaces de generar otros formatos de texto.

Podemos dividir el proceso de transformación en dos etapas básicas:

- La transformación estructural. En esta etapa los datos cambian su estructura, primeramente, se extraen de la estructura del documento XML de entrada, y posteriormente, son llevados a la estructura del nuevo documento del que formarán parte.
- La transformación de formato. En esta segunda etapa, la estructura de datos obtenida de la fase anterior, es procesada con la finalidad de asignarle el formato de texto que se requiere.

En las etapas de transformación de un documento XML, como se puede ver en la figura 4.3, interviene un procesador XSLT, el cuál, es orquestado por una hoja de

[41] Un procesador es el programa que interpreta el contenido de las hojas de estilo XSLT y genera el resultado de dicha interpretación.
[42] Del término en inglés "*working draft*" que significa el paso previo a convertirse en un estándar.

estilo XSLT donde el programador le indica a éste que partes debe transformar. Dicho de otra manera, la hoja de estilo contiene una serie de instrucciones, donde cada una especifica un segmento del documento XML, que puede ir desde una sola etiqueta hasta todo el documento completo, y la salida que será asociada a este documento. Cabe mencionar que estás reglas no presentan ningún tipo de orden, ni tampoco tienen relación con el orden de salida de los elementos del documento resultante de la transformación. Dentro de la hoja de estilo se especifica, únicamente, que valores debemos tener a la salida cuando se presenten ciertos valores a la entrada. Esta característica singular es lo que hace de XSLT un lenguaje declarativo[43].

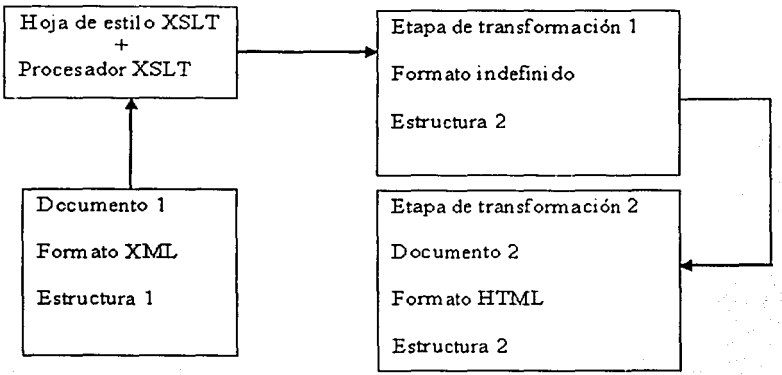

Figura 4.3: "*Las etapas de transformación de un documento XML*"

La transformación de un documento XML se pueden presentar en tres formas distintas:

- *El documento XML y la hoja de estilo XSLT asociada a éste son enviadas al navegador de la computadora cliente, el cual deberá posteriormente realizar la transformación y presentar el documento resultante al usuario.*
- *La computadora servidor realiza la transformación, generalmente a un formato HTML, y envía el documento ya transformado al navegador de la computadora cliente.*
- *El documento XML es transformado a otro formato, usualmente HTML, mediante la utilización de algún programa de software. Posterior a esto, el documento transformado se coloca en la computadora servidor [Rusty 2000].*

[43] Un lenguaje declarativo es aquel que pertenece al paradigma de programación funcional.

Podemos decir que en los primeros dos casos la transformación del documento se realiza de manera dinámica, o sea, instantes antes de la visualización del documento; mientras que en el tercer caso, la transformación del documento se realiza antes de la visualización del mismo.

4.2.1 PROCESADORES XSLT

En la actualidad existen diversos tipos de procesadores XSLT, sin embargo, la mayor parte de estos esta todavía en fase de desarrollo. Debido a que el papel principal de todos estos procesadores es básicamente el mismo, aplicar una hoja de estilo a un documento XML fuente para obtener otro documento resultado a la salida, en este capítulo hablaremos solamente de tres de los más utilizados:

- *Saxon*
- *XT*
- *Microsoft MSXML3 [Crane 2000].*

SAXON

Saxon es una herramienta de código abierto[44]. Su característica principal es que puede ser invocado desde la línea de comandos, por lo que no se requiere un Navegador. Saxon genera a la salida un documento en formato HTML, el cual puede ser posteriormente visualizado desde cualquier otro programa que maneje este formato. Además, este procesador puede ser ejecutado desde cualquier sistema operativo con el único requerimiento de que tenga instalado Java.

Ejemplo:

Si quisiéramos ejecutar una transformación con Saxon, por ejemplo desde un sistema operativo Windows, haríamos lo siguiente:

- Bajar el programa saxon.exe[45] para Windows.
- Instalar este programa en un directorio específico.
- Desde la línea de comandos, teclearíamos lo siguiente:

```
saxon documento1.xml documento2.xsl
```

Este procedimiento desplegaría a la salida en pantalla un documento HTML. Si quisiéramos generar físicamente el documento, ahora tendríamos lo siguiente:

[44] Del término en inglés:"*Open Source Software*". Para más información véase: *http://www.opensource.org/docs/osd-spanish.html*
[45] Este programa se puede obtener en: *http://users.iclway.co.uk/mhkay/saxon/*.

saxon documento1.xml documento2.xsl > documento3.html

Ahora, podemos abrir el documento documento3.html desde cualquier navegador.

XT

XT también es un procesador XSLT de código abierto. Saxon y XT no sólo son similares en sus características de diseño, sino también en su funcionamiento como veremos a continuación.

Para ejecutar XT desde Windows tendríamos los siguientes pasos:

- Bajar el programa xt.exe[46] para Windows.
- Posterior a esto, el resto de la secuencia sería exactamente la misma que para Saxon (sólo que ahora tendríamos que cambiar el nombre del programa a ejecutar).

Si teníamos, para el caso se Saxon: saxon documento1.xml documento2.xsl, ahora ponemos, para el caso de XT: xt documento1.xml documento2.xsl. Incluso para generar el documento documento3.html el procedimiento es exactamente igual: xt documento1.xml documento2.xsl > documento3.html.

DIRERENCIAS ENTRE SAXON Y XT

XT ha sido sin duda una de las principales contribuciones a la especificación XSLT debido a la cercanía que siempre mantuvo con las recomendaciones de ésta. En consecuencia, XT también fue por mucho tiempo el procesador XSLT más usado, sin embargo, ya no lo es, debido a la falta actualizaciones que han impedido generar una versión de XT capaz de competir con los nuevos procesadores.

Por el contrario, Saxon ha logrado consolidarse como uno de los procesadores XSLT más avanzados y eficientes. Gracias a sus actualizaciones constantes para corrección de errores y aportaciones adicionales, Saxon es considerado por muchos el procesador XSLT más eficiente de la actualidad.

MSXML

De manera casi inmediata a la primera publicación en 1998, de lo que en ese entonces sería el primer borrador de la especificación XSL, Microsoft presentó una tecnología conocida como MSXSL, para usarse dentro de IE4[47] y posteriormente dentro de IE5. Este procesador soportaba sólo parte de dicha especificación, pero

[46] Este programa puede ser bajado de: *http://www.jclark.com/xml/xt.html*.
[47] Del término "Internet Explorer".

añadía a está numerosas aportaciones que salían por completo de la especificación oficial.

Para la publicación del primer borrador de la versión renovada XSLT 1.0, a finales de 1999, ya existían millones de copias del IE5 y IE5.5 por todo el mundo que incluían el "soporte" para la antigua especificación. Debido a este hecho, Microsoft presentó, pocos meses después, una actualización para la nueva especificación XSLT que llamó MSXML2, sin embargo, nuevamente, este soporte presentaba algunas aportaciones adicionales a la especificación oficial que estaban fuera de norma.

MSXML es un procesador XSLT que realiza las transformaciones de documentos XML de manera dinámica. A diferencia de XT o Saxon, MSXML realiza la transformación del documento en el momento en que el usuario solicita visualizarlo. Por lo tanto, es IE quien invoca a MSXML para que éste realice la transformación, una vez que la transformación se realiza, IE sólo muestra el documento ya transformado (por lo tanto, el documento XML original nunca aparece en la pantalla del IE). El documento XML debe contener, en una primera línea, la liga a la hoja de estilo para indicar donde se encuentra ésta.

Ejemplo de liga:

```
<?xml-stylesheet type="text/xsl" href="hoja_de_estilo.xsl"?>
< documento_xml>
</ documento_xml>
```

Actualmente MSXML está en su versión 4 y las mejoras observadas, en comparación a su primera versión, no son pocas. MSXML4 soporta gran parte de la recomendación XSLT 1.0. Sin embargo, el problema no es tan simple, porque como ya se dijo hay millones de copias de IE que no incluyen la última versión del MSXML.

Antes de empezar a construir hojas de estilo XSLT, es importante saber exactamente qué navegadores las soportan y a qué nivel. Hoy día, Internet Explorer es el único navegador comercial que soporta XSLT, siempre y cuando sea actualizado el MSXML, al menos, a la versión 3. Otro navegador del que se espera que en poco tiempo soporte las transformaciones XSLT es Mozilla, el cual, desde la versión 0.7 incluye un soporte parcial aunque todavía en versión beta[48].

[48] Versión previa a la versión final de la liberación de un programa de software.

4.2.2 HOJAS DE ESTILO XSLT

ELEMENTOS QUE INTEGRAN UNA HOJA DE ESTILO XSLT

Aunque las hojas de estilos XSLT son muy potentes y pueden llegar a ser algo complejas al aplicarse a documentos XML complejos, su estructura básica es muy clara. Una hoja de estilos XSL consta de una o más *plantillas* que describen las *reglas* de transformación, que se usan para decir como transformar el documento XML. Por lo tanto, podemos decir que una hoja de estilo XSLT consta de dos partes fundamentales que son:

- Plantillas.
- Reglas de transformación.

Una plantilla es una estructura XSLT que describe la salida a generar con base en ciertos criterios de coincidencia. La idea básica consiste en definir una serie de reglas de transformación que se apliquen a una cierta parte de un documento XML.

Las plantillas se definen en las hojas de estilos XSLT utilizando el elemento xsl:template, que es, principalmente, un elemento contenedor de las reglas y datos de transformación. El elemento xsl:template utiliza un atributo opcional llamado match para hacer coincidir los patrones en un documento XML. El atributo match especifica la parte del documento XML que se quiere transformar.

Para especificar todo el documento, por ejemplo, utilizaríamos el atributo "/", que indicaría que estamos seleccionando el documento XML desde la raíz o elemento inicial. Posteriormente, la regla de transformación que especificáramos sería aplicada a todo el documento XML y no solamente a una fracción de éste.

Ejemplo:

```
<xsl:template match='/'>
</xsl:template>
```

Haciendo una analogía podemos pensar en al atributo match de una plantilla como la instrucción de consulta de un lenguaje de bases de datos (los datos del documento XML que coinciden con el valor del atributo match se pasan a la plantilla para su procesamiento).

Ejemplo:

```
<xsl:template match="titulo">
<xsl:value-of/>
</xsl:template>
```

Lo anterior significa que en la plantilla sólo se estarían considerando los elementos que coincidieran en valor con la palabra *titulo.*

En resumen, podemos decir que las plantillas XSLT utilizan un parámetro que les permite determinar que porción del documento XML será tomada en cuenta durante la etapa de transformación. Este parámetro, para nuestro último ejemplo titulo, describe una porción del documento XML. No se debe olvidar que un documento XML esta formado por un conjunto de etiquetas perfectamente anidadas.

La sintaxis que utiliza para especificar este parámetro de selección es parecida a la que se usa para especificar la ruta de un archivo dentro de una computadora. Por ejemplo, el parámetro titulo/autores/nombre selecciona los elementos nombre que pertenecen a otro elemento llamado autores, que a su vez pertenece a otro elemento llamado titulo.

Existen diversos elementos o instrucciones XSLT, normalmente dentro de las plantillas, que nos sirven para especificar con exactitud las reglas de transformación que queremos utilizar para dar formato a los elementos que anteriormente fueron seleccionados en las plantillas.

A continuación veremos algunos de estos elementos, los más comunes:

- xsl:value-of
- xsl:if
- xsl:for-each

XSL:VALUE-OF

El elemento xsl:value-of se usa para insertar el valor de un elemento o atributo en la salida de la hoja de estilo, xsl:value-of proporciona el medio para transformar los documentos XML a cualquier otro formato, como por ejemplo dentro de marcas que pertenecen al formato HTML.

Veamos un ejemplo de la sintaxis de este elemento:

```
<xsl:value-of select="parametro_de_seleccion"/>
```

XSL:IF

El elemento xsl:if se usa para realizar selecciones condicionales dentro de las plantillas. Este elemento utiliza el elemento test para indicar una evaluación, si el resultado de esta evaluación es verdadero entonces se ejecutan las instrucciones contenidas dentro la etiqueta.

Veamos un ejemplo de la sintaxis de este elemento:

```
<xsl:if test="//etiqueta1[parametro1 = valor]">
[ejecuta algo]
</xsl:if>
```

XSL:FOR-EACH

El elemento xsl:for-each se utiliza para establecer un ciclo que recorre los elementos de un documento que cumplen con una condición. El atributo select determina la condición que los elementos deberán cumplir para que sean seleccionados como parte del recorrido del ciclo. Otro atributo importante del elemento xsl:for-each es order by, que especifica los criterios que se usan para ordenar la forma en la que aparecerán los datos a la salida.

Veamos un ejemplo de la sintaxis de este elemento:

```
<xsl:for-each select="//etiqueta1">
<xsl:sort order="ascending" />
[ejecuta algo]
</xsl:for-each>
```

EJEMPLO DE UNA HOJA DE ESTILO XSLT

A continuación se muestra un ejemplo completo y la descripción de su funcionamiento:

Documento XML

```
<?xml version="1.0"?>
<Tutorial >
<negritas>Hola mundo!</ negritas >
<rojo>(Esto es rojo)</rojo>
<italica>(Esto esta en letra italica)</italica>
</Tutorial>
```

Hoja de estilo XSLT

```
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match=" negritas">
<P><B><xsl:value-of select="."/></B></P>
</xsl:template>

<xsl:template match="rojo">
<P style="color:red"><xsl:value-of select="."/></P>
</xsl:template>

<xsl:template match="italica">
<P><i><xsl:value-of select="."/></i></P>
</xsl:template>
</xsl:stylesheet>
```

Documento HTML generado:

```
<P> <B> Hola mundo!</B></P>
<P style="color:red">(Esto es rojo)</P>
<P><i>(Esto esta en letra italica)</i></P>
```

Salida:

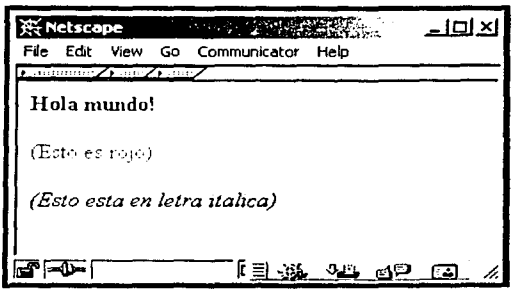

Figura 4.4: "*Ejemplo XSLT*"

La figura 4.4, muestra la salida resultante al ejecutar el ejemplo anterior con el procesador Saxon. El archivo resultante es visualizado, posteriormente, con el navegador Netscape.

Descripción del funcionamiento:

Esta hoja de estilo está formada por tres plantillas. Anterior a la primera, en lo que podríamos llamar la primer línea de código, que se muestra a continuación, se indica la etiqueta raíz o inicial y, de manera opcional, como parámetros adicionales de ésta se puede indicar la versión a partir de la cual fue generada la hoja y el sitio Web donde puede ser consultada la especificación oficial de esa versión.

<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

Algo importante de esta primer etiqueta, es que dentro de la misma deberá estar contenido el resto de las etiquetas, no olvidemos que una hoja de estilo XSLT también es un documento XML, por lo que la primer etiqueta debe anidar o contener el resto de las etiquetas de ese documento.

En la primer plantilla, o sea, las siguientes tres líneas de código, a diferencia de la primer línea, se describen instrucciones de transformación. En la primer línea de esta plantilla se hace la selección de todas las etiquetas llamadas negritas y, además, de las etiquetas que se encuentren anidadas dentro de éstas, para este caso es importante mencionar que la etiqueta negritas es única y, además, no contiene ninguna otra etiqueta anidada. En la segunda línea, utilizamos las etiquetas de formato de <p> <b> de HTML para indicar que queremos realizar un párrafo cuyo contenido estará escrito con letra en formato negrita.

A continuación, indicamos el contenido de estas etiquetas por medio del comando xsl:value-of select. Para este caso, el contenido está dado por '.', lo que significa, que tomará el contenido de todas las etiquetas que hayan sido seleccionadas previamente por la instrucción xsl:template. Si no quisiéramos seleccionar todo el contenido de todas las etiquetas, hubiéramos tenido que poner en vez del '.' el nombre de cualquier etiqueta que estuviera contenida dentro de la etiqueta negritas. Finalmente, lo único que resta es cerrar todas las etiquetas que utilizamos para esta instrucción. Estás serían las siguientes tres líneas de código que hemos usado hasta este momento:

```
<xsl:template match=" negritas">
<P><B><xsl:value-of select="."/></B></P>
</xsl:template>
```

Para las siguientes dos plantillas lo único que cambia es, para la segunda el contenido del párrafo tendrá color rojo y ya no será texto en negritas; y para la tercera, el contenido del párrafo estará con texto en formato itálico y no en color rojo. Finalmente, sólo se agrega una línea que indica el cierre de la etiqueta con la que iniciamos la hoja de estilo. A continuación se muestran el código de las últimas líneas:

```
<xsl:template match="rojo">
<P style="color:red"><xsl:value-of select="."/></P>
</xsl:template>

<xsl:template match="italica">
<P><i><xsl:value-of select="."/></i></P>
</xsl:template>
</xsl:stylesheet>
```

4.3 FUTURO DE LAS APLICACIONES XSLT

Es importante definir algunas de las áreas donde las aplicaciones XSLT tendrán mayor impacto, o sea, aquellas tareas para las cuales XSLT representará verdaderamente la herramienta de solución idónea.

Podemos hablar, principalmente, de dos escenarios dentro de los cuales XSLT parece ser una herramienta esencial:

- Conversión de texto de un formato a otro.
- Publicación de datos.

4.3.1 CONVERSIÓN DE TEXTO DE UN FORMATO A OTRO

La necesidad de convertir o pasar un conjunto de datos de un cierto formato a otro, es un problema tan complejo que resulta imposible pensar que el sólo hecho de la existencia de un lenguaje como XML ha resuelto el problema definitivamente. Si bien, es cierto que muchos de los problemas de intercambio de información entre organizaciones o aplicaciones han encontrado una solución en XML, el problema de la existencia de diferentes modelos de datos, o de diferentes formatos para representar una misma información, es un problema que todavía ve lejana una solución definitiva.

Por ello, requerimos de una herramienta eficiente, que nos permita pasar de un formato de datos a otro. Esta herramienta es XSLT.

La figura 4.5 muestra lo versátil que pueden llegar a ser las transformaciones de documentos XML, éstas pueden llegar a ser tan complejas y versátiles como se requiera. Por ejemplo, el envío de noticias, en tiempo real, a través de la telefonía móvil, se puede generar a partir de documentos XML[49].

[49] *"Voice-based XML comes to your car"*, es un artículo publicado sobre este tema que puede ser consultado en: *http://www.cnn.com/2000/TECH/computing/05/24/xml.car.idg/*.

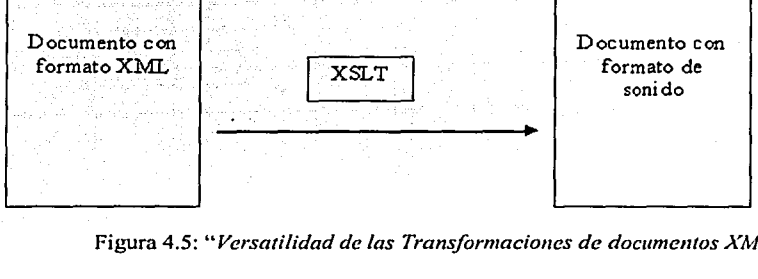

Figura 4.5: "*Versatilidad de las Transformaciones de documentos XML*"

XSLT es una herramienta que nos permite, de manera muy sencilla, extraer datos, reordenarlos, agregar o quitar atributos de éstos, etc. Si pensamos, por ejemplo, en convertir ciertas columnas de un texto a reglones, XSLT resulta una solución idónea para resolver este problema.

Sin embargo, si pensamos en ciertas tareas como, modificar el contenido de una cadena de texto, o la búsqueda y eliminación de espacios en blanco del contenido, XSLT resulta una herramienta poco práctica y que requerirá un trabajo mucho más elaborado que el que realizaríamos con otros lenguajes de programación como Javascript o Java. De cualquier manera, XSLT es capaz de invocar procedimientos escritos en estos lenguajes, siendo capaz de solucionar incluso muchos de los problemas para los cuales no fue pensado gracias a la compatibilidad que presenta para trabajar con otros lenguajes.

Una de las características más sorprendente de XSLT, es que este lenguaje constituirá una herramienta sumamente útil para convertir o pasar desde un formato distinto a XML a otro.

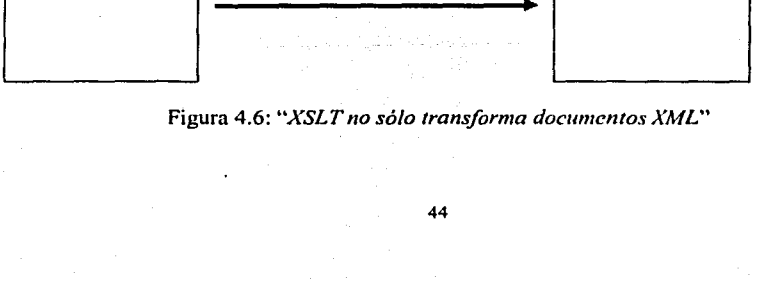

Figura 4.6: "*XSLT no sólo transforma documentos XML*"

Como se muestra en la figura 4.6, la transformación de un documento puede partir, incluso, de un formato de origen distinto de XML. Por ejemplo, en la actualidad, ya existen algunas aplicaciones capaces de transformar documentos con formatos como PDF o HTML en XML.

4.3.2 PUBLICACIÓN DE DATOS

Otra de las áreas que se verá altamente beneficiadas con la existencia de XSLT es la publicación de datos. La publicación de datos, a diferencia de conversión de formato cuya finalidad normalmente debería ser proporcionar un formato entendible para una aplicación de software, requiere proporcionar un formato entendible, o mejor dicho, legible para su lectura por seres humanos. Donde estos formatos irán, desde los ya tradicionales archivos multimedia, hasta la generación en línea de reportes de ventas, recibos de teléfono o cuentas de banco. Las aplicaciones XML XSLT serán ideales para cualquier tipo de publicación electrónica, tanto o más, de lo que HTML lo fue. Mientras que XML fue diseñado para mantener el contenido de la información independiente al formato, XSLT es ahora el medio para convertir esa información, sin presentación e inservible dentro del proceso de publicación electrónica, en una información sumamente versátil para su publicación.

Para la publicación electrónica la única problemática existente es convertir los datos de su formato inicial a cualquier formato especializado en la presentación de datos como HTML. Sin embargo, aunque HTML no es XML desde luego, las semejanzas son tantas que pasar de un formato al otro es sumamente simple. Es por ello que la aplicación más común de XSLT en la actualidad es precisamente esa, convertir un documento XML en otro documento HTML.

4.4 DESVENTAJAS DE XSLT

Una de las desventajas más serias de las transformaciones XSLT se presenta cuando se requiere trabajar con documentos de tamaño bastante grande y si ha esto agregamos el querer realizar sobre éstos transformaciones realmente complejas el problema aumenta considerablemente. En la actualidad cualquier procesador XSLT requiere poner todos los datos del documento en memoria durante el tiempo de ejecución de la transformación y, aunque la transformación de documentos considerablemente grandes puede realizarse en un periodo de tiempo razonablemente bueno, no hay manera de asegurar que siempre será así, por lo que debemos decir que algunos factores como la complejidad de la transformación o la cantidad de datos procesados pueden afectar considerablemente el rendimiento de la transformación.

Por ejemplo, si requiriéramos procesar enormes cantidades de datos para posteriormente extraer un cierto número de registros de manera selecta, entonces, estaríamos hablando de un problema para el cual XSLT ha dejado de ser la herramienta adecuada debido a su forma de operar. En la actualidad se está trabajando en el desarrollo de nuevas tecnologías que podrían trabajar en un futuro de manera conjunta con XSLT para dar solución a diversas problemáticas, que como ésta, actualmente están lejos del alcance de XSLT.

Un ejemplo de esto es el lenguaje de consultas XQuery:

> *El objetivo de XQuery consiste en proporcionar un lenguaje interactivo de consultas lo suficientemente flexible como para cubrir un amplio espectro de fuentes de información incluyendo Bases de Datos y documentos distribuidos a través del Web [Chamberlin et al., 2002].*

4.5 DISCUSIÓN

Actualmente XSLT se encuentra en la versión 1.0, basada en la recomendación aprobada el 16 de Noviembre de 1999 por el W3C. Sin embargo, no hay que olvidar que XSLT se encuentra todavía bajo desarrollo. El lenguaje ha cambiado de manera radical en el pasado y podemos estar casi seguros de que presentará nuevos cambios en el futuro. De cualquier manera, es muy probable que estos cambios, que se esperan mínimos, no repercutan de ninguna manera en los trabajos basados sobre la versión XSLT 1.0. Por lo que podemos afirmar que al liberarse la versión XSLT 2.0, misma sobre la que ya se está trabajando, todos los documentos basados en las versiones anteriores a ésta seguirán siendo perfectamente válidos.

Desafortunadamente, no todos los procesadores XSLT están basados en la versión XSLT 1.0. Particularmente, las versiones 5.0 y 5.5 del Internet Explorer de Microsoft, de las que actualmente se encuentran millones de copias por todo el mundo, solamente puede trabajar con una versión de XSL, muy antigua, que no se parece en nada a la recomendación oficial. Debido al gran impacto que tiene este producto de software dentro del mercado y a que la versión de Internet Explorer 6.0, con la especificación oficial, tardará todavía algún tiempo en llegar a la mayor parte de las computadoras del mundo; sería importante poner atención desde este momento al aspecto de normatividad de XSLT para no repetir lo ocurrido con HTML.

Finalmente, como ya se ha comentado, la razón fundamental por la que se decidió utilizar XSLT en el desarrollo de este sistema, se debió a que esta tecnología, a pesar de su reciente aparición, representa, como se estudió en los capítulos anteriores, la solución ideal para la manipulación de estructura y formato de los documentos XML que genera ELXI. Además, si comparamos XSLT con otras especificaciones de XML, como CSS o incluso la misma XSLFO, notamos que

estas últimas, se limitan a la transformación de datos y no a la de documentos como es el caso de XSLT.

Una vez finalizado el desarrollo teórico del presente trabajo, conformado por los Capítulos:

- Capítulo 2: Antecedentes de XML
- Capítulo 3: Conceptos básicos de XML
- Capítulo 4: XSLT: Lenguaje de Transformación basado en hojas de Estilo Extensible.

Dicho desarrollo pretende haber sentado las bases de conocimiento mínimas, para continuar con el estudio del desarrollo práctico. El siguiente capítulo, detalla, entonces; las fases de análisis, diseño e implementación llevadas acabo durante la construcción del sistema de consulta y visualización para de los documentos XML generados por ELXI.

CAPÍTULO 5

ARQUITECTURA DEL SISTEMA

5.1 INTRODUCCIÓN

A partir de este capítulo, el documento de tesis se enfocará al análisis, diseño e implementación de la aplicación. Es importante mencionar que estas tres etapas que se mencionan forman la columna vertebral de la metodología propuesta para la presente tesis y, además, están presentes también en algunas de las principales metodologías de desarrollo de software de la última década como veremos a continuación.

Metodología / Autor	Libros	Actividades
Object Oriented Design Grady Booch	Object Oriented Design with Applications Benjamming Cummings,1991	Análisis de Requisitos Análisis de Dominio Diseño
Objectory Ivar Jacobson	Object-Oriented Software Engineering: A Use Case Driven Approach Addison-Wesley,1992	Análisis de Requisitos Análisis de Robustez Diseño Implementación Pruebas
Object Modeling Technique James Rumbaugh	Object Oriented Modeling and Design Prentice Hall, 1991	Análisis Diseño del Sistema Diseño de Objetos Implementación

Figura 5.1: "*Metodologías*"

A finales de 1997 y gracias a la participación en un proyecto común, dentro de la empresa *Rational*, de los tres autores destacados en la figura 5.1, las metodologías de cada uno de estos fueron fusionadas en una sola. Sería a finales de 1998, cuando dichos autores publican la versión definitiva de la nueva metodología de desarrollo bajo la denominación del "Proceso Unificado para Desarrollo de Software". En general, la metodología del "Proceso Unificado", al igual que las metodologías que lo integran, está formada por el siguiente grupo de actividades [Booch et al., 1999]:

- Análisis de Requerimientos.
- Diseño del sistema.
- Implementación.

Tomando como base dichas actividades, la presente tesis propone las mismas como la guía para el desarrollo del sistema. Antes de iniciar con la metodología, es importante mencionar que el primer punto a tratar por el presente capítulo es un panorama general de ELXI, puntualizando sólo en las partes de éste que se requieren para comprender el sistema desarrollado durante este trabajo.

5.2 ELXI: EDITOR COLABORATIVO DE DOCUMENTOS XML EN INTERNET

El editor colaborativo ELXI, como ya se mencionó, fue desarrollado en el IIMAS dentro de un proyecto de investigación dirigido por el Dr. Manuel Romero Salcedo, mismo director de la presente tesis. El objetivo principal de dicho editor consiste, primordialmente, en proporcionar un medio para crear documentos de forma colaborativa.

En su primera versión, ELXI genera documentos XML de tipo artículo de investigación. Dichos documentos estarán organizados por grupos, por lo que todos los artículos generados pertenecerán a un solo grupo. Físicamente, los grupos están representados por directorios del sistema de archivos de la computadora donde está instalada la aplicación servidor de ELXI. Así, por cada grupo, un directorio llevará el mismo nombre de éste y contendrá todos los artículos que pertenecen al grupo, cada uno dentro de un documento XML. Todos los directorios estarán contenidos en el otro directorio padre llamado ELXI, y ningún grupo podrá tener más de un directorio.

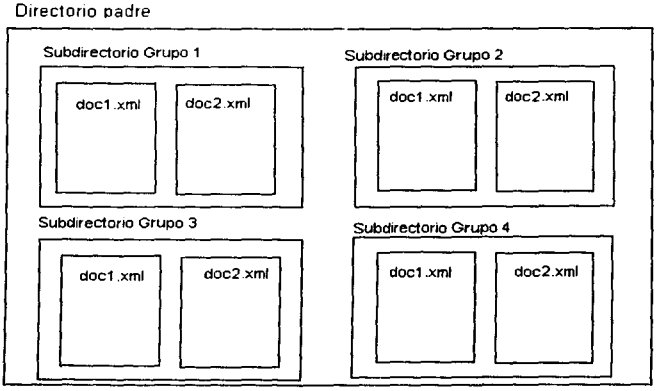

Figura 5.2: "*Organización de documentos realizada por ELXI*"

La figura 5.2 muestra la organización de documentos que realiza ELXI para almacenar los artículos generados. El *Directorio padre*, llamado ELXI dentro del sistema de archivos, contiene una serie de subdirectorios que representan los grupos en que están clasificados los artículos existentes. Dentro de cada directorio o grupo están contenidos todos los artículos que pertenecen a dicho grupo. Para este caso, se supone que se han creado cuatro grupos distintos de artículos, y que cada uno de estos contiene dos artículos.

Además del grupo al que pertenecen, cada uno de los artículos permanece en un estado de disponibilidad único. Los estados en los que podrá residir un documento son:

- *Disponible y visible.* Se puede ver la referencia y se puede hacer la consulta del artículo.
- *No disponible y visible.* Se puede ver la referencia pero no se puede hacer la consulta del artículo.
- *Invisible.* No se puede ver la referencia y, por lo tanto, tampoco se puede hacer la consulta del artículo.

A diferencia de estas peculiares características, los artículos generados por ELXI comparten todas las características propias de cualquier artículo de tipo investigación, como puede ser: un primer autor, varios o ningún segundo autor, índice de palabras clave, resumen, bibliografía, entre muchas más.

Una vez aclarados estos puntos sólo resta mencionar que en lo posterior del presente capítulo se estará haciendo referencia a estas características de visibilidad y disponibilidad de los artículos. Por lo que es importante mantenerlas presentes.

5.3 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA

5.3.1 DEFINICIÓN DEL PROBLEMA

Generar un sistema de consulta y visualización, basado en XSLT, para los diversos documentos XML generados por ELXI. Este sistema debe ser capaz, primero, de convertirse en un medio de acceso y búsqueda eficiente a estos artículos y, segundo, la vía para proporcionarles los distintos formatos que requieren para su publicación.

DIAGRAMA GENERAL DEL SISTEMA

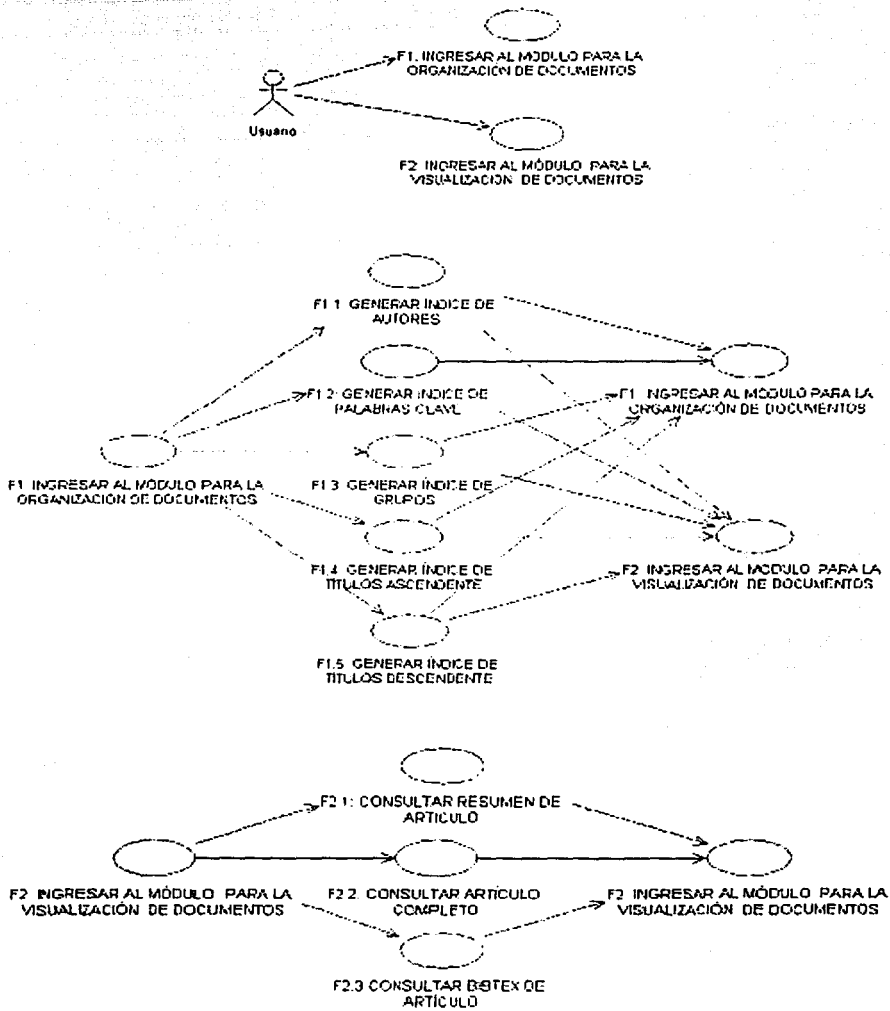

Figura 5.3: "*Diagrama general del sistema*"

5.3.2 REQUERIMIENTOS FUNCIONALES

La figura 5.3 muestra un panorama general de lo que deberá ser la arquitectura del sistema, la cuál estará formada por dos módulos independientes, uno para la organización y otro para la visualización de documentos. Cada uno de estos módulos, como se puede observar en la figura 5.3, presentará distintas funcionalidades. Si bien, se espera que dichos módulos presenten cambios mínimos en su etapa de implementación, podría no ser así con las funcionalidades que éstos presentan, debido a que esta es sólo una primer etapa de análisis.

DESCRIPCIÓN DEL DIAGRAMA GENERAL DEL SISTEMA (CASOS DE USO)

F1: INGRESAR AL MÓDULO PARA LA ORGANIZACIÓN DE DOCUMENTOS

Flujo de eventos principal:

- El usuario ingresa al módulo
- El sistema muestra una lista de los artículos disponibles para que el usuario seleccione el que desea consultar

F1.1: GENERAR ÍNDICE DE AUTORES

Flujo de eventos principal:

- El usuario solicita la creación de un índice de autores de los artículos disponibles
- El sistema genera el índice de autores de los artículos disponibles para que el usuario seleccione el que desea consultar

F1.2: GENERAR ÍNDICE DE PALABRAS CLAVE

Flujo de eventos principal:

- El usuario solicita la creación de un índice de palabras clave de los artículos disponibles
- El sistema genera el índice de palabras clave de los artículos disponibles para que el usuario seleccione el que desea consultar

F1.3: GENERAR ÍNDICE DE GRUPOS

Flujo de eventos principal:

- El usuario solicita la creación de un índice de grupos de los artículos disponibles
- El sistema genera el índice de grupos de los artículos disponibles para que el usuario seleccione el que desea consultar

F1.4: GENERAR ÍNDICE DE TÍTULOS ASCENDENTE

Flujo de eventos principal:

- El usuario solicita un ordenamiento ascendente por título de los artículos mostrados
- El sistema ordena los artículos mostrados para que el usuario seleccione el que desea consultar

F1.5: GENERAR ÍNDICE DE TÍTULOS DESCENDENTE

Flujo de eventos principal:

- El usuario solicita un ordenamiento descendente por título de los artículos mostrados
- El sistema ordena los artículos mostrados para que el usuario seleccione el que desea consultar

F2: INGRESAR AL MÓDULO PARA LA VISUALIZACIÓN DE DOCUMENTOS

F2.1: CONSULTAR RESUMEN DE ARTÍCULO

Flujo de eventos principal:

- El usuario solicita la consulta de uno de los artículos mostrados
- El sistema muestra el resumen del artículo solicitado

Flujo de eventos excepcional:

- Si el artículo no está disponible el sistema no habilita la funcionalidad para la consulta del mismo y muestra una leyenda indicando el suceso.

F2.2: CONSULTAR ARTÍCULO COMPLETO

Flujo de eventos principal:

- El usuario solicita la consulta del contenido completo del artículo mostrado
- El sistema muestra el artículo solicitado

F2.3 CONSULTAR BIBTEX DE ARTÍCULO

Flujo de eventos principal:

- El usuario solicita la consulta del Bibtex del artículo mostrado
- El sistema muestra el Bibtex del artículo

5.3.3 OTROS REQUERIMIENTOS

REQUISITOS DE HARDWARE

Básicamente, el sistema puede correr sobre cualquier PC con Internet Explorer 5, sin embargo, los requerimientos de hardware recomendables pueden resumirse en los siguientes puntos[50]:

- Procesador 166 MHz (Pentium o equivalente).
- 16 Mb de memoria RAM.
- Conexión a Internet.

REQUISITOS DE SOFTWARE

- Sistema operativo Windows versión 95/98/NT/2000/Me/XP.
- Internet Explorer versión 5 ó 5.5.
- Procesador XSLT MSXML versión 3 ó 4.

La plataforma sobre la que debe correr el sistema es una PC con sistema operativo Windows. Por otro lado, debido a la naturaleza del mismo sistema, es necesario que en dicha plataforma se cuente con el navegador Microsoft Internet Explorer en su versión 5 ó 5.5. Además, dicho navegador requiere estar actualizado para incluir el soporte de XSLT[51].

Es importante mencionar que ningún navegador Internet Explorer, en sus versiones 5 ó 5.5, cuenta con el procesador MSXML3 o MSXML4, indispensable para visualizar documentos transformados con XSLT, por lo tanto antes de poder trabajar con dichos documentos será necesario realizar la actualización. Por otro lado, la versión 6 de Microsoft Internet Explorer, actualmente liberada, no resulta muy recomendable debido a que a pesar de integrar el soporte completo para el

[50] Requerimientos mínimos para correr Internet Explorer 5.
[51] Para mayor información de cómo realizar la actualización visitar:
http://www.microsoft.com/Latam/msdn/articulos/2001/12/art03/default.asp

trabajo de XSLT, el sistema requiere, por razones de eficiencia, de un Script[52] que fue programado especialmente para su uso dentro de Microsoft Internet Explorer en cualquiera de sus versiones 5.x.

RECURSOS ESPECIALES

Para el desarrollo del proyecto han sido necesarias páginas Web, manuales especializados de programación en XSLT y libros que funcionan como fundamentos teóricos.

Otro de los recursos especiales que fueron requeridos para la realización del sistema, como se mencionó previamente, fue la utilización de un programa, cuyo código fuente fue modificado y adaptado a los propias necesidades del sistema. Este programa[53] permite realizar las transformaciones XSLT de manera dinámica, o sea, no se requiere cargar nuevamente el documento para realizar una transformación sobre él, dicha transformación puede ser consecuencia de los mismos eventos que ocurren dentro de la página.

LIMITACIONES DE DISEÑO

El sistema a desarrollar es afectado por las limitaciones presentadas a continuación:

- Las funcionalidades del sistema para realizar la organización de los documentos, dependen fundamentalmente de la generación de un índice donde se encuentran algunos atributos como dirección, autores y palabras claves de dichos documentos.
- El desempeño del sistema estará determinado por el ancho de banda de la conexión a Internet, así como de las capacidades de software y hardware de la computadora en la que se utiliza el sistema.

5.4 DISEÑO DE LA ARQUITECTURA DEL SISTEMA

Para el caso particular de este sistema el diseño de la arquitectura costa fundamentalmente de dos módulos:

- Módulo para la organización de los documentos
- Módulo para la visualización de los documentos

[52] Término que se utiliza para denotar algunos programas cuyo código fuente, usualmente, no va más allá de unas cuantas líneas de código.
[53] La versión original de este programa puede obtenerse en:
http://www.bayes.co.uk/xml/index.xml

5.4.1 MÓDULO PARA LA ORGANIZACIÓN DE LOS DOCUMENTOS

Este primer módulo está integrado por un documento XML, donde son almacenados algunos atributos de los documentos como son dirección, autor principal, grupo del documento y palabras claves. La finalidad de dicho documento, al que llamaremos índice, es proporcionar una referencia general de los atributos de todos los documentos generados por ELXI. Sobre el índice es importante mencionar que se tiene contemplada la creación de un pequeño programa que lo actualice periódicamente. Sin embargo, este hecho está planeado para futuras actualizaciones del sistema, y por el momento su actualización debe hacerse manualmente.

Adicional al índice, este módulo debe interactuar con un conjunto de programas XSLT, contenidos individualmente en archivos con extensión *"xsl"*. Son estos programas los que indican a Internet Explorer que documentos deben estar contenidos en el índice, como debe mostrarlos, como debe organizarlos y, finalmente, cuando se realizará la carga del módulo de visualización.

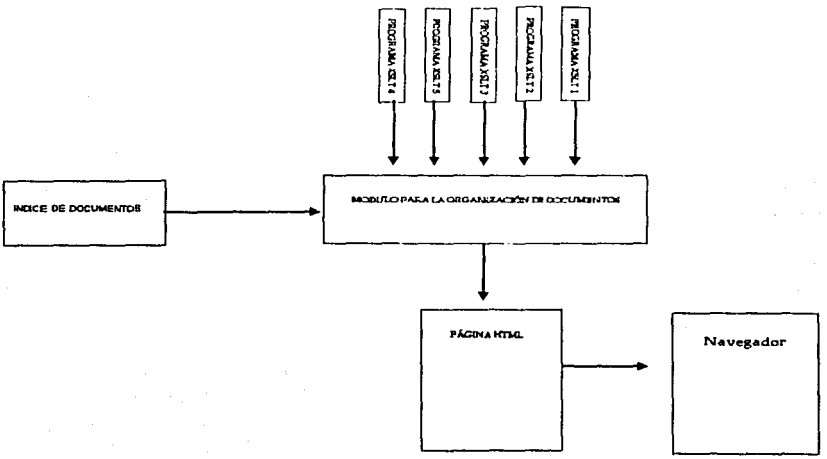

Figura 5.4: "*Diagrama de bloques del módulo para la organización de documentos*"

La figura 5.4 modela mediante un diagrama de bloques como está diseñado el módulo para la organización de los documentos. Dicho módulo debe recibir, por un lado, todos los datos referentes a los documentos contenidos en el índice; y por el otro lado, debe seleccionar el programa XSLT que le indicará que elementos de dicho índice deberá seleccionar y como debe mostrarlos dentro de la página HTML que mostrará Internet Explorer.

La función primordial de este módulo, como su nombre bien lo dice, es proporcionar una organización lógica que facilite al usuario la navegación por el conjunto de documentos generados por ELXI.

5.4.2 MÓDULO PARA LA VISUALIZACIÓN DE LOS DOCUMENTOS

El módulo para la visualización de los documentos está formado por un conjunto de documentos, los documentos previamente generados por ELXI, y al igual que el módulo anterior, también está integrado por un conjunto de programas XSLT que indican los diferentes estilos en que puede ser visualizado el documento.

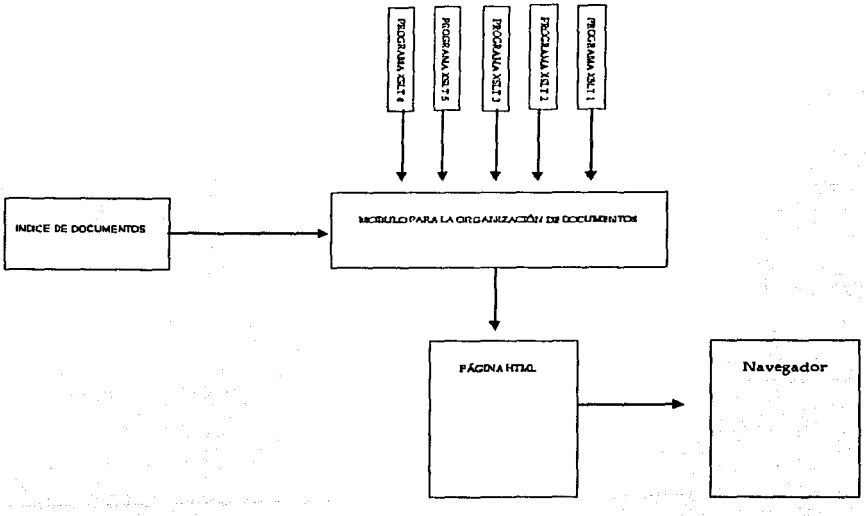

Figura 5.5: "*Diagrama de bloques del módulo para la visualización de documentos*"

Como principal tarea, este módulo será el encargado de proporcionar un estilo adecuado al documento visualizado. Para este caso en particular, donde se está trabajando con documentos de tipo artículos, es esté módulo lo que hace posible que el estilo de los documentos visualizados, a pesar de presentar un formato XML, puedan ser visualizados como cualquier otro artículo de investigación publicado en la Web.

La figura 5.5 muestra un diagrama de bloques con la descripción de la secuencia del funcionamiento de este módulo. Dicha secuencia está dada como sigue: inicialmente, este módulo parte de una selección previa del documento que se desea visualizar, esto quiere decir que, al momento de iniciar la carga del módulo ya se ha escogido uno de los documentos. La selección de dicho documento puede hacerse de dos formas, ingresando directamente al documento, o sea, abriendo el documento directamente con Internet Explorer, o bien, asistido por el módulo de organización de los documentos. Este módulo presenta la funcionalidad de seleccionar directamente uno de los documentos contenidos en el índice, y al realizarse esta selección se carga directamente el módulo de visualización de documentos con el documento seleccionado.

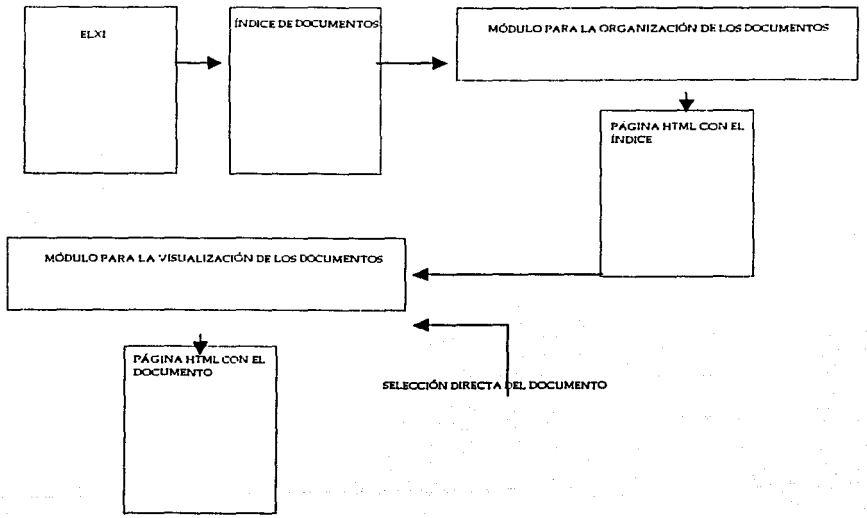

Figura 5.6: "*Diagrama de bloques de los módulo del sistema*"

La figura 5.6 describe todo el proceso que atraviesa un documento, posterior a su creación por ELXI, para poder ser visualizado. Primero, el módulo de organización de documentos toma el índice y lo presenta con una organización y un estilo adecuado para su correcta visualización. Este índice se muestra al usuario mediante un documento HTML generado dinámicamente por el módulo. Posterior a este proceso y una vez ya generado el índice, el usuario puede seleccionar entre el conjunto de documentos, sólo aquel documento que desea consultar. Una vez seleccionado el documento, el módulo para la visualización de los documentos proporcionará el estilo adecuado a éste y lo muestra dentro de una página HTML generada, nuevamente, al momento mismo de la selección. Es importante mencionar, como se ilustra en la figura 5.6, que la selección del documento puede hacerse directamente, o sea, sin la asistencia del índice, y la visualización del documento será exactamente la misma.

5.5 IMPLEMENTACIÓN DEL SISTEMA

5.5.1 MÓDULO PARA LA ORGANIZACIÓN DE LOS DOCUMENTOS

La implementación de este primer módulo consistió en proporcionar una organización lógica para el conjunto de documentos que pueden ser consultados por los usuarios que ingresen al sistema. Dicha organización se complementa con diferentes funcionalidades, que permiten al usuario realizar una búsqueda, eficiente y sencilla, de los documentos deseados.

Este módulo está constituido por un documento XML y un total de cinco hojas de estilo que contienen el código fuente de los programas XSLT que describen el conjunto de pantallas y funcionalidades que integran el módulo.

La figura 5.7 muestra la pantalla principal del sistema, esta pantalla está formada por una tabla que contiene el índice de los documentos. El índice que se genera en esta pantalla es el más completo en cuanto al contenido de información y se presenta al usuario mediante una tabla formada por cuatro columnas y tantos reglones como documentos visibles haya en ese momento. En las columnas, se despliega el título del documento, autor, grupo y palabras claves; en ese orden.

En el campo de "*Título*" de cada documento se presenta como información adicional el año de publicación y un botón que permite desplegar el resumen del artículo, para documentos disponibles, y sólo una leyenda en rojo de "*No disponible*" en caso de que el documento no esté disponible. En el campo de *"Autor"* se presentará en un segundo plano, la lista de segundos autores que participaron en la elaboración del artículo y sus respectivas direcciones electrónicas. Para el caso del campo de *"Grupo"* de los artículos, solamente se anexará una lista del total de artículos que también pertenecen al mismo grupo y para el caso de *"Palabras clave"* no se presentará ninguna información adicional.

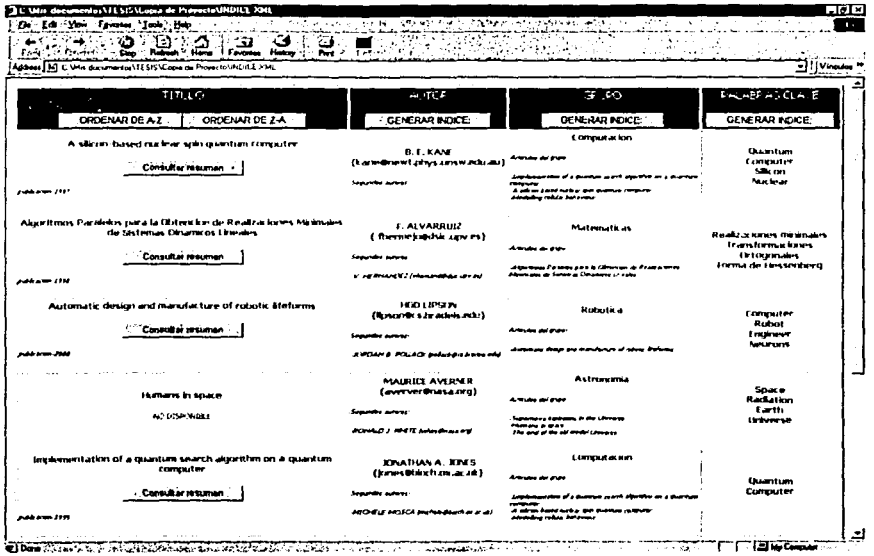

Figura 5.7: "*Pantalla principal del sistema*"

A partir de la pantalla principal del sistema podemos generar una serie de pantallas secundarias, con información adicional acerca del estado actual del conjunto de documentos, accionando los diferentes botones que en ésta se muestran. Esta pantalla inicial es una de las principales del sistema.

DESCRIPCIÓN DE FUNCIONALIDADES

BOTÓN PARA EL ORDENAMIENTO ASCENDENTE DEL ÍNDICE

Esta funcionalidad nos permite generar un índice de los documentos disponibles, jerarquizado por el orden alfabético del título de los documentos. El índice que se genera como resultado de está funcionalidad se muestra en la figura 5.7. La figura 5.8 muestra el botón que acciona el ordenamiento ascendente del índice. El resultado de está acción será equivalente a generar nuevamente la misma tabla, pero está vez asegurando que el ordenamiento de los documentos es ascendente con respecto al título. Por lo tanto, esta funcionalidad no requiere generar pantallas adicionales, sino que el resultado se desplegará en la misma pantalla.

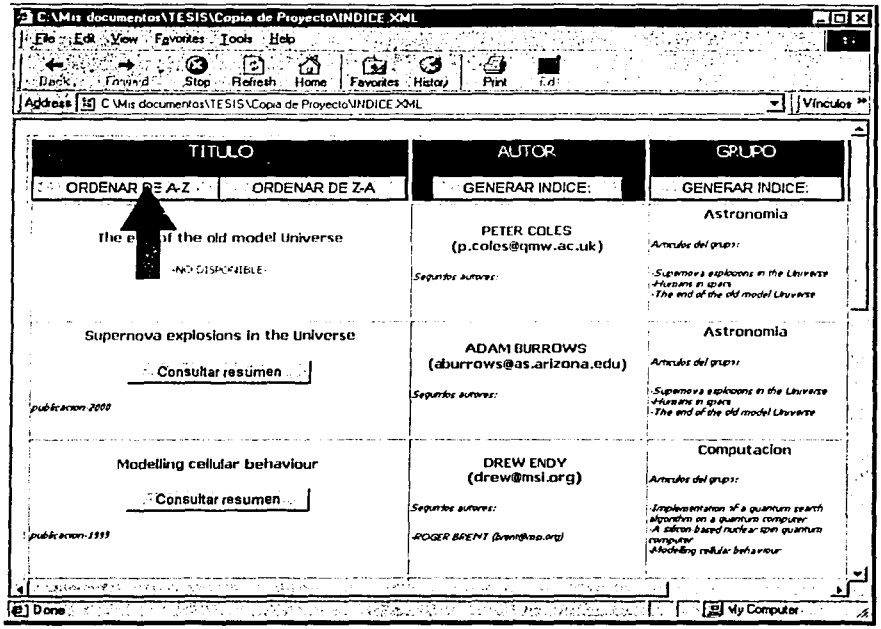

Figura 5.8: "*Generación de índice ascendente*"

También, la funcionalidad para el ordenamiento alfabético del índice de los documentos es invocada implícitamente al ingresar al sistema. Lo anterior significa que el usuario visualizará en la pantalla inicial del sistema un índice de documentos ordenado alfabéticamente por el título.

BOTÓN PARA EL ORDENAMIENTO DESCENDENTE

Esta funcionalidad nos permite generar un índice de los documentos jerarquizado, por el orden alfabético inverso del título de los documentos. De manera muy similar al anterior, sólo que para este caso el orden de los documentos será el inverso. La localización del botón que invoca esta funcionalidad se muestra en la figura 5.9 con una flecha.

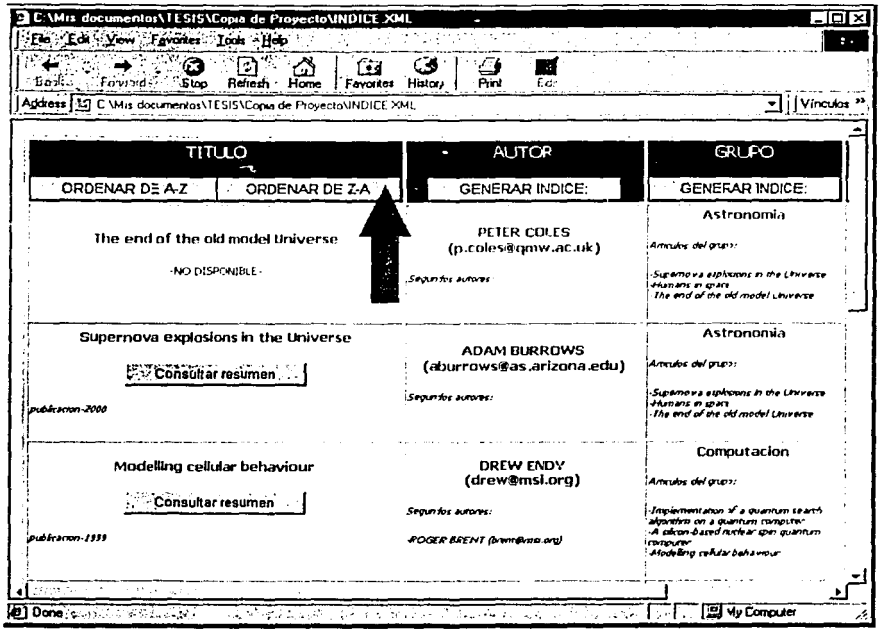

Figura 5.9: "*Generación de índice ascendente*"

La funcionalidad del sistema que nos muestra la figura 5.9 no genera pantallas adicionales y tanto su invocación como la presentación de resultados se muestran en la misma pantalla.

BOTÓN PARA LA GENERACIÓN DE UN ÍNDICE DE AUTORES

Está funcionalidad, a diferencia de las anteriores, genera una segunda tabla con características de formato y contenido de datos muy distintos a los de la pantalla principal. Por esta razón, el resultado de esta funcionalidad será presentado en una segunda pantalla, a la que llamaremos pantalla *Autor-Artículos*. Para este caso, la tabla sólo contendrá una columna de autores ordenada alfabéticamente y con los respectivos artículos de cada uno de ellos. En las figuras 5.10 y 5.11 respectivamente, se muestran ambas pantallas, la primera, desde donde será invocada la funcionalidad que genera el índice de autores y artículos, dentro de la

pantalla *Principal*, y la segunda, donde se despliega el resultado de invocar dicha funcionalidad.

En la figura 5.11 podemos observar el resultado de desplegar el índice de autores. Como se puede notar, para el caso de artículos disponibles, el índice contendrá un botón que nos permitirá hacer la consulta de dicho artículo de manera inmediata, y para el caso de artículos no disponibles, tendremos una leyenda en rojo indicando que el artículo no se encuentra disponible. Como una funcionalidad adicional, esta pantalla muestra un botón, en la parte superior izquierda, con la leyenda "*Regresar*". Donde dicha funcionalidad nos permite regresar a la pantalla *Principal*.

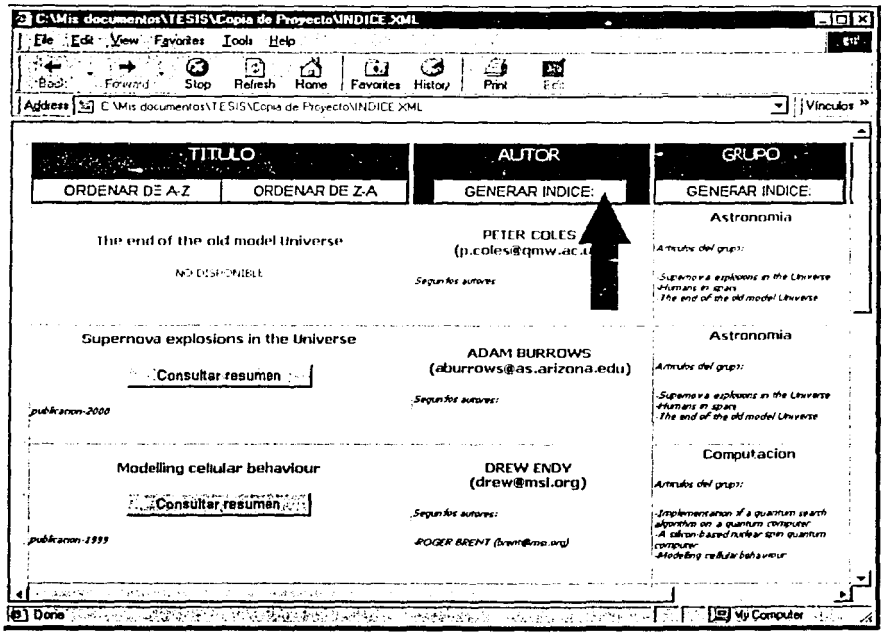

Figura 5.10: "*Generación de indice de autores*"

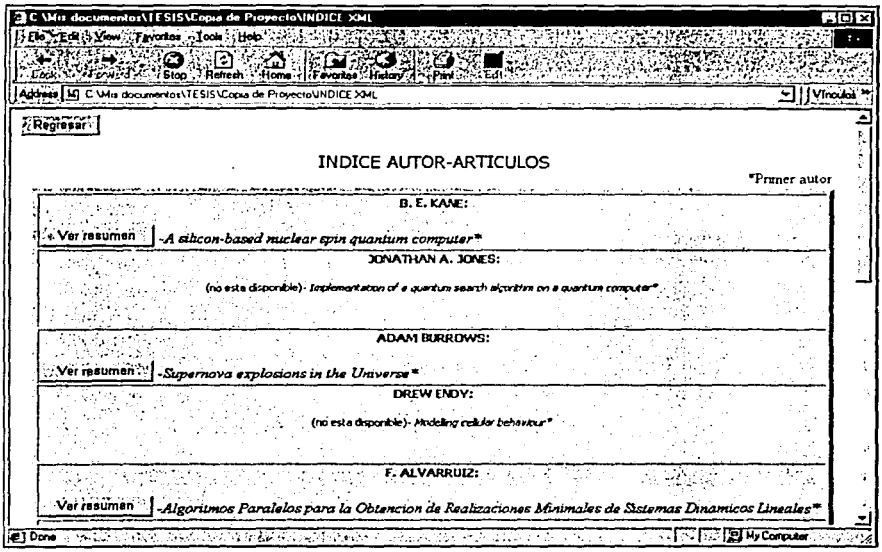

Figura 5.11: "*Pantalla de índice de autores*"

BOTÓN PARA LA GENERACIÓN DE UN ÍNDICE DE GRUPOS

Otra de las funcionalidades importantes que el usuario puede utilizar es la generación de un índice que permita agrupar el total de los artículos disponibles de acuerdo al grupo al que pertenecen. De la misma manera que el índice anterior, esta funcionalidad será invocada desde la pantalla *Principal* y el resultado será desplegado en una segunda pantalla, llamada *Grupo-Artículos*, con características de estilo muy similares a la pantalla *Autor-Artículos*.

La figura 5.12 señala el botón que da inicio a la acción de generar el índice de grupos, mientras que la figura 5.13 muestra los resultados de llevar acabo dicha acción. Aunque el índice es distinto en contenido, como muestra la figura 5.13, las funcionalidades y propiedades de estilo son muy parecidas a las vistas en la figura 5.11.

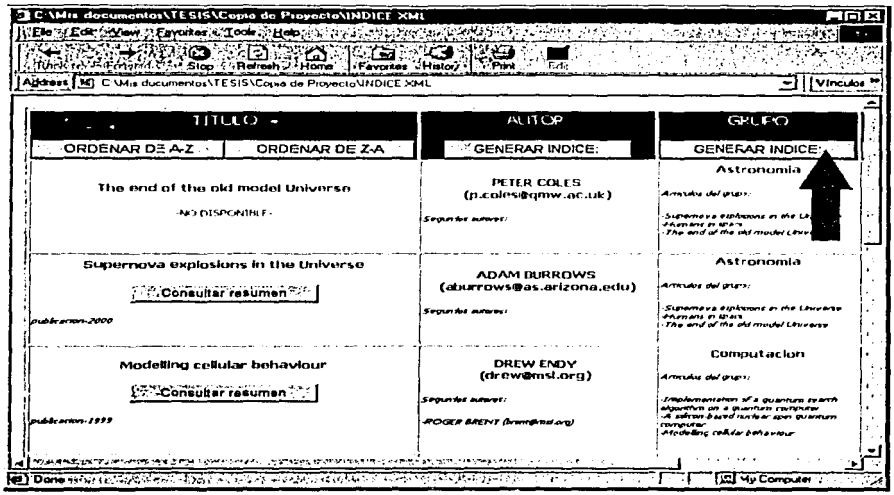

Figura 5.12: "*Generación de índice de grupos*"

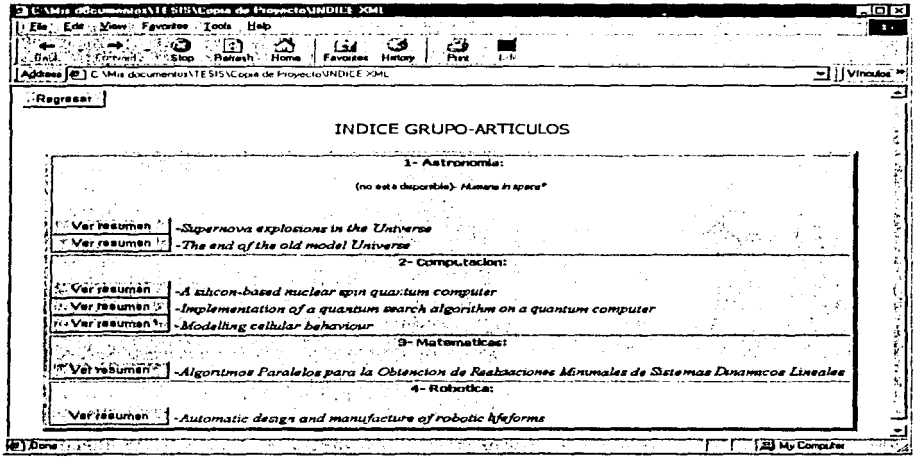

Figura 5.13: "*Pantalla de índice de grupos*"

BOTÓN PARA LA GENERACIÓN DE UN ÍNDICE DE PALABRAS CLAVES

La pantalla que contiene los resultados (figura 5.15) invocados por la funcionalidad que se muestra en la figura 5.14, es muy parecida a las pantallas vistas en las figuras 5.11 y 5.13. La única diferencia visible está, nuevamente, en el contenido. Para este caso el contenido del índice será una lista de las palabras claves contenidas en el sistema junto con el título del artículo al cual pertenecen.

La figura 5.15, entonces, muestra el resultado de generar un índice de palabras claves, y al igual que en las figuras 5.11 y 5.13, las funcionalidades que permiten consultar el resumen presionando el botón *"Ver resumen"*, o regresar al índice inicial presionando el botón *"Regresar"*, están presentes.

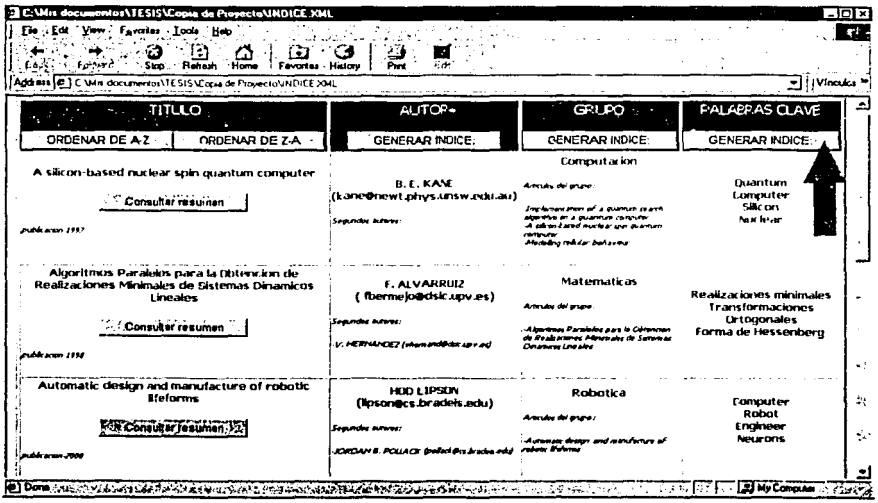

Figura 5.14: "*Generación de indice de palabras clave*"

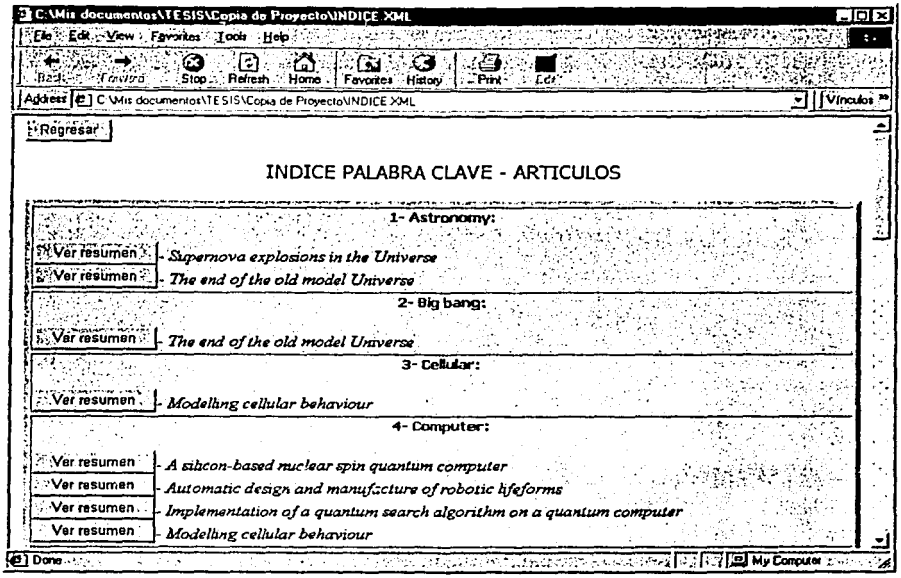

Figura 5.15: "*Pantalla de indice de palabras clave*"

5.5.2 MÓDULO PARA LA VISUALIZACIÓN DE LOS DOCUMENTOS

Este módulo, como hemos visto anteriormente, estará encargado de proporcionar a los artículos el estilo adecuado para su fase de publicación. El usuario realizará la consulta de artículos, que a pesar de tener un formato XML, presentan el mismo estilo que los publicados en formatos como PDF o HTML, formatos especializados para esta tarea.

La estructura del módulo está formada, primero, por el conjunto de documentos XML previamente generados por ELXI, y segundo, por un total de tres hojas de estilo que contienen el código fuente con la información sobre cómo deberá realizarse la asignación de estilo.

Antes de comenzar a describir las funcionalidades de este módulo, partimos de que se ha seleccionado la consulta de un artículo en particular. Recordemos que las únicas dos maneras de acceder a uno de los artículos generados es, primero, accediendo directamente a éste por medio de su dirección o ubicación, y segundo, habiendo accedido previamente al módulo para la organización de los artículos y

habiendo seleccionado éste a través de cualquiera de los índices que hay se presentan.

Así, debido a que todos los artículos tendrán un estilo asignado, previo a su consulta, se desplegará únicamente el resumen, independientemente de la vía que se haya seleccionado para acceder a éste. De acuerdo con esto, en la pantalla observaríamos algo muy parecido a lo que se muestra en la figura 5.16.

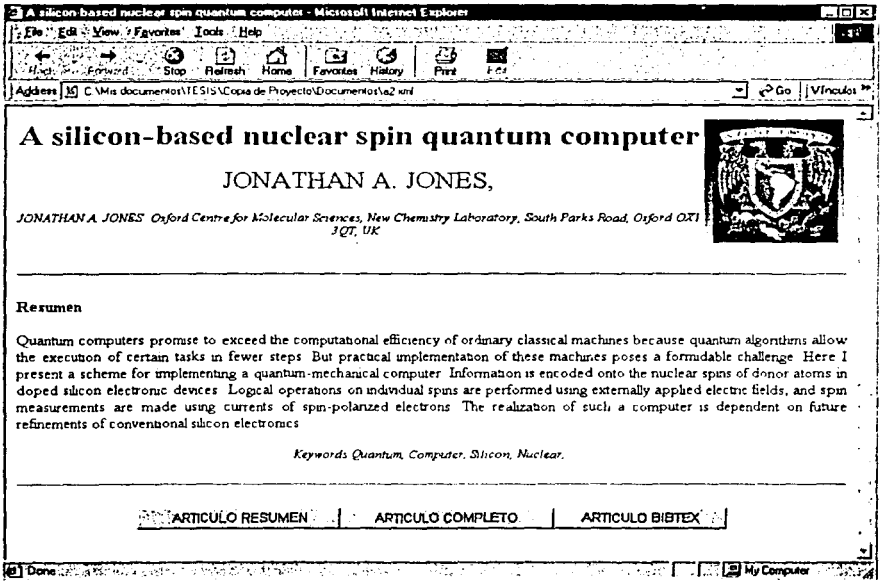

Figura 5.16: "*Resumen de artículo*"

Posterior a la consulta del resumen, el usuario deberá elegir entre alguna de las funcionalidades que se presentan para su elección, presionando alguno de los tres botones que se observan en la parte inferior de la figura 5.16.

La pantalla que muestra la figura 5.16, en caso de venir la selección del artículo desde un índice, se mostrará en una ventana nueva. Por tal razón, si el usuario después de leer el resumen desea consultar otro artículo, bastará con cerrar esta segunda ventana y seleccionar el artículo desde la pantalla que contiene el índice.

Algo importante de mencionar, es que el estilo asignado a la consulta de cualquier artículo será exactamente el mismo. Dicho de otra manera, cualquier artículo presentará título, autores, palabras claves, y todos los demás elementos con los mismos tamaños, colores y propiedades de estilo. Lo único que será distinto es el contenido de texto de dichos elementos, que dependerá obviamente del artículo que se esté consultando.

DESCRIPCIÓN DE FUNCIONALIDADES

BOTÓN PARA LA GENERACIÓN DEL RESUMEN DEL ARTÍCULO

Esta funcionalidad, como puede intuirse, nos permite mostrar el resumen del artículo que estamos consultando. En la figura 5.17, se muestra el resultado de llevar a cabo dicha acción, así mismo, se índica la ubicación del botón que la puede iniciar en cualquier momento.

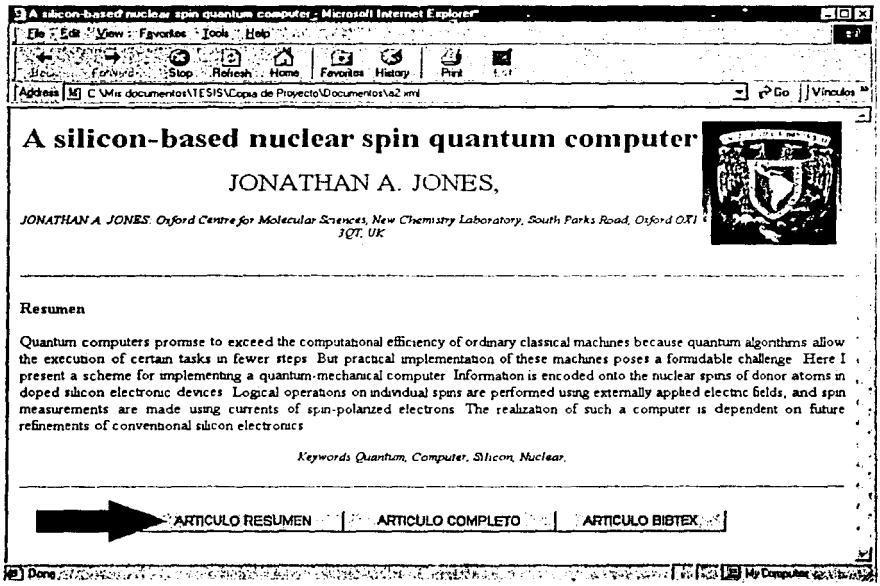

Figura 5.17: "*Generación del resumen de un articulo*"

Dentro de la publicación de artículos de investigación, la introducción a estos por medio de un resumen[54], es una alternativa muy usada, de ahí la importancia de presentar una funcionalidad que permita desplegar, de forma explícita, dicho resumen.

BOTÓN PARA LA GENERACIÓN DEL CONTENIDO COMPLETO DEL ARTÍCULO

Para desplegar un artículo completo en pantalla, el usuario sólo presionará el botón con la leyenda "*Artículo completo*". Dicho botón, se localiza en la parte inferior de la pantalla que contiene el resumen del artículo como se muestra en la figura 5.18.

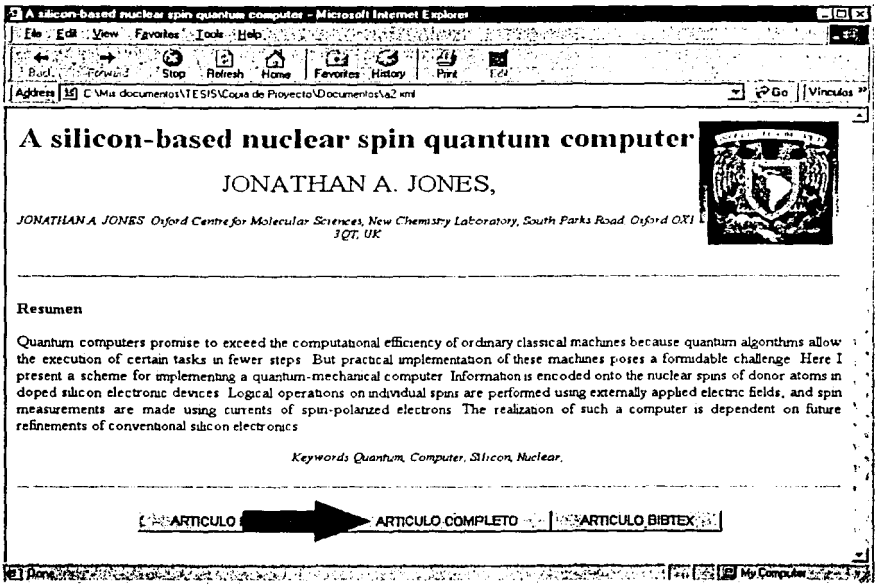

Figura 5.18: "*Generación del contenido completo de un artículo*"

[54] Usualmente se utiliza el término en inglés "*Abstract*".

Usualmente, después de leer el resumen de un artículo, si éste fue de su interés, el usuario solicitará la consulta del contenido completo del artículo. Es por ello que se ha agregado a este módulo dicha funcionalidad.

BOTÓN PARA LA GENERACIÓN DEL BIBTEX DEL ARTÍCULO

Este módulo presenta una funcionalidad que podemos llamar complementaria, la generación del Bibtex[55]. Si el usuario está interesado en generar la bibliografía del artículo que está consultando, en formato Bibtex, deberá presionar el botón que se indica en la figura 5.19.

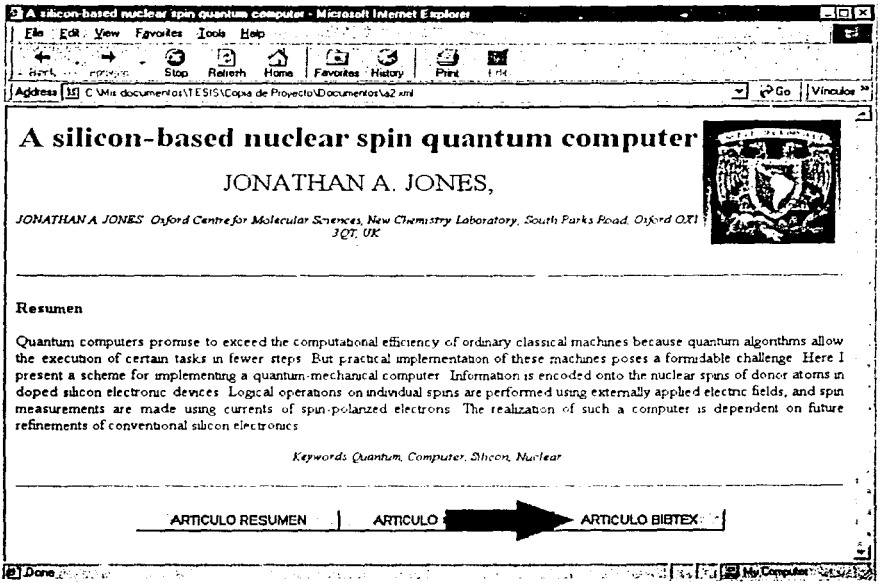

Figura 5.19: "*Generación del bibtex de un artículo*"

Como resultado de tal evento, en la pantalla observaremos algo parecido a lo que se muestra en la figura 5.20.

[55] BibTEX es un lenguaje de programación para manejar bases de datos de texto. Su principal aplicación es facilitar la entrada de referencias bibliográficas en archivos LATEX

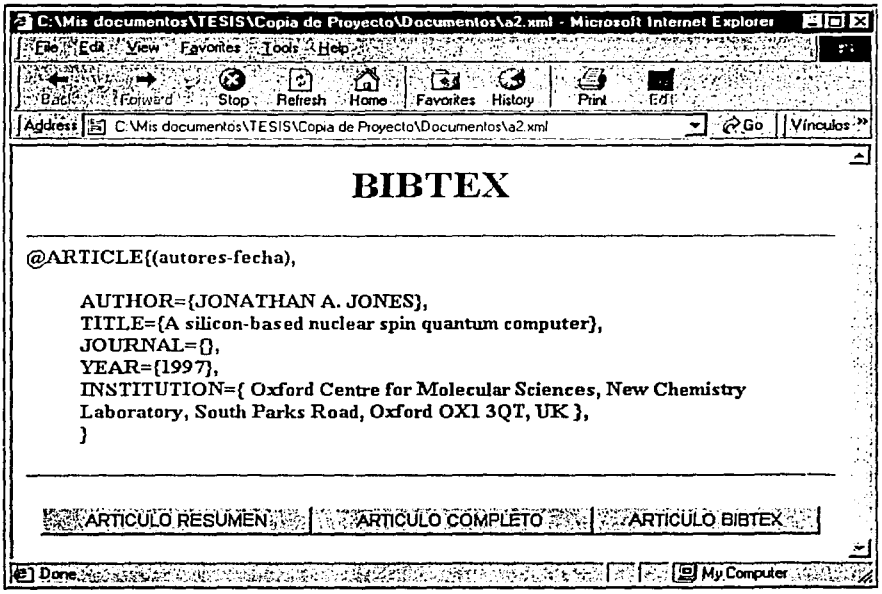

Figura 5.20: "*Generación del bibtex de un articulo*"

Una vez generado el Bibtex del artículo, el usuario podrá regresar la pantalla al contenido completo del artículo, presionando el botón "*Artículo completo*", o incluso, nuevamente al resumen del mismo, presionando el botón "*Artículo resumen*". Dichos botones se muestran en la parte inferior de la figura 5.20.

5.5.3 EVALUACIÓN DE RESULTADOS DEL SISTEMA

A lo largo del desarrollo de este trabajo se realizó el análisis, diseño e implementación de un sistema de visualización y consulta, basado en XSLT y XML, para la publicación de documentos tipo artículo de investigación. Dichos artículos, como se ha mencionado anteriormente, son generados por ELXI y, posteriormente, almacenados como documentos XML en el sistema de archivos del mismo servidor donde ELXI está residiendo.

En la actualidad, la existencia de un sistema de características similares a éste, que proporcione cualquier tipo de servicio implementando las tecnologías XML XSLT, es prácticamente nula. Razón por la cual resulta un tanto subjetivo evaluar los resultados del sistema desde un punto de vista comparativo. Sin embargo, creo que desde el punto de vista tecnológico, la utilización de XML y XSLT, tiene mucho que aportar a la evaluación de resultados del sistema:

A continuación, se listan algunos de los puntos más destacados de dicha evaluación:

- **Limitaciones de desarrollo.** La elección del lenguaje de desarrollo (XSLT) trajo consigo diversas consecuencias. Es importante decir que no existe hoy día ninguna otra alternativa para lenguaje de programación debido a las propias características de ELXI, que genera exclusivamente documentos en formato XML. Debido a que XSLT es un lenguaje de programación relativamente nuevo, y si bien, es cierto que su especificación se encuentra muy avanzada, no es así con las aplicaciones que soportan dicha especificación. Las cuales todavía son pocas y en la mayor parte de los casos con un número de funcionalidades muy limitado.
- **Portabilidad**. Debido a la necesidad de utilizar un navegador que permitiera visualizar el trabajo realizado con los documentos XML, que son transformados dinámicamente a HTML, se eligió Internet Explorer. Internet Explorer es actualmente el navegador con el mayor soporte para trabajar con la especificación XSLT. Sin embargo, el costo de utilizarlo es alto debido a que el sistema se vuelve dependiente, para su buen funcionamiento, del uso de Internet Explorer. Dependencia que además limita al sistema de funcionar bajo Unix o Linux, debido a que son sistemas operativos que no utilizan Internet Explorer. Una alternativa para estos sistemas operativos podría darse con la incorporación futura del soporte XSLT al navegador Mozilla.
- **Modularidad**. La división del sistema en dos módulos distintos permite la anexión o sustitución posterior de funcionalidades, o incluso módulos completos, en la arquitectura. Por ejemplo, la adición de una base de datos relacional que nos permita reestructurar el módulo para la organización de documentos, sin tener que modificar el módulo para la visualización de los mismos.
- **Escalabilidad**. Gracias a que casi todos los documentos y programas que integran el sistema pertenecen a la especificación XML, y algunas especificaciones derivadas de ésta, como XSLT o XPATH, las funcionalidades del sistema podrán ser extendidas a medida que se añada más soporte de estas especificaciones a los navegadores. Incluso, los códigos actuales del sistema podrían llevarse fácilmente a cualquier otro navegador que tuviera el soporte adecuado, debido a que fueron realizados con base en especificaciones oficiales.

5.6 DISCUSIÓN

De los puntos anteriores, podemos resumir que la evaluación del sistema gira fundamentalmente sobre un eje: XML y su entorno de desarrollo. El utilizar XML como el sustento de todo el desarrollo implicó ciertas limitaciones, sobre todo en algunas funcionalidades del sistema relacionadas con la consulta de documentos. La causa fundamental es que, hoy día, las herramientas que implementan funcionalidades de búsqueda en documentos XML son muy limitadas e incluso poco eficientes.

El problema fundamental no es que XML no sea capaz de realizar estos trabajos, sino que las herramientas que complementan las funcionalidades para realizarlo todavía no están "disponibles". El utilizar sólo unas cuantas herramientas de todo un conjunto, donde además muchas de éstas todavía no están disponibles, o al menos no en versiones finales o estables; representa una merma en el desarrollo de cualquier sistema.

Sin embargo, en términos generales, el potencial del limitado XML utilizado en el desarrollo del sistema, es suficiente para mostrar lo que esta sorprendente tecnología es capaz de hacer hoy mismo, cuando parece apenas empezar a mostrarse como una parte fundamental en el futuro de las aplicaciones Web.

Una muestra de esto son las transformaciones XSLT del sistema, utilizadas para proporcionar dinámicamente estilo a los artículos y que son funcionalidades controladas exclusivamente por datos, lo que las hace muy poderosas y versátiles. Gracias a esto, cualquier persona que genere un artículo de investigación en formato XML, similar al tipo de documento generado por ELXI, puede olvidarse del estilo que requiere para su publicación, debido a que ya existe una aplicación que realizara dicha tarea; y no otra persona o él mismo como podría suponerse.

Las ventajas de utilizar XML para un desarrollo de estas características no son pocas, como pudimos ver anteriormente, sin embargo, el costo en desventajas también puede llegar a ser considerable. Principalmente por tratarse de tecnologías que aún se encuentran en una primera etapa de desarrollo.

CAPÍTULO 6

CONCLUSIONES Y PERSPECTIVAS

6.1 CONTRIBUCIONES

La presente tesis ha cumplido con el objetivo de: "*generar un sistema de consulta y visualización, basado en XSLT, para los diversos documentos XML generados por ELXI*". Sin embargo, más allá del desarrollo del sistema[56], el cual representa la principal aportación de este trabajo, las contribuciones adicionales, como trabajo de investigación teórico y práctico, aportadas al proyecto: "*Ambiente Multipropósito de Edición Colaborativa en Internet e Intranet*", del cuál esta tesis forma parte, resultaron muy importantes.

- **El desarrollo teórico** comprende el estudio de dos de las tecnologías Web más recientes: XML y XSLT. Dicho estudio consiste, inicialmente, en mostrar un panorama general de lo que son cada una de estas tecnologías y, posteriormente, en analizar la convergencia de éstas hacia la solución del problema de la publicación de documentos.
- **El desarrollo práctico** aporta al proyecto toda la experiencia de trabajar con XSLT y XML, no sólo como una solución "viable" al problema de la publicación de documentos, sino como la nueva propuesta tecnológica que surge con la publicación de ambas tecnologías como especificaciones.

En términos generales, con este trabajo de investigación se constituye una base que pretende ser un referente importante en la evaluación de futuras decisiones acerca de la utilización de XML y XSLT.

6.2 LIMITACIONES

Como se comenta en la evaluación de resultados, las limitaciones que presenta el sistema de visualización y consulta no son pocas. En cuestiones de portabilidad, la fuerte dependencia a ciertas herramientas, como Internet Explorer, puede resultar sumamente nociva. Otra limitante importante, es la necesidad de actualizar el soporte que para XSLT proporciona Microsoft, actualización que, a mi parecer, puede resultar compleja para los usuarios.

[56] Las conclusiones arrojadas por la evaluación de resultados del sistema se presentaron en el capítulo anterior en la sección de *Discusión*.

En términos generales, las limitaciones más importantes del sistema son causa de un mismo factor: XML y su entorno de desarrollo. El utilizar tecnologías tan novedosas como la base de todo el desarrollo, implicó limitaciones importantes, sobre todo a corto plazo. Sin embargo, se espera que dichas limitaciones sean disueltas con la evolución y madurez de estas tecnologías.

6.3 PERSPECTIVAS

Hoy día, la vertiginosa evolución que puede presentar una herramienta en fase de desarrollo, sobre todo en materia de cómputo, es un factor a considerar en la utilización de ésta. La probabilidad de que una herramienta pueda ser sustituida por otra mejor, está latente en cualquier fase de su vida útil. Si hablamos del caso concreto de las herramientas que giran entorno a una tecnología tan novedosa como es el caso de XML, es obvio pensar que éstas presentan una evolución constante y, en consecuencia, la probabilidad de ser sustituidas por herramientas mejores estará siempre presente.

Por ejemplo, en el caso de Microsoft, el cual nos ofrecía un soporte "completo" para la utilización de XSLT a cambio de una dependencia absoluta hacia el uso de Internet Explorer, parecía ser, hace tan sólo unos meses, la solución más viable a la problemática resuelta por el presente trabajo, sin embargo, a la publicación del mismo, parece existir una nueva propuesta que ofrece notables mejorías.

El proyecto Apache, que es una comunidad de personas que crearon un servidor http[57] (o servidor Web), ha realizado un entorno de publicación y transformación[58] de documentos XML llamado Cocoon[59]. Cocoon funciona como un servlet[60], lo que implica, que en principio, se podrá ejecutar desde cualquier servidor que pueda contenerlo.

Cocoon sólo implementa un soporte parcial debido a que las versiones publicadas, hasta la fecha, se han hecho con un carácter demostrativo. Sin embargo, la versión final, todavía beta, está pensada para constituir una verdadera solución al problema de la publicación y transformación de documentos XML como veremos a continuación.

La versión final de Cocoon, permitirá a un servidor http realizar algunos servicios basados en XML como:

[57] De nombre *Apache*.
[58] Del término en inglés "*Publishing Framework*".
[59] Más información en: *http://xml.apache.org/cocoon/index.html*.
[60] Programa Java que se ejecuta del lado del servidor y cuyo resultado es usualmente enviado a un cliente.

- El envío de páginas XML a navegadores que lo soporten, por ejemplo, las últimas versiones de los navegadores Netscape, Mozilla o el mismo Internet Explorer.
- Las transformaciones XSLT a documentos XML. Estas transformaciones las aplica dinámicamente el servidor, resultando un documento HTML, XML, PDF, etc., que se envía posteriormente al cliente.
- Las XSP[61] (*Páginas de Servidor XML*), tecnología muy novedosa, equivalente a las ASP[62] de Microsoft o a las JSP[63] de SUN.

Como sugieren los dos primeros puntos, las limitaciones del sistema desarrollado podrían solventarse en su totalidad con la utilización de Cocoon. Por lo tanto, además de las aportaciones antes mencionadas, el presente trabajo pretende conformar la base para la creación de un nuevo proyecto. Dicho proyecto consistirá en implementar un entorno de publicación y transformación de documentos XML, dentro de un servidor http, basándose en el sistema de visualización y consulta, desarrollado en el presente trabajo, y en el entorno Cocoon de Apache.

[61] Del término en inglés "*XML Server Pages*".
[62] Del término en inglés "*Active Server Pages*".
[63] Del término en inglés "*Java Server Pages*".

BIBLIOGRAFÍA

[Booch et al., 1999]
Grady Booch et al.,
"The Unified Software Development Process", 1999.

[Bos 1999]
Bert Bos, W3C,
"XML in 10 points",
http://www.w3.org/XML/1999/XML-in-10-points

[Bosak et al., 2000]
Jon Bosak et al., Sun Microsystems,
"Extensible Markup Language (XML) 1.0 (Second Edition)",
W3C Recommendation 6 October 2000,
http://www.w3.org/TR/2000/REC-xml-20001006

[Chamberlin et al., 2002]
Don Chamberlin et al., IBM Research Center,
"XQuery 1.0: An XML Query Language",
W3C Working Draft 30 April 2002,
http://www.w3.org/TR/xquery/

[Clark et al., 1999]
James Clark et al., W3C,
"XML Path Language (XPath) Version 1.0",
W3C Recommendation 16 November 1999,
http://www.w3.org/TR/xpath

[Crane 2000]
Crane Softwrights,
"Practical Transformation Using XSLT and Xpath",
http://www.cranesoftwrights.com/training/

[Kay 1999]
Michael Kay,
"XSLT Programmer's Reference",
http://www.softwareag.com/xml/Techn_Links/xslt_book/contents.htm

[Kay et al., 2000]
Michael Kay et al., W3C,
"XSL Transformations (XSLT) Version 1.0",
W3C Recommendation 16 November 1999,
http://www.w3.org/TR/xslt

[Maler et al., 06/2001]
Eve Maler et al., Sun Microsystems,
"XML Linking Language (XLink) Version 1.0",
W3C Recommendation 27 June 2001,
http://www.w3.org/TR/2001/REC-xlink-20010627/

[Maler et al., 09/2001]
Eve Maler et al., Sun Microsystems,
"XML Pointer Language (XPointer) Version 1.0",
W3C Candidate Recommendation 11 September 2001,
http://www.w3.org/TR/xptr/

[Pemberton et al., 2000]
Steven Pemberton et al., CWI,
"XHTML The Extensible HyperText Markup Language,
A Reformulation of HTML 4 in XML Version 1.0",
W3C Recommendation 26 January 2000,
http://www.w3.org/TR/xhtml1/

[Raggett et al., 1999]
Dave Raggett et al., W3C,
"HTML The Extensible HyperText Markup Language Version 4.01"
W3C Recommendation 24 December 1999,
http://www.w3.org/TR/html4/

[Rusty 2000]
Elliotte Rusty Harold,
"XML Bible",
http://www.ibiblio.org/xml/books/bible/