Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

USO DE XML(LENGUAJE DE MARCADO EXTENSIBLE)
Y XSL(LENGUAJE DE ESTILO EXTENSIBLE) EN LA GENERACIÓN DE NUEVOS PROYECTOS Y NEGOCIOS SOBRE INTERNET CON CONTENIDOS EN TIEMPO REAL.

Tesis para optar al título de Ingeniero Civil en Informática

PROFESOR PATROCINANTE: SR. JORGE GANA LEAY

PROFESOR COPATROCINANTE: SRA. MARIA ELIANA DE LA MAZA

JORGE ANTONIO CANALES SOTO

VALDIVIA - CHILE

2002

FECHA 24/05/2002

DE : **Jorge Gana Leay**

A : SRA. MIGUELINA VEGA R., DIRECTORA ESCUELA INGENIERÍA CIVIL EN INFORMÁTICA

MOTIVO: **Evaluación Tesis.**

INFORME TRABAJO DE TITULACIÓN

Nombre Trabajo de Titulación: .....

"Uso de XML (Lenguaje de Marcado Extensible) y XSL (Lenguaje de Estilo Extensible) en la generación de nuevos proyectos y negocios sobre Internet con contenidos en tiempo real".

Nombre del Alumno: .....
Jorge Antonio Canales Soto. (12.993.690-8)

Nota : 7.0 (en números) Siete (en letras)

FUNDAMENTO DE LA NOTA:

Considerar :
- Cumplimiento del objetivo propuesto
- Satisfacción de alguna necesidad
- Aplicación del método científico
- Interpretación de los datos y obtención de conclusiones
- Originalidad
- Aplicación de criterios de análisis y diseño
- Perspectivas del trabajo
- Coherencia y rigurosidad lógica
- Precisión del lenguaje técnico en la exposición, composición, redacción e ilustración.

Excelente Trabajo. Muy bien estructurado, considera estado del arte, evolución aplicaciones, proyecciones y conclusiones. Muy completo.

Cordialmente,

NORTH SUPPLY CHILE

Valdivia, 18 de junio de 2002

DE : Prof. María Eliana de la Maza W.
Instituto de Informática

A : Sra. Miguelina Vega R.
Directora Escuela de Ingeniería Civil en informática

MOTIVO : Informar revisión y calificación del Proyecto de Título "Uso de XML (Lenguaje de Marcado Extensible) y XSL (Lenguaje de Estilo Extensible) en la generación de nuevos proyectos y negocios sobre Internet con contenidos en tiempo real", presentado por el alumno Jorge Antonio Canales Soto, que refleja lo siguiente:

Se logró el objetivo planteado de estudiar y analizar nuevas tendencias y tecnologías que se puedan aplicar en el desarrollo de futuros negocios para una empresa particular. Además se presentan variadas alternativas de aplicaciones, centrándose en el desarrollo de un Portal Educativo.

Se presenta en la tesis un completo estudio de los lenguajes y documentos XML y XSL, lo que transforma este documento en una excelente herramienta para la consulta y aprendizaje de estos temas.

En el documento se aprecia la aplicación de criterios adecuados de análisis y diseño. Sin embargo faltó una mayor rigurosidad y capacidad de síntesis, al momento de redactar el documento final.

Por lo anteriormente expuesto, y además en mi calidad de co-patrocinante, por conocer el trabajo realizado por el alumno, califico la tesis presentada con nota seis coma cuatro (6,4).

Con este particular, saluda atte. a Ud.,

María Eliana de la Maza W.
Profesora Instituto de Informática

Universidad Austral de Chile

Instituto de Informática

Valdivia, Jueves 11 de julio del 2002.-

DE : Luis A. Alvarez G.. Instituto de Informática.
A : Sra. Miguelina Vega. Directora Escuela Ingeniería Civil en Informática.

Motivo

Informar de Calificación de Trabajo de Titulación.

Nombre Trabajo de Titulación:
Uso de XML (Lenguaje de Marcado Extensible)Y XSL (Lenguaje de Estilo Extensible) en la Generación de Nuevos Proyectos y Negocios sobre Internet con Contenidos en Tiempo Real.

Nombre Alumno: Jorge Antonio Canales.

Nota: 5,0 (cinco coma cero)

FUNDAMENTO DE LA NOTA.:

Los primeros 6 capítulos corresponden a un estudio *y* no una investigación; como se indica en los objetivos; sobre XML. El capítulo 7 se hace un análisis cualitativo de la combinación entre ASP, XML y XSL. En el capítulo 8 se hace un buen estudio de las herramientas XML y XSL, pero falta un análisis comparativo. En el capítulo 9 se muestran alternativas de implementación, sin embargo, el punto 9.6 debió ser un capítulo aparte, por el aporte que significa. En las conclusiones no se menciona el grado de cumplimiento de los objetivos.

El principal aporte del Trabajo de Titulación corresponde a la implementación del portal, sin embargo, analizados los objetivos y las conclusiones no queda claro el grado de participación del alumno. De hecho no está mencionado en los objetivos .

Luis A. Alvarez G.

UNIVERSIDAD AUSTRAL DE CHILE
INSTITUTO DE INFORMATICA
FACULTAD DE CIENCIAS DE LA INGENIERIA

General Lagos 2086 · Campus Miraflores · Valdivia · Chile
Casilla 567 · Fono: 56 63 221427 · Fax: 56 63 293115 · instituto@inf.uach.cl · www.inf. uach.cl

AGRADECIMIENTOS

A ***mis padres***, por todo el apoyo y dedicación diaria que han puesto en mí, para lograr que siempre saliera adelante, pese a todas las dificultades y problemas que se presentaron. Gracias a ellos y a Dios estoy en este lugar cumpliendo mis sueños.

A ***mi hermana*** por su paciencia y por su alegría, que me llevaron y motivaron a continuar.

A ***mi Empresa*** *y a* ***mi Patrocinante*** por darme la posibilidad de seguir creciendo y enfrentar cada día nuevos desafíos.

A ***mi Copatrocinante***, por su responsabilidad y colaboración en el desarrollo de mi tema.

INDICE

SÍNTESIS

Se propone la investigación y el uso de XML (Lenguaje de Marcado Extensible) y XSL (Lenguaje de Estilo Extensible) para la generación de proyectos y perspectivas nuevas de negocios sobre Internet, relacionados al e-business (comercio electrónico), para la empresa North Supply Chile, División e-Business (eBusiness Technology), como una necesidad de la empresa para abrir el campo de nuevas posibilidades de proyectos informáticos, cuyo fin principal es organizar y dar un mejor formato a la información que se maneja en la actualidad.

Esta investigación se complementa con el trabajo realizado en el desarrollo del Portal Educacional para el Ministerio de Educación de Chile (http://www.educarchile.cl), utilizando dichos lenguajes.

XML fue concebido como solución definitiva, para brindar un mejor estructuramiento de la información en Internet. Este nuevo lenguaje dota de estructura a un área compuesta tanto por software de contenido (periódicos en línea y portales web), como por software de desarrollo (sistemas bancarios). Al hacerlo, XML amplía las posibilidades de dicho software, presentando además, como potencial del mismo, la posibilidad de interconectividad entre distintas plataformas y arquitecturas computacionales. Por su parte, XSL corresponde a una tecnología en desarrollo hasta el día de hoy, para dar estilo de presentación a un documento formado con XML.

Este Proyecto de Tesis tiene un carácter de investigación, y además práctico, documento clave para la empresa, que necesita resultados rápidos y mercados nuevos donde aplicar nuevas iniciativas, además de generar nuevos proyectos de aplicaciones en tiempo real.

Se presentan variadas alternativas de aplicaciones e implementaciones, como resultado para la empresa; presentando primeramente un estudio e investigación de dichos lenguajes, para llegar finalmente a las alternativas de solución planteadas, con la mejor implementación.

SYNTHESIS

The propose of this Memory is the investigation and the use of XML (Marked Language of Tensile), XSL (Language of Tensile Style) and the generation of projects and new perspective of Internet businesses, related to e-business (electronic commerce), for the company North Supply Chile, e-Business Division, like a necessity of the company to open the field of new possibilities of computer science projects, whose main aim is to organize and to give a better format to the information that is handled at the present time.

This investigation will complement with the work made in the development of the Educational Web Site for the Chilean Ministry of Education, using these languages. XML was conceived like definitive solution, to offer a better estructure of the information in Internet. This new language equips as much from structure to a compound area by content software (periodic in line and Web sites), like by development software (banking systems). When doing it, XML extends the possibilities of this software, displaying in addition, like potential of the same one, the computacionales possibility of interconnectivity between different platforms and architectures. On the other hand, XSL corresponds to a developing technology until today, to give style of presentation to a document formed with XML.

The Memory has an investigation character, and in addition practitioner, key document for the company, that needs fast results and new markets where to apply new initiatives, besides to generate new projects. It is there where this Thesis work will offer to a new perspective of businesses and application of the tools to study.

1. INTRODUCCIÓN

1.1 INFORMACIÓN GENERAL DEL PROBLEMA

En la actualidad, una empresa del rubro tecnológico, tiene como principal misión introducir soluciones nuevas, que generen expectativas de negocios dentro de un mercado común. El campo de la informática requiere resultados innovadores y rápidos para el caso de análisis gerencial, donde se proyectan nuevos acuerdos entre instituciones y empresas. Si se considera el ritmo actual de dichas aplicaciones, son de interés las que manejen el concepto de tiempo real, entendiéndose por esto, el uso de aplicaciones en un espacio de tiempo, para toda forma de transacción de la información, como por ejemplo, las aplicaciones web on-line (en línea): periódicos, portales, sistemas de información bancarios, etc. [Gua2000].

El gran problema que resulta es poder encontrar las herramientas informáticas precisas, que permitan implementar aplicaciones de alta conveniencia, para lo cual se plantea como una excelente solución para la empresa North Supply Chile, División e-Business el uso de XML en combinación con XSL.

Considerando el caso de integración de servicios en la Internet, con proyectos tales como los emprendidos por la empresa Microsoft Corporation [URL 2], se puede entender el potencial de XML, para la integración de arquitecturas computacionales distintas en el planeta. Esto lleva a pensar en la situación que se presenta en Chile, y en particular en las empresas del rubro de tecnologías de la información.

Se piensa que toda aplicación computacional en la actualidad está acotada por el tiempo, el cual es un factor importante en el manejo de los negocios. Mientras mayor sea la rapidez para brindar un servicio con una propuesta vanguardista, confiable e innovadora, la empresa llevará un paso al frente, en relación a sus pares, si combina sus tecnologías actuales y las adapta a las nuevas tendencias [Gua2000].

El tema contribuye al conocimiento actual de la gerencia de una empresa, al considerar las nuevas incorporaciones tecnológicas y vanguardistas que podrían resultar en

innovaciones para los negocios y apertura de nuevos campos de mercado ya sean nacionales e internacionales, modernizando no solo el concepto de logística de un proyecto: hardware nuevo, softwares potentes, si no más bien el campo y gama de aplicación de negocios.

Se pretende en este proyecto de Tesis, poder investigar los nuevos campos de aplicación de estas dos tecnologías XML, XSL y presentar además la integración de estas, junto a nuevas arquitecturas, que mediante pilotos básicos y prototipos podrán ejemplificar el nuevo espacio de campos de aplicaciones para la empresa.

1.2 ANTECEDENTES EXISTENTES

Actualmente, en Chile no existe referencia a un documento de estudio realizado sobre XML y XSL a profundidad, que permita o sugiera abrir nuevas perspectivas de negocios sobre Internet dentro de una empresa particular.

Se trabaja actualmente en el proyecto del Portal del Ministerio de Educación de Chile, Fundación Chile y Microsoft Chile: Educar Chile, segunda fase, con la herramienta propietaria de North Supply Chile *PubliXpress*®, aplicando los conceptos de XML y XSL, para el desarrollo de los contenidos en tiempo real, que será presentado en parte como el trabajo personal realizado.

1.3 IMPORTANCIA Y NATURALEZA DEL ESTUDIO

La importancia y naturaleza de este estudio se fundamenta en los siguientes puntos:

- Proporcionar una apertura de aplicaciones novedosas e innovadoras para el mercado.
- Introducir aplicaciones novedosas y de investigación para North Supply Chile, División e-Business.
- Introducir aplicaciones innovadoras y atractivas para el comercio electrónico y aplicaciones sobre Internet.

- Introducción de aplicaciones para el sector público y privado.
- Generar nuevos negocios en la empresa y expandirlos.
- Presentar las alternativas existentes para que North Supply Chile lleve la vanguardia en materia de negocios sobre Internet.
- Establecer arquitecturas viables y factibles a mínimo riesgo, para el desarrollo de aplicaciones nuevas, que permitan a la empresa sacar provecho de estas y así poder enfocarlas en un vasto sector que permita introducirse en el mercado con nuevas soluciones, aplicando todo el poder que presenta XML en combinación con XSL.
- Aprovechar la herramienta propietaria PubliXpress® para el desarrollo de nuevos negocios según las variantes ofrecidas en el marco de trabajo con XML y XSL para contenidos que necesitan respuestas y contenidos en tiempo real.
- Comandar nuevos proyectos de desarrollo informático para la empresa.
- Renovar las alternativas existentes en el plazo considerado para este trabajo de Tesis.

1.4 OBJETIVOS A LOGRAR

Los objetivos a lograr en esta Investigación de Tesis son los siguientes:

Objetivo general:

- Investigar los nuevos campos de aplicación de las tecnologías XML y XSL, para generar nuevos proyectos en Internet y estructurar de una mejor forma la información de la empresa; propósitos que en la actualidad son de particular interés para la empresa North Supply Chile, División e-Business.

Objetivos específicos:

- Investigar XML y XSL, realizando un estudio de los lenguajes, que incluya una definición de los mismos e integrarlos dentro de distintas plataformas.

- Evaluar y seleccionar las herramientas de software que puedan existir en el mercado para el desarrollo de XML y XSL, presentando diversas alternativas para la empresa, que incluyan aplicaciones de uso, tecnología actual y perspectivas.
- Dentro de cada alternativa nueva de aplicación de XML y XSL presentar módulos prácticos para la empresa, que faciliten el entendimiento y el verdadero campo de aplicación de los mismos.

1.5 CONCLUSIONES

Se pretende demostrar en esta Tesis que XML es una solución definitiva a la situación que se generó por varios años con el lenguaje HTML (Lenguaje de Marcado de Hipertexto) para la presentación de la información en las páginas web, que a su vez complementa el despliegue de la información con el Lenguaje de Estilo Extensible XSL; además, de hacer notar que hay una gran cantidad de campos de aplicación de los mismos dentro del mercado, que sugieren un poco más de profundidad en el conocimiento de dichos lenguajes, ya que se puede replantear la estructura de la información, así como de su manejo, pudiendo crear lenguajes de marcado muy estructurados con él y además aplicaciones nuevas e interesantes.

2. XML

2.1 INTRODUCCIÓN

El Lenguaje de marcado extensible (XML) es el lenguaje universal utilizado para los datos de la Web. XML proporciona a los programadores la eficacia para entregar datos estructurados de una amplia variedad de aplicaciones al escritorio con vistas al procesamiento y la presentación locales. XML permite la creación de formatos de datos únicos para aplicaciones específicas; también es un formato idóneo para la transferencia de datos estructurados entre servidores [Flo2000, 8].

En los años sesenta existía la necesidad de estructurar los documentos de forma organizada, con el fin de facilitar el intercambio y manipulación de la información. IBM creó GML (Lenguaje de Marcado Generalizado) para satisfacer las necesidades de los sistemas internos que sus sistemas de edición, así aumentaba la producción de sus libros, informes y documentos internos dentro de la compañía, a partir de un solo conjunto de archivos fuente. En empresas específicas se introdujeron soluciones de estructuración de información, pero no se hacía nada para solucionar el tema a gran escala.

El primer estándar fue SGML (Lenguaje de Marcado Generalizado Estándar), utilizado como tecnología de toda la información estandarizada y estructurada de cierta importancia, que también procedía de IBM, con el fin de dar formato y mantener la documentación legal, el cual, se fue propagando a los sectores informáticos como un estándar de información de propósito general [Flo2000, XI].

En 1986 SGML emergió como un estándar ISO (Organización de Estándares Internacionales, ISO 8879) y tenía como dificultad que se requería de una gran cantidad de software para procesarlo, por lo que SGML no supuso una alternativa al hipertexto en los primeros días de Internet [Cas99, 346].

En 1989, dos investigadores del Laboratorio Europeo de Física de Partículas (CERN), Tim Berners-Lee y Anders Berglund, crearon un lenguaje basado en etiquetas o tags, para marcar documentos técnicos a fin de compartirlos en Internet. Este lenguaje fue

ampliado en una aplicación simplificada de SGML llamada HTML (Lenguaje de Marcado de Hipertexto), que introdujo el primer formato de información estándar de la Web [Flo2000].

HTML tuvo mucho éxito y revolucionó el mercado. Cuando pasó la moda, tanto los particulares como las empresas empezaron a anhelar algo más que imágenes estáticas y texto, surgiendo así una serie de tecnologías tendientes a añadir interactividad a la Web, una de ellas: el conocido lenguaje JAVA [Flo2000].

HTML llevó a cabo una ímproba tarea satisfaciendo necesidades de los desarrolladores y programadores web, considerando sus limitaciones innatas. Sin embargo, los miembros del Consorcio de la World Wide Web (W3C) se dieron cuenta que sería necesaria una alternativa más amplia de lenguaje. Una limitación del HTML, pese a las mejoras que se han realizado, es el conjunto finito de etiquetas. Es imposible agregar etiquetas personalizadas, lo cual significaría una mayor utilidad al tratar con datos pertenecientes a una aplicación o sector específico [Flo2000].

SGML soporta completamente el uso de conjuntos personalizados de etiquetas, característica que se logró visualizar.

En 1996, el W3C (Consorcio Web), se propuso introducir el poder y la flexibilidad de SGML en el dominio de la web, ofreciendo tres ventajas claras e importantes que faltaban en el HTML:

- **Extensibilidad**
- **Estructura**
- **Validación**

Esto dejó en manifiesto que HTML es un lenguaje poco estructurado; así el W3C presentó un equipo de expertos en SGML, "SGML para la Web", con el objetivo de desarrollar una nueva tecnología de marcado con las ventajas de SGML y con la simplicidad del HTML [Mar2000].

En 1998 la W3C lanzó la especificación XML 1.0 con lo que los problemas se abrieron a un sinnúmero de soluciones [Flo2000].

Existen muchas ventajas en el uso de XML tanto en el Web como en aplicaciones de plataforma central [Mar2000]:

- **Proporciona información para el procesamiento local:**

Los datos proporcionados al escritorio están disponibles para el procesamiento local. Los datos se pueden leer con el analizador de XML y, a continuación, se entregan a una aplicación local, como un explorador, para la visualización y procesamiento adicionales. O bien, los datos se pueden manipular mediante secuencias de comandos u otros lenguajes de programación con el Modelo de objetos XML.

- **Proporciona a los usuarios una visión adecuada de los datos estructurados:**

Los datos entregados al escritorio se pueden presentar de muchas maneras. Es posible presentar un conjunto de datos locales en la vista situada a la derecha del usuario, dinámicamente, en función de factores como las preferencias del usuario y la configuración.

- **Habilita la integración de datos estructurados procedentes de varios orígenes en vistas lógicas comunes:**

Normalmente, se utilizarán agentes para integrar datos procedentes de bases de datos de servidor y otras aplicaciones en un servidor de plataforma central, por lo que los datos estarán disponibles para su entrega al escritorio o a otros servidores para su ampliación, procesamiento y distribución.

- **Describe datos de una amplia variedad de aplicaciones:**

Debido a que XML es extensible, se puede utilizar para describir los datos contenidos en una amplia variedad de aplicaciones, desde colecciones de páginas Web hasta registros de datos. Como los datos son autodescriptivos, se pueden recibir y procesar sin necesidad de una descripción integrada de los mismos.

- **Mejora el rendimiento mediante actualizaciones granulares:**

XML permite actualizaciones granulares. Los programadores no tienen que enviar todo el conjunto de datos estructurados cada vez que hay un cambio. Con las actualizaciones granulares, sólo es necesario enviar el elemento modificado del servidor al cliente. Los datos modificados se pueden presentar sin necesidad de actualizar la totalidad de la página o la tabla.

2.2 XML

XML es un subconjunto simplificado de SGML que incorpora muchas de las características de SGML. XML representa una nueva época para la Web, en la medida que representa una forma de transmitir datos estructurados [Flo2000]. En la realidad no es más que un formato de texto estandarizado que sirve para representar información estructurada en Internet y representa menos de una décima parte del tamaño de la especificación SGML. En otras palabras, XML es una especificación que permite diseñar lenguajes de marcado, por lo tanto es un Metalenguaje [Flo2000]. Esto implicaría que HTML que fue escrito como una aplicación específica de SGML, podría rediseñarse como aplicación de XML. De hecho, HTML ya está siendo diseñado como aplicación XML llamada XHTML, abordado en el capítulo 6.

XML es una tecnología general que al parecer sirve para toda aplicación en Internet. Sin embargo, a diferencia de HTML, XML no es una solución por sí mismo. XML define un marco que se puede usar para crear soluciones. La premisa que subyace a XML es la creación de conjuntos personalizados de etiquetas usadas para identificar tipos específicos de información [URL 15], por lo tanto no existe un visor XML genérico en el mismo sentido que un navegador web es un visor HTML. Existen visores XML genéricos, pero sólo permiten ver la estructura del marcado XML y no la visualización de la verdadera significación del contenido XML, para lo cual se debe describir de cierta forma cómo hay que presentarlo y desplegarlo.

XML se considera además como un lenguaje de metamarcado que ofrece un formato para la descripción de datos estructurados. Esto facilita unas declaraciones de contenido más precisas y unos resultados de búsquedas más significativos en diversas plataformas computacionales. Además, XML habilita una nueva generación de aplicaciones para ver y manipular datos basadas en la Web [Mar2000].

XML proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos:

- Documento normal.
- Registro estructurado, como un registro de citas o un pedido de compra.
- Objeto con datos y métodos, como el formulario permanente de un objeto Java o de un control ActiveX.
- Un registro de datos, como el conjunto de resultados de una consulta.
- Metacontenido sobre un sitio Web, como el formato de definición de canal (CDF).
- Representaciones gráficas, como la interface de usuario de una aplicación.
- Entidades y tipos de esquema estándar.
- Todos los vínculos entre datos y personas que hay en la Web.

Cuando los datos llegan al escritorio del cliente, se pueden manipular, editar y presentar en varias vistas, sin tener que regresar al servidor [Mar2000]. Ahora los servidores pueden ser más escalables, gracias a la reducción de las cargas de ancho de banda y computación. Además, dado que los datos se intercambian en el formato XML, se pueden combinar fácilmente desde distintas fuentes.

XML es muy valioso para Internet, así como para los entornos de intranets corporativas de gran tamaño, ya que proporciona interoperabilidad mediante un formato basado en estándares flexibles y abiertos, con formas nuevas de acceso a las bases de datos existentes y de entregar datos a clientes de la Web. Las aplicaciones se pueden generar

más rápidamente, su mantenimiento es más sencillo y pueden ofrecer fácilmente varias vistas de los datos estructurados [Sel2001].

2.3 FUNDAMENTOS DE LA SINTAXIS DE XML

2.3.1 ETIQUETAS

Un archivo (o documento) XML se compone de elementos XML, cada uno de los cuales consta de una etiqueta de inicio (**<título descriptor>**), de una etiqueta de fin (**</título descriptor>**) y de los datos comprendidos entre ambas etiquetas (el contenido). Al igual que los documentos HTML, un documento XML contiene texto acotado por etiquetas. Sin embargo, a diferencia de HTML, XML admite un conjunto ilimitado de etiquetas, no para indicar el aspecto que debe tener algo, sino lo que significa en concreto [Cas99, 345], [Har98, 3].

El autor del documento XML es quien decide qué tipo de datos va a utilizar y qué etiquetas son las más adecuadas. Por ejemplo se puede utilizar XML para describir el reporte del clima en la ciudad de Valdivia. Cabe señalar que los archivos XML se puede guardar con una extensión de XML, por ejemplo Tiempo_Valdivia.xml.

En el siguiente ejemplo (tabla 2-1) se mostrará la sintaxis de un documento XML, bien formado:

Tabla 2-1: Ejemplo de un documento XML.

```
<?xml version='1.0' encoding='ISO-8859-1' standalone="no"?>
<tiempo-ciudad>
  <fecha>30 de Marzo 2002</fecha>
  <hora>06:00</hora>
  <area>
    <ciudad>Valdivia</ciudad>
    <comuna>Valdivia</comuna>
    <region>Décima</region>
    <pais>CHILE</pais>
  </area>
  <medidas>
    <cielo>sol</cielo>
    <temperatura>27º</temperatura>
    <viento>
      <direccion>NORTE</direccion>
      <velocidad>10</velocidad>
    </viento>
    <humedad>22</humedad>
    <visibilidad>9</visibilidad>
  </medidas>
</tiempo-ciudad>
```

En lugar de describir el orden y la disposición de la presentación de los datos, las etiquetas indican qué significa cada elemento de datos, existiendo libertad para la creación. Cualquier receptor de estos datos puede descodificar el documento y utilizarlo para sus propios fines.

Los elementos XML pueden estar vacíos, lo cual significa que contienen datos de caracteres analizados sintácticamente. Las etiquetas vacías de XML pueden especificar etiquetas de inicio y final así como también de final, lo cual sería sintácticamente correcto en XML [Cas99, 347].

Cada documento XML está diseñado para ser "procesado" por procesadores XML.

El **procesador XML** es un módulo de software que lee un documento XML y que proporciona acceso a su contenido y estructura. Los procesadores procesan documentos XML en nombre de las aplicaciones. Una aplicación XML emplea un procesador XML para obtener acceso al contenido (lo que está al interior de las etiquetas) y a la estructura de los documentos. El **analizador XML** es el componente del procesador XML que

analiza el marcado XML y determina la estructura de los datos del documento [Mar2000, 33].

2.3.2 REFERENCIAS DE ENTIDADES

Las entidades son bloques de construcción de documentos XML y tienen como características:

- Son entidades en sí mismos.
- Se pueden formar con otras entidades a través de referencias a otras entidades.

Las referencias en XML se utilizan con el fin de asignar alias a piezas de datos. Una referencia de entidad sirve como nombre único para una pieza de datos XML [Cas99, 350].

La tabla 2-2 muestra algunas de las referencias usadas en XML:

Tabla 2-2: Ejemplos de referencias en XML .

▪ **&amp**	= caracter ampersand (**&**)
▪ **&apos**	= caracter apóstrofo (´)
▪ **&quot**	= caracter comillas(")
▪ **&lt**	= caracter menor que (<)
▪ **&gt**	= caracter mayor que (>)

Las referencias a entidades deben ser declaradas de forma explícita antes de poder ser utilizadas en un documento XML, sin embargo las referencias mostradas en la tabla 2-2 pueden ser invocadas en cualquier parte del documento XML .

2.3.3 COMENTARIOS

Los comentarios en un documento XML se utilizan para presentar información, que en forma técnica no es parte del contenido de un documento. Por lo general se utilizan para proporcionar descripciones de datos de documentos, según estime el usuario.

Los analizadores y aplicaciones XML suelen ignorar los comentarios. Los comentarios se pueden utilizar en cualquier parte de un documento XML en la que aparezcan datos de caracteres analizados sintácticamente.

Los comentarios empiezan con **<!--** y terminan con **-->** **,** la única limitante de los comentarios es que no se puede utilizar los guiones dobles (--) en el interior del comentario, ya que entraría en conflicto con la sintaxis de comentario XML [Cas99, 351]. La tabla 2-3 muestra el uso de comentarios en XML:

Tabla 2-3: Ejemplos de comentarios en XML

```
<!--  Este es un comentario correcto en XML   -->
<!--  Se puede utilizar este comentario en cualquier parte de un documento XML,
ya que la información contenida en estos comentarios no forma parte de los datos del
documento XML
 -->
<!-- Este es un comentario -- incorrecto en XML -->
```

2.3.4 INSTRUCCIONES DE PROCESAMIENTO

La sintaxis de XML incluye además instrucciones de procesamiento que son instrucciones muy especiales para ser utilizadas por la aplicación que está procesando el documento XML. Estas instrucciones indican si el documento XML está bien construido y que la estructura del documento es compatible con su DTD (Definición de Tipo de Documento) [Flo2000, 295], si es que existe.

Las instrucciones de procesamiento empiezan con un signo menor que y con el signo de interrogación (**<?**) y terminan con el signo interrogación y el de mayor que (**?>**).

Un ejemplo de instrucción de procesamiento puede verse en el listado 2-1, correspondiente al inicio de la cabecera del documento XML.

2.3.5 DECLARACION DE TIPO DE DOCUMENTO

Las declaraciones de tipo de documento son utilizadas en XML para especificar información de un documento particular, incluyendo el elemento de la raíz del mismo y de la **Definición de Tipo de Documento** (**DTD**) [Flo2000, 295].

Una declaración de tipo de documento es importante para saber si un documento XML está bien construido y a la vez que sea válido.

La declaración de tipo de documento, por lo tanto se encarga de:

- Especificar el elemento raíz de un documento XML.
- Definir elementos, atributos y entidades (DTD internas).
- Identificar una DTD externa en el documento.

La tabla 2-4 muestra la sintaxis de una DTD:

Tabla 2-4: Ejemplo de una Definición de Tipo de Documento (DTD) en XML.

```
<!ELEMENT librodedirecciones (contacto)+>
<!ELEMENT contacto (nombre, direccion+, ciudad, estado, telefono, email, web, empresa)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT estado (#PCDATA)>
<!ELEMENT telefono (voice, fax?)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT web (#PCDATA)>
<!ELEMENT empresa (#PCDATA)>
```

El elemento raíz del documento para la tabla 2-4 es el elemento **librodedirecciones**, que está especificado en la declaración de tipo de documento. La DTD se puede almacenar como un archivo con la extensión .**dtd**.

La gran ventaja de una DTD es que es reutilizable, es decir se puede utilizar para documentos más complejos, aprovechando así los atributos y las entidades. Si no se necesita una DTD externa reutilizable, es posible mover el contenido de la DTD externa directamente a la declaración de tipo de documento.

La tabla 2-5 muestra un documento XML invocando a la DTD correspondiente de la tabla 2-4:

Tabla 2-5: Ejemplo de un documento XML que invoca a la DTD correspondiente.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE librodedirecciones SYSTEM " librodedirecciones.dtd" [
<!ENTITY amp "&#38;#38;">
<!ENTITY apos "&#39;">
]>
< librodedirecciones >
  <!-- Acá está la dirección de un contacto, podrían existir más contactos en el libro de
       direcciones. -->
  <contacto>
    <nombre>Frank Rizzo Matuz</nombre>
    <direccion>1212 Providencia</direccion>
    <ciudad>Santiago de Chile</ciudad>
    <estado>Chile</estado>
    <telefono><!-- Sin teléfono momentáneamente por reinstalación--></telefono>
    <email>sr_cliente@empresa.cl</email>
    <web>http://www.webempresa.cl</web>
    <empresa>Empresas Frank&apos;s Ratchet </empresa>
  </contacto>
< librodedirecciones >
```

2.3.6 SECCION CDATA

Las secciones de datos de caracteres que no son analizados sintácticamente se denominan secciones CDATA y se emplean en los documentos XML para bloquear texto que tiene que ser puesto a un lado por un analizador XML. Lo que se define en el interior de una sección CDATA no se analiza nunca sintácticamente, normalmente se usa para citar fragmentos de código [Cas99, 354].

Se define una sección CDATA con los inicios de caracteres **<![CDATA[** y **]]>**.

2.4 ESTADO ACTUAL DE XML

Existen una serie de tecnologías emergentes que están asociadas y van de la mano con XML, como es el caso de las hojas de estilo que se analizará en el capítulo 4, que llevan a un gran número de aplicaciones novedosas; además, se está avanzando en la tecnología sobre espacios de nombres XML, que proporciona una forma de asignar nombres únicos a elementos XML. El Modelo de Objetos de Documentos (DOM) XML también representa una tecnología muy importante que proporciona un acceso pragmático a la estructura interna de los documentos XML [URL 10]. La tecnología XML Schema de Microsoft, que proporciona una alternativa muy potente al diseño de clases de documentos XML con las DTD tradicionales [URL 6]. Las tecnologías XLink y XPointer proporcionan una forma de establecer vínculos avanzados entre recursos de XML.

A finales de 2001 y principios de 2002, bCentral, el portal de Microsoft para pequeños negocios, comenzará a exhibir y a integrar servicios Web basados en XML, creando una de las primeras experiencias de amplia cobertura para el usuario de .NET, iniciativa que busca la integración de todos los servicios sobre la Internet, donde las empresas podrán incorporar directamente los servicios Web basados en XML presentados por el portal, así como tener sus procesos de negocios perfectamente bien integrados [URL 7]; además se realizó entre el 22 y 26 de Octubre de 2001 la Conferencia de Desarrolladores Profesionales (PDC) en el Convention Center de los Angeles, California, donde uno de los temas que se trataron fueron los servicios XML para la Web; esta misma conferencia fue abordada además por Microsoft [And2001], [Jac2001].

Como avance de investigación en el campo del software de integración de XML, un miembro de los .NET Enterprise Servers, BizTalk Server ofrece una solución comprensiva para instrumentar procesos de negocios de la empresa al Internet. Usando la suite de BizTalk de herramientas de instrumentación y servicios, se puede habilitar

XML a las aplicaciones actuales, conectándolo rápida y seguramente con otras aplicaciones internas y externas [URL 4].

En el campo de las bases de datos el W3C trabaja en desarrollar el Lenguaje de Consulta Extensible (XQL), para la extracción de información de colecciones de documentos XML por medio de una interface de Modelo de Objeto de Documento e interfaces XQL [Sel2001].

Los desafíos que se plantean a futuro con XML están en relación a la distribución geográfica, por medio de los protocolos como HTML y el Protocolo Simple de Acceso a Objetos (SOAP), además de la búsqueda de estructuras de datos heterogéneos y lenguajes, representaciones de atributos heterogéneos y semántica, Esquemas heterogéneos e identificación de objetos [And2001], [Jac2001].

2.4.1 ESTILO DE UN DOCUMENTO XML

XML es un metalenguaje basado en contenido, por lo que, para poder ver un documento XML se deberá proporcionar información de cómo verlo. Esto se realiza con las hojas de estilos, que están codificadas por medio de CSS o de XSL [Flo2000, 10].

CSS es una tecnología de hojas de estilos que fue creada para ser utilizada con HTML, que a su vez, funciona muy bien con XML. XSL es una tecnología de hojas más avanzada que se creó explícitamente para ser utilizada con XML. XSL es mucho más compleja que CSS, pero más poderosa [Flo2000, 149]. Estas tecnologías se analizarán con más detalle en el capítulo 4.

Aunque las hojas de estilos son la forma preferida de mostrar un documento XML, es posible también escribir código que procese directamente un documento XML y se ofrezca el resultado. Por ejemplo las aplicaciones XML más avanzadas utilizan esta solución para proporcionar un control exacto sobre la muestra del contenido.

2.4.2 EXAMINAR UN DOCUMENTO XML

Hace un par de años, el concepto de examinar documentos XML solo era un concepto como tal. La Web está en camino de soportar una red de documentos y servicios basados en XML [Jac2001], [URL 14]. Microsoft Internet Explorer 5.0 es el primer navegador web comercial que soporta XML 1.0, además de otras tecnologías XML estrechamente relacionadas, como el DOM XML (un subconjunto de XSL), la Recomendación 1.0 de Espacios de nombres XML y una visión preliminar de la tecnología para esquemas XML avanzados [URL 19], [Flo2000, 89]. Además Internet Explorer 5.0 soporta dos vocabularios XML: el Formato de Definición de Canal (CDF) y el Lenguaje de Marcado de Vector (VML).

Microsoft se ha tomado en serio el futuro de XML, aunque no ha cumplido la tarea de adherirse a los estándares XML que establece el Consorcio W3C [Flo2000, 9], [Mar2000, 20].

La empresa Nestcape tiene planes para soportar XML 1.0 en sus navegadores por medio de su iniciativa de origen abierto Mozilla, además está en avances de XUL, un vocabulario XML para describir interfaces de usuario gráficas de plataforma cruzada [URL 39].

El W3C también tiene un navegador llamado Amaya, que se puede usar para examinar documentos XML [URL 19]. Se emplea más como herramienta de creación, ya que soporta la edición de documentos Web, además de navegador.

2.4.3 ANALIZAR SINTACTICAMENTE XML

Según [Mar2000, 31], hay muchos analizadores XML disponibles para analizar sintácticamente el código XML, que proporciona el acceso al análisis de un documento e información de XML. Todos a su vez, funcionan como elementos de validación y de no validación, dependiendo de sí su documentación requiere o no validación. Adicionalmente, casi todos los analizadores XML disponibles están diseñados como

componentes de software que se pueden conectar de manera fácil a una aplicación. Es posible añadir soporte XML a una aplicación, conectándola a un analizador XML.

La tabla 2-1 muestra un listado de los analizadores XML más conocidos y disponibles.

Tabla 2-1: Ejemplo de analizadores XML.

▪ ***XML Lark y Larval para JAVA.***
▪ ***Project X para JAVA de Sun .***
▪ ***XML para JAVA de Datachannel***
▪ ***XML para JAVA de IBM***
▪ ***XML para JAVA de Oracle***
▪ ***XML AElfred para JAVA, de MicroStar***
▪ ***XML para C++ de IBM***
▪ ***XML para el constructor C++***
▪ ***XML parser 4*** **MSXML.DLL** ***de Microsoft.***

2.5 CONCLUSIONES

Para entender más abiertamente el concepto de XML se ha abordado el origen del mismo, además de sus relaciones ante HTML y SGML, entendiendo el concepto de un documento XML y su sintaxis correspondiente, dando una visión técnica necesaria de XML para su comprensión.

XML mantiene separación entre la interface de usuario y los datos estructurados. El lenguaje HTML especifica como visualizar datos en un navegador, XML define el contenido. Por ejemplo, HTML utiliza etiquetas para decirle al navegador que despliegue el contenido en negritas o itálicas; en XML solo se utilizan etiquetas para describir los datos.

XML, utiliza hojas de estilo, tales como el Lenguaje de Estilo Extensible (*Extensible Style Language*, XSL) y las Hojas de Estilo en Cascada (*Cascading Style Sheets*, CSS) para presentar los datos en un navegador. XML separa los datos de la presentación y el proceso, permitiendo desplegar y procesar los datos al aplicar diferentes hojas de estilo y aplicaciones.

Esta separación de datos de la presentación permite integración de datos de fuentes diversas. La información de clientes y cualquier otra información se puede convertir a XML en el nivel medio, permitiendo a los datos ser intercambiados en línea de manera muy conveniente. No es necesario ajustar información en formatos propietarios almacenados en bases de datos o documentos de mainframes, debido a que se usa el protocolo HTTP para transmitir el XML sobre la red, no se necesitan cambios para esta función.

En el Capítulo siguiente se analizará el concepto de Definición de Tipo de Documento y de los Esquemas, elementos claves para la formación de documentos XML correctos y bien formados.

3. DEFINICIÓN DE TIPO DE DOCUMENTO (DTD) Y ESQUEMAS

3.1 INTRODUCCIÓN

Este capítulo presenta el modelamiento de datos con DTD y con Esquemas XML según lo expuesto en [Flo2000], [Mor2000] y complementado con las referencias a los sitios de la Web. Un esquema XML describe un modelo de datos de un documento XML determinado [Mor2000, 20]. Existen dos soluciones fundamentales a la hora de definir los esquemas XML:

- Las Definiciones de Tipo de Documento, DTD.
- Los Esquemas XML

Además hay que distinguir dos tipos de documentos XML, los válidos y bien formados.

Bien formados, son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas, sin estar sujetos a unos elementos fijados en un DTD. De hecho los documentos XML deben tener una estructura jerárquica muy estricta, y los documentos bien formados deben cumplirla.

Los XML **válidos** siguen una estructura y una semántica determinada por un DTD, sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

Se abordará este capítulo distinguiendo los fundamentos del modelado de datos XML, describiendo la estructura de los documentos XML para fines de validación. También se compara las DTD con los Esquemas XML y proporciona información sobre cuál de las dos soluciones es más conveniente. Se abordará como conclusión que ambas soluciones, tanto los DTD como los Esquemas XML son suficientes para la mayoría de los documentos, ofreciendo cada cual sus ventajas respectivas.

3.2 MODELAMIENTO DE DATOS XML

Un documento XML es, en esencia un medio estructurado para almacenar información. Para valorar la validez de un documento en XML, se debe establecer a la estructura que debe adherirse la información que hay en el documento. Por medio de un esquema, modelo usado para describir estructura de la información de un documento XML, se puede modelar una clase de datos. Cuando el modelo de datos está colocado para una clase de datos se pueden crear documentos XML bien estructurados que se adhieran al modelo planteado como se ilustra en la figura 3-1 y basándose en [Flo2000, 295]:

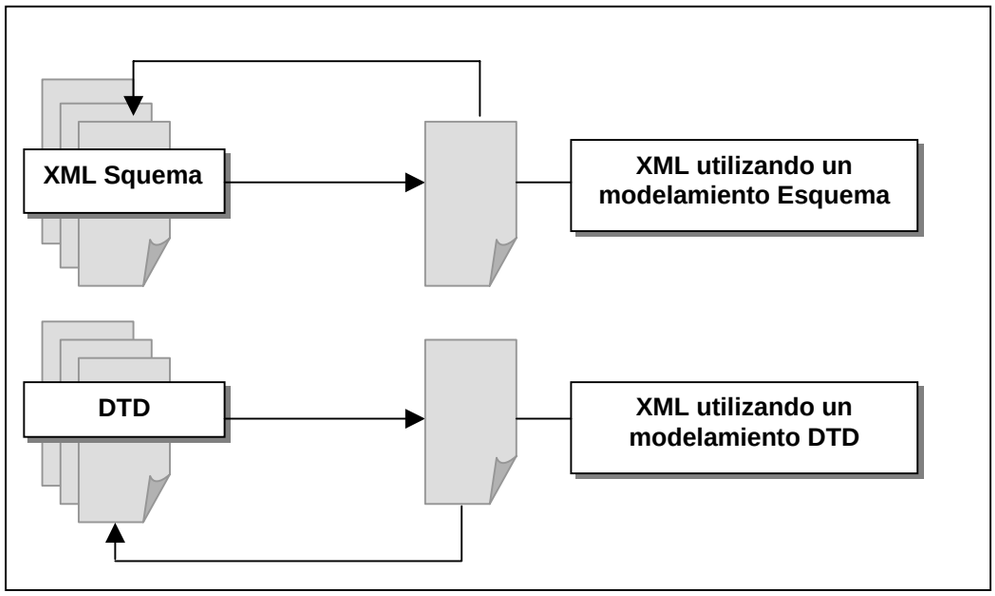

Figura 3-1: XML adheridos a sus modelos de datos respectivos.

La mejor forma de comprender el rol de un Esquema para la validación de un documento construido en XML es considerando un documento XML sin Esquemas.

Sin un Esquema, el documento XML bien construido debe adherirse a la sintaxis XML. Según la recomendación 1.0 de XML, propuesta por el W3C, un documento XML debe adherirse a la sintaxis XML para ser considerado un documento XML [URL 19], [Mor2000, 21]. No obstante, se pueden crear documentos XML que sean abiertos en términos de cómo se pueden estructurar y organizar.

Un Esquema establece como regla la estructura de un documento XML estableciendo restricciones en los datos de marcado y de caracteres que existen en el interior y que se utilizan para modelar clases de datos, lo que se denomina además **vocabularios XML**. Las DTD, por su parte no incluyen una utilidad para restringir tipos de datos, presentando una gran ventaja según [URL 17].

Los esquemas establecen restricciones en la estructura del contenido de un documento XML, esto se realiza de la siguiente forma:

- Establece un modelo de contenido de los documentos, dando el orden y anidado de cada elemento.
- Establece el tipo de datos del documento.

3.3 MODELAMIENTO DE DATOS CON DTD

Las DTD tienen el origen en el SGML [URL 17] y además son usadas como mecanismo estándar de esquemas para validar los documentos SGML. Puesto que XML es un subconjunto de SGML, figura 3-2, solo tiene sentido utilizar la misma solución de esquemas. Una de las grandes ventajas del uso de las DTD como esquemas es que las herramientas SGML existentes pueden fácilmente modificarse para soportar XML, ya que soportan las DTD en SGML.

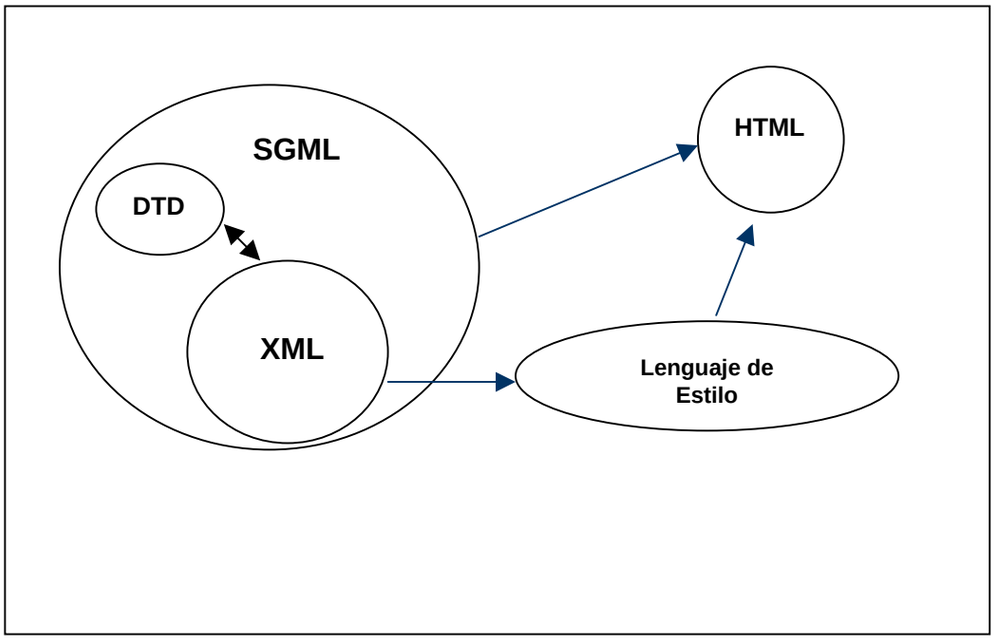

Figura 3-2: XML en el contexto

Sin embargo se pueden detectar algunas limitaciones, como conclusión, de las DTD, basándose en [Mor2000, 24]:

- Las DTD descansan sobre una sintaxis especializada, que describe una estructura para los vocabularios XML, esto puede aparecer como un punto en contra, ya que

XML proporciona la sintaxis suficiente y adecuada para describir los datos de cualquier tipo, incluyendo la estructura de los documentos.

- La sintaxis DTD es compacta, sin embargo la forma en que se describe la estructura de los documentos es bastante críptica y además es poco comprensible, sin embargo crear una DTD es un proceso fácil y mecánico.

Haciendo referencia a la figura 2-4, del capítulo anterior, donde se listaba una DTD para un XML específico, se puede notar que una DTD permite mostrar la mecánica básica de como se podría definir un modelo de documento por medio de la sintaxis DTD. El signo (+) indica que el elemento se puede producir más de una vez, mientras que el signo (**?**) indicará que el elemento solo es opcional, sin embargo, si este aparece, debe aparecer solo una vez.

Se pueden agrupar los elementos en una especie de libreta de direcciones para que la DTD sea más compacta, tal como se ilustra en la tabla 3-1:

Tabla 3-1: Ejemplo de una Definición de Tipo de Documento (DTD) compacta.

```
<!ELEMENT librodedirecciones (contacto)+>

<!ELEMENT contacto (nombre, direccion+, ciudad, estado, telefono, email, web, empresa) (#PCDATA)>

<!ELEMENT estado (codigo_postal, pais_origen?)>
```

Todos los elementos restantes en una DTD se enumeran como #PCDATA, los cuales corresponden a datos de caracteres analizados sintácticamente [URL 17].

Se puede elegir entre agrupar elementos o dejarlos expandidos según se requiera, lo cual no alterará la sintaxis de una DTD.

La Definición de Tipo de Documento (DTD) corresponde a la base de los documentos válidos, ya que establecerá siempre la gramática de un vocabulario XML, que además establecerá la estructura de los documentos XML. Una DTD sirve para realizar la

validación de documentos que es además muy importante en el desarrollo de contenido en XML.

Las características importantes de las DTD son las siguientes:

- La estructura de un documento XML viene determinada por la declaración de tipo de documento; en las declaraciones de tipo de documento, las declaraciones de marcado establecen la gramática de un documento o clase de documento.
- Proporciona un inicio para describir la estructura de un documento (un documento DTD se puede guardar como un archivo acompañando al XML correspondiente con la extensión **.dtd**).
- Un procesador XML utiliza la DTD de un documento para determinar el esquema del documento, que es la definición de la estructura del documento,
- Contiene declaraciones de marcado en el prólogo del documento, DTD local o DTD interna.
- Hace referencia a declaraciones externas de marcado que se llaman subconjunto de la DTD externa, o DTD externa.
- Las DTD se dividen en dos subconjuntos, uno interno y otro externo, cuando se hace mención a la DTD se hace referencia al subconjunto interno y externo, razón que tiene que ver con la flexibilidad.
- El **subconjunto externo** de la DTD permite describir la estructura general del documento de una clase de documentos, el **subconjunto interno** de la DTD suele describir una estructura específica de un documento. Los procesadores XML dan prioridad al subconjunto interno de la DTD. Esto da como característica extra que las DTD tengan una forma de herencia a un nivel.
- Se puede incluir en las DTD los tipos de elementos que se permiten en el documento y los modelos de contenido.
- Se puede incluir en las DTD los atributos que se pueden asignar a cada elemento.
- Se puede incluir las entidades.

- Se puede incluir las notaciones que se permiten para usarse como entidades externas.
- Las declaraciones de marcado que hay en la DTD describen las distintas partes de un tipo de documento, lo cual permite validar los documentos frente al tipo de documento.

Es posible describir la DTD de un documento utilizando uno de los dos subconjuntos conocidos:

- **DTD interna**
- **DTD externa**
- **Combinación de ambas**

Las DTD internas deben declararse en la declaración de tipo de documento, mientras que se debe hacer referencia a las DTD externas. La sintaxis de la declaración de tipo de documento se puede apreciar en la tabla 3-2:

Tabla 3-2: Sintaxis de la declaración de tipo de documento.

<! DOCTYPE RootELem SYSTEM ExternalDTDRef [InternalDTDDecl]>

ExternalDTDRef hace referencia a la DTD externa, que es el URI (Uniform Resource Identifier, Identificador Uniforme de Recursos) de un archivo que contiene la DTD externa.

La DTD interna, se declara entre corchetes (**[ ]**) y corresponde a **InternalDTDDecl** de la sintaxis de la declaración de tipo de documento.

RootELem identifica el elemento raíz de la clase de documento en la sintaxis de la declaración de tipo de documento.

La tabla 3-3 ejemplifica el uso de estos elementos:

Tabla 3-3: Aplicación de la sintaxis de la declaración de tipo de documento.

```
<!DOCTYPE direcciones SYSTEM "librodedirecciones.dtd" [
<!ELEMENT  persona (#PCDATA)> ]>
```

El elemento raíz de la clase de documentos es direcciones, lo que significa que todos los documentos de este tipo deberán estar dentro del elemento direcciones. La declaración de tipo de documento hace referencia a una DTD externa almacenada en **librodedirecciones.dtd**. Además se declara un elemento persona como parte de una DTD interna.

Al hacer referencia a la tabla 2-4, **<?xml version="1.0" standalone="no"?>**, nos encontramos con el segmento de código: **standalone**, el cual significa que el documento no es autónomo por lo que cuenta con una DTD a la que hace referencia, en cambio la sintaxis **<?xml version="1.0" standalone="yes"?>** permite definir que el documento XML es autónomo, por lo tanto no se apoya en fuentes de información externas.

A la hora de elegir una DTD, sea externa o interna, se prefiere colocar declaraciones de marcado de documento en una DTD externa, por las siguientes razones:

- Es preferible cuando se está creando un documento o documentos que deben ser válidos.
- Si se están creando múltiples documentos de la misma clase
- Se prefiere utilizar una DTD existente.
- Se prefiere para hacer documentos más concisos.

La tabla 3-4 resume los elementos para construir una DTD:

Tabla 3-4: Elementos de la DTD.

Descripción:	**Sintaxis:**	**Característica:**
Trabajar con elementos:	**<u>Sintaxis:</u>** **<!ELEMENT *ElementName* Type>**	Los elementos se declaran en una DTD con declaraciones de elementos.
Elementos vacíos:	**<u>Sintaxis:</u>** **<!ELEMENT *ElementName* EMPTY>**	Los elementos vacíos no tienen contenido.
Elementos de sólo elementos: 1. **()**: Engloba una secuencia o un grupo de elementos secundarios. 2. **,** : Separa los elementos de una secuencia y establece el orden en el que deben aparecer. 3. **|** : Canalización, permite separar los elementos de un grupo de alternativas. 4. Sin símbolo: Indica que debe aparecer un elemento secundario una sola vez. 5. **?**: Indica que un elemento secundario debe aparecer solamente una vez, o que no debe aparecer en absoluto.	**<u>Sintaxis:</u>** **<!ELEMENT *ElementName* ContentModel >** **<u>Ejemplo:</u>** **<!ELEMENT resumen (intro,(educacion | trabajo | experiencia+)+, hobby?, referencias*) >**	Corresponde a la forma para declarar el modelo de contenido de un elemento.

6. ***** : Indica que un elemento secundario puede aparecer cualquier número de veces, +: Indica que un elemento secundario debe aparecer por lo menos una vez.		
Elementos Mixtos:	**<u>Sintaxis:</u>** **<!ELEMENT *ElementName* (#PCDATA)>** **<u>Ejemplo:</u>** **<!ELEMENT banda (#PCDATA)>** <banda>Los Beatles</banda>	Los elementos mixtos contienen datos de caracteres y elementos secundarios. Suele decirse que es un elemento de sólo texto. Los elementos de solo texto se declaran de esta forma.
Elementos ANY:	**<u>Sintaxis:</u>** **<!ELEMENT *ElementName* ANY>**	Este elemento no tiene estructura, puede contener datos de caracteres, tipos de elementos declarados o una mezcla de ambos.
Atributos:	**<u>Sintaxis:</u>** **<!ATTLIST ElementName**	Los atributos se utilizan para

AttrName: Nombre atributo. **AttrType:** Tipo. **Default:** valor predeterminado, puede tomar los valores siguientes: ▪ #REQUIRED, atributo obligatorio. ▪ #IMPLIED, atributo opcional. ▪ #FIXED value, atributo valor fijo. ▪ default, valor predeterminado del atributo.	**AttrName AttrType Default>**	especificar información adicional acerca de un elemento. Dentro de un elemento, los atributos son usados para formar un par de nombres-valor, que en cierto modo describe una propiedad particular del elemento.
Atributos de cadena: ▪ **CDATA**	**<u>Ejemplo:</u>** **<!ATTLIST jugador equipo CDATA #REQUIRED >**	Indica que el atributo posee una cadena sencilla de texto.
Atributos enumerados:	**<u>Ejemplo:</u>** **<!ATTLIST jugador posicion (centro | atras | defensa | alero) "centro" >**	Se componen de atributos cuyos valores están limitados a una lista de cadenas de texto predefinidas.

Atributos con símbolos: ▪ **ENTITY:** Referencias a entidades. ▪ **ENTITIES:**Referencias a entidades. ▪ **ID:** Identificador, relacionados a identificadores únicos. ▪ **IDREF:** Identificador, relacionados a identificadores únicos, permiten referenciar al ID. ▪ **IDREFS:** Similar que IDREF. ▪ **NMTOKEN:** Especificar atributos que contienen valores de símbolos con nombre. ▪ **NMTOKENS:** Similar que NMTOKEN.	**Ejemplo:** **<!ATTLIST imagen formato ENTITY #IMPLIED >**	Atributos que son procesados como símbolos por un analizador XML.

A continuación, la tabla 3-5, muestra una DTD para una colección de elementos:

Tabla 3-5: DTD válida.

```
<!ELEMENT peliculas_tv_cable (pelicula)+>
<!ELEMENT pelicula (titulo, escritor+, productor+, director+, actor*, comentarios?)>
<!ATTLIST pelicula
  type (drama | comedia | aventura | ficcion | misterio | terror | romance |
    documental) "drama"  rating (G | PG | PG-13 | R | X) "PG"  revision (1 | 2 | 3 | 4 | 5)
"3"  año CDATA #IMPLIED>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT escritor (#PCDATA)>
<!ELEMENT productor (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT comentarios (#PCDATA)>
```

El código de la DTD expuesto en el ejemplo anterior muestra como ensamblar los elementos y los atributos para formar una DTD completa.

La estructura de un documento XML viene determinada por la declaración de tipo de documento, que contiene la definición de tipo de documento o DTD. La DTD servirá como plantilla de una clase de documentos y además se compone de declaraciones de marcado que establecen la gramática de la clase de documentos. Son fundamentales además en el diseño y desarrollo de los documentos XML, ya que presentan la funcionalidad de prestar una forma de validar documentos XML, según lo expuesto en [Mor2000, 53].

3.4 MODELAMIENTO DE DATOS CON XML SCHEMA

De acuerdo con el W3C, un esquema es *"un conjunto de reglas que sirve para forzar la estructura y la articulación de un conjunto de documentos XML".* Por lo tanto, las DTD están capacitadas como Esquemas XML [Mor2000, 74]. XML Schema, esquema XML, es una tecnología creada por Microsoft que se basa en XML-Data y DCD (Document Content Description, Descripción del Contenido del Documento), tomada del vocabulario RDF (Resource Description Framework, Estructura de Descripción de Recursos), tomando la definición de [Mor2000, 75]. Microsoft se refiere a XML Schema como *"un subconjunto de la presentación de XML-Data que se corresponde al conjunto de características propuestas para la DCD".*

Con XML-Data y DCD, Microsoft y otros fabricantes proponen los vocabularios para que un documento XML exprese el Schema XML que utiliza. Esto permite que sean los datos XML los que pueden describir su propia estructura. La expresión de Schema dentro de XML aumenta la potencia del formato XML, ya que permite que el software examine determinada información, para comprender su estructura, sin necesitar ninguna descripción previa incorporada de la estructura de los datos, como conclusión de [URL 11].

Con un Schema XML, una persona puede definir exactamente qué nombres de elementos se permiten en un documento y, dentro de cada elemento, qué sub-elementos, atributos y relaciones se admiten. Se puede importar fragmentos de otros XML Schemas, así como ampliar tipos a través de la herencia. Todo permite hacer relaciones complejas, entre elementos sin perder la simplicidad de la estructura.

Con la solución brindada por Microsoft, se plantea la inquietud de que las empresas que quieren utilizar XML necesitan una forma sencilla de buscar la información del esquema que se acomode a sus necesidades, los documentos y los procesos empresariales, compatibles además con otras aplicaciones empresariales, que eventualmente pueden ser distintas o similares. Plantea además el problema del costo que supondría para los consumidores y las empresas, si para cada empresa se definiera su propia forma de publicar la información, al aprender a crear y manipular fuentes de datos XML [URL 12].

Además se plantea que de no haber ningún límite al número de empresas que podrían publicar dicha información, la falta de estándares que definan el modo de publicarla de una forma segura y controlada llevaría como consecuencia miles y miles de implementaciones, enfoques de exploración de la web y profundidad del contenido distintos. La carga financiera que supondría permitir la propagación de este entorno llegaría hasta los consumidores. Como consecuencia, se ha implementado la organización Biztalk, que consiste en un conglomerado de empresas que comparten su información en un portal del tipo Business to Business (Negocio a Negocio) [URL 2].

Según [Mor2000, 79], los elementos que conforman el vocabulario para el modelamiento XML Schema son los siguientes:

1. **Schema:** Sirve como elemento raíz de los documentos XML Schema y actúa como contenedor del resto del contenido de los esquemas. Incluye los atributos name y xmlns.

- **name**: Nombre del esquema.
- **xmlns**: El espacio de nombres del esquema, debe configurarse como **urn:schemas-microsoft-com:xml-data**, para usar la implementación de Microsoft.
- Sintaxis utilizada: <Schema name="Nombre_Esquema" xmlns="urn:schemas-microsoft-com:xml-data" >

2. **Datatype:** Describe los tipos de datos en los elementos y los atributos.

- **Sintaxis utilizada:** <Schema name=" Nombre_Esquema" xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes">
- <datatype dt:type="tipo_de_dato">

3. **ElementType:** Describe un tipo de elemento. Dentro de la etiqueta además se definen:

- **name:**nombre de elemento
- **model:** Si el contenido del modelo está abierto o cerrado
- **content:** El tipo de contenido que hay en el elemento
- **order:** orden de los elementos secundarios y grupos.
- **dt:type:** El tipo de elemento.
- **Sintaxis utilizada:** <ElementType name="nombre_variable" content="textOnly" dt:type="tipo_de_dato"/>

4. **Element:** Identifica un elemento que puede producirse en un tipo de elemento. Este elemento incluye tres atributos, que se pueden utilizar para escribir información adicional acerca de una instancia de elementos:

- **type**: Tipo de elementos.
- **minOccurs**: El número mínimo de veces que el elemento debe producirse.
- **maxOccurs**: El máximo número de veces que el elemento debe producirse.
- **Sintaxis utilizada:** <element type="tipo" minOccurs="1" maxOccurs="1"/>

5. **<u>Group</u>:** Organiza los elementos en grupos, con fines organizativos. Este elemento incluye tres atributos:

- **order:** Orden de los elementos secundarios que hay en el grupo. Los valores aceptables de este atributo:
  - *one*: Permite una serie de elementos en el grupo.
  - *seq*: Los elementos deben producirse en la secuencia del grupo.
  - *many*: Los elementos pueden producirse cualquier número de veces y en cualquier orden del grupo.
- **minOccurs**: El número mínimo de veces que se produce el grupo.
- **maxOccurs**: El máximo número de veces que se produce el grupo.

6. **<u>AttributeType</u>:** Describe un atributo, permite definir un tipo de atributo.

- **name**: Nombre del atributo.
- **dt:type**: Tipo de datos del tipo de atributo.
- **dt:values**: La lista de valores posibles de un atributo, se aplica cuando dt:type está establecido a **enumeration.**
- **default**: El valor predeterminado de un atributo.
- **required**: Indicador que muestra si debe proporcionarse el atributo en el elemento.
- **Sintaxis utilizada:** <AttributeType name="nombre" dt:type="tipo_dato" dt:values="valor_a valor_b valor_c ... valor_n"/>

7. **<u>Attribute</u>:** Identifica un atributo que puede producirse en un tipo de elemento. Este elemento cuenta con tres atributos:

- **type**: Tipo del atributo.
- **default**: Valor predeterminado del atributo.
- **required**: Indicador que muestra que debe darse el atributo en el elemento.

8. **Descripción:** Proporciona documentación para un elemento o atributo. Puede ser utilizado como comentario. Va acompañado entre las etiquetas **<description>** y **</description>**

Los XML Schemas, se identifican por sus etiquetas de inicio y final:

<Schema name="TrainSchema" xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes"> y **</Schema>**. Lo que hay al interior de estas etiquetas corresponde al esquema en si.

Los tipos de datos utilizados por XMLSchemas son los descritos en la Tabla 3-6, que corresponden además a los soportados por el estándar XML 1.0:

Tabla 3-6: Tipo de datos soportados por XML Schema.

▪ **char**	Carácter longitud uno
▪ **boolean**	Booleano 0 o 1
▪ **int**	Entero
▪ **float**	Número flotante
▪ **number**	Número real
▪ **fixed.14.4**	Número real 14 dígitos enteros y 4 fraccionarios.
▪ **i1**	Entero de un byte
▪ **i2**	Entero de dos bytes
▪ **i4**	Entero de cuatro bytes
▪ **r4**	Número real de cuatro bytes
▪ **r8**	Número real de ocho bytes
▪ **ui1**	Entero no firmado de un byte
▪ **ui2**	Entero no firmado de dos byte

▪ **ui4**	Entero no firmado de cuatro byte
▪ **bin.hex**	Número hexadecimal, base 16.
▪ **bin.base64**	Número base 64
▪ **date**	Fecha
▪ **dateTime**	Fecha con hora opcional
▪ **dateTime.tz**	Fecha con hora opcional y configuración regional.
▪ **time**	Hora
▪ **time.tz**	Hora y configuración regional, sin fecha.
▪ **uri**	Identificador Uniforme de Recursos
▪ **uuid**	Identificador global
Tipos de datos primitivos:	**Descripción:**
▪ **string**	Tipo cadena de caracteres.
▪ **enumeration**	Un tipo enumerado.
▪ **notation**	Un tipo NOTATION.
▪ **entity**	Un tipo ENTITY.
▪ **entities**	Un tipo ENTITY.
▪ **id**	El tipo ID.
▪ **idref**	El tipo IDREF.
▪ **idrefs**	El tipo IDREFS.
▪ **nmtoken**	El tipo NMTOKEN.
▪ **nmtokens**	El tipo NMTOKENS.

3.5 COMPARACION DEL MODELAMIENTO

Las principales ventajas de XML Schema, en comparación con las DTDS son:

- XML Schema se basa en XML y no en una sintaxis especializada.
- XML puede ser analizado sintácticamente y manipulado como cualquier otro documento XML
- XML Schema soporta tipos de datos (int, float, boolean, date, etc.).
- XML Schema presenta un modelo de datos abierto, lo que permite ampliar vocabularios y establecer relaciones de herencia entre los elementos.
- XML Schema soporta la integración de los espacios de nombres , lo cual permite asociar nodos individuales de un documento.
- XML Schema soporta grupos de atributos.
- XML Schema soporta combinaciones de atributos.
- Los modelos de contenido de la DTD son cerrados.
- La DTD soporta sólo sus tipos de datos propios.
- Se permiten construir XML Schemas válidos para negocios personales.

3.6 CONCLUSIONES

XML Schema es una tecnología de Microsoft que se implementa como un subconjunto de la XML-Data al W3C, que se aproxima a la funcionalidad de la nota DCD del W3C. XML Schema proporciona importantes ventajas sobre la solución DTD tradicional a la hora de describir la estructura de los documentos XML, que se ha heredado de SGML.

La gran ventaja es que un XML Schema es reutilizable, con posibilidad de generar Schemas propios para negocios o un área en particular.

Los documentos XML se adhieren perfectamente a los XML Schemas, al igual que en las DTDs, sin embargo la gran ventaja ofrecida por los Schemas es que se pueden definir tipos de datos que en la práctica resultarían útiles en los contenidos de los documentos XML, para especificar un modelamiento de datos más granular y específico.

La comunidad Internet está presta a cambiar las DTD por la utilización de los esquemas; esta implementación parcial de Microsoft ha servido como detonante para que los desarrolladores de XML trabajen con esquemas.

Se verá en el capítulo 4, como es posible dar formato de salida a un documento XML y la importancia que tiene un documento bien estructurado, para la representación de salida de la información.

4. FORMATO DE ESTILOS DE UN DOCUMENTO XML

4.1 INTRODUCCION

XML se utiliza para marcar y estructurar el contenido de un documento, por lo que no se preocupa de cómo se presenta el contenido en un navegador web o en otra aplicación.

Cuando se requiere realizar esta tarea, se necesita enfocar ciertos detalles, tales como los tamaños de las fuentes, colores asociados con los distintos elementos y atributos, la disposición de la información propiamente tal. El mecanismo usado para aplicar estilos de formato a los documentos XML se denomina *hojas de estilo.*

Actualmente existen dos técnicas para dar estilo a un documento XML, CSS (Cascading Style Sheets, Hojas de Estilos en Cascada) y XSL (eXtensible Style Languaje, Lenguaje de Estilo Extensible) [Flo2000].

La comunidad Web en general y la comunidad XML en términos particulares ha comprendido que la combinación del marcado de contenido con la presentación de los estilos de diseño puede ser difícil y además costosa en mantenimiento, flexibilidad y complejidad. HTML es el ejemplo más común de lo que ocurre cuando se combina el contenido y la presentación en un solo lenguaje, ya que HTML nunca fue diseñado para ser un lenguaje de presentación, (en sus inicios, estaba diseñado para permitir que los físicos compartieran notas técnicas) [Flo2000, 5].

Las hojas de estilo, por lo tanto, hicieron que fuera posible volver a adaptar HTML y ordenar el caos que había en el contenido y la presentación.

Este capítulo analizará estas dos técnicas fundamentales a la hora de poder representar la información de salida de un documento XML, sobretodo en las aplicaciones sobre Internet, donde la mayoría son aplicaciones para la Web. Se enfocará en el Lenguaje de Estilo Extensible XSL, debido a que presenta una mejor solución para dar formato de salida a un documento XML y sobretodo en la representación de contenidos.

4.2 XSL Y CSS

Los primeros sistemas SGML utilizaban hojas de estilo para separar el contenido de la presentación. Incluso los primeros navegadores utilizaban hojas de estilo para determinar cómo mostrar los datos de los documentos HTML.

En 1995 se comenzó a hablar de CSS, cuando el W3C lo propuso formalmente, convirtiéndose en una recomendación en el año 1996 denominándose CSS1, Nivel uno de las CSS. En 1998 se adoptó el CSS2, sin embargo no se ha dado un empuje por parte de los fabricantes de navegadores [Mor2000, 134].

Microsoft soportó una primera versión de CSS1 en Internet Explorer 3.0 EN 1996, incluso utilizado en Explorer 4.0; en cambio Explorer 5.0 (en la actualidad, versión 6.0) no soportaba completamente la recomendación CSS1 [Mor2000, 133].

XSL es una tecnología más emergente que CSS y no tiene la misma compatibilidad con los navegadores. Esto se debe a que solo existe un navegador que soporta XSL: Internet Explorer 5.0 y sus posteriores versiones.

XSL nace en 1997 por medio de una nota de Microsoft y ArborText, publicándose el primer borrador del trabajo en julio de 1998 [Mor2000, 134].

Según [Mor2000, 134], XSL fue creado como una forma de dar respuesta a las necesidades de XML, basándose en DSSSL (Document Style Semantic and Specification Languaje, Lenguaje de Especificación de la Semántica de Estilo de Documentos) y las CSS [URL 16].

4.3 FUNDAMENTOS

Los fundamentos para los dos estilos existentes son:

- **CSS:**

Según la definición de [Mor2000, 135], CSS es un lenguaje de hojas de estilos que está pensada y diseñada como una forma de dotar de estilo a los documentos HTML,

permitiendo que los desarrolladores web separen el contenido de la presentación. Antes de CSS existía el DHTML (Dynamic HTML, HTML Dinámico).

Cuando una hoja de estilos CSS es aplicada a un documento XML utiliza una estructura de árbol XML como base para la aplicación de reglas.

- **XSL:**

Según la definición de [Mor2000, 135], XSL es una tecnología de hojas de estilos que toma dos características de los documentos XML, la transformación de documentos de un tipo a otro (XSLT) y dar estilo a los documentos en base a reglas de formato (objetos con formato XSL). Dado a que ambos componentes se implementan como vocabularios de XML, las hojas de estilos que se crean a partir de ellos son documentos XML, así se pueden crear hojas de estilos XSL utilizando sintaxis XML. Cuando un documento XML se transforma con XSL, la transformación se aplica a la estructura lógica del documento XML, por lo que el resultado de una transformación XSL es un árbol de datos XML. Cuando queremos transformar XML en HTML usamos XSLT, la transformación XSL que mejora la necesidad de soporte para navegador XSL.

La figura 4-1 muestra en el contexto la forma de operar con CSS y XSL.

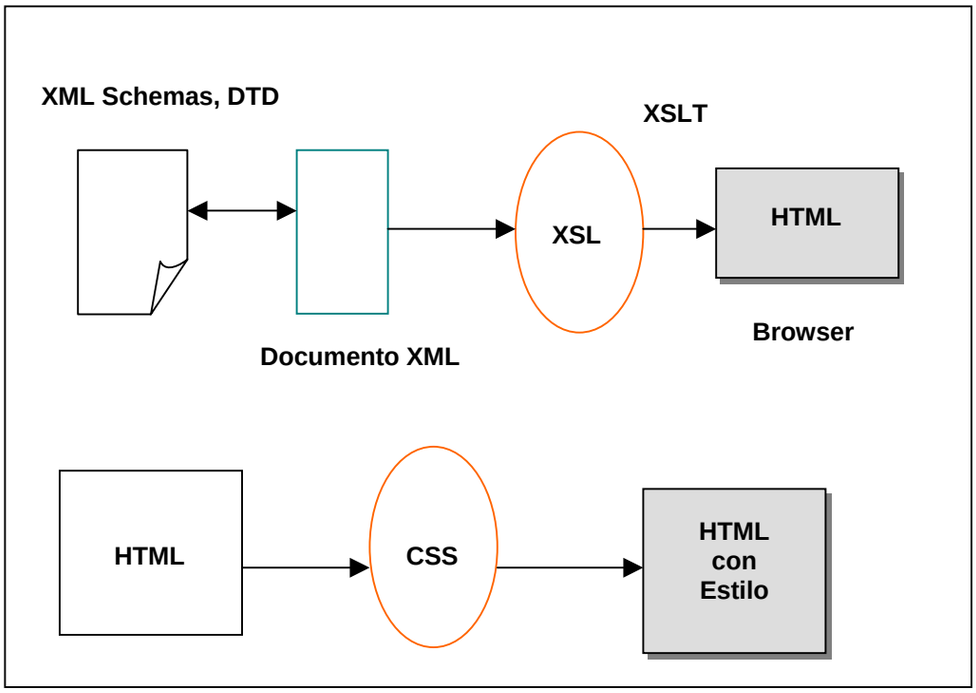

Figura 4-1: CSS y XSL en el contexto.

4.4 EVOLUCION DE LAS CSS Y XSL

Como ya se discutió anteriormente el avance de XSL y CSS, cabe mencionar que aún son tecnologías que están en fase de investigación y desarrollo, esperando nuevas mejoras a futuro, para las aplicaciones específicas de la Web y dispositivos.

Actualmente el estándar adoptado por la W3C, CSS1, está siendo trabajado por los principales navegadores, sin embargo el estándar CSS2 (Nivel 2 de las CSS) supone un avance en lo referente a añadir flexibilidad al CSS. CSS2 por el momento solo pasa a ser una buena idea [Mor2000, 134].

XSL por su parte es el lenguaje de estilo para un documento XML, ya que algunos navegadores no soportan XSL (característica que se puede observar en la práctica, como en el proyecto Educar Chile), sino que soportan XSLT (XSL= XSLT + Objeto con formato XSL, XSLFO), con el cual se resuelve el tema de la incompatibilidad entre los navegadores.

Con la aparición de Internet Explorer 6.0 (ocupado como navegador de prueba en el proyecto Educar Chile), de la empresa Microsoft Corporation, el 27 de agosto del 2001, se abre una nueva plataforma para aplicaciones basadas en DHTML, CSS1, XML, XSLT, Document Object Model (DOM) Nivel 1, con lo cual CSS no ha tenido mayor evolución.

4.5 COMPARACIÓN DE XSL Y CSS

Basándo la comparación según [Mor2000, 136], existen dos diferencias claves entre CSS y XSL:

- Las CSS se pueden utilizar para dar estilo a los documentos HTML, mientras que XSL no.
- XSL se puede usar para transformar documentos XML, mientras que las CSS no.

Algunas limitantes de CSS:

- No puede tomar una parte de los datos del documento y reutilizarla en otra parte.
- No conceptúan las relaciones hermanas entre los nodos.
- No soporta las estructuras de decisión (condicionales).
- No puede calcular cantidades o almacenar valores en las variables.

Como se demuestra, XSL es una tecnología más potente que CSS, pero la potencia añadida se incorpora a costa de una complejidad que también debe ser añadida.

Este trabajo, investiga las herramientas XML y XSL, por lo que no abordará en profundidad el tema CSS, dejando el debate abierto para algún trabajo posterior.

4.6 XSL

4.6.1 PROCESAR UNA HOJA DE ESTILOS XSL

Según Mor[2000, 158], dos puntos importantes para que un procesador XSL pueda procesar documentos XML son: la representación del árbol XML del documento y la hoja de estilos XSL.

La representación del árbol XML de un documento se obtiene analizando sintácticamente el documento, lo que significa que el procesador XSL debe emparejarse con un analizador XML para que funcione.

El procesador XSL comienza con el nodo raíz del árbol, utilizándolo para llevar a cabo el cotejo de los patrones en la hoja de estilos. Una hoja de estilos XSL está formada por plantillas que pueden utilizar patrones y así determinar qué partes de un documento XML tienen que ser formateadas. El procesador analiza estas plantillas y los patrones asociados para procesar las distintas partes del árbol. Cuando se produce una coincidencia, la parte del árbol que coincide con el patrón determinado se pasa a la plantilla de la hoja de estilos para su procesamiento. El procesador XSL sigue las reglas de la plantilla para generar un árbol de resultados. Este punto es útil, ya que se toma un árbol como entrada y se genera otro árbol como salida.

El árbol de resultados que genera el procesador XSL puede contener objetos de formateo XSL, para dar formato al árbol de resultados. Ver figura 4-4.

Una observación muy importante es que no es posible utilizar código HTML tradicional como base de un árbol de resultados XSL, ya que el HTML tradicional no es un vocabulario XML. Sin embargo, existe una nueva versión de HTML, el XHTML, para añadir más estructura a HTML y para convertirlo en un vocabulario XML.

La figura 4-2 muestra el despliegue en un browser de un documento XSL que toma sus datos a partir de un XML, mostrando así la estructura de árbol que se representa en la salida, en este caso no se ha aplicado XSLT para desplegar en el browser el formato de salida HTML. El código está disponible en el anexo 4-1.

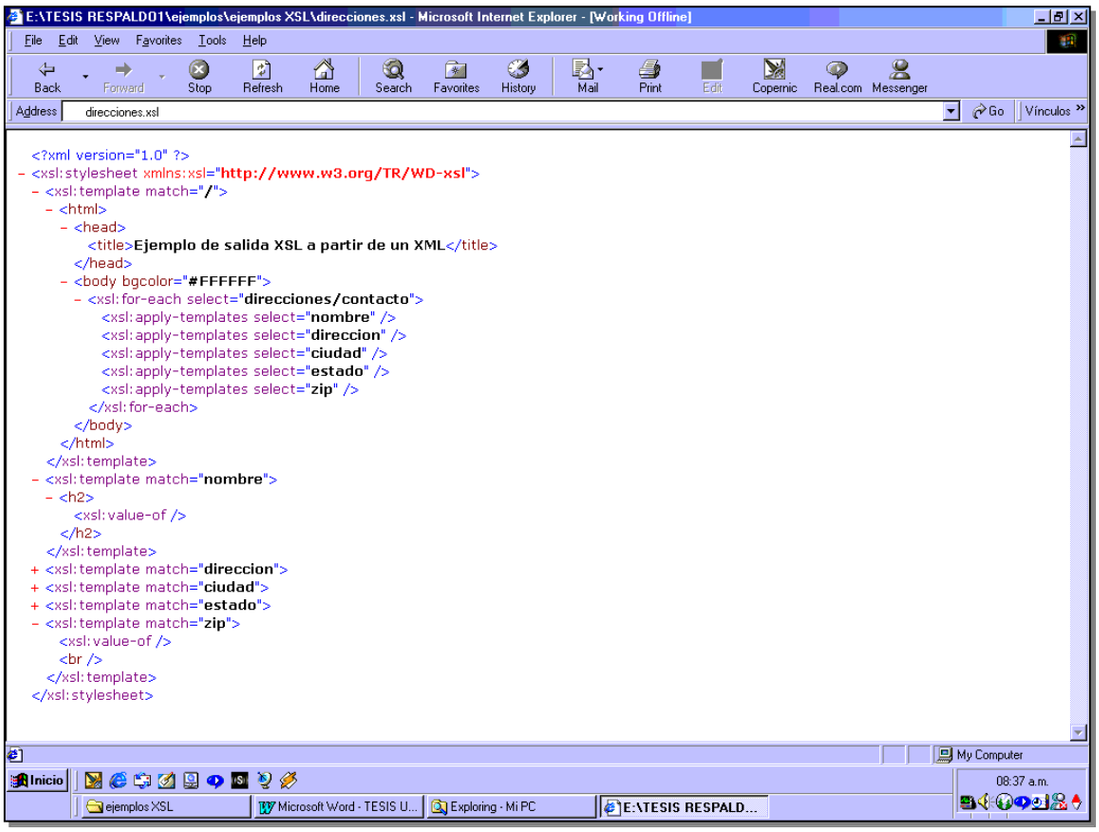

Figura 4-2: direcciones.xsl

4.6.2 ARQUITECTURA DE XSL

El W3C ha realizado un gran trabajo para diseñar la arquitectura de XSL, para que éste sea lo más flexible y extensible posible. Aunque XSL ha empezado como una tecnología individual, ha evolucionado en múltiples tecnologías, las cuales para ser comprendidas deben ser analizadas de acuerdo a dos acciones del procesador XSL [Mor2000, 159]:

- La construcción de un árbol de resultados a partir de un árbol de origen.
- La interpretación del árbol de resultados para fines de formateo.

El primer papel del procesador XSL corresponde a transformar un árbol de origen en un árbol de resultados, lo que se conoce como *" transformación de árboles"*, este proceso consiste en transformar contenido XML de un vocabulario a otro. La segunda acción del

procesador XSL implica examinar el árbol para dar resultados y así ofrecer información de formateo, para posteriormente dar formato al contenido de cada nodo.

Con XSL existe una libertad total de amoldar como se requiera el documento de origen durante la transformación del procesamiento del documento.

Esta transformación da un nivel amplio de flexibilidad, muy útil para ofrecer visualización de documentos XML.

Las dos tareas fundamentales que lleva a cabo el procesador XML, se corresponden con dos tecnologías XSL:

- XSL Transformation (XSLT, Transformación XSL)
- Objetos de Formateo (XSLFO)

4.6.3 XSL TRANSFORMATION

Según [Mor2000, 160], XSL Transformation (XSLT) es el componente de transformación de la tecnología de hojas de estilos XSL. XSL consta de un vocabulario XML que se usa para crear hojas de estilos para transformar documentos XML. Funciona en datos XML analizados sintácticamente en forma de árbol y muestra un árbol de resultados formado por los datos transformados.

Utiliza un patrón para seleccionar secciones de un documento XML para su transformación.

XSLT utiliza otra tecnología XML para proporcionar sus características de transformación de documentos. La tecnología XPath se usa en XSLT para seleccionar elementos para su procesamiento y generación de texto.

4.6.4 XPATH

Según la definición de [Mor2000, 161], XPath es un lenguaje de expresión, no XML, utilizado para enfocarse en partes de un documento XML, pero que no se implementa como vocabulario XML. Esto se debe a que las expresiones XPath se usan en

situaciones en las que el marcado XML no se aplica en realidad, como los valores de los atributos. XPath proporciona una forma abstracta para dirigirse a las partes de un documento XML, y forma la base del direccionamiento de documentos en XSLT. La sintaxis que utiliza XPath está diseñada para ser utilizada en los URI y en los valores de los atributos XML.

XPath es un nombre usado además como notación de ruta para dirigirse a documentos XML. XPath funciona suponiendo que un documento XSL ha sido analizado sintácticamente en un árbol de nodos.

XSLT utiliza XPath para recorrer un documento XML y llegar a un nodo determinado. Cuando se evalúan las expresiones XPath, acaban convirtiéndose en un objeto de datos de un tipo específico, como booleano o número. La forma en que se evalúe una expresión XPath depende completamente del contexto de la expresión, que no lo determina XPath.

Según lo expuesto anteriormente, la figura 4-3 muestra la relación de XPath, como va a buscar un nodo en XML:

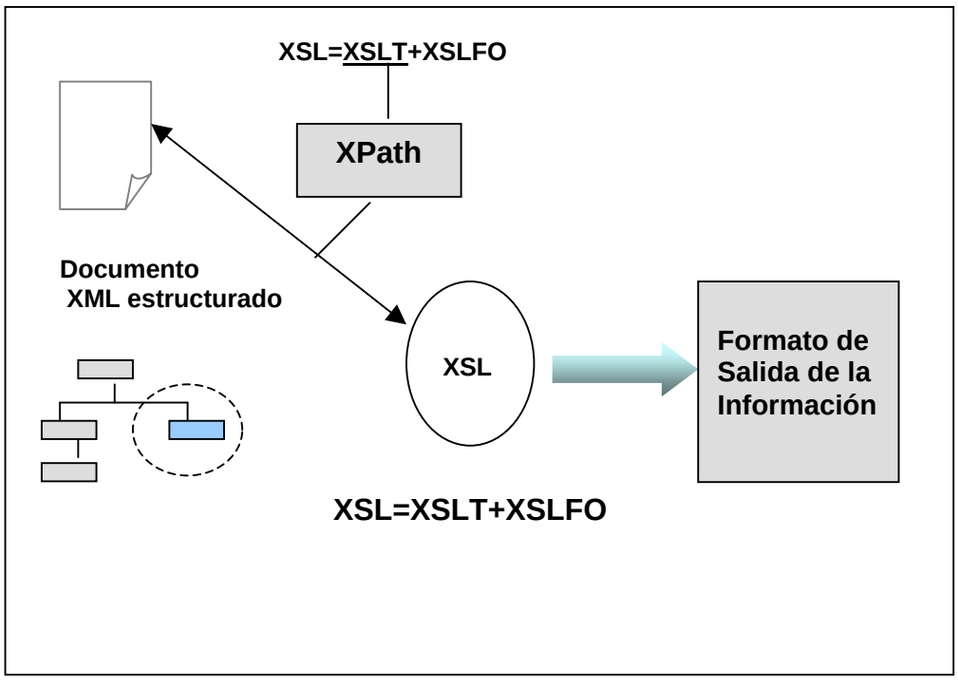

Figura 4-3: Relación de XPath.

4.6.5 OBJETOS DE FORMATEO XSL (XSLFO)

Considerando [Mor2000, 161] y [URL 20], los Objetos de Formateo XSL constituyen el componente del formateo de la tecnología XSL; es un vocabulario XML el cual opera en un árbol de datos XML, que puede ser analizado sintácticamente a partir de un documento, o bien, ser transformado desde un documento por medio de XSLT. Por razones de formateo, los Objetos de Formateo XSL tratan cada uno de los nodos del árbol como objetos de formateo, donde cada nodo soporta una amplia variedad de estilos de presentación. Se puede aplicar estilos estableciendo atributos en un determinado elemento (nodo) del árbol.

Cuando el procesador XSL procesa un objeto de formateo, se asigna a un área rectangular en la superficie de la pantalla. Las propiedades del objeto determinan qué formato tiene, junto con los parámetros del área en la que está asignada.

Los XSLFO serán una tecnología más interesante cuando haya una implementación para navegador.

4.6.6 CREAR HOJAS DE ESTILOS XSL

Según el capítulo 11 de [Mor2000] y [Flo2000,156], las hojas de estilos XSL se usan para transformar el contenido de los documentos y dar formato, para mostrarlo en navegadores web. Microsoft se ha tomado la libertad de soportar una versión anterior de XSLT en Internet Explorer 5.0, lo que da a los desarrolladores web una buena oportunidad de jugar con XSLT y empezar a crear hojas de estilos XSL.

Debido a las diferencias entre la implementación XSL de Microsoft y el borrador de trabajo XSL establecido por el W3C, algunos desarrolladores XSL han optado por extensiones del tipo .msxsl , en hojas de estilos XSL para usarlas con Internet Explorer 5.0, en vez de la extensión .xsl normal.

Una hoja de estilos XSL está formada por una o más plantillas que describen los patrones, que se utilizan para adherir el contenido XML en relación a los estilos.

Las dos construcciones de una hoja de estilos XSL son:

- Plantillas
- Patrones

que serán definidas en los puntos siguientes.

El elemento **stylesheet** es el elemento de documento de las hojas de estilo XSL. Este elemento forma parte de los espacios de nombres XSL, junto con los demás elementos y atributos que conforman XSL. Es necesario declarar el espacio de nombres XSL para poder usar los elementos y atributos XSL. La sintaxis de declaración de espacios de nombres XSL dentro del elemento **stylesheet** es la siguiente:

- **<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">** como cabecera de inicio.
- **</xsl:stylesheet>** como cabecera final.

El prefijo del espacio de nombres se establece a xsl, que es la solución común que se usa en las hojas de estilos XSL, se antepone en todos los elementos y atributos XSL.

4.6.6.1 PLANTILLAS XSL

Una plantilla XSL, es una estructura que describe la salida que se genera en base a "*criterios de coincidencia de criterios*". Consiste en definir una estructura de transformación que se aplique a una cierta parte de un documento de XML [Mor2000, 166].

Es posible crear hojas de estilos compuestas por una sola plantilla, así se manejarán los detalles de la navegación por el árbol XML para dar contenido al formato específico.

También se pueden utilizar múltiples plantillas, en cuyo caso cada plantilla se corresponde con una parte del árbol XML.

Las plantillas se definen en las hojas de estilos XSL utilizando el elemento **xsl:template**, el cual utiliza el atributo opcional **match**, para hacer coincidir los patrones en un documento XML. El match o coincidencia más amplia posible de un documento consiste

en establecer el atributo match a "/", lo que indica que la raíz del árbol tiene que coincidir. El resultado es que en la plantilla se selecciona la totalidad del árbol:

```
<xsl:template match="/">
    ...
</xsl:template>
```

Por ejemplo **<xsl:template match="direcciones">**, hará una coincidencia en los elementos de direcciones.

4.6.6.2 PATRONES XSL

Los patrones se usan en XSL simplemente para determinar que porciones de un documento XML se pasan a través de una determinada plantilla para su transformación.

Un patrón describirá además una rama de un árbol XML, consistente en nodos jerárquicos. Para seleccionar un árbol de documento completo, se usa el patrón raíz, que consta de una sola barra (/). El patrón raíz se presupone que está en otros patrones si no se sustituye [Mor2000, 166].

La figura 4-4 muestra un diagrama de cómo es procesado un documento XML y el rol que tiene la hoja de estilo XSL.

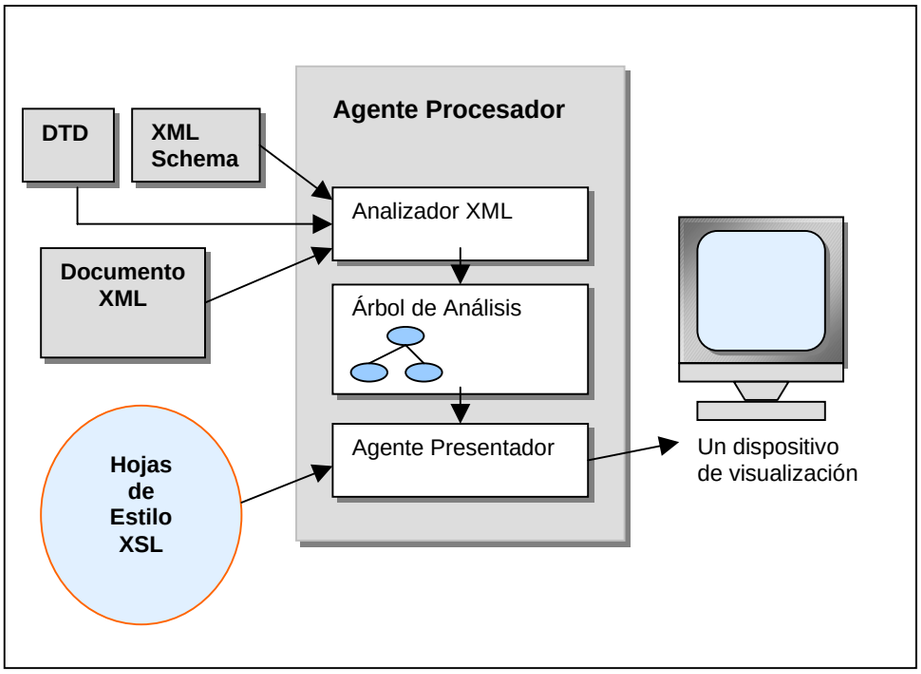

Figura 4-4: Procesamiento de un documento XML

El vocabulario XSLT define varias construcciones de plantillas que controlan la aplicación de las plantillas en las hojas de estilos XSL [Mor2000,168].

Algunos de los elementos XSLT más usados son los mencionados en la tabla 4-1:

Tabla 4-1: Elementos XSLT para construcción de plantillas.

▪ **xsl:value-of** Usado para insertar el valor de un elemento o atributo en la salida resultante de la hoja de estilos.	**Ejemplo:** <xsl:template match="nombre"> <!-- bold en negrita --> <xsl:value-of/> <!-- comentario en XSLT --> </xsl:template>
▪ **xsl:if** Usado para llevar a cabo sentencias condicionales.	**Ejemplo:** <xsl:if match="valor=5"> <xsl:apply-templates select="valor_a_seleccionar"/>

	</xsl:if>
▪ **xsl:for-each** Establece un bucle para recorrer los elementos de un documento.	**Ejemplo:** <xsl:for-each select="libretadirecciones/contacto"> <xsl:apply-templates select="nombre"/></td> <xsl:apply-templates select="direccion"/></td> <xsl:apply-templates select="ciudad"/></td> <xsl:apply-templates select="estado"/></td> <xsl:apply-templates select="pais"/></td> </xsl:for-each>
▪ **xsl:apply-templates** Se utiliza para aplicar plantillas que se definen en una hoja de estilos. Soporta los atributos: ▪ **select** ▪ **order-by**	**Ejemplo:** <xsl:apply-templates select="nombre"/>

4.7 CONCLUSIONES

Considerando los avances de XSL, en el futuro se podrá utilizar para transformar y dar formato a los documentos XML, pero en la actualidad se pueden hacer cosas interesantes con XSL. Por ejemplo, XSLT puede ser utilizado para transformar documentos XML en documentos HTML muy estructurados, lo que se ve muy fácilmente en los navegadores web que se presentan.

XSL representa el mejor lenguaje de estilos para un documento XML bien formado, en contraposición con CSS, tecnología que servirá sólo para dar formato a documentos HTML. XSL en cambio, lleva a pensar que es posible tener muchas aplicaciones que

separan el contenido del formato de despliegue de salida, los cuales se pondrán en análisis en los capítulos posteriores, cuando se analicen las diversas aplicaciones de respuesta en tiempo real.

Además se verá que la mejor forma de aplicación de XML en combinación con XSL es utilizando una combinación con las páginas de servidor activas, ASP, que permitirán un control total de documentos XML para construir niveles de negocios sobre Internet.

5. MODELO DE OBJETO DE DOCUMENTO (DOM)

5.1 INTRODUCCION

El Modelo de Objeto de Documento (DOM) corresponde al medio con el cual se puede acceder y además manipular un documento XML cualquiera y que expone el contenido del mismo [URL 10].

Un problema con XML es que las aplicaciones que implementan estándares tienen la tendencia de producir métodos que son propios para la manipulación.

El estándar de un DOM en HTML, por ejemplo, es el modelo que soporta el lenguaje de programación JavaScript, pero con el inconveniente de que no permite el acceso a todas las partes del documento.

Un DOM debe ser completo, es decir, que debe soportar la reconstrucción de un documento completo desde el propio modelo y debe además permitir el acceso a cualquier parte de dicho documento.

El W3C tiene un grupo de trabajo que se preocupa de la construcción del DOM para HTML y XML; la versión actual no se encuentra terminada en algunas secciones, sin embargo, se designan los siguientes niveles de soporte que permiten accesar la gran mayoría de un documento XML [Mor2000, 268]:

- Nivel 0, es una redeclaración de la sintaxis del lenguaje JavaScript.

- Nivel 1, permite acceder a las partes de un documento XML; no permite acceder a las DTD o a las hojas de estilo asociadas.
- Nivel 2, corresponde al modelo que se trabaja en la actualidad en el W3C, se establecerán accesos a las DTD.

Este capítulo intenta dar un enfoque necesario y general al DOM de XML, analizando el tema en cuestión de una forma global, para posteriormente dar inicio al capítulo de aplicación de XML en la Internet, el cual se entenderá totalmente si se maneja el concepto global de DOM en XML.

5.2 DOM XML

En la actualidad existen diversos tipos de *Modelos de Objeto de Documentos* (DOM), que son en definitiva una serie de planes que proporcionan un modelo del documento; por ejemplo, Microsoft denomina MSXML DOM a su propia **.dll** que define su DOM para codificación de la información [URL 9].

Un DOM puede ser completo, permitiendo el modelado y el acceso a la totalidad de un documento, o puede ser incompleto, permitiendo el acceso parcial al documento [Mor2000, 269].

El modelo puede seguir un patrón común:

- **Modelos lineales:** Es el modelo más sencillo, detallando el documento de una forma lineal. La desventaja de este modelo es que toda alteración de un modelo lineal en la primera parte de un documento, invalida cualquier referencia de las secciones posteriores.
- **Modelos de árbol:** Describe los documentos con los términos de un árbol, donde cada uno de los elementos del árbol se llama *nodo* y cada elemento final se denomina hoja.

- **Modelos de objetos:** Es un modelo útil y menos sensible a los cambios; con un modelo de objetos, cada sección de un documento tendrá una propiedad con un nombre determinado.

A continuación se verá la definición de árboles de documentos y árboles de análisis.

5.3 ÁRBOLES DE DOCUMENTOS Y ARBOLES DE ANÁLISIS

Un ***árbol de documentos*** se puede definir de la siguiente forma:

Según [Mor2000, 271], corresponde a un árbol de representación del documento, donde la raíz del árbol es el propio documento. La raíz posee tres nodos secundarios.

- Un nodo de declaración de versión.
- Un nodo de comentario.
- Un nodo de elemento, también conocido como raíz del documento.

Un árbol de documento se ilustra de una mejor forma en la figura 5-1:

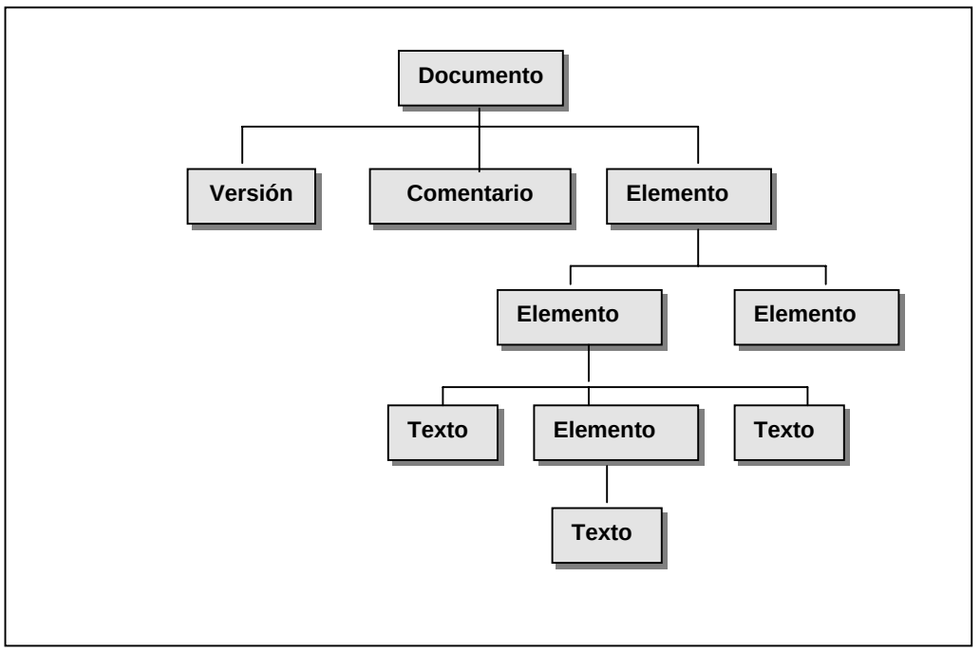

Figura 5-1: Diagrama de árbol de documentos XML.

Un documento XML válido para el diagrama de árbol de documentos como el representado en la figura 5-1 es el ilustrado en la tabla 5-1:

Tabla 5-1: Un documento XML válido para el árbol de documentos 5-1.

```
<?xml version="1.0"?>
<!- - Documento válido para el árbol de análisis - ->
<documento>
  <id_directorio_calles>
      Ñuñoa
      <valor>
        Calle Vicuña Mackenna 750
      </valor>
      Santiago
  </id_directorio_calles>
<antecedentes type="dircomercial"/>
 </documento>
```

En definitiva, el árbol de documentos, es la representación en modelo de árbol de un posterior documento XML bien formado en base a esta estructura.

El resultado de este árbol, en diagramación corresponde al ***árbol de análisis***, que permitirá analizar el recorrido completo del documento.

Para recorrer un árbol DOM, se utiliza la siguiente metodología descrita:

Hay dos formas de acceder a un nodo en el DOM [Mor2000, 274]. Se puede recorrer el árbol o acceder al nodo directamente por el nombre. En la práctica se accede por una combinación de ambas. Se deben seguir los siguientes pasos:

1. Empezar desde cualquier parte del árbol.
2. Utilizar un método de la interfaz del nodo del DOM:

- ***parentNode():*** Método que accede al nodo primario.
- ***firstChild():*** Accede al primer secundario del nodo, si no existe toma el valor null.
- ***nextSibling():*** Este método accede al siguiente nodo igual, en caso contrario toma el valor null.

- ***previousSibling():*** Este método accede al nodo igual anterior, si no existe toma el valor null.

5.4 FUNDAMENTOS DEL DOM DEL W3C

El DOM de la W3C utiliza el lenguaje del OMG IDL (Object Management Group Interface Definition Language, Lenguaje de Definición de Interfaz de Grupo de Administración de Objetos) para describir sus trabajos [Mor2000, 272].

Este lenguaje permite que las aplicaciones se comuniquen entre sí, incluso si están escritas con otros lenguajes diferentes de programación (de ahí la característica importante de XML que permite comunicación entre distintas plataformas computacionales). Los objetos exponen distintas formas de interfaces. Cada interfaz posee una lista de atributos que describen las distintas propiedades del objeto que se oculta detrás de la interfaz. El objeto además puede ser manipulado por una serie de métodos que activarán las funciones que exista tras esa interfaz, y estos métodos devolverán un resultado que apuntará a la aplicación que las solicita.

En conjunto, los atributos y métodos constituyen la API (Interfaz de Aplicación del Programa) de ese objeto.

La aplicación que utiliza un objeto, utiliza además la sintaxis general de la sección.

Para acceder y recuperar una propiedad del objeto y leerla en una variable de aplicación se usa la siguiente sintaxis:

variable= nombre_objeto.nombre_interfaz.nombre_atributo

Para ejecutar un método del objeto y capturar el resultado en una variable:

variable_nombre=nombre_objeto.nombre_interfaz.interfaz_metodo(parametros)

Lo anterior pueden aclararse con la figura 5-2:

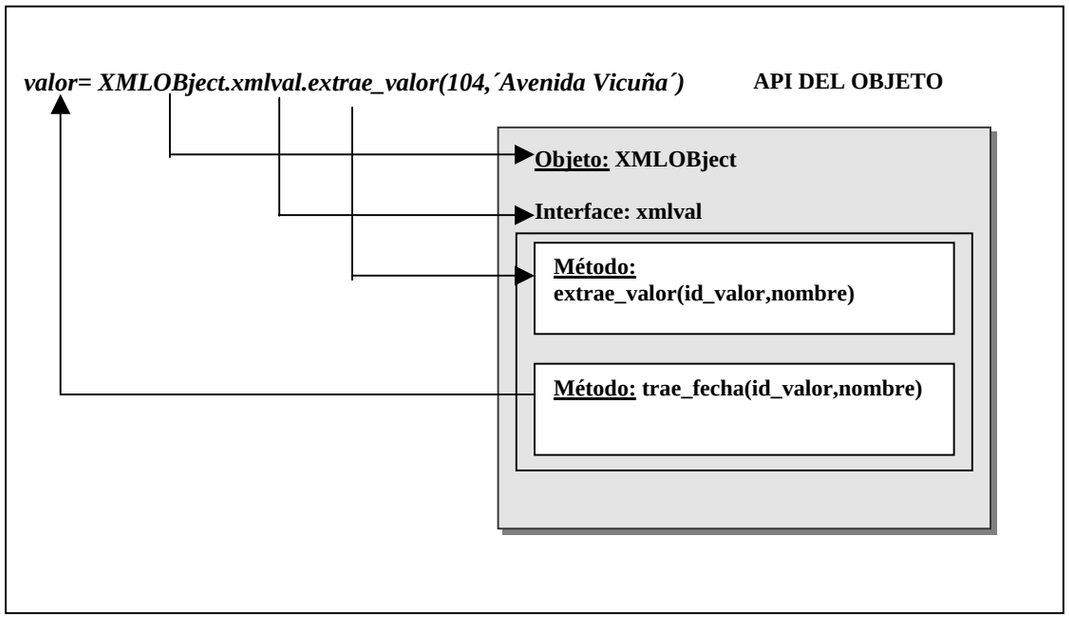

Figura 5-2: Objetos y métodos

5.4.1 OBJETOS DOM

Los objetos DOM tienen las siguientes características:

Todo objeto que soporta el DOM debe ser capaz de cargar un documento XML en sí mismo [Mor2000, 272].

- Se debe exponer todas las interfaces con los atributos y métodos apropiados de acuerdo con la especificación del DOM de la W3C.
- La aplicación a la que se conecte deberá ser capaz de acceder a documentos enteros XML, utilizando la sintaxis necesaria.

5.4.2 INTERFACES DEL DOM

Según [Mor2000, 273], las interfaces del DOM se pueden resumir en los siguientes métodos y atributos que se describen brevemente en esta sección:

- **DOMImplementation:** Es una consulta al propio objeto y es independiente del documento que se cargue. Devuelve información acerca del nivel del DOM que puede soportar un objeto.

- ***DocumentFragment:*** corresponde a la parte que permite trabajar con un fragmento del documento.
- **Document:** Proporciona información acerca del propio documento.
- **Node:** Todo en un documento puede ser considerado nodo, elementos, comentarios y así respectivamente.
- **NodeList:** Conjunto ordenado de nodos.
- **NameNodeMap:** Colección de nodos a los que se puede acceder por nombre.
- **CharacterData:** Interfaz que trata sobre los datos de caracteres o texto.
- **Attr:** Atributos XML de un nodo.
- **Element:** permite interactuar con los elementos y sus atributos.
- **Text:** Trata sobre el contenido de texto de un elemento.

5.5 NODOS Y OBJETOS

Como plantea [Mor2000, 273], el DOM del W3C utiliza una referencia doble con el fin de describir objetos de documentos.

Todo el contenido en un modelo de objeto es necesariamente:

- nodo
- tipo de objeto con su nombre respectivo.

En resumen los nodos y los objetos están representados en el árbol de documentos.

5.6 ACCEDER A LOS NODOS EN EL DOM

La forma de acceder a los nodos en el XML tiene dos variantes:

Una forma es mediante el recorrido del documento, y la otra forma es mediante un método. Para acceder a los nodos en el DOM, se debe utilizar el método **getElementsByTagName(nombre_elemento)**, el cual permite obtener una lista de todos los elementos que contienen el nombre en el documento XML o en cualquier parte del documento [Mor2000, 275], [Flo2000, 79].

5.7 TIPOS DE DATOS DEVUELTOS POR EL DOM

Siempre que se utilice un método DOM se devolverá un tipo de dato o de objeto, los principales que son devueltos se pueden resumir en la tabla 5-2:

Tabla 5-2: Datos y tipos de Objetos usados en el DOM.

Tipo de dato	Descripción
unsigned short	Todos los tipos de nodos son enteros cortos.
unsigned long	La longitud de una lista de nodos o el número de elementos que contiene es un entero largo.
NodeList	Objeto que corresponde a una lista indexada de objetos de nodo.
Node	Un objeto que contiene los detalles del nodo.
NamedNodeMap	Un objeto que es una lista no indexada de objetos de nodo.
DOMString	Cadena de caracteres.
Boolean	Verdadero o falso
Void	Métodos que no devuelven valor.

5.8 SOAP

SOAP, Protocolo Simple de Acceso de Objetos (Simple Object Access Protocol) es un protocolo para el intercambio de información en un entorno descentralizado y distribuido, como ejemplo, se podría usar en aplicaciones de plataformas de integración de medios de pagos u entornos computacionales típicos, que presenten una plataforma distribuida [Jac2001]. Es un protocolo basado en XML que consiste de tres partes típicas, según [URL 40]:

- Una envoltura que establece un punto de trabajo, usado para describir el contenido de un mensaje y la forma de procesarlo.
- Un grupo de reglas de codificación, usadas para expresar las instancias o métodos de tipos de datos definidos por la aplicación.
- Una convención usada para representar las llamadas y las respuestas a procedimientos remotos.

Uno de los objetivos principales de SOAP es la simplicidad y la capacidad de extensión. Existen por lo tanto, varias características de los sistemas de mensajería tradicionales y de objetos distribuidos que no forman parte de la especificación principal de SOAP. Entre estas características, se incluyen según [URL 40]:

- Recolección de elementos no utilizados.
- Carga o procesamiento por lotes de mensajes.
- Objetos por referencia (que precisa recolección de elementos no utilizados.)
- Activación (que precisa objetos por referencia.)

Principalmente, los mensajes SOAP son transmisiones unidireccionales desde un emisor a un receptor y además se suele combinarlos para implementar patrones, como ***solicitud-respuesta***. El DOM también es aplicable a SOAP, ya que SOAP de por sí es un documento XML [URL 40].

El mensaje SOAP es un documento XML que contiene:

- **Envoltura obligatoria:** **<SOAP-ENV:Header> </SOAP-ENV:Header>**.
- **Encabezado opcional:** **Header**, puede ser un nombre nemotécnico.
- **Cuerpo obligatorio:** utilizando una sintaxis similar a la de XML.

Este protocolo es bastante usado en aplicaciones de solicitudes de respuestas en tiempo real y de comunicación entre plataformas, como es en el caso del proyecto Educar Chile, del Ministerio de Educación.

5.9 CONCLUSIONES

Actualmente los motores DOM implementados con mayor éxito son los implementados por Microsoft y por IBM. El primero, utiliza Internet Explorer 5.0, está disponible en la última **dll** (Aplicación de Objeto compilado) llamada MSXML.dll (actualmente en la versión 4.0) y como objeto Active X; mientras que en IBM los analizadores en el lenguaje Java y C soportan de forma perfecta el DOM.

Con este capítulo se intentó dar un enfoque general del DOM, necesario para comprender la lógica que hace que un documento XML pueda ser compatible con múltiples plataformas, lo cual le da la característica de ser el mejor estándar que hoy existe para trabajar en Internet. Se ha abordado de una forma general, sin entrar en profundidad en el tema, para poder dar paso así a futuras investigaciones que se puedan generar con este apartado de tema. Además se abordó en forma general el concepto del protocolo de comunicación SOAP, necesario para entender el enlace con otras plataformas computacionales distribuidas, al cual también se puede aplicar el DOM. El interés principal fue dar a conocer el DOM y en que consiste, para así dar paso al siguiente capítulo y poder tener una mejor comprensión, cuando se aborden conceptos más detallados en XML.

Como conclusión principal de este capítulo se puede establecer que el nivel 1 del DOM propuesto por el W3C es el que permite manipular un documento XML, y de esta forma además poder utilizar XML como un almacén de datos, lo cual lleva a pensar en bases de datos creadas con documentos XML bien formados, ahorrando un costo de licencias de software muy grande a las empresas hoy en día.

A continuación se abordará el tema de XML en la Web y posteriormente las aplicaciones que se pueden destinar a la administración y publicación de contenidos en tiempo real, utilizando como base XML y XSL, en el despliegue de la información.

6. XML EN LA WEB

6.1 INTRODUCCION

En la actualidad, la Web sólo es posible de ver a través de un navegador como Internet Explorer o Netscape Navigator. XML resuelve muchos aspectos de HTML y los documentos XML bien formados son tan importantes como los navegadores donde pueden ser desplegados.

Si no existiera un navegador sólido, XML no sería más que una tecnología que no llegaría a ningún destino. Actualmente uno de los retos que se enfrenta en North Supply Chile división e-Business (NSCH), es la incompatibilidad de los navegadores Web, ya que el soporte que ofrecen los navegadores a XML es muy cuestionable.

Microsoft Internet Explorer es el único navegador comercial que soporta XML, y dicho soporte dista mucho de ser el mejor. Netscape, por su parte con su versión 6.0 y su motor de diseño Gecko tiene un soporte sólido para XML, sin embargo no ofrece una confiabilidad completa.

Para que XML tenga éxito, es necesario que los fabricantes de navegadores web se puedan complementar con los estándares propuestos por la W3C.

El problema del despliegue de XML por medio de XSL en un navegador web se resuelve de una sola forma hasta el momento, y es el camino adoptado, incluso para NSCH, que consiste en utilizar XSLT bien formado, para que la salida HTML de despliegue en un navegador sea compatible con todos los navegadores.

Sin embargo existen otros problemas, considerando que un web consta de applets, javascript, programación asp, entre otras, lo cual hace que los resultados entre distintas plataformas, por mencionar algunas: Apple, Microsoft o Linux sean prácticamente distintos; para lo cual se verá en el desarrollo posterior cual es la mejor alternativa de solución actual , para trabajar con XML y XSL.

Se analizará en este capítulo la relación entre XML y el estado con los navegadores actuales, en qué consisten las llamadas aplicaciones de Internet de tiempo real y las de

administración de contenidos en tiempo real; se definirá el concepto de XHTML, y cómo poder manipular datos con XML, y acceder a datos remotos.

6.2 APLICACIONES DE TIEMPO REAL

Basándose en algunos productos de NSCH, como es el caso de PubliXpress® (herramienta de administración de contenidos en tiempo real), se puede deducir que una ***aplicación de tiempo real*** puede definirse de la siguiente forma: Toda aplicación que necesite un resultado visible para el usuario, en un intervalo de tiempo crítico mínimo y que se pueda obtener de una forma segura y confiable.

En un "tiempo crítico acotado", significa en la realidad que se desea obtener un resultado instantáneo, sin embargo en las aplicaciones computacionales, el concepto de tiempo real va de la mano con una suerte de dualidad que altera la definición:

- Las aplicaciones computacionales nunca se transmiten en un tiempo exacto, para catalogarlas de "tiempo real".
- Las aplicaciones de tiempo real (verificar estado del tiempo, valores de UF, Dólar, aplicaciones de pago y transacciones con tarjetas de crédito, aplicaciones de voz, de multimedia, entre otras), requieren un intervalo de retraso mínimo, que a veces suele tomar un par de minutos, lo cual hace que el concepto en sí, no sea tan aplicable.

Una *aplicación de administración de contenidos en tiempo real* puede definirse como: Toda aplicación que integre una administración de algún tipo de contenido, que pueda clasificarse según un criterio o rol determinado y que el resultado pueda reflejarse instantáneamente en una aplicación final para un usuario concreto.

Se denomina *contenido* a algún tipo concreto de información que se pueda administrar:

- Texto.
- Imagen.
- Videos en algún formato (mpeg, avi, etc.).
- Archivos en algún formarto (.doc, .pdf, .txt, etc.).
- Etc.

La segunda definición es la más acertada para las aplicaciones y arquitecturas de solución que se abordarán en el capítulo 9 que serán en definitiva las nuevas posibilidades de aplicación de negocios para North Supply Chile División e-Business.

Se puede ver en la figura 6-1 un esquema que hace la diferencia entre estos dos conceptos planteados:

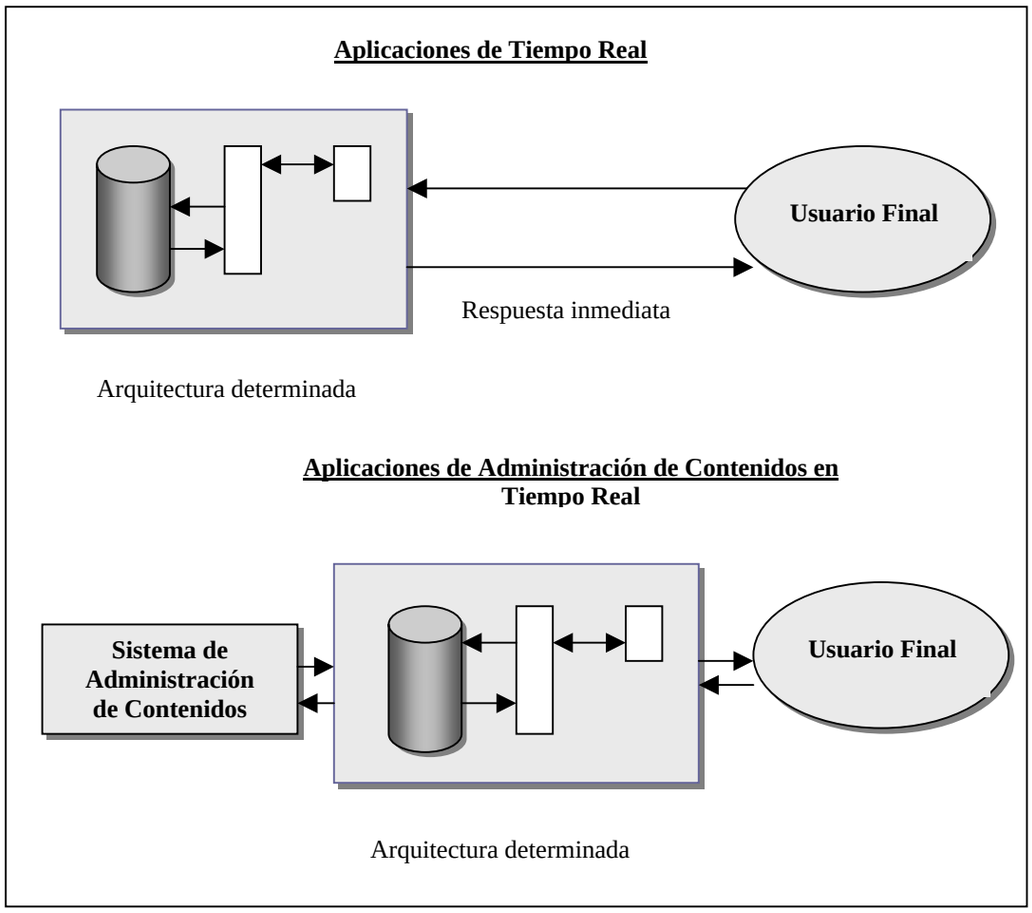

Figura 6-1: Aplicaciones de Tiempo Real versus Aplicaciones de Administración de Contenidos en Tiempo Real.

Para estos dos tipos de aplicaciones existen los mismos problemas de incompatibilidad de navegadores web antes planteados.

A continuación se verán algunas opciones de navegadores que soportan XML como solución al problema de incompatibilidad y se verán algunas de las más importantes características de cada uno.

6.3 NAVEGAR CON XML

En el mercado Chileno, XML poco a poco se está situando como una de las tecnologías web más recientes y atractivas por una razón particular: XML es una herramienta poderosa. Sin embargo, el soporte de navegador web que hay actualmente es muy débil.

Si se considera cuanto tiempo lleva existiendo la especificación XML 1.0, desde 1998 a la fecha, parece algo poco claro que los distintos navegadores web sigan dando poco apoyo al tema.

Microsoft llegó a tiempo con Internet Explorer 5.0, pero lo ha hecho muy mal en lo relacionado con **aplicar** los estándares de XML.

Para que XML tenga éxito como estándar en Internet, es obligatorio que los fabricantes de navegadores se adhieran a los estándares XML del W3C, ya que XML constituye una posición interesante sobretodo para las aplicaciones de negocio en el mercado.

Los navegadores web que soportan XML de una forma correcta son los siguientes:

- Microsoft Internet Explorer.
- Netscape Navigator (Mozilla).
- CITEC DocZilla.
- W3C Amaya.

6.3.1 XML Y EL ESTADO DE LOS NAVEGADORES WEB

Microsoft Internet Explorer: Microsoft fue la primera empresa que lanzó un navegador comercial con soporte para XML, Internet Explorer 5.0 (actualmente en la versión 6.0 del navegador) [Flo2000, 89]. El problema principal con el soporte que da Internet Explorer a XML radica en la solución original planteada por Microsoft al

tratamiento de ciertos temas XML. Cuando se lanzó Internet Explorer (IE), XSL todavía estaba en una etapa de borrador para el W3C. Microsoft siguió dando soporte a una versión de XSL modificada, sin ocuparse de los efectos de permitir que los desarrolladores basaran el código en una tecnología que todavía no era un estándar oficial [Mor2000, 347].

Una de las características de Internet Explorer es que se puede ver código XML directamente en el navegador, figura 4-2. Cuando el documento XML no tiene una hoja de estilos asociada, Internet Explorer presenta un documento en forma de árbol.

El árbol contiene signos de **menos** y **más** junto a cada rama y que permite además expandir y contraer las porciones del árbol, en distintos colores que permiten hacer una diferencia entre las etiquetas XML, los datos y las instrucciones del procesador.

Internet Explorer es una buena herramienta a la hora de detectar errores en los documentos XML, que se traduce en el despliegue del error si el documento XML no consta de una DTD, XML Schema, un archivo específico, o no se escribe bien una etiqueta de inicio o fin en un documento XML, en este aspecto IE es bastante exigente.

Uno de los puntos débiles de IE además es el que tiene relación con las CSS; Microsoft excluyó parte de las funcionalidades de CSS lo cual aseguró la incompatibilidad con otros navegadores, proporcionando extensiones propias. Las CSS no son completamente compatibles con el estándar de la W3C.

Netscape Navigator: El futuro de Netscape Navigator se basa en el movimiento open source (código abierto) llamado Mozilla [Mor2000, 352].

En enero de 1998, Netscape declaró que serían gratuitas las versiones futuras de Netscape Navigator, por lo cual se lanzó una versión de desarrollo de Navigator que incluía código fuente completo, del cual nació Mozilla, una herramienta de Netscape que contempla el desarrollo open source colectivo para Navigator. Gecko corresponde al nombre del código del motor de presentación que utiliza el navegador Mozilla/Navigator.

La versión actual, 6.0 del Netscape Navigator presenta características de soporte para XML [URL 39]. Netscape ha tomado ventaja sobre Microsoft al no incluir un soporte para XSL en Mozilla; en vez de esta situación, Netscape se quedó con el estándar CSS que es suficiente para dar estilo a los documentos XML, dejando que XSL sea más profundizado por el W3C.

CITEC DocZilla: Corresponde al primer producto no comercial de Netscape que integra toda la iniciativa de open source de Mozilla. La empresa CITEC tiene experiencia en el desarrollo de las herramientas de navegador como documentación técnica. Su navegador Multidoc Pro SGML es considerado el navegador SGML más rápido que existe hasta el día de hoy. DocZilla es el sucesor de CITEC a Multidoc Pro. Tiene soporte SGML y XML entre otros [Mor2000, 353].

W3C Amaya: Amaya es una herramienta de navegación, con fines de prueba, creada por el W3C para apoyar la demostración y pruebas de nuevos protocolos y de formatos web [Mor2000, 355].

Opera: Opera corresponde a un navegador que aún no soporta estándar XML [Mor2000, 356], sin embargo tiene una implementación estable de CSS, lo que le da el apoyo suficiente para dar soporte XML. Actualmente está en la versión para Windows 5.12. También incluye versiones del navegador para dispositivos portátiles y diversas plataformas computacionales como MAC, OS2, Linux/Solaris, Symbian OS.

6.3.2 XHTML

Según [Mor2000, 358], una de las principales causas de confusión entre los principiantes de XML es la relación entre XML y HTML.

- XML es un metalenguaje.
- HTML no es un metalenguaje, es una aplicación específica de SGML.

El W3C está tendiendo a dar una mejor estructuración al HTML con el denominado XHTML , una versión de HTML que se ha reformulado como aplicación XML. XHTML tiene similitud al HTML 4.0, que representa el futuro de HTML.

En la actualidad la Web es un caos de código HTML con muy poca estructura, presentando modelos de negocios a veces incorrectos debido a la falta de estructuración de la información que se presenta.

La respuesta que se podría generar como solución es convertir HTML a XML con hojas de estilos para mostrarlos, en este caso XSL utilizando la transformación XSLT. El problema es que HTML está bastante integrado para hacerlo a un lado. Además aunque XML y XSL sugieren una ventaja estructural enorme en comparación con la típica página web HTML en cuanto a su presentación, implican mucho más trabajo. En este caso hay que diferenciar siempre las situaciones en las que no importa si el contenido está o no separado de cómo se muestra, en cuyo caso HTML representa una solución más eficaz.

Al expresar HTML como aplicación XML (XHTML) se pueden adoptar mejoras con poco esfuerzo en la implementación; los navegadores así deberían exigir un esquema HTML para dar pie a que los documentos estén bien construidos y que sean válidos.

Según el borrador de XHTML del W3C planteado en [Mor2000, 359]:

- Los documentos XHTML se basan en la sintaxis XML, lo que significa que se pueden editar y validar con herramientas XML estándares.
- Los documentos XHTML pueden servir como tipos de medios text/html.
- Los documentos XHTML pueden servir como tipos de medios text/xml.
- Los documentos XHTML pueden servir como tipos de medios application/xml (aplicación/xml), con un soporte adecuado para las hojas de estilos.
- Los documentos XHTML pueden utilizar tecnologías de programación, los cuales ocupan el Modelo de Objeto de Documento HTML o el Modelo de Objetos de Documento XML.

- Todo documento que se adapte a XHTML podrá operar de una mejor manera con los distintos entornos XHTML.

Las características y reglas principales para construir documentos XHTML son las siguientes:

1. Los documentos deben estar bien construidos.
2. Los nombres de los atributos y los elementos deben ir con minúsculas.
3. Las etiquetas de cierre son necesarias para los elementos no vacíos.
4. Los elementos vacíos deben estar formados por un par de etiquetas de inicio-cierre o de un elemento vacío.
5. Los valores de los atributos deben ir entre comillas.
6. Los nombres de los atributos no pueden minimizarse (utilizar sin un valor).
7. Un espacio de nombres XHTML debe declararse en el elemento html.
8. Los elementos head y body no se deben omitir.
9. El elemento title debe ser el primer elemento del encabezado head.
10. Todos los elementos script y style deben ir dentro de secciones CDATA.
11. Los documentos deben utilizar el atributo id para definir identificadores de fragmentos.

La figura 6-2 muestra el uso de XHTML para producir un Web Site de clasificación de productos frutales de exportación, como un ejemplo de implementación XHTML. La página XHTML fue construida con el editor Front Page 2000 de Microsoft y tiene como objetivo demostrar la estructuración de HTML, utilizando las reglas de XML. Podría resultar bastante interesante para una empresa poder comenzar a trabajar en código XHTML para dar una mejor estructuración a sus páginas. La gran mayoría de las empresas pequeñas y medianas prefieren una Web plana sólo con el objeto de dar a conocer su rubro, donde no existe una interactividad con algún sistema de información. Sin embargo sería recomendable poder aplicar el concepto de XHTML en vez de HTML

por una simple razón: "Mantener un orden estructurado de la información que se maneja".

El código fuente está incluido en el anexo correspondiente al capítulo 6, anexo 6-1 y la información respectiva de salida está definida en la siguiente lista:

- **Frutas**
- **Precios de Exportación por Unidad**
- **Unidades**

Navel Oranges - Naranjas tradicionales

US$2.49

pound -Kilo

Strawberries - Fresas

- US$2.99
- quart - cuarto

Limes - Limones

- US$1.49
- pound-Kilo

Granny Smith Apples - Manzanas Granny Smith

- US$1.99
- pound-Kilo

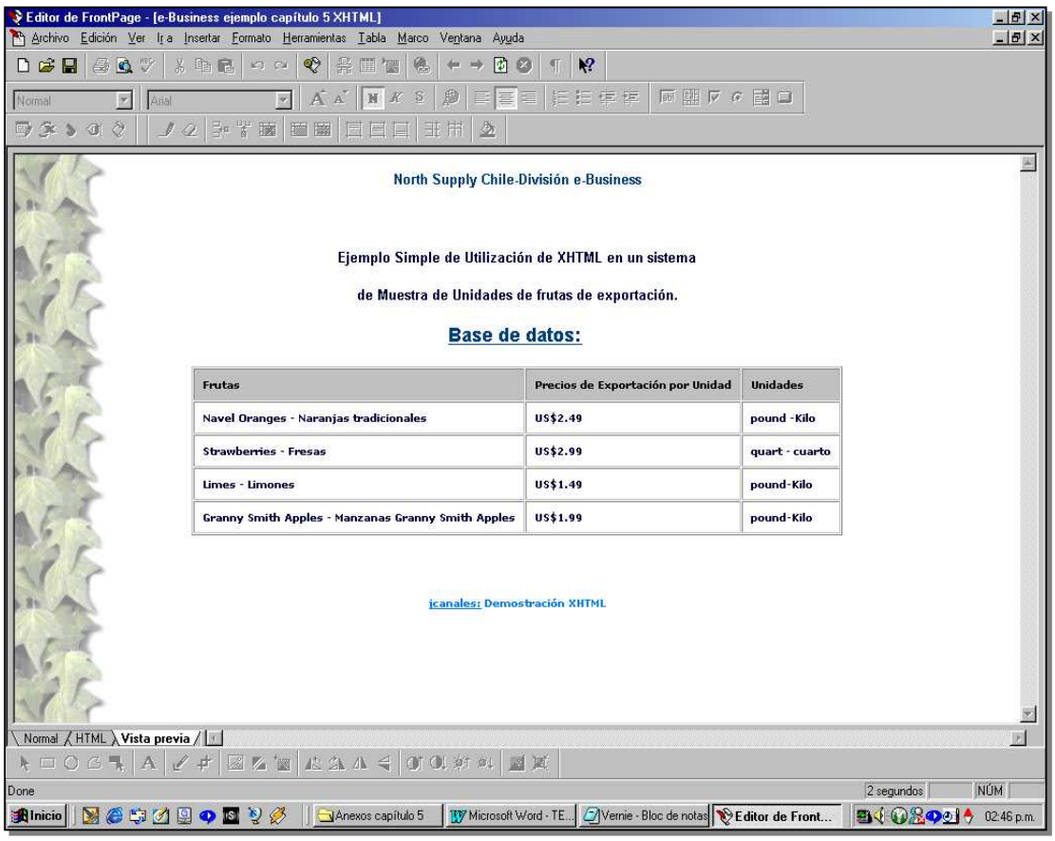

Figura 6-2: Una implementación de XHTML.

Se puede demostrar con este ejemplo que es posible trabajar al más alto nivel en la construcción de sitios web de una forma simple, incluso para la aplicación de negocios, tomando el concepto de **XHTML** y estructurando de una forma adecuada la información. En este caso no se aplica una DTD ni un XML Schema, y sirve para ejemplificar el concepto de estructuración y mejora del HTML.

Como conclusión directa se puede afirmar que XHTML es un lenguaje que sirve para dos propósitos:

- **Estructurar de mejor forma un documento HTML.**
- **Dar un formato de salida, tal como una hoja de estilo, agrupando de una mejor forma la información.**

6.4 MANIPULACION DE DATOS CON XML

6.4.1 XQL

XQL (XML Query Language) es una notación para manejar los elementos y texto de los documentos XML. XQL (o SQLX) es una extensión natural del sistema de patrones XSL [Mar2000, 175].

Según [Mar2000] es un lenguaje declarativo, con el que uno sólo tiene que describir los tipos de nodos que busca, usando una notación de tipo "directorio" (por ejemplo, seccion/recursos_humanos representa una búsqueda de elementos de documento de tipo " recursos_humanos " contenidos en elementos de tipo "seccion").

XQL extiende las posibilidades de XSL, permitiéndonos identificar clases de nodos añadiendo lógica booleana, filtros e indexado de los nodos.

XQL está diseñado para ser usado en diversos contextos: al ser una extensión de XSL, se puede aplicar para linkear nodos, buscar en repositorios, y para muchas otras aplicaciones, aunque XSL si bien es concebido para presentar el documento, también posee elementos que llevarían a considerarlo un lenguaje de consultas. Para evitar esta confusión se denominó XQL al lenguaje de consultas que permite hacer esto dentro de un documento XML [Mar2000, 188].

XQL es una notación para obtener información de un documento. La información puede ser un conjunto de nodos, información sobre las relaciones entre nodos, o valores derivados.

Dependiendo de la implementación, el resultado de una consulta será un documento XML o un árbol. En otros casos, podría ser una estructura diferente, tal como un conjunto de punteros a nodos.

Por lo tanto, si se quiere pensar en almacenar bases de datos sin necesidad de una licencia de software cara, como es el caso de SQL, se puede concluir que XML es un poderoso almacén estructurado de información. Tal vez no al más grande nivel, en lo que respecta a vistas, y todas las posibilidades que presenta una herramienta de base de

datos; pero si, a un nivel que perfectamente puede emplearse en aplicaciones específicas para el manejo de la información.

La analogía que se hace entre XQL y XSL es muy buena, en el sentido que XSL de por sí también es un lenguaje de consulta, y se puede pensar que la diferencia es solo de término, porque ambas en concreto, realizan consultas sobre un documento XML; con la diferencia que XSL está encargada de la capa de presentación del documento; vale decir, del cómo se desplegará en un navegador; muy distinto a XQL , que como resultado al usuario, despliega los valores de XML.

XQL como concepto, daría paso para otra investigación, que dejaría abierta la posibilidad de otra Tesis dedicada al tema de ahorro de licencias de softwares de bases de datos con XQL y XML.

6.5 CONCLUSIONES

Se presentó en este capítulo un recorrido al estado actual de XML y la Web, donde se pudo hacer notar que el estado de los navegadores, aún deja mucho que desear, ya que cada uno está apostando por sus propios estándares y no se basan en definitiva en los "*estándares propuestos"* por la W3C.

Se explica el concepto de aplicaciones en tiempo real y aplicaciones de administración de contenidos en tiempo real, los cuales son importantes de diferenciar a la hora de definir una aplicación sobre Internet y cuáles parámetros serán usados para poder realizar una arquitectura clave en un proyecto determinado.

Se presentó XHTML, una mejora al HTML, que presenta un alto nivel de estructuramiento y presenta una mejor disposición de agrupamiento de la información, a diferencia de HTML; por lo que se puede establecer que sería un buen punto migrar el código HTML a XHTML, sin embargo es una tarea imposible de realizar, porque Internet está lleno de HTML. Se propone a este problema, poder utilizar XHTML cada

vez que se requiera simple HTML, con lo que se gana en estructuración de la información y el código se hace inmediatamente portable con XML.

Se mostró en un rango general el concepto de XQL, estableciéndose que una combinación de XML con XQL podría ayudar bastante en aplicaciones de bases de datos y ahorrar una buena cantidad en licenciamiento de software. Las licencias de bases de datos en la actualidad son bastante caras por ejemplo para las PYMES (Pequeñas y Medianas Empresas), con XQL, en cambio, se puede tener un buen nivel de trabajo en administración de bases de datos, tal como lo haría un DBMS (Data Base Management System, Sistema de Administración de Base de Datos).

En resumen este capítulo fue realizado en etapas claves:

- Presentar el estado actual de XML en la Web.
- Presentar la diferencia entre aplicaciones de tiempo real y de administración de contenidos en tiempo real.
- Presentar la mejora de HTML en Internet (XHTML)
- Presentar XQL a diferencia de XSL que transforma a XML con XQL en una excelente alternativa para trabajar en una solución de base de datos.

Estos puntos ayudarán mucho a la hora de poder definir un modelo de negocios, basándose en XML y XSL como alternativa de solución a implementar, ya que da una idea real del estado en que podrán ser aplicados y de las limitantes que se presentan.

En el siguiente capítulo se verá como es posible aplicar negocios en Internet utilizando XML y XSL.

7. GENERACION DE NEGOCIOS EN INTERNET CON XML Y XSL

7.1 INTRODUCCIÓN

Basándose en [Flo2000, 209], en la actualidad, al investigar el mercado de los servidores Web, se pueden dividir en dos grandes campos:

- Apache sobre Unix
- Internet Microsoft Server (IIS)

Cuando se trata de una intranet, se piensa en servidores Microsoft, en la gran mayoría, ello se debe fundamentalmente a las numerosas tecnologías que pone Microsoft, tales como: COM, ActiveX, Scriplet, conectividad ODBC (Object Data Base Connect, Conexión a Base de Datos por Objetos), son las que dominan en la mayoría de las ocasiones, donde Microsoft integra estas herramientas en todos sus productos.

En el núcleo de Microsoft se encuentran las páginas ASP, Páginas de Servidor Activo, que permiten a los desarrolladores generar páginas HTML dinámicamente desde el servidor. Una página ASP puede conectarse directamente con una base de datos, por ejemplo la conectividad ODBC o conectarse mediante componentes Active X.

De esta forma las páginas ASP constituyen un gran punto de inicio para el desarrollo de aplicaciones y negocios en la Web.

En este capítulo se propone como mejor alternativa de solución de negocios y construcción de aplicaciones para la Web, la combinación entre ASP, XML y XSL (para representar el formato de salida utilizando XSLT), en la actualidad.

Eventualmente, existen otras combinaciones de tecnologías, que no dejan de ser excelentes a la hora de medir el performance y el desarrollo involucrado; sin embargo ASP es una tecnología que hace un año atrás, en Chile, se atrevía a apostar que desaparecería de la Web, al aparecer nuevas tendencias de lenguajes, como C#, VB .Net, ASP .NET, .NET Framework, etc., propuestas por su mismo creador, Microsoft [And2001], lo cual, hasta el día de hoy, **no ha ocurrido**.

En Chile se puede distinguir que el campo de aplicaciones con ASP aún tiene para bastante tiempo, debido a que hoy en día las aplicaciones "a la medida" que se requieren, son siempre aplicables a una tecnología conocida, como ASP, que ha resultado en un importante éxito anterior en la Web.

La propuesta de utilizar ASP, es la de poder hacer una combinación con XML y XSL, que anteriormente no se utilizaba, para así otorgar una mejor solución que cuando se utilizaba ASP con HTML puro. No es posible hacer hoy una migración de aplicaciones **ASP-HTML** a **ASP-XML-XSL**, en la Web, debido a que sería un trabajo muy complejo (la Web está llena de contenido en HTML y ASP).

Al proponer una combinación ASP, XML y XSL, se gana en una mejor **estructuración de la información** y en un excelente **modelamiento de la información** y **comunicación entre las aplicaciones**, motivos suficientes para optar por dicho camino [Sel2001]. En el capítulo 9, se plantearán las nuevas tendencias de aplicaciones con XML y XSL para poder generar nuevos campos de aplicación con estas dos tecnologías, donde se podrá tener una visión a futuro de las posibles nuevas generaciones de aplicaciones que se vendrán venir en la Web y esperando que, para North Supply División e-Business, signifique una información altamente estratégica a futuro, que permita abrir el campo de posibilidades de incorporación de negocios.

Otro punto importante en este capítulo, es que ASP es una tecnología que cuenta con soporte y respaldo de una empresa (Microsoft), a diferencia de su real competencia en el mundo Linux y Open Source, como PHP (Hypertext Preprocessor), donde las aplicaciones recién están comenzando a tener un giro comercial, por un tema de ahorro en licenciamiento de software; pero que en definitiva no proporciona un nivel de seguridad y respaldo de las aplicaciones que se desarrollan, lo cual puede verse reflejado al trabajar proyectos dentro de NSCH; se prefiere contar con una tecnología que brinde respaldo y soporte.

Se podría exponer un caso particular, para ejemplificar lo anterior: En caso de que una aplicación desarrollada en Linux literalmente "se caiga" (deje de funcionar, en términos informáticos), no existe hoy en Chile una empresa sólida y consolidada, que brinde el soporte o la garantía de sus aplicaciones bajo esta plataforma; a lo más, se podría tener un respaldo en documentación o en apoyo de programación neta.

Lamentablemente la propuesta de aplicaciones en Linux, donde también se puede combinar XML con otras tecnologías, todavía necesita un período de crecimiento real en Chile. Sin embargo, la propuesta de Linux en el País, es en el campo de aplicaciones en materia de investigación; lo cual corresponde a un excelente inicio para esta plataforma y a nivel universitario, donde se requieren de más investigaciones al respecto.

La gran mayoría de las aplicaciones se ejecutan en servidores con tecnología Microsoft a la hora de generar negocios sobre Internet; son muy pocas las empresas que se atreven a apostar por plataformas Linux únicamente por el respaldo de una empresa de por medio; es tentador y resulta bastante conveniente una plataforma totalmente gratuita, donde se ahorrarían millones en tema de licencias; pero en aplicaciones reales, aún la plataforma Microsoft representa la mayoría que predomina y que se conoce, para gusto de algunos o discrepancia de otros.

Básicamente, una página ASP que reside en un servidor recibe una solicitud de un cliente , procesa esa solicitud, accede a un posible almacén de datos y devuelve un flujo HTML (o XML) al servidor. Esto lo hace por medio de una serie de objetos integrados en el motor ASP.

El documento ASP se diferencia del documento HTML por el sufijo **.asp**. Cuando el navegador (que admite ASP) recibe una solicitud de una página con un sufijo .htm o .html seguirá las siguientes etapas:

- Buscará la página en la dirección que se especifica.
- Cuando encuentra la página, se envía al cliente con el encabezado http apropiado.

- Si hay problemas, se genera un mensaje de error.

Como se puede notar, ASP es una tecnología exitosa y confiable, la propuesta es implementarla en conjunto con dos tecnologías que permiten plantear soluciones mejor estructuradas y de un excelente nivel de organización y presentación de la información.

A continuación se expone la implementación que representa una mejor solución para aplicación de negocio en Internet, con XML y XSL.

7.2 XML Y ASP COMO UNA BUENA ALTERNATIVA DE SOLUCION

XML actúa como el ASCII de Internet [URL 13], permitiendo la interacción entre servidores sin importar el sistema operativo que se considere, los administradores de bases de datos o formatos de datos. Al añadir ASP y el Lenguaje de Hoja de Estilo Extensible se pueden escribir aplicaciones de red impresionantes sin siquiera tomar en cuenta a Java, C++, sockets u otras complejas herramientas que existen actualmente.

ODBC es una tecnología de Microsoft diseñada para crear un conjunto común de métodos y rutinas para acceder a almacenes de datos. En gran medida la sustituye OLE-DB (Base de datos - Incrustación y Vinculación de Objetos), que es mucho más rápido y fácil de utilizar. Los Objetos de Datos Active X (ADO) son la interfaz que se sitúa en la parte superior de OLE-DB, y proporciona métodos y propiedades que se usan para realizar una conexión y manipulación de la información. La figura 7-1 muestra la arquitectura de la tecnología OLE-DB en relación a ASP.

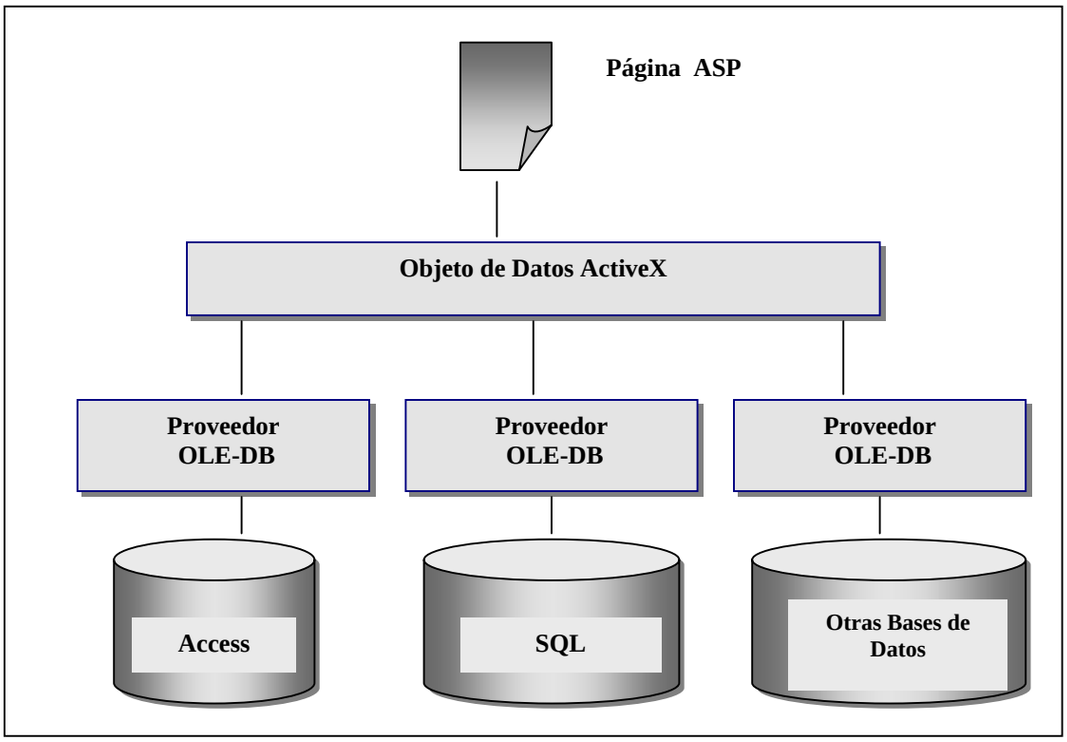

Figura 7-1: Arquitectura de la tecnología OLE-DB

Las bases de datos tienen motores que les permiten buscar, filtrar y cotejar los registros en la memoria. Por ejemplo, al hacer una búsqueda sencilla, un motor de base de datos empieza por lo general con el primer registro y busca por todos los registros en orden, probando el criterio de la solicitud de búsqueda. En un documento XML es fácil también hacer una búsqueda similar. Al usar las ASP o un motor similar, XML puede compartir las mismas virtudes.

Actualmente existen cuatro usos principales de XML en la administración de las bases de datos:

- Duplicar y archivar un almacén de datos.
- Como paquete para enviar datos entre distintas bases de datos y almacenes de datos.
- Como paquetes para mostrar información.
- Como almacenaje de información.

XML es útil para pasar paquetes de información de una base de datos en Internet o en una intranet. Se puede convertir en un reducido número de recordsets (conjunto de registros) a XML y pasarlos a Objeto de Origen de Datos (DSO) en el lado del servidor para la manipulación. Se puede también pasar información de una base de datos ORACLE de un servidor A a una base de datos en SQL de un servidor B, mediante una arquitectura de alta disponibilidad [Lip2001].

Ya que se puede utilizar XML como almacén de datos, existen por lo tanto dos formas de poder recuperar un documento XML:

- Como archivo sin formato de flujo de texto.
- Como objeto DOM XML

Una arquitectura ideal para la aplicación de negocios en Internet utilizando ASP, XML y XSL es la representada en la figura 7-2, basado en [URL 13], donde se agrupan:

- Configuración de aplicaciones multinivel.
- ASPs empleadas como una aplicación de nivel de servicios de negocios (la mayoría de las aplicaciones multinivel utilizan componentes COM en este nivel intermedio).
- Peticiones (*queries*) y obtención de datos en tiempo real desde varias fuentes, en vez de mantener una base de datos centralizada y posiblemente no actualizada.
- Obtención desde bases de datos utilizando un motor ADO.
- Utilización de XML para estandarizar datos de diversas fuentes.
- Conversión manual de XML a HTML por medio de XSLT.
- Manipulación del Modelo de Objeto Documento XML (*Document Object Model*, DOM) con el lenguaje JScript®.
- Formateo de XML utilizando XSL en el cliente y en el servidor.

Esta arquitectura es ideal para una combinación de ASP, XML y XSL cuando se trabaja en aplicaciones de búsqueda con agentes inteligentes sobre Internet.

La solución de arquitectura presentada muestra como el XML y las ASP se pueden combinar para crear poderosas aplicaciones multinivel. Además, el uso de scripts en el lado del servidor con ASP simplifica aún más tales soluciones, dado que no se necesitan lenguajes adicionales (tales como el lenguaje Java) en ningún punto.

La mejor solución para enfrentar una buena alternativa de combinación con ASP, XML y XSL es presentar siempre tres capas de modelado:

- **Capa de Almacenamiento de datos:** XML, Bases de Datos.
- **Capa de Negocio:** Componentes COM, accesar mediante ASP
- **Capa de Aplicación y Presentación**: XSL o WML (Wireless Markup Language, Lenguaje de Marcado Inalámbrico, utilizado para aplicaciones de protocolo WAP en telefonía).

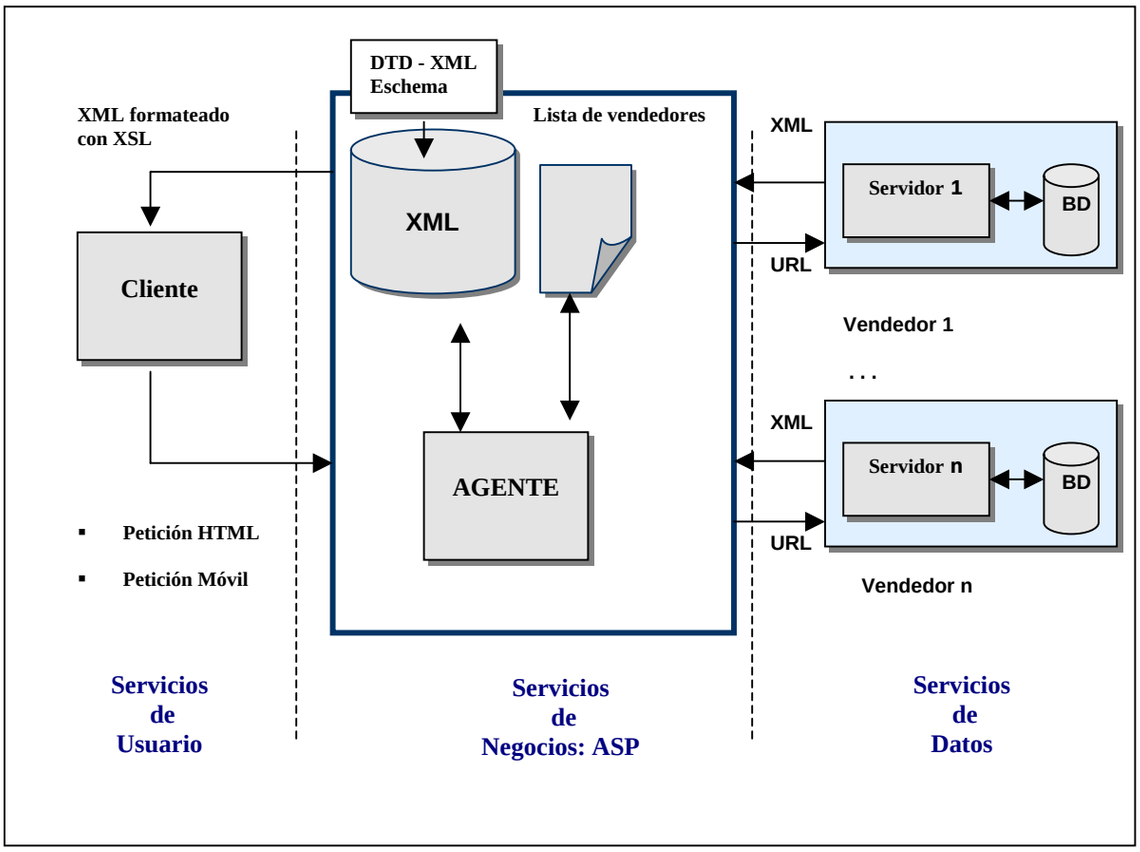

Figura 7-2: Una Arquitectura de aplicación ideal para combinar un nivel de negocio con ASP, XML y XSL.

En el Anexo 7-1 se pueden ver segmentos de código que utilizan combinaciones de XML con ASP y a su vez código XSL que hace referencia a este XML. Se incluye además una sección de código del portal educativo **Educar Chile**, para ilustrar de mejor forma el trabajo entre ASP, XML y XSL.

7.3 CONCLUSIONES

En este capítulo se ha visto que XML hace una buena combinación con las bases de datos, en materia de almacenamiento estructurado de la información.

Se ha presentado y expuesto que la mejor alternativa para la aplicación de negocios sobre Internet en la actualidad es haciendo una combinación de XML, ASP y XSL, debido a que se pueden distinguir y clasificar perfectamente tres secciones o capas de aplicación, que son **necesarias** para implementar una buena arquitectura de solución, que sea confiable y robusta.

Se expuso en la introducción, que la mayoría de las aplicaciones de negocios en Chile aún son predominadas por las herramientas Microsoft, por lo cual el éxito de las páginas de servidor activo, ASP proporciona una tecnología confiable, exitosa y comprobada, que conviene combinar con XML y XSL para una mejor estructuración de la información. Corresponde a una tecnología exitosa en implementación; sin embargo, cabe mencionar que en una PYME (Pequeña y Mediana Empresa) el ahorro de licencias de Software es significativo, si el negocio se implementa bajo plataforma Linux, lo cual no deja de ser una buena alternativa en empresas pequeñas.

Con este capítulo se pretende conocer y saber que a la hora de aplicar un gran negocio, utilizando XML y XSL, es recomendable hacerlo sobre el lenguaje ASP y el servidor Web Microsoft IIS, Internet Information Server, Servidor de Información de Internet, para obtener un resultado con probabilidad de éxito. El éxito a demostrar, se basará en la *Calidad del Servicio Final* que se proporcione al Cliente, independiente de la plataforma

a utilizar, aunque es recomendable contar con una tecnología nueva (XML y XSL), que se apoye en una conocida (ASP).

Se menciona además que existen otras arquitecturas que son excelentes para trabajar con XML y XSL, sin embargo a la hora de prestar un respaldo y soporte, no se encuentra nada al respecto. Se espera que en Chile se avance en materia de aplicaciones bajo plataformas OpenSource (Código Abierto).

Se expondrán en el capítulo siguiente las mejores herramientas existentes para trabajar con XML y XSL, las cuales se deberán identificar de acuerdo a una clasificación de herramientas para cierto tipo de aplicaciones: *Herramientas de Creación y Herramientas de Administración.*

8. ANÁLISIS DE HERRAMIENTAS XML Y XSL

8.1 INTRODUCCIÓN

Las herramientas XML han sido en gran parte las responsables de lo rápido que XML ha pasado a formar parte del incremento de los desarrolladores de la Web.

En este capítulo se darán a conocer las mejores herramientas disponibles para trabajar con XML y a la vez con XSL. Este punto es importante y de interés para North Supply Chile División e-Business y en general para cualquier empresa que desee obtener una información acerca de las herramientas de software con las que se cuenta en la actualidad, para poder construir archivos XML y además diseñar plantillas XSL.

XML al ser un descendiente de SGML, el que a su vez es el padre de HTML, permite deducir que se cuenta con numerosas herramientas de software para el desarrollo.

Se analizarán las principales herramientas existentes y se hará una clasificación de ellas en dos grandes campos:

- **Herramientas de creación XML y XSL.**
- **Herramientas de administración de contenidos.**

Esta información es vital para una empresa que desea buscar una herramienta, para comenzar con el desarrollo en XML y XSL y adoptarla además como un elemento clave en el desarrollo de sus negocios, ya que la elección de una herramienta precisa, dará el punto inicial de partida, para comenzar con un desarrollo eficiente en el futuro, considerando que cada día se ofrecen nuevos productos de software.

8.2 MEJORES HERRAMIENTAS PARA XML Y XSL

Cuando se hace referencia a una herramienta de software XML, se habla de un elemento de software que ayudará a "facilitar" la tarea de crear o manipular código XML o XSL , dependiente del soporte de la misma. A diferencia de HTML, donde una herramienta de desarrollo se considera como un **"editor HTML"**, una herramienta XML es un concepto más extenso, porque XML es una tecnología más compleja que HTML y XSL es un lenguaje de estilo más complejo que HTML.

Un documento XML puede tener una DTD y esquemas asociados, hojas de estilos y códigos scripts complejos.

Existen dos tipos para clasificar estas herramientas :

- Herramientas de creación.
- Herramientas de administración de contenido.

Las *herramientas de creación*, equivalen a los editores tradicionales HTML, en la dirección que se puede crear o editar documentos.

Las *herramientas de administración de contenido*, se utilizan para compartir, administrar y almacenar datos; por lo general, estas herramientas suelen implicar cierto tipo de sistemas de administración de bases de datos (DBMS, Data Base Management System), una herramienta de control de versiones.

Las mejores herramientas de creación en la actualidad corresponden a las siguientes:

1. SoftQuad XMetaL.
2. Adobe FrameMaker+SGML.

3. Arbortext ADEPT-Editor.
4. Stilo WebWriter.
5. IBM Xeena.
6. Vervet Logic XML Pro.
7. Bluestone Visual-XML.
8. Microsoft XML Notepad.
9. XMLWriter

Las mejores herramientas de administración de contenidos XML:

1. Poet Content Management Suite.
2. Arbortext Epic.
3. Chrystal Astoria.
4. Oracle 8i - Oracle 9i.
5. PubliXpress ® 2.0 de North Supply Chile.
6. Colabra.
7. Real Info.

En la figura 8-1 se puede observar la diferencia entre estos dos tipos de herramientas:

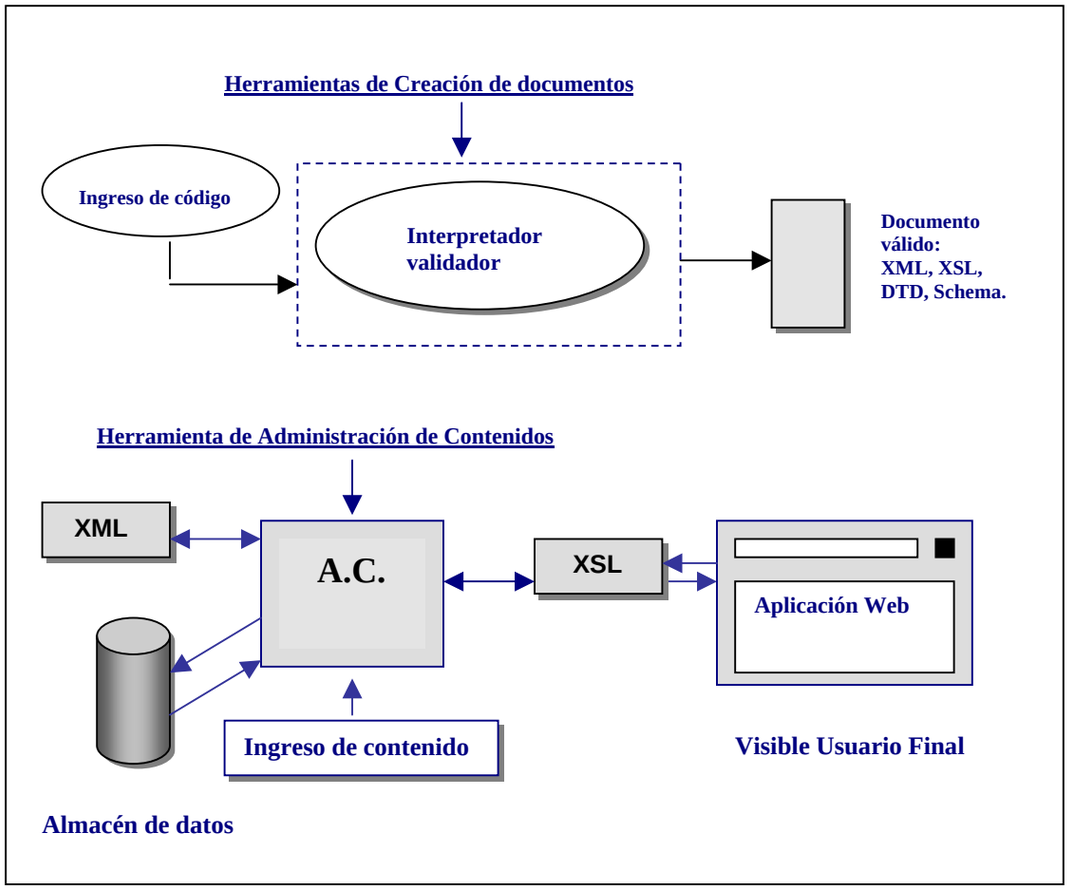

Figura 8-1: Herramientas de creación de documentos XML - XSL versus herramientas de Administración de Contenidos.

8.3 HERRAMIENTAS DE CREACIÓN Y ADMINISTRACIÓN DE CONTENIDOS

A continuación se definirán las herramientas de creación XML:

SoftQuad XMetaL: XMetaL es una herramienta de creación, desarrollada por SoftQuad, que está orientada al desarrollo de documentos XML validados, que se despliegan finalmente en la Web. Soporta el uso de las hojas de estilo en cascada CSS y XSL para mostrar documentos XML. XMetaL también soporta Windows Scripting Host, que consiste en un motor de automatización de Windows que soporta lenguajes como Java Script y VB Script. La propia herramienta XMetaL proporciona una herramienta gráfica de usuario (GUI), que sirve para la creación y edición de documentos XML. Esta interfaz soporta tres vistas distintas de un documento XML:

- Vista Texto, que permite ver el código XML.

- Vista Etiquetas, que permite ver el documento como árbol XML.
- Vista Normal, que permite ver el documento con un estilo de salida.

Una característica importante en esta herramienta es que se requiere que los documentos XML estén bien formados y sean válidos, lo que implica basarse en una DTD.

XMetaL soporta SGML, proporcionando un buen soporte para proyectos de migración de SGML a XML. XMetaL proporciona un medio de separación de contenido y la presentación de los datos.

XMetaL está disponible para las siguientes plataformas:

- Windows 98
- Windows NT
- Windows 2000

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 22].

Adobe FrameMaker+SGML: Adobe Frame Maker es una popular herramienta que se utiliza para ensamblar información en forma impresa y electrónica. FrameMaker+SGML es una extensión de FrameMaker que soporta creación de documentos estructurados XML o en SGML.

Proporciona una GUI (Interfaz Gráfica de Usuario) completa que manipula documentos SGML/XML. Una característica particularmente atractiva es que se pueden abrir documentos no válidos y que FrameMaker+SGML guía en la reparación de estos problemas, además la herramienta permite integrarse en paquetes de gestión de contenido XML para ofrecer una solución XML completa para una empresa.

Las plataformas para esta herramienta son las siguientes:

- Windows 98.
- Windows NT.
- Windows 2000.
- Macintosh

- Unix (Solaris, HP-UX,IBM AIX,SGRI IRIX)

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 23].

Arbortext ADEPT- Editor: Esta herramienta hereda su soporte XML de SGML, que consiste en un editor de validación SGML/XML que incluye una GUI personalizable. Tiene como principales características:

- Visualización de documentos y código XML.
- Visualización de documentos XML como árboles analizados.
- Visualización preliminar.
- Permite un editor de hojas de estilos XSL para crear hojas de estilos que ofrezcan contenido XML.

Las plataformas para esta herramienta son:

- Windows 98/NT/2000
- Unix (Solaris, HP-UX, IBM AIX)

Esta herramienta fue utilizada como herramienta de creación SGML antes de usarlo para XML.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 24].

Stilo WebWriter: WebWriter es una herramienta de creación XML basada en Windows, de Stilo, que fue diseñada para creación de documentos XML. Al igual que la gran mayoría de herramientas, WebWriter adopta una solución gráfica a la construcción de documentos XML. Soporta la creación de documentos XML bien construidos y válidos, dependiendo si se le suministra o no una DTD; si no se facilita, se puede generar automáticamente en base a la estructura de documentos bien formados. Esta característica es importante para quien quiera aprender a crear una DTD. La GUI WebWriter es sensible al contexto, lo que significa simplemente que los menús y los

cuadros de diálogo sólo muestran comandos y opciones que son aplicables al contenido actual.

WebWriter funciona bajo plataforma Windows 98/NT/2000. La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 25].

IBM Xeena: Esta herramienta está diseñada para ser un editor de documentos XML válidos. Fue completamente desarrollada en Java, por lo que requiere de un intérprete Java para ejecutarse. Se soporta además bajo cualquier plataforma computacional que soporte Java.

Xeena proporciona una GUI para hacer la edición visual de documentos XML válidos. Lee una DTD y crea una paleta de elementos que se pueden usar para la creación de documentos. La primera solución de la edición de Xenna es un árbol XML que muestra la relación jerárquica que existe entre los elementos.

Xeena incluye un visor de orígenes de XML, para examinar un código XML.

Las plataformas computacionales para esta herramienta son las siguientes:

- Windows 98.
- Windows NT.
- Windows 2000.
- Macintosh.
- UNIX.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 26] o en el sitio de investigación Alpha Works [URL 1].

Vervet Logic XML Pro: XML Pro es una herramienta de creación XML de la empresa Vervet Logic que presenta contenido XML en forma de árbol XML jerárquico. XML Pro además muestra los datos asociados con el nodo seleccionado del árbol, como los atributos y sus respectivos valores. La GUI de XML Pro es muy clara y fácil de utilizar, e incluye opciones para arrastrar y soltar asistentes para elementos y atributos. También

pueden buscar datos específicos en un documento XML y crear documentos XML válidos identificando a una DTD.

XML Pro es muy útil para el desarrollo basado en pequeñas transacciones.

XML Pro está disponible para las siguientes plataformas:

- Windows 98.
- Windows NT.
- Windows 2000.
- Linux
- Solaris.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 27].

Bluestone Visual-XML: Visual - XML es una herramienta de creación XML de Bluestone Software, que apunta a la creación de soluciones de bases de datos empresariales que incluyen XML. La herramienta está formada por Visual -XML y XML Server.

XML Server genera código XML haciendo de interfaz con las bases de datos en segundo plano, mientras que Visual-XML proporciona un método de crear documentos XML que forman una aplicación empresarial completa.

Visual-XML es una herramienta XML visual que proporciona una GUI para editar contenido XML, permite acceder y ver ambas bases de datos y documentos XML, así como desarrollar y ejecutar comandos SQL y procedimientos almacenados.

Esta herramienta funciona bajo cualquier plataforma que soporte Java, ya que la herramienta está completamente escrita en Java. En la actualidad Bluestone Software es parte de la empresa HP (Hewlett Packard).

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 28].

Microsoft XML Notepad: El Bloc de notas XML de Microsoft se sitúa en la parte más baja de las herramientas de creación XML. Corresponde al equivalente al block de notas tradicional, pero esta herramienta descansa en un árbol jerárquico para proporcionar una vista de un documento XML.

Se pueden validar documentos XML siempre que se tenga instalado Internet Explorer 5.0 o superior (actualmente IE 6.0). Funciona bajo plataforma Windows.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 5].

XMLWriter: Esta es una herramienta freeware que permite la escritura de documentos XML, DTS, XSL, Esquemas, permite validar documentos y consta de un editor de texto que permite organizar de una mejor forma las etiquetas o tags, para una mayor velocidad de escritura.

Lamentablemente la desventaja que existe hoy en las herramientas de creación XML y XSL, es que no existe una herramienta totalmente visual, que permita crear código tal como una herramienta CASE que abriría la posibilidad de facilitar la creación de Templates XSL. Actualmente se trabaja a nivel de código manual y como mejora se han incorporado clasificación de etiquetas, validadores, que ayudan a resumir el esfuerzo de Ingeniería que significa editar una página XML o una plantilla XSL en un proyecto que involucre una solución XML.

La herramienta que más se aproxima a estas características es la proporcionada por Exelon Stylus [URL 29], sin embargo no permite crear templates XSL de forma visual.

Se definirán a continuación las herramientas de **administración de contenidos XML**, que resultarán de gran ayuda estratégica a North Supply Chile, para identificar las mejores herramientas actuales de administración de contenidos:

Poet Content Management Suite: Esta herramienta consiste en un paquete de administración de contenidos que está construida en torno a una base de datos orientada a objetos que almacena contenidos XML. Estos componentes del paquete proporcionan soporte para administrar, reutilizar y enviar contenido XML en un entorno de colaboración. Aparte de soportar el almacenamiento de documentos XML, la base de datos orientada a objetos del paquete puede almacenar archivos de datos de herencia de cualquier tipo, incluyendo contenido del tipo multimedia.

Esta herramienta soporta XML y SGML y canaliza el proceso de publicación de colaboración a través de sus componentes. En soluciones XML basadas en la Web, Poet incluye un componente de fábrica Web que automatiza el proceso de creación de documentos HTML que pueden ser usados por el navegador a partir de documentos XML.

Las plataformas de trabajo de esta herramienta son:

- Windows 98/NT/2000.
- Solaris (servidor).
- HP-UX (servidor).

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 30].

Arbortext Epic: Epic de Arbortext, es una herramienta que sirve como un sistema completo de administración de contenidos para publicar documentos XML en la Web, CD-ROM, o imprimir. Epic soporta la colaboración entre los desarrolladores de contenido a través de un sistema de versiones que requiere que los desarrolladores comprueben los documentos. También incluye una utilidad basada en la Web donde los usuarios pueden hacer sugerencias si están conectados automáticamente con un número de control para la identificación.

Epic incluye un editor XML que está basado en procesadores modernos de texto.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 31].

Chrystal Astoria: Astoria es una herramienta de Chrystal Software, que permite la administración de contenidos XML que fue desarrollada desde la base para soportar SGML/XML. Está destinada a organizaciones que deben crear y manipular almacenes de información XML que contienen documentos estructurados, como la documentación técnica y la de referencia.

La base de datos que soporta es ObjectStore de ObjectDesign, que es una base de datos orientada a objetos que soporta una gran variedad de tipos de datos, entre los que se incluye el contenido multimedia.

Las plataformas para esta herramienta son:

- Windows 98/NT/2000
- Solaris (servidor)

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 32].

Oracle 8i - Oracle 9i:

Oracle está a la cabeza de los Sistemas de Administración de Bases de Datos, DBMS y la versión de Oracle: Oracle 8i , actualmente 9i, ofrece soporte para XML [Hal2001]. Un sistema de administración de contenidos puede construirse perfectamente bajo esta plataforma ya que Oracle define su DBMS como la "e-Business Database" [Feu2001], [Lip2001].

Posee una arquitectura de alta disponibilidad, para la integración con aplicaciones reales de clúster de información, contiene además un analizador XML que fue desarrollado en Java. La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 33].

Microsoft, por su parte no se queda atrás con SQL Server 2000, que es la competencia directa para Oracle en este campo.

Herramientas de administración y publicación de contenidos en Chile:

PubliXpress ® 2.0 de North Supply Chile División e-Business:

Esta herramienta propietaria está orientada a facilitar la publicación de contenidos en la WEB, permitiendo la creación de Sitios Corporativos, Publicaciones de Diarios y Revistas, en general cualquier tipo de Portal en que el contenido proviene de diferentes fuentes y cambia frecuentemente.

Una de las principales características de este producto es que permite que el sitio WEB pueda ser actualizado en línea directamente por los profesionales que generan el contenido. Para realizar esto PubliXpress maneja perfiles de Editor y Redactor, cuyos privilegios permiten al primero (Editor), revisar el contenido generado por el Redactor, antes de publicarlo en Internet.

Esta herramienta está compuesta por un conjunto de Módulos lo que permite personalizar el sitio de acuerdo a los requerimientos del Cliente.

Es una herramienta propia de la Empresa North Supply Chile, División e-Business para trabajar proyectos sobre Internet. La información de la herramienta puede encontrarse en los sitios Web de la compañía [URL 34], [URL 35].

Colabra: Este producto es otro que se puede clasificar como administrador de contenidos, dentro de Chile, este producto tiene sub-clasificaciones que ofrecen soluciones a distintos campos de negocios:

- **OpenSite:** Plataforma dinámica de administración de contenido y acceso a aplicaciones Web. Permite crear portales de entrada a la información y a los servicios. Ofrece Estilos dinámicos de despliegue de información y perfilamiento, facilitando la administración a los usuarios.

- **OpenFile 2.0:** Permite organizar en forma estructurada la documentación de la organización. Provee utilidades de búsqueda y clasificación de información. Su administración dinámica permite crear y modificar bases de conocimiento que se desarrollen con la organización.
- **OpenProject 2.0** Permite administrar la información en la ejecución de proyectos. Provee herramientas de gestión y retroalimentación para equipos de trabajo. Sus funcionalidades facilitan la colaboración, el control y el registro de actividades

La plataforma requerida es la siguiente:

- S.O.: Windows NT o 2000
- Web Server: IIS 4.0 o superior
- B.D.: Sql Server 6.5 o superior, Oracle 7.x o superior, Sybase 10.x o superior.
- Intel Server Pentium III 300 MHZ o superior.
- 6 GB en Disco.
- 256 MB en memoria.

La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 36].

Real Info: Es una plataforma perteneciente a la empresa *Soluziona*, que permite la implementación de un Sitio Web basándose en el modelo de un periódico electrónico virtual. Tiene mucha similitud con las características de PubliXpress ® 2.0 ya que se basa en un administrador de contenidos que permite separar la parte gráfica de despliegue, respecto de la generación de contenido. Real Info**,** consta de:

- Página principal
- Secciones especializadas.
- Herramientas para editores.
- Periodistas.
- Redactores.

- Diagramadores.

Su perfilación de usuarios otorga derechos y restricciones, permitiendo la segmentación de información presentada a visitantes. La información de la herramienta puede encontrarse en el sitio Web de la compañía [URL 37].

No se puede concluir este capítulo, sin antes mencionar al margen de esta clasificación de herramientas, para creación y administración, una herramienta de integración XML muy importante para conectividad de plataformas distintas y distribuidas, basándose en tecnología XML: **Microsoft BizTalk Server 2000**.

BizTalk Server 2000 proporciona un ambiente de desarrollo y ejecución muy eficaz que permite organizar procesos empresariales, tanto internos como entre empresas. BizTalk Server se puede utilizar para tramitar transacciones comerciales que pueden llegar a durar semanas e incluso, meses.

Entre las ventajas de BizTalk Server 2000 se incluyen la posibilidad de definir especificaciones de documentos comerciales y la forma en la que éstos se deben modificar al pasar entre aplicaciones, así como la capacidad de supervisar y registrar la actividad del servidor.

El servidor proporciona una puerta de enlace estándar para el envío y la recepción de documentos a través de Internet, así como para la obtención de una variedad de servicios que garantizan la seguridad, el envío y la integridad de los datos. Utiliza el protocolo XML de forma interna, para describir sus documentos comerciales y emplea protocolos estándares de Internet, como HTTP (Protocolo Web, Hiper Text Transfer Protocol) y SMTP (Protocolo para Correo, Simple Mail Transfer Protocol), para enviar dichos documentos a sus destinos correspondientes; por tanto, permite interoperar con las diferentes aplicaciones que se estén utilizando en cualquier otro ambiente siempre y cuando éstas admitan estándares de Internet. Se pueden enviar documentos a BizTalk Server como archivos XML, EDI (Exchange Data Interface, Interface de Intercambio de Datos) o archivos planos [Woo2001], [URL 4].

Otro producto interesante resulta ser el de la empresa Vinet Communications, Inc. con el producto **Vinet**, que consiste en una solución completa de administración de contenidos a un gran nivel de construcción, sin embargo el costo de la licencia resulta bastante lejano para implementar una solución Web de administración de contenidos en Chile. En la actualidad COPESA (Diario La Tercera) posee esta herramienta para el desarrollo de su sitio de contenidos y noticias [URL 38].

8.4 CONCLUSIONES

Una de las conclusiones directas que se puede asumir de este capítulo es que las herramientas para la construcción de documentos XML o plantillas XSL aún están en desarrollo y no existe en la actualidad una verdadera herramienta visual, que permita crear plantillas XSL y construcción XML como una herramienta del tipo CASE. Por lo tanto, aún se invierte tiempo en construcción del tipo manual, en muchas secciones de un documento XML y en un template XSL. Se invirtió bastante tiempo en localizar las verdaderas herramientas que ofrecen una mejor solución en el mercado y que fueron expuestas en este capítulo.

El objetivo principal fue dar una muestra de las herramientas existentes en el mercado y dar a conocer las herramientas existentes en Chile, de forma particular, en materia de administración de contenidos.

Estas herramientas cada día se van multiplicando, ya que XML representa un estándar que permite generar muchas soluciones de negocios como se expondrá en el Capítulo a continuación.

En el siguiente Capítulo, además se expondrán nuevas aplicaciones de tiempo real y de "administración de contenidos en tiempo real", que representan nuevos y futuros campos de aplicación de negocios sobre la Internet en Chile y que sirve de información estratégica para la Empresa North Supply Chile, División e-Business.

9. APLICACIONES XML CON CONTENIDOS EN TIEMPO REAL

9.1 INTRODUCCIÓN

En este capítulo se mostrarán diversas alternativas de implementación y de negocios posibles sobre Internet, tomando como referencia XML y XSL, además de los sub-lenguajes basados en XML, ya que algunos corresponden a vocabularios XML, es decir, que están basados en XML. El carácter de "Contenidos en Tiempo Real" se interpreta en este capítulo como la información que puede administrarse y presentarse al usuario final, en un intervalo de tiempo variante, ante un suceso dado. Se piensa en aplicaciones que tengan una respuesta "inmediata" al usuario, donde la respuesta **tiempo - información** es un factor preponderante e importantísimo a la hora de tomar decisiones.

Ya que XML presenta una buena alternativa de implementación de negocios es posible aplicarlo en 4 áreas de interés:

- Marcado de documentos.
- Metacontenidos.
- Bases de Datos.
- Mensajería.

Estas áreas se pretenden ejemplificar en forma global, para poder encontrar un camino de nuevas opciones de implementación de negocios, basándose en esta nueva tecnología. Cada día aparecen nuevas aplicaciones interesantes, donde se pueden aplicar lenguajes basados en XML.

Este Capítulo pretende servir a la Empresa North Supply Chile, División e-Business como una documentación de apoyo estratégico y compacto, para incorporar nuevas aplicaciones en el mercado, basándolas en la tecnología XML y la presentación al usuario con XSL.

Se pretende dar sólo un marco de inicio a nuevas alternativas de proyectos que pueden servir para abrir más los sectores de mercado, que aún no han sido explotados, dentro de la Empresa en particular. No se menciona el tema de aplicaciones de protocolo WAP

(Wireless Aplication Protocol, Protocolo para Aplicaciones Inalámbricas), muy relacionado a XML, ya que en North Supply Chile no es un tema que deba ser investigado por razones de "objetivos próximos" inmediatos en el mercado. WAP requiere una gama de celulares en Chile que soporten el protocolo, como por ejemplo el Samsung modelo SCH-6100 (SMARTCOM). El mercado para aplicaciones WAP no se considera masificado en Chile y no representa un atractivo económico aún dentro de NSCH (una alternativa a las aplicaciones WAP son las aplicaciones SMS, Short Message Service o Servicio de Mensajería Corta, para la actual generación de celulares existentes en el país).

9.2 IMPLEMENTACIÓN DE CONTENIDOS CON CDF

9.2.1 INTRODUCCIÓN

Según lo expuesto en [Mor2000, 677], las primeras aplicaciones del XML incluyen el llamado "Formato de Definición de Canal", de Microsoft (Channel Definition Format, CDF), que describe un canal, es decir, una porción de un sitio web que se ha bajado al disco duro y se actualiza periódicamente conforme cambia la información.

Un archivo CDF concreto contiene datos que especifican una página web inicial y la frecuencia con que se actualiza.

Una aplicación es el programa ChartWare, que usa el XML como una forma de describir tablas médicas para que los médicos puedan compartirlas. En Chile, por ejemplo, se espera que existan aplicaciones relacionadas con la actividad bancaria, adquisiciones en comercio electrónico, perfiles de preferencias personales, órdenes de compra, documentos de juicios, listas de refacciones, listas de compras y muchas otras aplicaciones, que aún distan de contar con un modelo en tiempo real, planteado con CDF. Se pueden presentar una gran variedad de aplicaciones si se logra implementar aplicaciones concretas sobre esta tecnología. El mejor ejemplo de aplicación que podría darse en este lenguaje, es la publicación de eventos noticiosos de alta demanda de

concurrencia; al presentar contenido HTML plano (para evitar la carga en el servidor), en el canal determinado, actualizable en intervalos de tiempo proporcionados por CDF.

9.2.2 PRESENTACIÓN DEL SISTEMA

Existen herramientas que permiten la automatización de la creación de canales y documentos CDF, como el CDF Generator de Microsoft, que además incluye un analizador sintáctico de XML y proporciona una interfaz visual de fácil utilización.

Se presenta la implementación de un pequeño documento en CDF, realizado con la herramienta de creación XML: XML Writer, la cual comúnmente se aplica en North Supply Chile, para la creación de documentación XML, por lo cual se debe tener en consideración ciertos elementos de este lenguaje que se pueden ver en la tabla 9-1.

El código resultante de esta pequeña aplicación permite definir un canal de noticias, que debe guardarse con la extensión cdf, para que Internet Explorer lo pueda reconocer; además se puede ver el código resultante en el Anexo 9-1.

Tabla 9-1: Elementos del lenguaje CDF.

Elemento	**Explicación**
1. <Channel>	Es el elemento de los documentos CDF que sirve como elemento raíz y que puede albergar otros elementos.
2. <Title>	Describe el título de un canal.
3. <Abstract>	Proporciona una descripción detallada de un canal.
4. <Author>	Se utiliza para identificar el canal del Autor y su contenido.
5. <Publisher>	Se utiliza para identificar el Editor del canal y su contenido.
6. <Copyright>	Se utiliza para suministrar información sobre quién es el propietario de los derechos de un canal y de su contenido.
7. <Date>	Establece la fecha de publicación de un canal.

8. <LastMod>	Se utiliza para describir la fecha de la última modificación de un canal.
9. <Schedule>	Se utiliza para describir la programación de un canal.

La plataforma empleada solo depende de contar con un editor XML para la creación del documento CDF.

9.2.3 CONCLUSIONES

Mediante CDF, los usuarios pueden suscribirse a canales, y recibir informaciones, notificaciones o actualizaciones, siempre y cuando cambie el contenido de un canal. CDF es un buen ejemplo de XML y de aplicación concreta, ya que está aceptado y tiene un uso general.

Esta tecnología se asemeja mucho en la difusión de noticias e información, versus la notificación a los usuarios de los cambios producidos en el contenido de un sitio Web.

Por ejemplo se puede pensar en aplicaciones tales como la actualización de listas de contactos dentro de una Intranet, por ejemplo en un área comercial o de ventas.

Se presentó una pequeña implementación de documento CDF, que permite servir de base inicial, si se desea profundizar en este tema dentro de North Supply división e-Business. En la Actualidad CDF podría utilizarse para ahorros de licencias, al trabajarlo manualmente, ya que el software Microsoft Site Server (Commerce Edition), cuenta con características propias de Publicación y CDF.

9.3 IMPLEMENTACIÓN Y MANEJO DE SEGURIDAD CON P3P

9.3.1 INTRODUCCIÓN

Cuando se trabaja en la construcción de un sitio Web dentro de un proyecto, por lo general no se evalúa como se manejará, ni menos como se administrará la información de carácter "***confidencial"*** del sitio. Lo más importante a la hora de definir una aplicación de negocios sobre la Web, que maneje información relevante, es definir una

política de "Privacidad de la Información" necesaria, para proteger la integridad de la misma.

El WWW Consortium (W3C) estableció un protocolo de estandarización, conocidos como Plataformas para Proyectos de Preferencias de Privacidad o **P3P**, la cual sugiere una infraestructura en el intercambio de información [Mor2000, 770].

P3P fue diseñado para proveer un método para proteger información de abusos. Permite a los usuarios proporcionar a los sitios de confianza cierta información personal, tal como el nombre, dirección o número telefónico, sin necesidad de escribirlo en cada sitio. El enfoque está en la privacidad del usuario y no en la validación de quién es el usuario, un concepto que en Chile aún no se aborda con profundidad.

Un ejemplo de implementación se encuentra en la organización TRUSTe , la cual trabaja para hacer cumplir normas de seguridad, realizando comprobaciones regulares de quienes aceptan contar con dicho control (usualmente los sitios utilizan un logo TRUSTe distintivo).

Cuando se habla en términos de P3P, los sitios Web se denominan servicios. Se podría interpretar P3P como un intermediario que permite que los usuarios y los servicios en particular concluyan acuerdos sobre el cuidado y la utilización de la información del usuario.

P3P está basado en XML y se le denomina "vocabulario armonizado". Está diseñado de tal forma que una organización cualquiera podría proporcionar un conjunto de propuestas de privacidad, basadas en distintos servicios ofrecidos por su Sitio Web. Por ejemplo, un comercio electrónico podría ofrecer artículos a los usuarios que sólo se hayan registrado como miembros y que hayan proporcionado información personal. Se puede contar con una propuesta de privacidad que anunciará lo que forma la información de los miembros, junto con el modo en que la organización piensa utilizar dicha información, como por ejemplo ofertas especiales, acceso a descuentos, etc.

Los otros usuarios podrían navegar en el sitio bajo la propuesta general de privacidad, pero no tendrían acceso a la página de ofertas o descuentos, como se planteaba en el ejemplo anterior [Mor2000, 771].

Se analizará en esta sección una implementación realizada con una herramienta de creación de documentos P3P: **IBM Policy Editor** y se analizarán las conclusiones respectivas.

9.3.2 PRESENTACIÓN DEL SISTEMA

Antes de presentar la implementación se debe tener en cuenta que los programas clientes que, en principio, soportarán P3P serán los navegadores; ya es el caso de Internet Explorer 6 de Microsoft, plug-ins para éstos, servidores proxy, applets JAVA o monederos electrónicos. Estos programas buscarán las especificaciones P3P en las cabeceras HTTP y en las etiquetas HTML de las páginas web para, posteriormente, compararlas con las preferencias que el usuario haya configurado y mostrarán sus contenidos al mismo. Por otra parte, los sitios Web deberán proporcionar las cabeceras o etiquetas (tags) para que los clientes puedan localizar sus políticas de privacidad para el sitio Web o personalizarlas según sus diferentes secciones.

Se presenta una implementación de P3P, utilizando la herramienta P3P Policy Editor, que se puede obtener en el sitio de IBM: Alpha Works. El P3P Policy Editor es una herramienta visual con una interfaz cómoda de utilizar, la cual permite crear **Políticas de Privacidad** o **Privacy Policy** en lenguaje P3P.

La plataforma utilizada es:

- **Windows 2000 server.**
- **IIS 5.0 (Internet Information Server 5.0, incluido en Windows 2000)**
- **IBM P3P Policy Editor.**

El primer paso para la instalación del IBM P3P Policy Editor, es contar con el JVM, Java Virtual Machine, que se puede bajar del sitio de SUN Microsystem. Para este caso

se utilizó el JAVA(TM) 2 Runtime Environment, Standar Edition 1.3.3_02, donde el primer paso es instalar el archivo: **j2re-1_3_1_02-win.exe**, el cual puede ser bajado vía FTP (File Transfer Protocol) desde java.sun.com y en el caso de no poder bajarlo vía FTP, se maneja la opción de bajarlo vía HTTP. Se seleccionó la herramienta IBM Policy Editor, del sitio AlphaWorks, debido a que presenta un entorno simple y bastante completo para el desarrollo de documentos p3p.

Una vez bajado el archivo j2re-1_3_1_02-win.exe, se procede a hacer doble click en él, en un directorio destinado a la aplicación, así se instalará el JVM, necesario para poder instalar el IBM Policy Editor.

Luego de instalar el JVM, el Setup del IBM Policy Editor se puede ejecutar correctamente, con lo que se podrá instalar fácilmente el programa en el PC (cuenta con un wizard de instalación), figura 9-1 y figura 9-2.

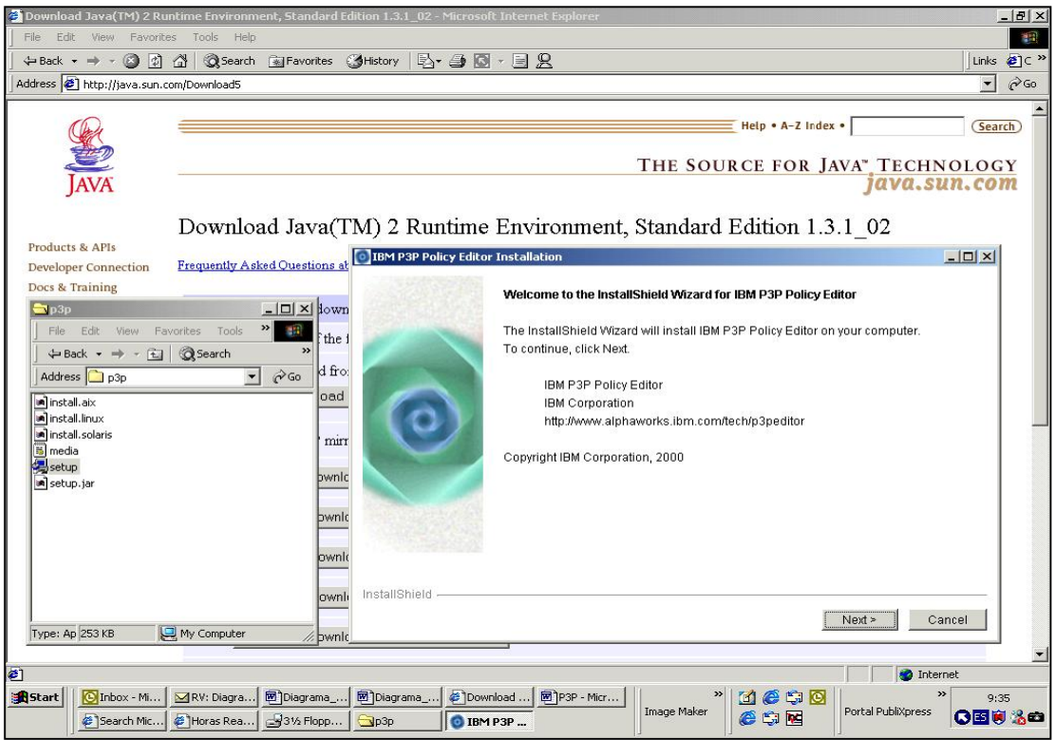

Figura 9-1: Instalación del IBM Policy Editor.

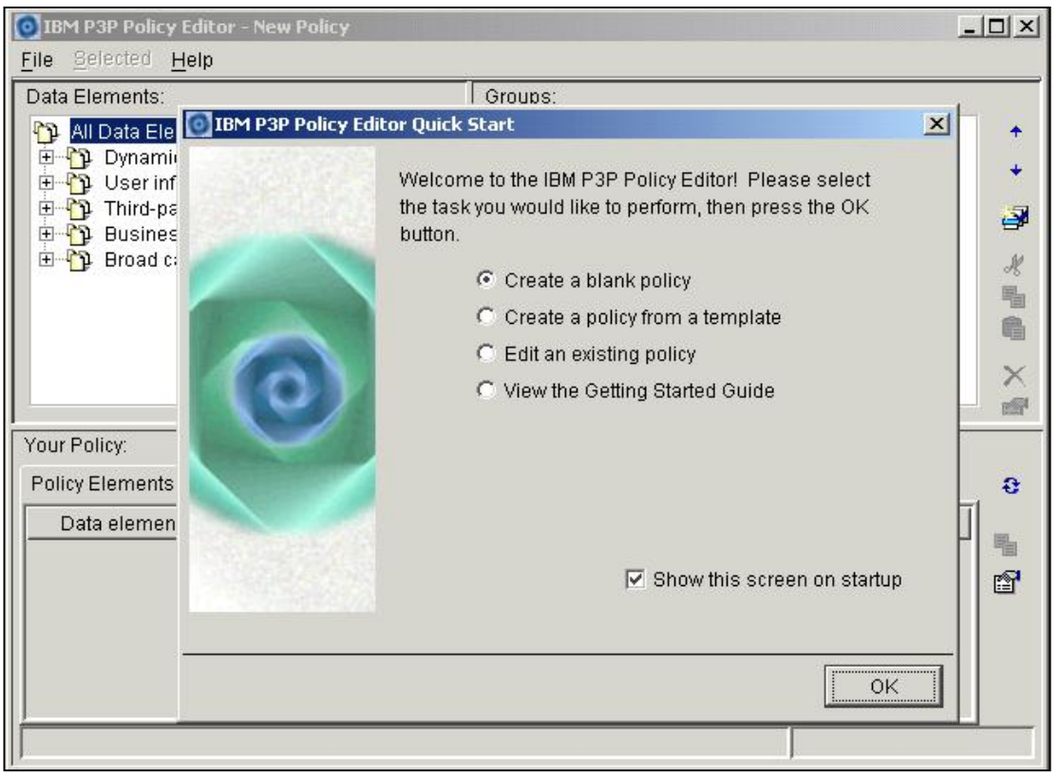

Figura 9-2: IBM Policy Editor.

La metodología a utilizar en la creación del documento P3P, mediante esta herramienta es la siguiente:

1. Definir y Planear la Política.
2. Definir las Propiedades Globales.
3. Declarar los datos.
4. Revisar y refrescar, hacer un update de la Política.

Realizar los siguientes pasos dentro del IBM P3P Policy Editor:

- Ir a **File**, y hacer un **New document**, para comenzar con una política en blanco.
- Seleccionar el tab Error en el panel principal, esta sección desplegará los errores y warnings.
- Hacer click en **File**, **Policy Properties**. Se desplegará: P3P Privacy Properties.
- Ingresar el Nombre de la Organización: en este caso **North Supply Chile**.

- Ingresar al menos un tipo de Información de Contacto: (e-mail, número de fono, o direcciones de mailing).
- Seleccionar el tab **Web Sites** e ingresar el URL del human-readable privacy policy (política de privacidad leíble a la persona). Esta sección permite brindar al Usuario la información acerca de su organización y como será usada.
- Seleccionar el tab **Assurances**, la cual despliega una lista de servicios y procedimientos que permiten declarar que la información en la organización está siendo monitoreada o verificada.
- Hacer click en **Add**, esta opción permite agregar o modificar un procedimiento o un servicio de resolución.
- Ingresar los **URL** de los servicios a los clientes o páginas de organizaciones independientes, de índole relevante o de leyes aplicables para medir si la política de privacidad está siendo seguida.
- Hacer click en **Ok** para cerrar el panel **Disputes Properties**.
- Hacer click en **Ok** para cerrar el panel de las propiedades de Política de Privacidad P3P.
- Abrir el panel **Groups**, hacer un click con el botón derecho en **New Group**, y seleccionar **Propiedades**. El Grupo es usado para especificar el propósito o recipiente de uno o más elementos de información.
- Seleccionar el tab **Purpose**, y seleccionar uno o más ítems que describen porqué la información está siendo reunida.
- Seleccionar el tab **Recipients** y seleccionar uno o más recipientes de información y finalmente **Aceptar**.
- En Data Elements del menú principal, seleccionar un conjunto de datos o elementos de datos para copiarlas al grupo (info@northsupply.cl, por ejemplo).
- Luego hacer un **Refresh Policy.**
- **Guardar Policy como,** se guardará un documento con extensión **.p3p**

- Se puede guardar con extensión .xml para web servers que no están configurados para reconocer archivos p3p.

El código resultante que se construyó, puede verse en el Anexo 9-1. La página resultante en HTML que genera automáticamente el IBM Policy está en Inglés, la cual tiene como ventaja que puede ser adaptada fácilmente al idioma que se requiera, mediante un editor como Front Page, Figura 9-3.

Privacy Statement for North Supply

North Supply has created this privacy statement in order to demonstrate our firm commitment to privacy. The following discloses the information gathering and dissemination practices for this Web site: North Supply

Information Automatically Logged

We use your IP address to help diagnose problems with our server and to administer our Web site. Your IP address is also used to help identify you and your shopping cart. *(** ELABORATE here on the purpose(s) stated above and explain here what other purposes you use the logged information for. **)*

Advertisers

We use an outside ad company to display ads on our site. These ads may contain cookies. While we use cookies in other parts of our Web site, cookies received with banner ads are collected by our ad company, and we do not have access to this information.

Order Forms

Our site uses an order form for customers to request information, products, and services. We collect contact information (like their email address), financial information (like their account or credit card numbers), unique identifiers (like their social security number), and demographic information (like their zip code, age, or income level).

Contact information from the order forms is used to ship purchases, information about our company. The customer's contact information is also used to get in touch with the visitor when necessary.

Figura 9-3: HTML resultante del IBM Policy Editor

Según lo anterior, la arquitectura empleada por la aplicación dentro de un entorno webconstará de la sección de desarrollo y de despliegue de la misma; se puede apreciar de una mejor forma en la figura 9-4:

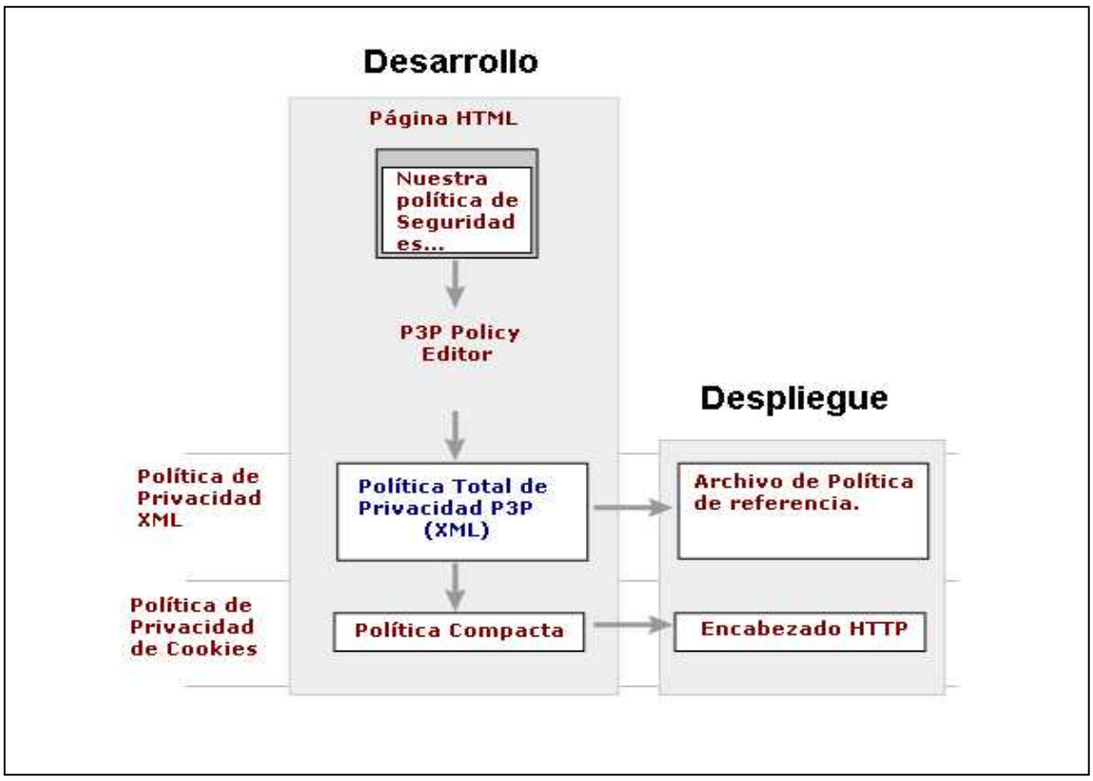

Figura 9-4: Arquitectura P3P en un entorno Web.

9.3.3 CONCLUSIONES

De las muchas áreas de la Web, una que está avanzando lentamente es la relacionada a las normas de estandarización; donde las normas de privacidad tienen un interés particular para los usuarios.

Si se revisa la gran mayoría de los sitios corporativos que manejan información de Clientes o Usuarios en el país, pocos de estos son los que utilizan una Política de Privacidad de la Información. Es recomendable como Control de Calidad de la misma aplicación, desarrollada o a desarrollar, el poder aplicar una normativa de privacidad en el sitio, aplicando este Protocolo P3P, que se basa en XML, independiente de las políticas de seguridad que se puedan aplicar, ya que son dos conceptos distintos. La Política de Privacidad de la Información se basa en los derechos de la información como tal, por lo que, implementar una buena política, implicará tener una aplicación de negocio de mejor calidad.

El campo de aplicación de la misma es grande (incluso para protección de las denominadas cookies o galletas), ya que en la mayoría de los proyectos no se utilizan prácticas de política de privacidad; implementar una política de seguridad en P3P resulta verdaderamente fácil y conveniente, ya que se adapta perfectamente a los estándares planteados por compañías y organizaciones como TRUSTe, Microsoft, W3C, YouPowered, etc.

Un campo de aplicación estratégico para la Empresa North Supply Chile, específicamente la división eBusiness, sería poder ofrecer servicios de Consultoría en Políticas de Privacidad de la Información, en cualquier aplicación que esté desarrollada. Puede aplicarse en sitios Web, como en cualquier otra aplicación. El costo de implementar políticas de privacidad pasa por contar con una herramienta que ayude a trabajar con P3P, de la cual se expuso en este caso el IBM Policy Editor, el cual es de distribución gratuita.

P3P está basado en XML por lo que ofrece una gran ventaja al poder adaptarse a cualquier plataforma y el lenguaje en sí no ofrece una mayor complejidad. Se intentó en esta sección demostrar que mediante XML es posible implementar además un protocolo de privacidad de la información necesario para desarrollos de calidad (y que pueden ser mostrados en la capa de presentación al Cliente como HTML, que perfectamente podría ser XHTML o XSL expuestos en capítulos anteriores).

9.4 IMPLEMENTACIÓN PARA APLICACIONES DE VOZ CON VOXML

9.4.1 INTRODUCCIÓN

Si alguna vez se llama a una línea de asistencia técnica o se intenta comprobar dónde estaba un paquete que se esperaba, sin duda se ha estado en contacto con aplicaciones de respuesta de voz que solicitaba que se pulsara 1 para una opción determinada o 2 para otra opción según [Mor2000, 832].

Las aplicaciones interactivas de voz, o aplicaciones de voz, constituyen las aplicaciones de respuesta por voz más avanzadas, e interpretan respuestas; respuestas habladas por el usuario y que generan voz a partir de texto. La combinación de estas dos características permite escribir código que conduce completamente una aplicación interactiva de voz. Una solución para el mercado del eBusiness es **Voice Extensible Markup Language (Lenguaje de Marcado Extensible de Voz ,VoxML)**, un vocabulario basado en XML introducido por la empresa Motorola, que se utiliza para crear aplicaciones interactivas de voz.

Por medio de VoxML, es posible definir una serie de interacciones del usuario, o cuadros de diálogo, que consultan la información al usuario y que responden a sus preguntas. VoxML es muy simple y considerablemente potente, cuando se trata de pensar lo que supone generar voz a partir del texto, para luego procesar las respuestas habladas por el usuario. Son las tecnologías subyacentes las que llevan a cabo estas tareas, pero VoxML las agrupa a todas. Basándose en XML y XSL para la estructuración de la salida de la información, es posible generar un nuevo campo de negocios y nuevas aplicaciones Web - Voz, que perfectamente pueden ser planteadas en North Supply Chile en la División eBusiness, debido a que no se han investigado diversas alternativas tecnológicas relacionadas.

Se presentará en esta sección la arquitectura necesaria para comenzar a trabajar aplicaciones de esta magnitud en un ambiente de desarrollo.

9.4.2 PRESENTACIÓN DEL SISTEMA

La alternativa más completa para trabajar aplicaciones de voz, se pueden conseguir gratuitamente para desarrollo e investigación en el sitio de Motorola o IBM, sin embargo son opciones que definen características de hardware muy altas para cualquier aplicación que se quiera desarrollar.

La arquitectura necesaria para instalar **Mobile ADK 2.0,** disponible en Motorola es la siguiente:

Hardware:

Pentium II 266 MHz PC (o superior)

- 80 MB de espacio libre en el disco duro.
- 64 MB de memoria RAM (128 MB o más se requieren para ejecutar aplicaciones de voz)
- Una tarjeta de video que soporte 1024 x 768 de resolución y un modo de 16-bit. Una tarjeta de sonido compatible con Windows, para ejecutar las aplicaciones de voz.
- Un set de parlantes compatibles con Windows, para ejecutar las aplicaciones de voz.

Software:

- Windows 95, 98, 2000 o NT 4.0.
- Microsoft JVM versión 5.00.3186 o superior (Windows 2000 necesita Microsoft JVM versión 5.00.3234 o superior)
- Microsoft Internet Explorer 5.0 (o superior)
- Service pack 3 o superior (requerido para Windows NT 4.0)

Para instalar una arquitectura similar de trabajo, se puede considerar el **IBM WebSphere Voice Toolkit**, otro producto para desarrollar aplicaciones VoxML. Se requiere la siguiente arquitectura:

Hardware:

- 550 MHz Intel® Pentium® con 256 MB de RAM o superior.
- Disco Duro: 500 MB mínimo y 500 MB libres para bajar e instalar.
- Tarjeta de sonido y micrófono.

Requerimientos de Software:

- Microsoft Windows® 2000
- WebSphere Voice Server SDK 2.0 .

Los elementos del Lenguaje VoxML que se deben considerar, son los siguientes:

1. **Elemento Dialog:** Es el elemento raíz que sirve como mantenedor para describir el contenido de una aplicación de voz.
2. **Elemento Step:** Es el elemento que se utiliza para definir un estado único en una aplicación VoxML.
3. **Elemento Prompt:** Se utiliza para definir el verdadero contenido de un paso, que consta de texto que se comunica al usuario.
4. **Elemento Help:** Se utiliza para especificar una oración de ayuda que proporciona información adicional cerca de un paso.
5. **Elemento Input:** Se utiliza para definir respuestas aceptables a entradas del usuario.
6. **Elemento Value:** Se utiliza para especificar el valor de una respuesta por parte del usuario.

Para desarrollar una aplicación específica se debe seguir el siguiente orden:

- **Los datos que se disponen en el sistema:** Se plantea un sistema de compra automática de automóviles usando VoxML: **Ford, Volvo, Chevrolet, Peugeot.**
- **El conjunto mínimo de entradas requeridas por el usuario:** Después de la bienvenida se pregunta por el tipo de marca que se desea consultar, para una vez introducida, volver a preguntar por el modelo de la marca en cuestión. Al elegirlo, el sistema nos describirá las características de dicho modelo. El siguiente paso será volver a la pantalla de saludo o bien abandonar la aplicación, perfectamente se puede integrar con otra tecnología (por ejemplo ASP) y permitir comunicación con una base de datos para elaborar un sistema de compra automatizado por medio de un teléfono móvil celular.
- **Las salidas que el sistema debe devolver:** Características del modelo.

- **<u>Tiempo aceptable de respuesta del sistema:</u>** Dependerá mucho de la implementación de Hardware y Software, lo más conveniente es asignar una plataforma dedicada de alta disponibilidad.
- **<u>Identificar el mejor método de diálogo con el usuario según sus características:</u>** Es decir, orientar el sistema hacia el usuario final que al fin y al cabo es el que va a tener que utilizarlo.

El sistema planteado puede verse en la figura 9-5:

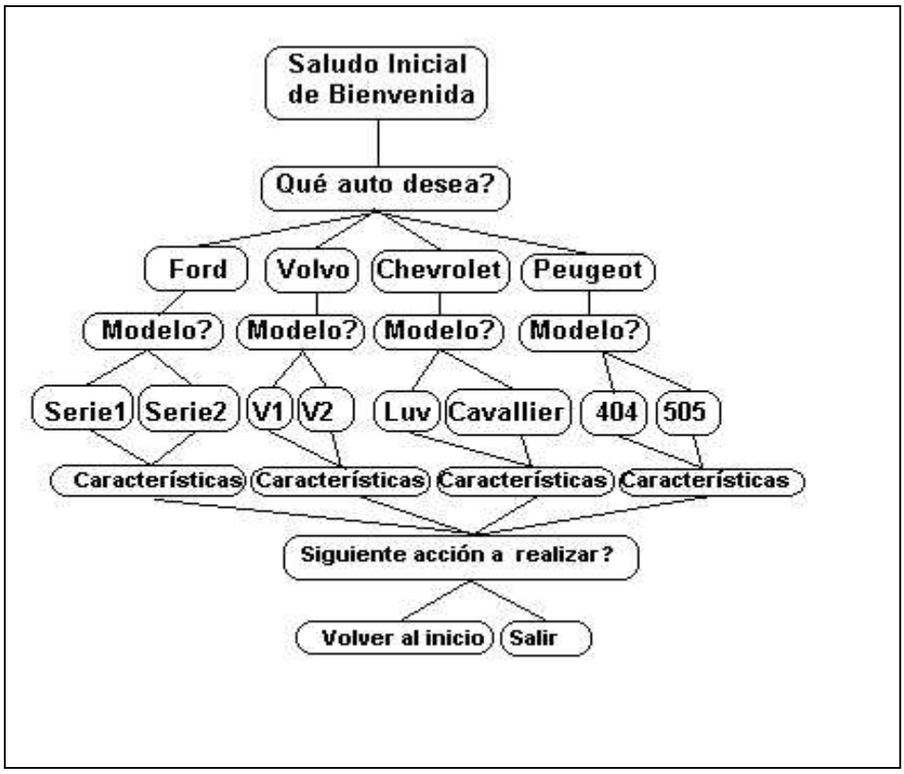

Figura 9-5: Un modelo planteado para una aplicación de venta de automóviles en VoxML.

El archivo realizado en VoxML debe guardarse con la extensión **.vml.** El modelo planteado se trabajó con el editor **HotMetal Pro**, debido a que no se cuenta con una plataforma adecuada de software para probar este tipo de aplicaciones en la División eBusiness de North Supply. El código resultante se puede ver en el Anexo 9-3.

9.4.3 CONCLUSIONES

VoxML está basado en XML y se puede decir que es una variante del voiceXML (Voice eXtensible Markup Language) y sus especificaciones las dirige el VoiceXML Forum (creado por AT&T, IBM, Motorola y Lucent Technologies). Su objetivo es el de proporcionar a los desarrolladores de aplicaciones móviles una herramienta para crear aplicaciones cuyo elemento fundamental de trabajo sea la voz, de modo que la navegación, la entrada de datos e incluso la salida se basen en este elemento y además las aplicaciones, se puedan integrar con entornos Web, según lo expuesto en [URL 18].

Se pretendió en esta sección, dar un marco de inicio a aplicaciones donde se utilicen estas herramientas para automatización de procesos o logística, que perfectamente pueden resultar en negocios interesantes para North Supply Chile.

9.5 XML Y .NET

9.5.1 INTRODUCCIÓN

.*NET* es una iniciativa de la empresa Microsoft, (partner de North Supply Chile) que corresponde a una plataforma de servicios Web basados en XML de Microsoft [And2001]. Los servicios Web XML son aquellos que permiten a las aplicaciones la comunicación y compartir los datos a través de Internet, independiente de la plataforma computacional con la que se cuente, esto es:

- Sistema Operativo.
- Dispositivos.
- Lenguajes de programación.

La plataforma Microsoft .NET proporciona las herramientas necesarias para crear servicios Web y además poder unificarlos.

Se puede entender que la Web modifica la forma de comunicación entre usuarios y aplicaciones. XML, por su parte, está revolucionando la comunicación entre aplicaciones, comunicación entre equipos (utilizando protocolos estándar, como SOAP,

Protocolo de Acceso a Objetos Simple y UDDI, Descripción, descubrimiento e integración universales), porque ofrece un formato de datos que es universal; la ventaja inmediata de esta característica es que permite adaptar fácilmente la información.

Los servicios Web permiten que las aplicaciones puedan compartir la información, además son unidades discretas que se encargan de un conjunto limitado de tareas, se basan en XML, por lo tanto, pueden utilizarse en cualquier sistema operativo.

.NET en definitiva proporciona ventajas aplicables a los negocios, como por ejemplo:

- Facilita la comunicación entre socios.
- Ofrece experiencias mejor personalizadas, por medio de la nueva gama de dispositivos inteligentes.
- Ahorra tiempo y dinero, reduciendo el ciclo de creación.
- Aumenta el flujo de los ingresos, porque proporciona servicios Web XML propios que quedan a disponibilidad de otros.

9.5.2 PRESENTACIÓN DEL SISTEMA .NET

.NET está concebido para unificar servicios Web XML, los cuales pueden aplicarse a:

- Traductores de formatos.
- Servicios de Promociones.
- Servicios de Reservaciones.
- Servicios para la compañía, como notificaciones, mensajería, correspondencia.
- Servicios de notificación.
- Servicios de calendarios y agendas comunes.
- Servicios de compra y venta.
- Monitoreo y logística.
- Servicios de contabilidad.

Los servidores .NET existentes son:

- **Application Center 2000:** Permite desarrollar y administrar aplicaciones Web de alta disponibilidad y escalabilidad.
- **BizTalk Server 2000:** Permite construir una base de procesos de negocios con el protocolo XML, tanto para aplicaciones, como para negocios interempresas, que se comunican finalmente en un mismo lenguaje.
- **Commerce Server 2000:** Para construir aplicaciones de e-commerce en forma rápida y escalable.
- **Content Management Server 2001:** Para administrar contenido para sitios Web dinámicos.
- **Exchange 2000 Server:** Para habilitar mensajería y colaboración, en cualquier lugar y en cualquier parte.
- **Host Integration Server2000:** Permite integrar información y aplicaciones en sistemas del tipo legacy.
- **Internet Security and Aceleration Server 2000:** Para proporcionar seguridad y conectividad rápida en Internet.
- **Mobile Information 2001 Server:** Para habilitar aplicaciones soportada por dispositivos móviles, tales como teléfonos celulares.
- **SharePoint Portal Server 2001:** Permite un medio de comunicación del tipo workflow muy fluido y un manejo de documentación interna de una compañía, para publicar información de negocios.
- **SQL Server 2000:** Motor de Base de datos que permite almacenar y analizar información estructurada XML.

Mediante estos servidores, que en definitiva son los encargados del manejo de la información en este tipo de arquitectura de integración de la Internet como Servicios y basándose en el concepto de Servicios Web, se puede entender el concepto de .NET según la figura 9-6:

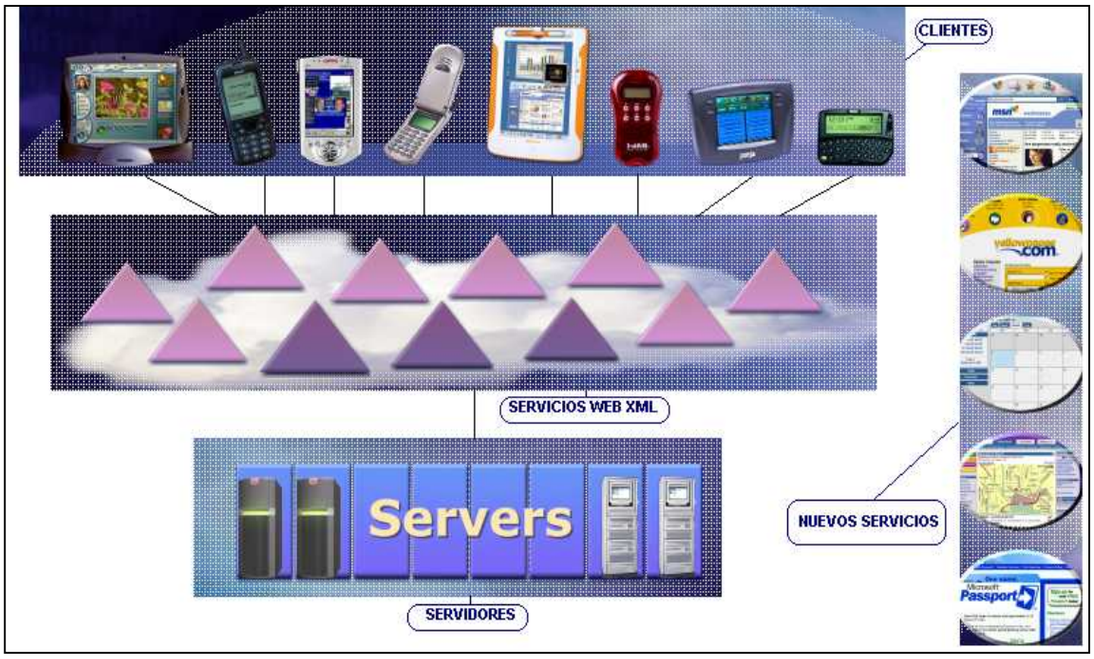

Figura 9-6: Un modelo planteado de arquitectura para .NET, dividido en sección de cliente, servicios Web XML y servidores de aplicación.

Las herramientas de desarrollo propuestas por Microsoft, para aplicar esta arquitectura de solución, desarrollo e integración de servicios Web XML son:

- Visual Studio .NET
- C#
- Visual Basic .NET (Versión posterior a Visual Basic 6.0)
- ASP .NET

En la actualidad, North Supply Chile evalúa estas herramientas que aún están en fase de investigación. Sin embargo dado el rumbo de negocios actual en materia de Telecomunicaciones, que es en definitiva el real interés de North Supply Chile; para la División e-Business como tal, es conveniente o presenta una buena alternativa manejar aplicaciones direccionadas en ese rumbo, aplicables al negocio, por lo que una visión de

solución excelente es aprovechar este marco de trabajo y arquitectura definida por .NET e integrarla a soluciones en la línea de desarrollo de la compañía.

.NET se puede integrar a otra tecnología para dispositivos que pueden resultar en una gama de nuevas aplicaciones bastante rentables, como es el caso de la tecnología **Bluetooth Wireless**, de 3COM [URL 3], la cual es una tecnología inalámbrica de estándar global abierto para enlaces de radio, que ofrece conexiones inalámbricas económicas entre computadoras portátiles, dispositivos de mano, teléfonos celulares y una amplia gama de dispositivos, así como acceso a otros recursos en red. Una buena alternativa de negocios es apostar por una aplicación pequeña (small business) que combine una arquitectura de servicios como .NET con un tipo de conexión XML y un protocolo como WebDAV, usado en servicios Web XML [URL 8] y una tecnología como Bluetooth la cual se puede masificar y vender a un precio bastante "conveniente" para las ambiciones y metas de la empresa North Supply.

9.5.3 CONCLUSIONES

Se presentó en esta sección una arquitectura basada en servicios Web XML, .NET, la cual puede definir una gama de servicios específicos para proporcionar un conjunto de soluciones más personalizables al usuario final.

Se plantea además una combinación de esta arquitectura en conjunto con una tecnología que permita brindar soluciones a dispositivos, como Bluetooth de la empresa 3COM.

Para North Supply Chile, y en definitiva para la División e-Business, una buena alternativa de negocio está en el campo de las aplicaciones móviles específicas, basadas en servicios Web. Plantear un Small Business y desarrollarlo, para proporcionar un servicio Web XML en un dispositivo, del tipo Palm, o celular, representaría un buen inicio de nuevos negocios aplicables a los clientes actuales de la Empresa.

Estas soluciones representan en la actualidad una inversión fuerte en tecnología nueva, pero debería significar un retorno bastante significativo en materia económica, si se

logra introducir un proyecto con estas nuevas tecnologías y replicarlo a distintos clientes (masificarlo).

9.6 IMPLEMENTACIÓN Y APLICACIÓN DE EDUCACION A DISTANCIA Y PORTALES EDUCATIVOS

9.6.1 INTRODUCCIÓN

El objetivo de esta sección es proponer la solución y visión de solución, para los requerimientos de implementación de un Portal de Educación, tomando en consideración lo realizado en North Supply Chile División e-Business, para Fundación Chile, en conjunto con Microsoft Chile.

Se presenta un resumen de las secciones más relevantes de este proyecto (se reserva la información estratégica para la empresa), junto a la arquitectura de solución para la construcción del portal Web del Ministerio de Educación, Educar Chile (http:// www.educarchile.cl).

Se presenta una descripción general de la implementación, además de la arquitectura de la solución definitiva. Con esta sección se puede demostrar que aplicar una herramienta de administración de contenidos, que sea la adecuada y correcta para combinarlas con XML y XSL, representa el ideal de solución para estructurar el gran contenido y volumen que puede llegar a tener un portal de estas características y dimensiones a un nivel de incremento alto y proporciones de nivel sudamericano.

La creación de sitios web escolares aparece como una gran oportunidad para que los establecimientos educacionales puedan, por una parte, proyectarse hacia la comunidad en la búsqueda de promoción y mejorar sustancialmente la comunicación entre los diferentes componentes de la comunidad educativa y, por otra parte, encuentren en ellos un nuevo espacio para las actividades educativas en cuanto al desarrollo de contenidos y la formación. Actualmente el proceso de educación por Internet, concepto denominado **e-*learning*** (aprendizaje electrónico) aún está en fase experimental, ya que es un tema complejo de

abordar. La corporación en Chile que permanentemente ha apostado al e-learning es TELEDUC, de la Pontificia Universidad Católica de Chile. Sin embargo se pueden tener avances en materia de aprendizaje a distancia, con herramientas **de apoyo a la educación**, como presenta el perfil de este portal web de educación que está considerado por el Ministerio de Educación de Chile (MINEDUC), como un logro y apoyo a la educación chilena en todo sus ámbitos.

El proyecto pone a disposición de cada establecimiento educacional los servicios de:

- Hosting (hospedaje de páginas).
- Una herramienta especialmente diseñada para la construcción del sitio en la que se entregan diferentes diseños a través de plantillas XSL.
- Herramientas funcionales para la interacción de los usuarios (constructores y visitantes) con el sitio.
- Herramientas para el docente, alumno, estudiante, investigador y escritorios respectivos personalizados.
- Secciones de registro, bolsa de trabajo, sitios para colegios y personas, correo electrónico, ciberplaza.

Se buscan los siguientes propósitos:

- Entregar una herramienta mediante la cual los establecimientos educacionales puedan presentarse y promocionar sus actividades.
- Dotar de un espacio de comunicación para los diferentes actores de la comunidad educativa.
- Dotar de un espacio de comunicación entre establecimientos educacionales.
- Dotar de un espacio de apoyo a las actividades curriculares tradicionales.
- Fomentar el uso de Internet en los establecimientos educacionales.
- Permitir la ampliación del espacio de educación a un **Medio Virtual**.

La Figura 9-7 resume en un diagrama de contexto general las interacciones externas para un submódulo del proyecto para la sección docente y colegios:

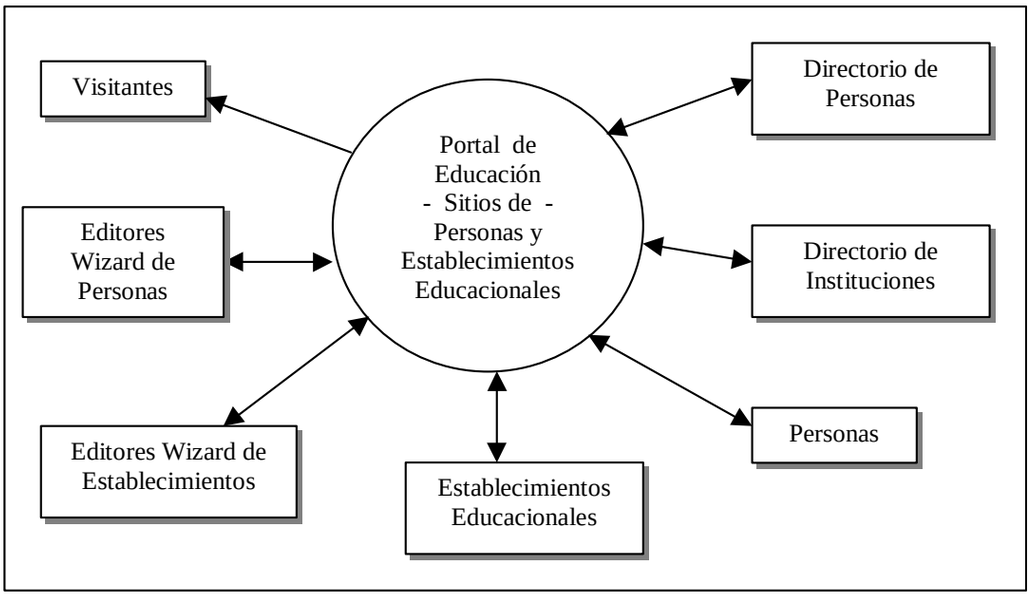

Figura 9-7: Diagrama de contexto portal educativo submódulo para sitios de colegios.

- **Establecimientos Educacionales:** Corresponde a los establecimientos educacionales definidos en el directorio de instituciones, que utilizando un "Wizard de Sitios de Establecimientos", permite crear Sitios Web Corporativos, compuestos de una portada y artículos, aportando de esta forma con información al Portal Educativo.
- **Personas**: Corresponde a personas definidas en el directorio de personas, que utilizando el "Wizard de Sitios de Personas", tiene la posibilidad de crear sitios Web propios, compuestos de una portada y artículos, aportando de esta forma con información al Portal Educativo.
- **Visitantes**: Son aquellas personas que navegan por los sitios de personas y establecimientos educacionales creados a través de los asistentes.
- **Editores Wizard de Personas**: Son funcionarios del Portal Educativo que son responsables de administrar la información presentada en el "Wizards de Sitios de Personas".

- **Editores Wizard de Establecimientos**: Son funcionarios del Portal Educativo que son responsables de Administrar la información presentada en el "Wizards de Sitios de Establecimientos".
- **Directorio de Instituciones**: La interacción con el directorio de instituciones del Portal Educativo, corresponde a obtener información de establecimientos educacionales y actualizar la URL, dirección de la Web de acceso a cada sitio de establecimiento creado, utilizando un asistente.
- **Directorio de Personas**: La interacción con el directorio de personas del Portal Educativo, corresponde a obtener información de personas y actualizar la URL de acceso a cada sitio de persona creado, utilizando un asistente.

Se presenta en la figura 9-8 el diagrama de escenario para la sección de establecimientos.

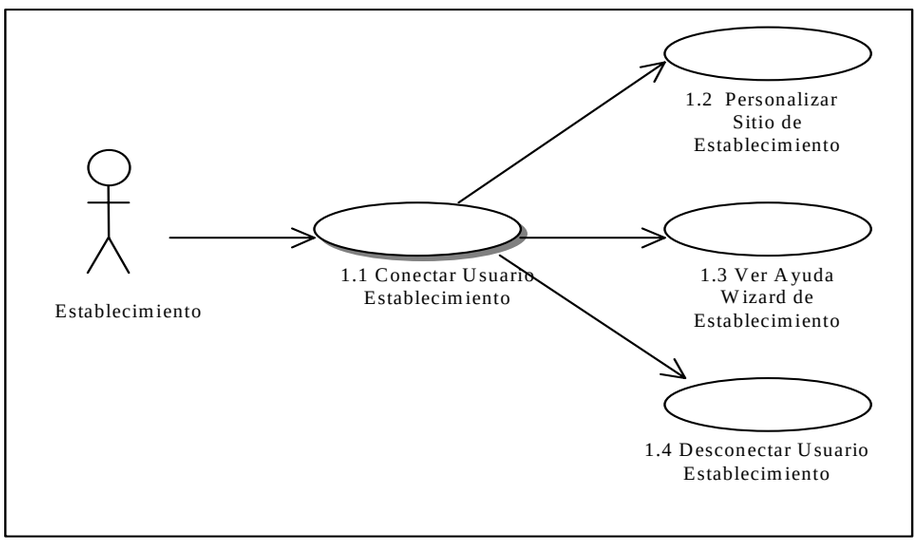

Figura 9-8: Diagrama de escenario para sección de establecimientos.

9.6.2 PRESENTACIÓN DEL SISTEMA

La descomposición de subproyectos se muestra en la tabla 9-2, que resume las funcionalidades asociadas a los subproyectos solicitados y su relación con los módulos de la arquitectura del Portal Educativo en su totalidad.

Como puntos críticos a considerar para la solución de implementación de un Portal de Educación, se deben considerar los siguientes:

- Se considera como mejor forma de abordar la solución, en materia de organizar los volúmenes de información, **XML**, por su gran poder de estructuración. Se requiere implementar un nivel de solución basado en Script y ASP, en combinación con XSL, para lo cual se necesita utilizar XSLT para transformar a HTML la salida y despliegue de los datos, que a su vez, sea compatible con la gran mayoría de los distintos navegadores web existentes en los colegios de Chile.
- Se utiliza el producto PubliXpress versión 2.0 como mejor herramienta para "Administración y Publicación de Contenidos en Tiempo Real" para la implementación de los sub-proyectos, manejo de XML y XSL; modificando y adaptando sus mantenedores y módulos principales.
- Se requiere que la entrega del diseño gráfico se realice durante la etapa de análisis, diseño y planificación. Se considera el diseño en formato de página HTML bien formado, para aplicar la transformación XSLT, incluyendo toda la producción gráfica, estilos de texto, imágenes y así evitar incompatibilidades en los navegadores o browsers con soporte de XML.
- Se requiere que la instalación de la plataforma de hardware y software básica de desarrollo por parte del cliente, se realice a más tardar al término de la etapa de análisis, diseño y planificación.
- Se debe considerar una alternativa de implementación con **Clúster de Datos**, ya que implementar un Portal Educativo de gran magnitud, en cuanto a volumen de la información, requiere una arquitectura de alta disponibilidad.

Un cuadro que resume todas las funcionalidades del proyecto Educar Chile se puede representar en la tabla 9-1:

Tabla 9-1: Presentación del sistema Educar Chile, http://www.educarchile.cl.

Nº	Funcionalidad	MÓDULOS								SUBPROYECTOS							
		Escuela	Comunidades	País	Investig.	Editorial	La Red	Escritorio Docente	Escritorio Estudte.	Adm. de Contenido	Centro de Servicios	Sitios Colegios y Personas	Migración BD	Gestión Escolar	Comunicaciones	Capacitación a Distancia Asíncrona	Comercio Electrónico
1	Adm. de Unidades de Contenido									x							
2	Máquina para hacer Clases (Adm. Mód. Didácticos)							x		x							
3	Máquina para crear pruebas (Adm. de pruebas)							x		x							
4	Máquina para hacer Tareas (Adm. de Mis Tareas)								x	x							
5	Búsqueda de Contenidos				x					x							
6	Recomendar Unidad de contenido									x							
7	Calificar Unidad de Contenido									x							
8	Editorial					x				x							
9	Deporte Escolar	x								x							
10	Radio Escolar	x							x	x							
11	Sitio CRAD	x						x	x	x							
12	Sitios Inducidos	x		x						x							
13	Sitios Licitados	x								x							
14	Mesa de Ayuda	x							x		x						
15	Bolsa de Trabajo										x						
16	Avisos Clasificados										x						
17	Buscador en Web (Utilitarios)							x	x		x						
18	Consulta de Directorios (Adm. de Directorios)		x		x			x	x		x						
19	Sitios Establecimientos	x										x					
20	Sitios Personales	x										x					
21	Directorios de Sitios	x					x	x	x								
22	Ventana con contenido destacado para profesores							x									
23	Ventana con contenido destacado para estudiantes								x								
24	Sitios de Capacitación	x															
25	Burbujas	x													x		
26	Gestión Escolar	x						x									
27	Sitios Referidos	x															
28	Herramientas de Comunicación (e-mail, foros, chat)		x					x							x		
29	Sitio de Comunidades		x														
30	Pág. Web de asociaciones gremiales y Partidos Políticos			x													
31	Ayuda acerca de uso de Internet						x										
32	Download de SW.						x										
33	Capacitación							x									
34	Agenda compartida							x							x		
35	Ventana con contenido destacado para profesores																
36	Ventana con contenido destacado para estudiantes																

La arquitectura de solución para la implementación del portal, se puede apreciar en la figura 9-9:

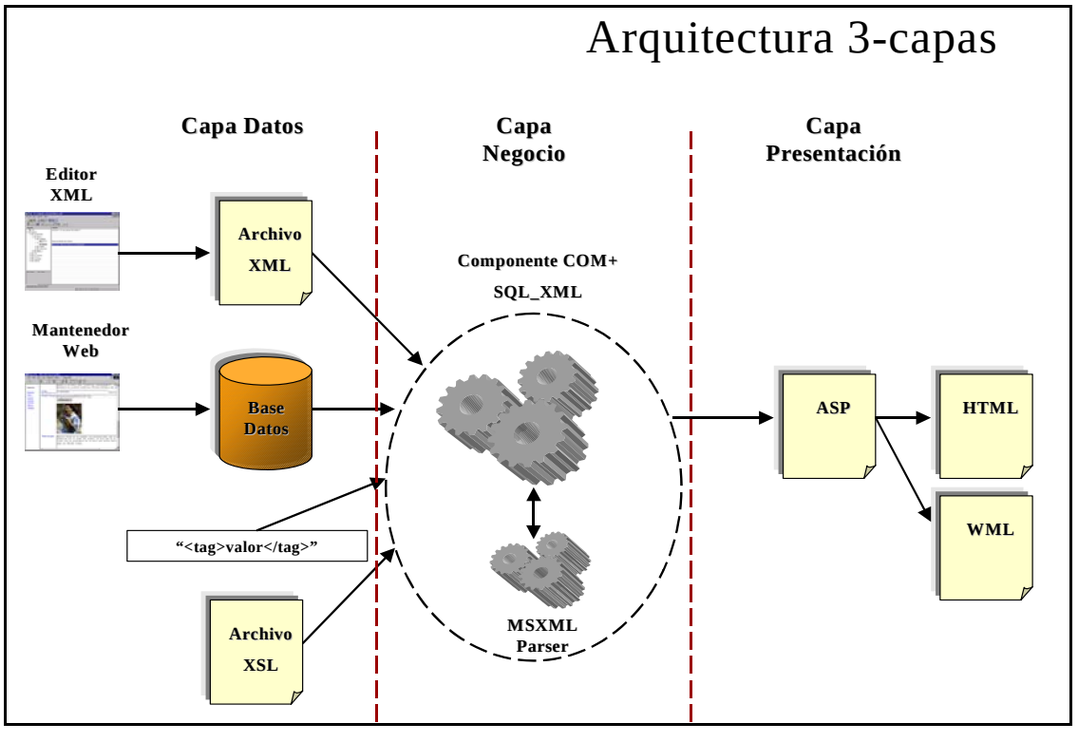

Figura 9-9: Mejor Arquitectura de solución para implementar un Portal Educativo basándose en XML y XSL.

La figura 9.10 muestra la arquitectura de la solución propuesta para los subproyectos solicitados y la estructuración de los volúmenes de la información:

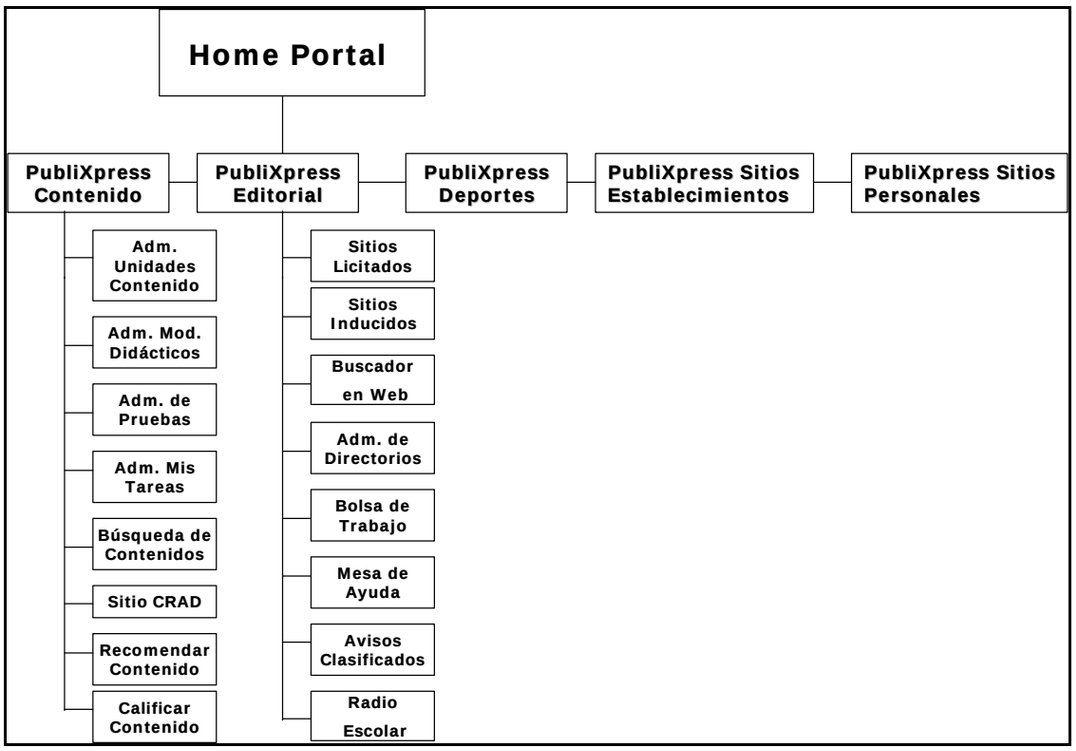

Figura 9-10: Arquitectura de la solución propuesta en el portal Educar Chile.

Esta solución contempla la instalación y adaptación de instancias de PubliXpress para el manejo de la información requerida.

La solución considera un único sistema de membresía que permite manejar distintos perfiles de usuarios definidos:

- Docentes.
- Estudiantes.
- Generadores de Contenido.
- Editores.
- Administradores del Sitio.
- Periodistas.

Las secciones del Portal Educativo pueden resumirse de la siguiente forma, considerando un orden y metodología de solución:

ADMINISTRACIÓN DE DIRECTORIOS

Descripción Solución:

Módulo que permite mantener la información actualizada de colegios, instituciones, docentes, Investigadores, Centros de Investigación y sitios Web.

1. Se adaptan mantenedores XML para el ingreso de esta información (rol generador de contenido o administrador del sitio).
2. Se provee un mecanismo simple de búsqueda XML de información.

BOLSA DE TRABAJO

Descripción Solución:

Módulo simple que permite ofrecer y buscar empleo.

1. Se provee de un formulario para ingresar información de empleos ofrecidos (clasificados bajo un criterio).

2. Se provee de un formulario para ingresar información de empleos buscados (clasificados bajo un criterio).
3. Se provee de un formulario para realizar una búsqueda simple de empleos ofrecidos. Una vez seleccionado el empleo se envía la postulación a través de un formulario.
4. Se provee de un formulario para realizar una búsqueda simple de empleos buscados. Una vez seleccionado el empleo, se permite el envío de un email al oferente a través de un formulario.
5. Se adapta administrador de privilegios con el fin de permitir la edición de los contenidos, los resultados se van mostrando en templates XSL.

MESA DE AYUDA

Descripción Solución:

Módulo que permite realizar preguntas al estilo de un FAQ simple.

1. Se provee de un formulario para el ingreso de preguntas clasificadas por sector. A través de otro formulario se asigna la pregunta a algún especialista.
2. Una pregunta pasa por los siguientes estados: ingresada, asignada (a un especialista) y resuelta.
3. Se provee de un formulario para realizar una búsqueda simple de las preguntas realizadas.
4. Para logra estos objetivos se adaptan los mantenedores XML y el administrador de privilegios de modo de permitir la asignación y el seguimiento simple de las preguntas.

AVISOS CLASIFICADOS

Descripción Solución:

1. Módulo simple que permite ofrecer o solicitar productos y/o servicios.

2. Se provee de un formulario para ingresar información de productos o servicios ofrecidos (clasificados bajo un criterio).
3. Se provee de un formulario para ingresar información de productos o servicios requeridos (clasificados bajo un criterio).
4. Se provee de un formulario para realizar una búsqueda simple de productos o servicios ofrecidos.
5. Se provee de un formulario para realizar una búsqueda simple de productos o servicios requeridos.
6. Para lograr este objetivo es necesario adaptar los mantenedores de la herramienta de Administración de contenidos PubliXpress y el administrador de privilegios.

DEPORTES

Descripción Solución:

1. Se realiza la integración gráfica con XSL y la adaptación de los mantenedores (interfaz de PubliXpress), con el fin de lograr una interfaz amigable y adhoc al tema en cuestión.

SITIOS ESTABLECIMIENTOS

Descripción Solución:

1. Cada colegio corresponderá a un canal. El generador de contenidos será el responsable de la creación inicial de estos canales.
2. Una vez creado el canal en XML, cada colegio (a través de un directivo o docente) podrá generar su portada y artículos.
3. La generación de portadas y artículos se realiza mediante el uso de wizard.
4. Se proveen 3 plantillas XSL para la portada y 3 plantillas XSL para los artículos.
5. Se provee de un mecanismo simple de búsqueda de información.

6. Se realiza integración gráfica con XSL y la adaptación de mantenedores XML con el fin de lograr una interfaz amigable y adhoc al tema en cuestión.

SITIOS PERSONALES

Descripción Solución:

1. Cada persona será un canal. El generador de contenidos será el responsable de la creación inicial de estos canales.
2. Una vez creado el canal, modificando XML, cada persona podrá generar su portada y artículos, los que serán validados por un editor.
3. La generación de portadas y artículos se realiza mediante el uso de wizard.
4. Se proveen 3 plantillas para la portada y 3 plantillas para los artículos.
5. Se proveerá de un mecanismo simple de búsqueda de información.
6. Se realiza integración gráfica con XSL, proceso XSLT y adaptación de mantenedores con el fin de lograr una interfaz amigable y adhoc al tema en cuestión.

9.6.3 CONCLUSIONES

Una implementación exitosa, a la hora de utilizar una herramienta de administración de contenidos, dentro de un portal Web, debe basarse en una tecnología que brinde confianza. En el caso del proyecto Educar Chile, para el Ministerio de Educación y Fundación Chile, bajo tecnología Microsoft, se eligió incorporar **el Modelo de Tres Capas**; es decir, proporcionar una capa de **Datos**, de **Negocios** y de **Presentación**.

La capa de Datos, se basó en XML el cual permite hacer una excelente clasificación de los canales del Portal, lo cual lleva a ser la característica y opción más óptima; para la capa de Negocios, se utilizaron componentes (dll's), los cuales permitían hacer más optima la llamada a funciones y procedimientos comunes dentro del Portal, utilizando para las llamadas en VBScript y lenguaje ASP; finalmente, la capa de Presentación,

desarrollada con tecnología XSL, que permite desplegar el contenido en HTML bien formado.

Se podría pensar que un modelo de tres capas, para un portal Web, tal vez no sea lo óptimo ante sucesos inesperados: aumento de visitas concurrentes ante noticias de impacto e imprevistos. Ante los sucesos noticiosos, ocurridos en el mundo en septiembre del año 2001 (caída de Las Torres Gemelas en el World Trade Center), muchos periódicos on-line y portales, que basaban sus administraciones de contenidos en el modelo de tres capas, simplemente colapsaron; fue el caso de muchos portales en Chile y en el extranjero, incluso en el Portal Educativo; por lo cual, se opta por presentar simplemente páginas planas en formato HTML, en lugar de presentar el portal completo. Esto se debe a que este modelo no está pensado ante una carga masiva de usuarios concurrentes, y ante todo bajo este tipo de "stress de concurrencia masiva". Para dar una idea de este factor, se requiere y se recomienda siempre hacer una prueba de carga y de stress al portal Web, o manejar una opción de balanceo de carga mediante un clúster de información, una vez que se está en ambiente de producción. Además contar con a lo menos una alternativa de contingencia, de reemplazo del portal y del modelo en sí.

El modelo de tres capas representa sin embargo una buena **alternativa** de solución, y se concluye además que XML representa la mejor alternativa para modelamiento de los datos y clasificación de los mismos. XSL por su parte, al ser un lenguaje para interpretar XML y presentarlo al usuario como HTML, utilizando su conjunto de transformadas XSLT, presenta una buena opción para usarlo en combinación con ASP, el cual se encarga de direccionar los parámetros y llamadas a objetos. En este capítulo se ha mencionado el concepto global del proyecto del Portal Educativo y sus distintas secciones. La aplicación funcionando puede verse en la dirección: http://www.educarchile.cl, donde se puede demostrar que un portal de tal magnitud es muy fácil de administrar, siempre que se haya elegido la mejor alternativa de solución,

en este caso se utilizó XML, XSL, ASP, Scripts, programación de componentes, un buen motor de base de datos y un diseño de tres capas.

9.7 CONCLUSIONES

En este capítulo se pretendió dar una visión de aplicaciones y tendencia de las aplicaciones basadas en XML y XSL. XSL va a seguir siendo la tecnología de representación de la salida de la información, la parte visible al usuario, mientras que XML la de organización de la información; la combinación es ideal. Existen muchas alternativas y una gran variedad de aplicaciones, que en este capítulo quedaron fuera, por razones de objetivos para North Supply. Se presentaron aplicaciones que se direccionan al rubro de negocios de la Empresa, por lo cual se cumple el objetivo principal de este capítulo, brindar una visión nueva de campos de negocios que se pueden implementar y a la vez resultar de gran impacto en el comercio.

10. CONCLUSIONES FINALES

XML es una especificación de marcado que se utiliza además para crear lenguajes de marcado, por lo que no debería llamar la atención la gran variedad de tecnologías cuyos nombres terminan en "ML". Al igual que ocurre con prácticamente todas las nuevas tecnologías que se basan en la Web, XML sigue adelante. La norma XML existe desde hace relativamente poco tiempo, existiendo además organizaciones, que se encargan del desarrollo e investigación de esta especificación, como por ejemplo la **Organización XML** [URL 21] y no es probable que cambie mucho, además las que están vinculadas a XML siguen evolucionando. La única forma de estar al día en el futuro, no obstante las muchas y variadas tecnologías existentes en cuanto a XML se refiere, es por medio de la investigación constante.

La Web está llena de información sobre XML, es el mejor lugar donde se puede estar al tanto de novedades respecto al tema.

Aunque es sencillo de entender, XML tiene la potencia suficiente para expresar estructuras de información que son bastante complejas. En función de la aplicación que se utilice, la información deberá estar estructurada siempre de una determinada forma. Para muchas aplicaciones actuales, una estructura de árbol es lo suficientemente general y robusta para expresar información con cierto grado de complejidad, alternando equilibrio y sencillez de la representación de la información. Por todo esto XML permite expresar estructuras complejas de datos que satisfacen las exigencias de casi todas las aplicaciones, y más aún representa una buena alternativa para las aplicaciones de negocios.

Se agruparon los capítulos en dos grandes áreas, la primera presentó XML y XSL a un nivel más detallado, para poder comprender estas tecnologías, mientras que la segunda mitad proporcionó una visión de las nuevas aplicaciones en Internet con administración y generación de contenidos en tiempo real. Cada capítulo contó con su sección de conclusiones o resúmenes finales, para dar una mejor pauta de entendimiento a quién lea

las secciones respectivas. De todos los capítulos, la gran conclusión de la tecnología XML se puede resumir en tres claras características:

- **Sencillez.**
- **Variedad de estructuras de datos.**
- **Excelente tratamiento y comunicación de la información.**

En esta Investigación se proporcionó un aporte significativo y estratégico a la **Empresa North Supply Chile**, División e-Business, para lograr un apoyo de nuevas tendencias y tecnologías que se pueden aplicar en materia de negocios, cumpliendo los objetivos claves planteados en el Capítulo 1.

Se pretende contar con este material como patrimonio y documentación interna de la Empresa, que pueda ayudar y apoyar significativamente los nuevos desarrollos y expansión de los mismos, así como para los estudiantes y docentes de último año de Ingeniería Civil en Informática, de la Universidad Austral de Chile, para que permita brindar un nuevo campo y enfoque en materia de enseñanza, ya que este estándar XML es el que se aplicará, por lo menos, durante muchos años más.

11. BIBLIOGRAFÍA

Libros:

[Cas99] CASTAGNETO, J., RAWAT, H., SCHUMANN, S., SCOLLO, C., VELIATH, D. (1999). *Professional PHP Programming, Chapter 14, XML*: Wrox.

[Flo2000] FLOYD, M. (2000). *Creación de sitios Webs con XML*: Pearson Education, S.A.

[Gua2000] GUADALAJARA, J. (2000). *Cómo crear un portal en Internet*: E-dita, grupo e-Xtra.

[Har98] HAROLD, E. (1998). *XML Extensible Markup Languaje*: IDG Books Worldwide, Inc.

[Mar2000] MARUYAMA, H., TAMURA, K., URAMOTO, N. (2000): *Creación de sitios Webs con XML y Java*: Pearson Education, S.A.

[Mor2000] MORRISON, M. (2000). *XML Al descubierto*: Prentice Hall.

Revistas:

[Feu2001] FEUERSTEIN, S. Understand how XML and PL/SQL can work together. *Oracle Magazine,* XV- 4: 93-96 July/August 2001.

[Hal2001] HALL, R. XML Support in Oracle Technology. *Oracle Magazine*, XV- 5: 33 September/October 2001.

[Lip2001] LIPSON, S. A Giant leap in e-Business database evolution. *Oracle Magazine,* XV - 4: 61-69 July/August 2001.

[Sel2001] SELIGMAN, L., ROSENTAL, A. XML's Impact on Databases and Data Sharing. *IEEE,* 0018-9162 - 01: 59-67 June 2001.

Congresos:

[And2001] ANDERS, M. Introducing .NET. Microsoft .Net arquitecture partner submmit. TECHNICAL TRAINING EVENT. Washington DC., USA 2001.

[Jac2001] JACOBS, R. COM+ Services In .NET. TECHNICAL TRAINING EVENT.

Washington DC., USA 2001.

[Woo2001] WOODGATE, S. BizTalk Server 2000: Orchestrating Business Processes Across .NET And Legacy Systems. TECHNICAL TRAINING EVENT Washington DC. , USA 2001.

Revisión Web:

[URL 1] ALPHA WORKS IBM (2001/2002). Application Development, XML.
http://www.alphaworks.ibm.com/tech/xmlgenerator

[URL 2] BIZTALK ORGANIZATION (2001/2002).
http://www.biztalk.org/News/news.asp

[URL 3] 3COM Tecnología Bluetooth
http://lat.3com.com/lat/products/wireless/bluetooth/faq.html

[URL 4] MICROSOFT CORPORATION (2001). Microsoft Biz Talk Server.
http://www.microsoft.com/biztalk/evaluation/trial/default.asp

[URL 5] MICROSOFT CORPORATION (2001). Microsoft XML Notepad.
http://msdn.microsoft.com/library/
default.asp?url=/library/en-us/dnxml/html/xmlpadintro.asp

[URL 6] MICROSOFT CORPORATION (2001). Porqué XML.
http://www.microsoft.com/latam/msdn/articulos/2000/03/art02/default.asp

[URL 7] MICROSOFT CORPORATION (2001). ¿Qué es .NET?.
http://www.microsoft.com/latam/net/defined/default.asp

[URL 8] MICROSOFT CORPORATION (2000). Communicating XML Data over the Web with WebDAV.
http://msdn.microsoft.com/library/en-us/dnxml/html/xmlandwebdav.asp

[URL 9] MICROSOFT CORPORATION (2000). How to Encode XML Data.
http://msdn.microsoft.com/library/en-us/dnxml/html/xmlencodings.asp

[URL 10] MICROSOFT CORPORATION (1999). A Beginner's Guide to the XML DOM
http://msdn.microsoft.com/library/en-us/dnxml/html/beginner.asp

[URL 11] MICROSOFT CORPORATION (1999). A Guide to XML and Its Technologies
http://msdn.microsoft.com/library/en-us/dnxml/html/xmlguide.asp

[URL 12] MICROSOFT CORPORATION (1999). Del HTML al XML.
http://www.microsoft.com/latam/msdn/articulos/1999/11/art03/default.asp

[URL 13] MICROSOFT CORPORATION (1999). XML: The ASCII of the Future?
http://msdn.microsoft.com/library/en-us/dnxml/html/xmlfinal.asp

[URL 14] MICROSOFT CORPORATION (1998). XML: Enabling Next-Generation Web Applications.
http://msdn.microsoft.com/library/en-us/dnxml/html/xmlwp2.asp

[URL 15] MICROSOFT CORPORATION (1997). Elementary XML
http://msdn.microsoft.com/library/en-us/dnxml/html/elxml.asp

[URL 16] MICROSOFT CORPORATION (1997). XML: Data the Way You Want It.
http://msdn.microsoft.com/library/en-us/dnxml/html/xmldata.asp

[URL 17] ORGANIZACIÓN RAMÓN (2001). XML, el lenguaje universal.
http://www.ramon.org/xml/articulos/intro_xml-html.htm

[URL 18] VOICE XML PLANET (2001) Evaluating the Benefits of VoiceXML for eBusiness.
http://voicexmlplanet.com/articles/benefits.html

[URL 19] W3C (2000). Extensible Markup Language (XML) 1.0 (Second Edition)
http://www.w3.org/TR/2000/REC-xml-20001006

[URL 20] W3C (1999). XSL Transformations (XSLT) Version 1.0
http://www.w3.org/TR/xslt

[URL 21] XML ORGANIZATION (2001/2002).
http://www.xml.org/xml/industrySectorList.jsp?CATEGORY=61

[URL 22] (2001/2002) XML XSL herramientas
http://www.softquad.com

[URL 23] (2001/2002) XML XSL herramientas

http://www.adobe.com

[URL 24] (2001/2002) XML XSL herramientas

http://www.arbortext.com

[URL 25] (2001/2002) XML XSL herramientas

http://www.stilo.com

[URL 26] (2001/2002) XML XSL herramientas

http://www.alphaworks.ibm.com/tech/xeena

[URL 27] (2001/2002) XML XSL herramientas

http://www.vervet.com

[URL 28] (2001/2002) XML XSL herramientas

http://www.bluestone.com

[URL 29] (2001/2002) XML XSL herramientas

http://www.exelon.com

[URL 30] (2001/2002) XML XSL herramientas

http://www.poet.com

[URL 31] (2001/2002) XML XSL herramientas

http://www.arbortext.com

[URL 32] (2001/2002) XML XSL herramientas

http://www.chrystal.com

[URL 33] (2001/2002) XML XSL herramientas

http://www.oracle.com

[URL 34] (2001/2002) XML XSL herramientas

http://www.northsupply.cl

[URL 35] (2001/2002) XML XSL herramientas

http://www.ebt.cl

[URL 36] (2001/2002) XML XSL herramientas

http://www.colabra.cl

[URL 37] (2001/2002) XML XSL herramientas

http://www.realinfo.cl

[URL 38] (2001/2002) XML XSL herramientas

http://www.vinet.com

[URL 39] (2001/2002) XML XSL herramientas

http://www.netscape.com

[URL 40] (2001/2002) XML XSL, SOAP

http://www.w3.org/TR/2000/NOTE-SOAP-20000508

12. ANEXOS:

CAPÍTULO 4:

Anexo 4-1: Listado de códigos para la figura 4-2.

direcciones.dtd

```
<!ELEMENT direcciones (contacto)+>
<!ELEMENT contacto (nombre, direccion+, ciudad, estado, zip, fono, email, web, empresa)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT estado (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT fono (voice, fax?)>
<!ELEMENT buzon (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT web (#PCDATA)>
<!ELEMENT empresa (#PCDATA)>
```

direcciones.xml

```
<?xml version="1.0"?>
<?xml-stylesheet href="direcciones.xsl" type="text/xsl"?>
<!DOCTYPE addressbook SYSTEM "direcciones.dtd" [
<!ENTITY amp "&#38;#38;">
<!ENTITY apos "&#39;">
]>
<direcciones>
  <contacto>
    <nombre>Jorge</nombre>
    <direccion>Mackenna 750 depto 1B </direccion>
    <ciudad>Santiago</ciudad>
    <estado>Chile</estado>
    <zip>10011</zip>
    <telefono>
      <buzon>02 6345591</buzon>
      <fax>212-555-1213</fax>
    </telefono>
    <email>jcanales@northsupply.cl</email>
    <web>http://www.northsupply.cl</web>
    <empresa>North Supply Chile ATI</empresa>
  </contacto>
</direcciones>
```

direcciones.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html><head><title>Ejemplo de salida XSL a partir de un XML</title></head>
      <body bgcolor="#FFFFFF">
        <xsl:for-each select="direcciones/contacto">
          <xsl:apply-templates select="nombre"/>
          <xsl:apply-templates select="direccion"/>
          <xsl:apply-templates select="ciudad"/>
          <xsl:apply-templates select="estado"/>
          <xsl:apply-templates select="zip"/>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="nombre">
    <h2><xsl:value-of/></h2>
  </xsl:template>
  <xsl:template match="direccion">
    <xsl:value-of/><br/>
  </xsl:template>
  <xsl:template match="ciudad">
    <xsl:value-of/>,
  </xsl:template>
  <xsl:template match="estado">
    <xsl:value-of/>
  </xsl:template>
  <xsl:template match="zip">
    <xsl:value-of/><br/>
  </xsl:template>
</xsl:stylesheet>
```

CAPÍTULO 6:

Anexo 6-1: Listado para el ejemplo de salida de la figura 6-2.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html xmlns="http://www.w3.org/TR/xhtml1">

<head>
<title>e-Business ejemplo capítulo 6 XHTML</title>
</head>

<body BACKGROUND="Ivy.gif" BGCOLOR="#FFFFFF">

<p ALIGN="center">
<font face="Arial" color="#004080">
<small>
```

```
 <strong>North Supply Chile-División e-Business</strong>
</small>
</font>
</p>

<p ALIGN="CENTER"> </p>

<!-- Comentario 01 -->
<!-- Código identado, es el ideal en XML  pero los softwares editores no identan así-->
<!-- lo  importante es  respetar las etiquetas de inicio y  final. -->

<p ALIGN="CENTER">
  <font face="Arial" color="#000040">
   <strong>
     <small>
         Ejemplo Simple de Utilización de XHTML en un sistema
     </small>
   </strong>
  </font>
</p>

<p ALIGN="CENTER"><font face="Arial" color="#000040"><strong><small>de
Muestra de Unidades
de frutas de exportación.</small></strong></font></p>

<p ALIGN="CENTER"><font face="Arial" color="#004080"
size="4"><u><strong>Base de datos:</strong></u></font><font
face="Arial" color="#004080"><small>  </small></font></p>

<div align="center">
<center>

<table BORDER="1" CELLPADDING="8" CELLSPACING="1">
  <tr>
    <td bgcolor="#C0C0C0"><font face="Verdana" size="1"
color="#000000"><strong>Frutas</strong></font></td>
    <td bgcolor="#C0C0C0"><font face="Verdana" size="1"
color="#000000"><strong>Precios de
    Exportación por Unidad </strong></font></td>
    <td bgcolor="#C0C0C0"><font face="Verdana" size="1"
color="#000000"><strong>Unidades</strong></font></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>Navel
    Oranges - Naranjas tradicionales </small></strong></small></small></font></td>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>US$2.49</small></strong></small><<
/small></font></td>
    <td bgcolor="#FFFFFF"><font face="Verdana"
color="#000040"><small><small><strong><small>pound
    -Kilo</small></strong></small></small></font></td>
  </tr>
```

```
  <tr>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>Strawberries
    - Fresas</small></strong></small></small></font></td>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>US$2.99</small></strong></small><<
/small></font></td>
    <td bgcolor="#FFFFFF"><font face="Verdana"
color="#000040"><small><small><strong><small>quart
    - cuarto</small></strong></small></small></font></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>Limes
    - Limones</small></strong></small></small></font></td>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>US$1.49</small></strong></small><<
/small></font></td>
    <td bgcolor="#FFFFFF"><font face="Verdana"
color="#000040"><small><small><strong><small>pound-
Kilo</small></strong></small></small></font></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFFF"><font color="#000040"
face="Verdana"><small><small><strong><small>Granny
    Smith Apples - Manzanas Granny Smith
Apples</small></strong></small></small></font></td>
    <td bgcolor="#FFFFFF"><small><small><font color="#000040"
face="Verdana"><strong><small>US$1.99</small></strong></font></small></small><<
/td>
    <td bgcolor="#FFFFFF"><font face="Verdana"
color="#000040"><small><small><strong><small>pound-
Kilo</small></strong></small></small></font></td>
  </tr>
</table>
</center>
</div>

<p align="center"> </p>

<p align="center"><font face="Verdana"
color="#0080FF"><small><small><strong><u>jcanales:</u>
Demostración XHTML</strong></small></small></font></p>

</body>
</html>
```

CAPÍTULO 7:

Anexo 7-1: Listados para ejemplificar un uso de XML y ASP.

- **menu_inferior.xml:**

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<menu_inferior>
	<canal>
		<nombre>Corporativo</nombre>

	<referencia>/modulos/noticias/constructor/corporativo.asp?id_canal_menu=34</r
eferencia>
	</canal>
	<canal>
		<nombre>Productos</nombre>

	<referencia>/modulos/noticias/constructor/servicios_moviles.asp?id_canal_menu
=35</referencia>
	</canal>
	<canal>
		<nombre>Mercado</nombre>
		<referencia>


/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=82&amp;id_canal_menu=
3
	  </referencia>
	</canal>
	<canal>
		<nombre>Servicio Cliente</nombre>
		<referencia>http://www.amarillas.cl</referencia>
	</canal>
	<canal>
		<nombre>Tienda Movil</nombre>
		<referencia>

/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=211&amp;id_canal_menu
=38</referencia>
	</canal>
</menu_inferior>
```

- **menu_inferior.xsl:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                                xmlns:user="http://www.nsb.cl"
                                version="1.0"
>
<!-- Se especifica que XML se va utilizar →
<xsl:template match="/recordset" mode="menu_inferior">
<table width="760" border="0" cellspacing="0" cellpadding="0"
```

```
bgcolor="164591" height="30">
  <tr align="center" valign="Middle">
        <td>
                <table width="470">
                        <tr align="center" valign="Middle">
                                <td align="center"><a href="/"><font
class="menu">Home</font></a></td>

<!-- entra en un ciclo mientras exista el tag CANAL en el XML anterior →
                                <xsl:for-each select="/recordset/menu_inferior/canal">
                        <td width="5"></td>
                                <td align="center">
                        <!—Se colocará un link →
                                        <a>
                                <!—se toma el valor que se encuentra en el TAG
referencia del XML
                                dejándolo como el href del link →
                                  <xsl:attribute name="href">
                                      <xsl:value-of select="referencia"/>
                                  </xsl:attribute>
                                <font class="menu">
                                 <!—se toma el valor que se encuentra en el TAG
nombre del XML
                                dejándolo como el texto que posee el link →
                                    <xsl:value-of select="nombre"/>
                                </font>
                               </a>
                              </td>
<!—FIN del Ciclo → </xsl:for-each>
                        </tr>
                </table>
        </td>
  </tr>
</table>

</xsl:template>
</xsl:stylesheet>
```

- **docente.asp: (http://www.educarchile.cl/modulos/noticias/constructor/docente.asp)**

```
<%@ EnableSessionState=False%>
<!-- #INCLUDE VIRTUAL = "/modulos/seguridad/persona_RS.asp" -->
<%

Dim rsPersona, id_directorio, dir_actual, strParametros

' Comentario: DECLARACION Y RECEPCION DE QUERYSTRING

XSL = Request.QueryString("XSL")

if XSL="" then
```

```
        XSL= "docente"
end if

select case XSL
        case "estudiante"
                dir_actual = 1
        case "docente"
                dir_actual = 2
        case "investigador"
                dir_actual = 3
        case "familia"
                dir_actual = 13
        case else
                dir_actual = 37
end select

' Comentario: CREACIÓN CONEXIÓN RecordSet
Set rsPersona  = crear_rsPersona()
rsPersona.AddNew
rsPersona("Str_Conexion") = Application("Contenido_ConnectionString")
rsPersona.Update

' Comentario:  Se crea el objeto para extraer datos de la persona

Set objPersona = CreateObject("EBT_Persona12.persona")
id_directorio = objPersona.get_id_directorio

If id_directorio <> 0 then
        id_sub_tipo_directorio = objPersona.get_id_sub_tipo_directorio
        set rs_user = objPersona.buscar(rsPersona)
        nombre = rs_user("nombre") & " " & rs_user("ap_paterno")
        user_owa = rs_user("username") &  Application("str_dominio_owa")
        pass_owa = rs_user("password")
        server_owa = Application("str_servidor_exchange")
        username = rs_user("username")
else
        id_sub_tipo_directorio = 0
End If

Set objPersona=nothing
Set rsPersona= nothing

' Comentario:  Parametros a ingresar para paso de correo OWA: Outlook Web
Access
If id_directorio<>0 then
%>
  <!-- #INCLUDE VIRTUAL = "/modulos/contenidos/include/include_owa.asp" -->
<%
end if
' Comentario: Paso de parametros al XSL respectivo por XML
strParametros = "<parametros><escritorio>" & XSL & "</escritorio>"
strParametros = strParametros & "<id_sub_tipo_directorio>" & id_sub_tipo_directorio
& "</id_sub_tipo_directorio>"
```

```
strParametros = strParametros & "<ip_server>" & Application("str_servidor_owa") &
"</ip_server>"
strParametros = strParametros & "<username>" & username & "</username>"
strParametros = strParametros & "<correos>" & intMensajes & "</correos>"
strParametros = strParametros & "<XSL>" & XSL & "</XSL>"
strParametros = strParametros & "<nombre>" & nombre & "</nombre></parametros>"

' Comentario: Se obtiene el texto XML
Set xmlObj = Server.CreateObject("NSB_sql2xml5.sql_xml")
xmlObj.Str_Conexion = Application("Contenido_ConnectionString")

' Comentario: Se incluyen archvivos XML
xmlObj.XML_1 = "docente"
xmlObj.XML_2 = "portada"

' Comentario: Se especifica el template
xmlObj.XSL = "docente"
xmlObj.Str_Sql1   = "edt_busca_servicios " & id_directorio & ", " & dir_actual
xmlObj.Str_tabla1 = "escritorio"
xmlObj.Texto_XML = strParametros
xmlObj.transformar()
' Comentario: Se destruye el objeto
Set xmlObj = nothing
%>
```

- **docente.xsl: Recibe los parámetros de docente.asp y da el formato de la salida.**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:user="http://www.nsb.cl"
                version="1.0"
>
<xsl:output method="html" version="4.0" encoding="ISO-8859-1"/>
<xsl:template match="/">
<HTML>
<HEAD>
        <title>Educarchile</title>
        <link rel="stylesheet" href="/StyleSheets/ms.css" type="text/css"/>
        <script language="JavaScript" src="/modulos/javascript/general_grafica.js"/>
        <script language="JavaScript" src="/modulos/javascript/java_esqueleto.js"/>
        <script language="JavaScript" src="/modulos/javascript/Javacontenido.js"/>
        <script language="JavaScript" src="/modulos/javascript/new_windows.js" />
</HEAD>
<body  bgcolor="#FFFFFF" text="#000000" leftmargin="0" topmargin="0"
marginwidth="0" marginheight="0">
<xsl:attribute name="onLoad">
        <xsl:choose>

<xsl:when test="/recordset/parametros/escritorio =
'estudiante'">MM_preloadImages('/imagenes/estudiantes/txt_mapadelsitio_on.gif','/ima
```

```
genes/estudiantes/txt_ayuda_on.gif','/imagenes/estudiantes/txt_contactenos_on.gif','/ima
genes/estudiantes/busavanzadaover.gif','/imagenes/estudiantes/agendaover.gif','/imagene
s/estudiantes/contactoover.gif','/imagenes/estudiantes/tareasover.gif','/imagenes/estudiant
es/correoover.gif','/imagenes/estudiantes/portadaover.gif','/imagenes/estudiantes/mapaov
er.gif','/imagenes/estudiantes/ayudaover.gif','/imagenes/estudiantes/contactenosover.gif')
</xsl:when>

<xsl:when test="/recordset/parametros/escritorio =
'docente'">MM_preloadImages('/imagenes/familia/txt_mapadelsitio_on.gif','/imagenes/
familia/txt_ayuda_on.gif','/imagenes/familia/txt_contactenos_on.gif','/imagenes/docentes
/busavanzadaover.gif','/imagenes/docentes/agendaover.gif','/imagenes/docentes/contacto
over.gif','/imagenes/docentes/tareasover.gif','/imagenes/docentes/correoover.gif','/imagen
es/docentes/portadaover.gif','/imagenes/docentes/mapaover.gif','/imagenes/docentes/ayu
daover.gif','/imagenes/docentes/contactenosover.gif')</xsl:when>

<xsl:when test="/recordset/parametros/escritorio =
'familia'">MM_preloadImages('/imagenes/familia/portadaover.gif','/imagenes/familia/m
apaover.gif','/imagenes/familia/ayudaover.gif','/imagenes/familia/contactenosover.gif','/i
magenes/familia/busavanzadaover.gif','/imagenes/familia/agendaover.gif','/imagenes/fa
milia/contactoover.gif','/imagenes/familia/tareasover.gif','/imagenes/familia/correoover.g
if')</xsl:when>

<xsl:when test="/recordset/parametros/escritorio =
'investigador'">MM_preloadImages('/imagenes/investigadores/txt_mapadelsitio_on.gif'
,'/imagenes/investigadores/txt_ayuda_on.gif','/imagenes/investigadores/txt_contactenos_
on.gif','/imagenes/docentes/busavanzadaover.gif','/imagenes/docentes/agendaover.gif','/i
magenes/docentes/contactoover.gif','/imagenes/docentes/tareasover.gif','/imagenes/docen
tes/correoover.gif','/imagenes/docentes/portadaover.gif','/imagenes/docentes/mapaover.gi
f','/imagenes/docentes/ayudaover.gif','/imagenes/docentes/contactenosover.gif')</xsl:whe
n>

<xsl:when test="/recordset/parametros/escritorio =
'generico'">MM_preloadImages('/imagenes/home/txt_mapadelsitio_on.gif','/imagenes/h
ome/txt_ayuda_on.gif','/imagenes/home/txt_contactanos_on.gif','/imagenes/home/txt_po
rtada_on.gif')</xsl:when>

</xsl:choose>
</xsl:attribute>
<!--     <xsl:choose>-->
<!-- Comentario: NICIO div Y MENU DERECHO-->
<!--                <xsl:when test="/recordset/parametros/escritorio = 'estudiante'"><div
id="lyrmnuder" style="LEFT: 615px; POSITION: absolute; TOP: 90px; WIDTH:
136px; Z-INDEX: 1"><xsl:apply-templates
mode="menu_derecho_estudiante"/></div></xsl:when>
                <xsl:when test="/recordset/parametros/escritorio = 'docente'"><div
id="lyrmnuder" style="LEFT: 616px; POSITION: absolute; TOP: 100px; VISIBILITY:
visible; WIDTH: 108px; Z-INDEX: 1"><xsl:apply-templates
mode="menu_derecho_docente"/></div></xsl:when>
                <xsl:when test="/recordset/parametros/escritorio = 'familia'"><div
id="lyrmnuder" style="position:absolute; width:136px; z-index:1; left: 615px; top:
90px"><xsl:apply-templates mode="menu_derecho_familia"/></div></xsl:when>
                <xsl:when test="/recordset/parametros/escritorio = 'investigador'"><div
id="lyrmnuder" style="LEFT: 614px; POSITION: absolute; TOP: 90px; VISIBILITY:
```

```
visible; WIDTH: 136px; Z-INDEX: 1"><xsl:apply-templates
mode="menu_derecho_investigador"/></div></xsl:when>
                    <xsl:when test="/recordset/parametros/escritorio =
'generico'"><xsl:apply-templates mode="menu_derecho_generico"/></xsl:when>
          </xsl:choose> -->

<TABLE border="0" cellPadding="0" cellSpacing="0" width="760" valign="top">
  <TBODY>
  <TR>
    <TD colSpan="2">

    <!-- Comentario: xxxxxxxxxxxxxxxx inicio menu superior
xxxxxxxxxxxxxxxxxxxx-->

    <xsl:choose>
                    <xsl:when test="/recordset/parametros/escritorio = 'estudiante'">
                                        <xsl:apply-templates mode="menu_superior_estudiante"/>
                    </xsl:when>
                    <xsl:when test="/recordset/parametros/escritorio = 'docente'">
                                        <xsl:apply-templates mode="menu_superior_docente"/>
                    </xsl:when>
                    <xsl:when test="/recordset/parametros/escritorio = 'familia'">
                                        <xsl:apply-templates mode="menu_superior_familia"/>
                    </xsl:when>
                    <xsl:when test="/recordset/parametros/escritorio = 'investigador'">
                                        <xsl:apply-templates
mode="menu_superior_investigador"/>
                    </xsl:when>
                    <xsl:when test="/recordset/parametros/escritorio = 'generico'">
                                        <xsl:apply-templates mode="menu_superior_generico"/>
                    </xsl:when>
    </xsl:choose>
    <!--xxxxxxxxxxxxx fin  menu superiorxxxxxxxxxxxxxxxxx-->
    </TD>
    </TR>
  <TR>
    <TD vAlign="top" width="615" align="rigth" >

    <!-- Comentario:xxxxxxxxxxxxx inicio central xxxxxxxxxxxx-->

<table width="615" border="0" cellspacing="0" cellpadding="0">
        <tr>

          <!--<a
href="javascript:busc_newWindow2('/funchile/invita/personalizar.htm','name')">
          prueba </a>-->

          <td width="1" bgcolor="#3366FF"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
          <td width="21"><img src="/imagenes/0.gif" width="1" height="10"/></td>
          <td colspan="3" width="593"><img src="/imagenes/0.gif" width="1"
height="10"/></td>
        </tr>
```

```
      <tr>
        <td width="1" bgcolor="#3366FF"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
        <td width="26"><img src="/imagenes/0.gif" width="26" height="1"/></td>
        <td width="196">
          <table width="196" border="0" cellspacing="0" cellpadding="0">
            <tr align="center">

              <td class="homefamilia" colspan="2">
              <table border="0" width="139" height="32">
                                                  <tr>
                                                          <td class="homefamilia"
width="138" height="29"><img src="/imagenes/noticias1.gif" width="134"
height="11"/><img src="/imagenes/noticias2.gif" width="11" height="70"/>

                                                          <xsl:if
test="/recordset/titulares/titular_principal/imagen[normalize-space(.)!=']">




<a>
<xsl:if test="string(number(/recordset/titulares/titular_principal/referencia))!='NaN'">
                                                                                  <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of select="/recordset/titulares/titular_principal/referencia" disable-output-escaping =
"yes"/>&amp;xsl=articulo_docente&amp;esc=docente</xsl:attribute>

                                                                                  </xsl:if>

                                                                                  <xsl:if
test="string(number(/recordset/titulares/titular_principal/referencia))='NaN'">
                                                                                  <xsl:attribute
name="href">http://<xsl:value-of
select="/recordset/titulares/titular_principal/referencia" disable-output-escaping =
"yes"/></xsl:attribute>

                                                                                  </xsl:if>

                                                                                  <img width="70"
height="70" border="0">
                                                                                  <xsl:attribute
name="src"><xsl:value-of
select="/recordset/titulares/titular_principal/imagen"/></xsl:attribute>
                                                                                  </img>
                                                                                  </a>

                                                          </xsl:if>

                                                          </td>
                                          </tr>
                                          </table>
                                  </td>
```

```
            </tr>
            <tr>
              <td class="homefamilia" width="16"><img
src="/imagenes/docentes/cuadradoplomo.gif" width="12" height="8"/></td>
                                        <td class="docentestitnoticia" width="180">
                                        <a class="docentestitnoticia">
                                        <!--xsl:attribute name="title">
                                        <xsl:value-of
select="/recordset/titulares/titular_principal/titulo" disable-output-escaping = "yes"/>
                                        </xsl:attribute-->
                                        <xsl:attribute
name="HREF">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:valu
e-of select="/recordset/titulares/titular_principal/referencia"/></xsl:attribute><xsl:value-
of select="/recordset/titulares/titular_principal/titulo" disable-output-escaping =
"yes"/></a>
</td>
            </tr>
            <tr>
              <td class="normal" width="16"> </td>
              <td class="normal" width="180"><xsl:value-of
select="/recordset/titulares/titular_principal/bajada" disable-output-escaping =
"yes"/></td>
            </tr>
            <tr>
              <td class="normal" width="16"> </td>
              <td class="docentestitnoticia" width="180"> </td>
            </tr>
            <tr>
              <td class="normal" width="16"><img
src="/imagenes/docentes/cuadradoplomo.gif" width="12" height="8"/></td>
                                        <td class="docentestitnoticia" width="180"><a
class="docentestitnoticia">
                                        <!--xsl:attribute name="title">
                                        <xsl:value-of
select="/recordset/titulares/titular1/titulo" disable-output-escaping = "yes"/>
                                        </xsl:attribute-->
                                        <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of select="/recordset/titulares/titular1/referencia"/></xsl:attribute><xsl:value-of
select="/recordset/titulares/titular1/titulo" disable-output-escaping = "yes"/></a>
</td>
            </tr>
            <tr>
              <td class="normal" width="16"> </td>
              <td class="normal" width="180"> </td>
            </tr>
            <tr>
              <td class="normal" width="16"><img
src="/imagenes/docentes/cuadradoplomo.gif" width="12" height="8"/></td>
                                        <td class="docentestitnoticia" width="180">
                                        <a class="docentestitnoticia">
                                        <!--xsl:attribute name="title">
```

```
                                        <xsl:value-of
select="/recordset/titulares/titular2/titulo" disable-output-escaping = "yes"/>
                                        </xsl:attribute-->
                                        <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of select="/recordset/titulares/titular2/referencia"/></xsl:attribute><xsl:value-of
select="/recordset/titulares/titular2/titulo" disable-output-escaping = "yes"/></a> </td>
          </tr>
          <tr>
            <td class="normal" width="16"> </td>
            <td class="normal" width="180"> </td>
          </tr>
          <tr>
            <td class="normal" width="16"><img
src="/imagenes/docentes/cuadradoplomo.gif" width="12" height="8"/></td>
                                        <td class="docentestitnoticia" width="180"><a
class="docentestitnoticia">
                                        <!--xsl:attribute name="title">
                                        <xsl:value-of
select="/recordset/titulares/titular3/titulo" disable-output-escaping = "yes"/>
                                        </xsl:attribute-->
                                        <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of select="/recordset/titulares/titular3/referencia"/></xsl:attribute><xsl:value-of
select="/recordset/titulares/titular3/titulo" disable-output-escaping = "yes"/></a> </td>
          </tr>
          <tr>
            <td class="normal" width="16"> </td>
            <td class="normal" width="180"> </td>
          </tr>
          <tr>
            <td class="normal" width="16"><img
src="/imagenes/docentes/cuadradoplomo.gif" width="12" height="8"/></td>
                                        <td class="docentestitnoticia" width="180">
                                        <a class="docentestitnoticia">
                                        <!--xsl:attribute name="title">
                                        <xsl:value-of
select="/recordset/titulares/titular4/titulo" disable-output-escaping = "yes"/>
                                        </xsl:attribute-->
                                        <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of select="/recordset/titulares/titular4/referencia"/></xsl:attribute><xsl:value-of
select="/recordset/titulares/titular4/titulo" disable-output-escaping = "yes"/></a> </td>
          </tr>
          <tr>
            <td class="normal" width="16"> </td>
            <td class="normal" width="180"> </td>
          </tr>
        </table>
      </td>
      <td width="37"><img src="/imagenes/0.gif" width="37" height="1"/></td>
      <td valign="top" width="355">
        <table width="355" border="0" cellspacing="0" cellpadding="0">
```

```
            <tr>
              <td width="355">
                <table width="355" border="0" cellspacing="0" cellpadding="0">
                  <tr>
                    <td align="right" width="270"><img
src="/imagenes/docentes/destacados1.gif" width="115" height="11"/></td>
                    <td rowspan="2" width="83" align="left">
                  <!--inicio de la seccion nueva destacados-->


                    <table border="0" width="83" height="1">
                                                <tr>
                                                 <td width="90" height="1" align="left">
                                                 <xsl:if
test="/recordset/titulares/destacados/titular/imagen[normalize-space(.)!=']">

                                                 <a><xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of
select="/recordset/titulares/destacados/titular/referencia"/>&amp;xsl=articulo_docente&
amp;esc=docente</xsl:attribute>
                                                 <img width="70" height="70" border="0">
                                                 <xsl:attribute name="src"><xsl:value-of
select="/recordset/titulares/destacados/titular/imagen"/></xsl:attribute>
                                                 </img>
                                                 </a>
                                                 </xsl:if>
                                                 <img src="/imagenes/destacados1.gif"
width="10" height="53"/>
                                                 <img src="/imagenes/destacados2.gif"
width="80" height="12"/>
                                                 </td>
                                                </tr>
                                          </table>


                    <!-- Comentario: fin nueva seccion destacados-->
                    </td>
                  </tr>
                  <tr>
                    <td width="270">
                      <table width="270" border="0" cellspacing="0" cellpadding="0">
                        <tr>
                          <td colspan="2"><img src="/imagenes/0.gif" width="1"
height="10"/></td>
                        </tr>
                        <tr>
                          <td width="158">
                            <table width="100%" border="0" cellspacing="0" cellpadding="0">
                              <tr>
                                <td align="left"><img src="/imagenes/docentes/cuadrados.gif"
width="60" height="6"/></td>
                              </tr>
```

```
                    <tr>
                      <td bgcolor="#0033CC"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
                    </tr>
                  </table>
                </td>
                              <td class="homedocentes" width="117"><a
class="homedocentes">
                              <!--xsl:attribute name="title">
                              <xsl:value-of
select="/recordset/titulares/destacados/titular/titulo" disable-output-escaping = "yes"/>
                              </xsl:attribute-->
                              <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of
select="/recordset/titulares/destacados/titular/referencia"/>&amp;xsl=articulo_docente&
amp;esc=docente</xsl:attribute>
                                        <xsl:value-of
select="/recordset/titulares/destacados/titular/titulo" disable-output-escaping =
"yes"/></a></td>
                  </tr>
                  <tr>
                    <td colspan="2" class="normal"><xsl:value-of
select="/recordset/titulares/destacados/titular/bajada" disable-output-escaping =
"yes"/></td>
                  </tr>
                </table>
              </td>
            </tr>
          </table>
        </td>
      </tr>
      <tr>
        <td width="355" align="left"><img src="/imagenes/0.gif" width="1"
height="10"/><br/>
          <table width="320" border="0" cellspacing="0" cellpadding="0">
            <tr>
              <td>
                <table width="100%" border="0" cellspacing="0" cellpadding="0">
                  <tr>
                    <td align="left"><img src="/imagenes/docentes/cuadrados.gif"
width="60" height="6"/></td>
                  </tr>
                  <tr>
                    <td bgcolor="#0033CC"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
                  </tr>
                </table>
              </td>
                                    <td class="homedocentes"><a
class="homedocentes">
                                    <!--xsl:attribute name="title">
                                    <xsl:value-of
```

```
select="/recordset/titulares/destacados/titular1/titulo" disable-output-escaping = "yes"/>
                                                                    </xsl:attribute-->
                                                                    <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of
select="/recordset/titulares/destacados/titular1/referencia"/>&amp;xsl=articulo_docente
&amp;esc=docente</xsl:attribute>
<xsl:value-of select="/recordset/titulares/destacados/titular1/titulo" disable-output-
escaping = "yes"/></a></td>
                  </tr>
                  <tr>
                                                                    <td colspan="2"
class="normal"><xsl:value-of select="/recordset/titulares/destacados/titular1/bajada"
disable-output-escaping = "yes"/></td>
                  </tr>
                 </table>
               </td>
             </tr>
             <tr>
               <td width="355"><img src="/imagenes/0.gif" width="1" height="10"/></td>
             </tr>
             <tr>
               <td width="355">
                 <table width="320" border="0" cellspacing="0" cellpadding="0">
                   <tr>
                     <td>
                       <table width="100%" border="0" cellspacing="0" cellpadding="0">
                         <tr>
                           <td align="left"><img src="/imagenes/docentes/cuadrados.gif"
width="60" height="6"/></td>
                         </tr>
                         <tr>
                           <td bgcolor="#0033CC"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
                         </tr>
                       </table>
                     </td>
                                                                    <td class="homedocentes"><a
class="homedocentes">
                                                                    <!--xsl:attribute name="title">
                                                                    <xsl:value-of
select="/recordset/titulares/destacados/titular2/titulo" disable-output-escaping = "yes"/>
                                                                    </xsl:attribute-->
                                                                    <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of
select="/recordset/titulares/destacados/titular2/referencia"/>&amp;xsl=articulo_docente
&amp;esc=docente</xsl:attribute>
<xsl:value-of select="/recordset/titulares/destacados/titular2/titulo" disable-output-
escaping = "yes"/></a></td>
                  </tr>
                  <tr>
                                                                    <td colspan="2"
```

```
class="normal"><xsl:value-of select="/recordset/titulares/destacados/titular2/bajada"
disable-output-escaping = "yes"/></td>
                  </tr>
                </table>
              </td>
            </tr>
            <tr>
              <td width="355"><img src="/imagenes/0.gif" width="1" height="10"/></td>
            </tr>
            <tr>
              <td width="355">
                <table width="320" border="0" cellspacing="0" cellpadding="0">
                  <tr>
                    <td>
                      <table width="100%" border="0" cellspacing="0" cellpadding="0">
                        <tr>
                          <td align="left"><img src="/imagenes/docentes/cuadrados.gif"
width="60" height="6"/></td>
                        </tr>
                        <tr>
                          <td bgcolor="#0033CC"><img src="/imagenes/0.gif" width="1"
height="1"/></td>
                        </tr>
                      </table>
                    </td>
                                                        <td class="homedocentes"><a
class="homedocentes">
                                                        <!--xsl:attribute name="title">
                                                        <xsl:value-of
select="/recordset/titulares/destacados/titular3/titulo" disable-output-escaping = "yes"/>
                                                        </xsl:attribute-->
                                                        <xsl:attribute
name="href">/modulos/noticias/constructor/detalle_noticias.asp?id_noticia=<xsl:value-
of
select="/recordset/titulares/destacados/titular3/referencia"/>&amp;xsl=articulo_docente
&amp;esc=docente</xsl:attribute>
<xsl:value-of select="/recordset/titulares/destacados/titular3/titulo" disable-output-
escaping = "yes"/></a></td>
                  </tr>
                  <tr>
                    <td colspan="2" class="normal"><xsl:value-of
select="/recordset/titulares/destacados/titular3/bajada" disable-output-escaping =
"yes"/></td>
                  </tr>
                </table>
              </td>
            </tr>
            <tr>
              <td width="355"><img src="/imagenes/0.gif" width="1" height="10"/></td>
            </tr>
            <tr>
                                                <td width="355" class="normalazul"
align="right">
```

```
                                        <a
href="/modulos/noticias/constructor/buscar.asp?id_canal=6&amp;XSL=docente_listado
" class="normalazul">otros artículos  </a><img
src="/imagenes/docentes/minibox.gif" width="9" height="9"/>  </td>
            </tr>
            <tr>
              <td width="355"><img src="/imagenes/0.gif" width="1" height="10"/></td>
            </tr>
          </table>
        </td>
      </tr>
    </table>



   <!-- Comentario: xxxxxxxxxxxxxxxfin  centralxxxxxxxxxxxxxxxxxx-->
   </TD>
   <!-- Comentario: xxxxxxxxxxxxxinicio background xxxxxxxxxxxx-->
   <xsl:choose>
                        <xsl:when test="/recordset/parametros/escritorio = 'estudiante'">
                                <td width="145" valign="top" align="center"
bgcolor="#BEF841" background="/imagenes/estudiantes/fondomenu.gif">
                                        <xsl:apply-templates
mode="menu_derecho_estudiante"/>
                                        <!--img src="/imagenes/estudiantes/fondomenu.gif"
width="145" height="351"/-->
                                </td>
                </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'docente'">
                                <td width="145" valign="top"  align="center"
background="/imagenes/docentes/mnuder_00.gif">
                                <xsl:apply-templates mode="menu_derecho_docente"/>
                                <!--IMG height="351"
src="/imagenes/docentes/mnuder_00.gif" width="145"/-->
                                </td>
                </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'familia'">
                                <td width="145" valign="top" bgcolor="#F6F6D1"
background="/imagenes/familia/mnu_der00.gif">
                                                <xsl:apply-templates
mode="menu_derecho_familia"/>
                                        <!--img src="/imagenes/familia/mnu_der00.gif"
width="145" height="351"/-->
                                </td>
                </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio =
'investigador'">
                                <td width="145" valign="top" bgcolor="#F0DFFB"
background="/imagenes/investigadores/fondomenu.gif">
                                        <xsl:apply-templates
mode="menu_derecho_investigador"/>
                                        <!--img
src="/imagenes/investigadores/fondomenu.gif" width="145" height="331"/-->
```

```
                                        </td>
                        </xsl:when>
                                <xsl:when test="/recordset/parametros/escritorio = 'vitrina'">
                                        <td valign="top" align="right" bgcolor="#FEF2D9"
background="/imagenes/mediateca/mt_mnuder.gif">
                                                <xsl:apply-templates
mode="menu_derecho_vitrina"/>
                                                <!--img src="/imagenes/mediateca/mt_mnuder.gif"
width="145" height="548"/-->
                                        </td>
                            </xsl:when>
                            <xsl:when test="/recordset/parametros/escritorio = 'generico'">
                                        <td valign="top" align="right" bgcolor="#FEF2D9"
background="/imagenes/mediateca/mt_mnuder.gif">
                                                <xsl:apply-templates
mode="menu_derecho_generico"/>
                                                <!--img src="/imagenes/mediateca/mt_mnuder.gif"
width="145" height="548"/-->
                                        </td>
                            </xsl:when>
    </xsl:choose>
    <!-- Comentario: xxxxxxxxxxxxx fin  background xxxxxxxxxxxx-->
  </TR>
  <TR>
    <TD colSpan="2">
    <!-- Comentario: xxxxxxxxxxxxxxxxinicio inferiorxxxxxxxxxxxxxxxxxxxxxxx-->
      <xsl:choose>
                                <xsl:when test="/recordset/parametros/escritorio = 'estudiante'">
                                        <xsl:apply-templates mode="menu_inferior_estudiante"/>
                        </xsl:when>
                                <xsl:when test="/recordset/parametros/escritorio = 'docente'">
                                        <xsl:apply-templates mode="menu_inferior_docente"/>
                        </xsl:when>
                                <xsl:when test="/recordset/parametros/escritorio = 'familia'">
                                        <xsl:apply-templates mode="menu_inferior_familia"/>
                        </xsl:when>
                                <xsl:when test="/recordset/parametros/escritorio =
'investigador'">
                                        <xsl:apply-templates
mode="menu_inferior_investigador"/>
                        </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'generico'">
                                        <xsl:apply-templates mode="menu_inferior_generico"/>
                        </xsl:when>
      </xsl:choose>

      <!--fin  inferior-->
      </TD>
  </TR>
  </TBODY>
  </TABLE>

      <!-- Comentario: asocia los maps para cada uno de los escritorios-->
```

```
        <xsl:choose>
                        <xsl:when test=" /recordset/parametros/escritorio = 'estudiante' ">
                                <MAP  name="Map">
                                <AREA shape="RECT" coords="16,17,139,54"
href="/modulos/noticias/constructor/estudiante.asp" />
                                <AREA shape="RECT" coords="189,1,351,44" href="/"
/>
                                </MAP>

                        </xsl:when>
                        <xsl:when test=" /recordset/parametros/escritorio = 'docente'">
                                <MAP  name="Map">
                                <AREA shape="RECT" coords="11,4,191,57" href="/" />
                                <AREA shape="RECT" coords="368,16,587,66"
href="/modulos/noticias/constructor/docente.asp"/>
                                </MAP>
                        </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'familia' ">
                                <MAP  name="Map">
                                <AREA shape="RECT" coords="85,13,261,65"
href="/modulos/noticias/constructor/familia.asp" />
                                <AREA shape="RECT" coords="319,10,496,584"
href="/"/>
                                </MAP>
                        </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'investigador'
">
                                <MAP  name="Map">
                                <AREA shape="RECT" coords="133,12,309,57" href="/"
/>
                                <AREA shape="RECT" coords="337,14,564,62"
href="/modulos/noticias/constructor/investigador.asp"/>
                                </MAP>
                        </xsl:when>
                        <xsl:when test="/recordset/parametros/escritorio = 'generico' ">
                                <MAP  name="Map">
                                <AREA shape="RECT" coords="46,16,213,63" href="/"
/>
                                </MAP>
                        </xsl:when>
                        <xsl:otherwise>
                                no resulto Map
                        </xsl:otherwise>
                </xsl:choose>

</body>
 </HTML>
</xsl:template>
<xsl:include href="menu_superior_estudiante.xsl"/>
<xsl:include href="menu_derecho_estudiante.xsl"/>
<xsl:include href="menu_inferior_estudiante.xsl"/>
<xsl:include href="menu_superior_docente.xsl"/>
<xsl:include href="menu_derecho_docente.xsl"/>
```

```
<xsl:include href="menu_inferior_docente.xsl"/>
<xsl:include href="menu_superior_familia.xsl"/>
<xsl:include href="menu_derecho_familia.xsl"/>
<xsl:include href="menu_inferior_familia.xsl"/>
<xsl:include href="menu_superior_investigador.xsl"/>
<xsl:include href="menu_derecho_investigador.xsl"/>
<xsl:include href="menu_inferior_investigador.xsl"/>
<xsl:include href="menu_superior_generico.xsl"/>
<xsl:include href="menu_derecho_generico.xsl"/>
<xsl:include href="menu_inferior_generico.xsl"/>
</xsl:stylesheet>
```

CAPÍTULO 9:

Anexo 9-1: Listado para ejemplificar el uso de CDF.

```
<?xml version="1.0" encoding="UTF-8"?>
<Channel HREF="http://www.northsupply.cl/canal_noticias.html" IsClonable="NO">
  <Title VALUE="Canal Noticias"/>
  <Abstract VALUE="Un canal que permite mantener informado en cada momento del
acontecer Nacional."/>
  <Author VALUE="Jorge Canales"/>
  <Publisher VALUE="NSCH"/>
  <Copyright VALUE=" NSCH "/>
  <PublicationDate VALUE="2002.01.19"/>
  <LastMod VALUE="2002.01.20"/>
  <Schedule>
    <StartDate VALUE="2002.01.19"/>
    <EndDate VALUE="2002.02.01"/>
    <IntervalTime DAY="7"/>
    <EarliestTime HOUR="2"/>
    <LatestTime HOUR="6"/>
  </Schedule>
  <Logo HREF="http://www.northsupply.cl/imagen/logo01.gif"
    TYPE="REGULAR"/>
  <Logo HREF=" http://www.northsupply.cl/imagen/logo02.gif "
    TYPE="WIDE"/>
  <Logo HREF=" http://www.northsupply.cl/imagen/logo03.gif "
    TYPE="SMALL"/>
</Channel>
```

Anexo 9-2: Listados para ejemplificar el uso del P3P Policy Editor.

```
<?xml version="1.0" ?>
<!--DOCTYPE PROPOSAL SYSTEM
"http://privacy.linkexchange.com/xml/syntax.dtd" -->

<PROPOSAL
  xmlns:VOC="http://privacy.linkexchange.com/xml/vocab.dtd"
  xmlns:DATA="http://privacy.linkexchange.com/xml/basedata.dtd"
```

```
  realm="http://www.northsupply.com"
  entity="North Supply">
<USES>
  <STATEMENT VOC:purp="4 "
   VOC:recpnt="0 "
   VOC:id="1" >
    <DATA:REF category="0"/>
    <DATA:REF category="1"/>
  </STATEMENT>
  <STATEMENT VOC:purp="0 1 "
   VOC:recpnt="0 "
   VOC:id="1"   >
    <DATA:REF category="2"/>
  </STATEMENT>
  <STATEMENT VOC:purp="0 "
   VOC:recpnt="0 2"
   VOC:id="1"   >
    <DATA:REF category="3"/>
    <DATA:REF category="6"/>
  </STATEMENT>
  <STATEMENT VOC:purp="2 "
   VOC:recpnt="0 "
   VOC:id="0"   >
    <DATA:REF category="7"/>
  </STATEMENT>
  <STATEMENT VOC:purp="0 1 "
   VOC:recpnt="0 "
   VOC:id="0" >
    <DATA:REF category="4"/>
    <DATA:REF category="5"/>
  </STATEMENT>
  <STATEMENT VOC:purp="0 2"
   VOC:recpnt="0 "
   VOC:id="1" >
    <DATA:REF category="9"/>
  </STATEMENT>
</USES>
<VOC:DISCLOSURE discURI="http://www.northsupply.com/policy.htm"
 access="1" other="0"/>
</PROPOSAL>
```

Anexo 9-3: Listados para ejemplificar el uso de VoxML en el modelo de venta de automóviles para una aplicación de voz.

```
<?xml version="1.0"?>
<DIALOG>
<CLASS NAME="help_generic">
<HELP> La posibles elecciones son<OPTIONS/>. </HELP>
</CLASS>
<STEP NAME="init" PARENT="help_generic">
  <PROMPT> Bienvenido a North Supply applications. Puede elegir la compra de su
```

```
modelo de automóvil
  </PROMPT>
   <INPUT TYPE="OPTIONLIST">
      <OPTION NEXT="#Ford"> ford </OPTION>
      <OPTION NEXT="#Volvo"> volvo </OPTION>
      <OPTION NEXT="#Chevrolet"> chevrolet </OPTION>
      <OPTION NEXT="#Peugeot"> peugeot </OPTION>
      <OPTION NEXT="top.vml#top"> menu principal </OPTION>
   </INPUT>
</STEP>

<STEP NAME="Ford" PARENT="help_generic">
<PROMPT> Cual es su modelo favorito en la serie</PROMPT>
<INPUT TYPE="OPTIONLIST">
 <OPTION NEXT="#serie1"> serie1 </OPTION>
 <OPTION NEXT="#serie2"> serie2 </OPTION>
</INPUT>
</STEP>

<STEP NAME="serie1" PARENT="help_generic">
<PROMPT> Esta serie Ford incluye ABS y el costo de este modelo es de quince
millones
<BREAK SIZE="LARGE"/> </PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
</STEP>

<STEP NAME="serie2" PARENT="help_generic">
<PROMPT> Esta serie Ford no incluye ABS y el costo del modelo es de nueve
millones<BREAK SIZE="LARGE"/>
</PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
 </STEP>

<STEP NAME="Volvo" PARENT="help_generic">
<PROMPT> Cual es su modelo favorito en la serie</PROMPT>
<INPUT TYPE="OPTIONLIST">
 <OPTION NEXT="#V1"> v1 </OPTION>
 <OPTION NEXT="#V2"> v2 </OPTION>
</INPUT>
</STEP>
<STEP NAME="V1" PARENT="help_generic">
<PROMPT> Esta serie Volvo incluye ABS, el costo de este modelo es de catorce
millones
<BREAK SIZE="LARGE"/> </PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
</STEP>

<STEP NAME="V2" PARENT="help_generic">
<PROMPT>Esta serie Volvo incluye opciones  de color. <BREAK SIZE="LARGE"/>
</PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
 </STEP>
<STEP NAME="Chevrolet" PARENT="help_generic">
```

```
<PROMPT> Cual es su modelo favorito en la serie</PROMPT>
<INPUT TYPE="OPTIONLIST">
 <OPTION NEXT="#Luv"> luv </OPTION>
 <OPTION NEXT="#Cavallier"> cavalier </OPTION>
</INPUT>
</STEP>

<STEP NAME="Luv" PARENT="help_generic">
<PROMPT> Esta serie Luv incluye acoplado y el costo de este modelo es de siete
millones
<BREAK SIZE="LARGE"/> </PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
</STEP>

<STEP NAME="Cavallier" PARENT="help_generic">
<PROMPT> Este modelo Chevrolet  dispone de ABS y airbag <BREAK
SIZE="LARGE"/>
</PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
 </STEP>

<STEP NAME="Peugeot" PARENT="help_generic">
<PROMPT> Cual es su modelo favorito en la serie</PROMPT>
<INPUT TYPE="OPTIONLIST">
 <OPTION NEXT="#404"> 404 </OPTION>
 <OPTION NEXT="#505"> 505 </OPTION>
</INPUT>
</STEP>

<STEP NAME="404" PARENT="help_generic">
<PROMPT> Esta serie Peugeot incluye ABS y el costo de este modelo es de trece
millones
<BREAK SIZE="LARGE"/> </PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
</STEP>
<STEP NAME="505" PARENT="help_generic">
<PROMPT>Este modelo incluye ABS y  variedad de cinco colores <BREAK
SIZE="LARGE"/>
</PROMPT>
<INPUT TYPE="NONE" NEXT="#init"/>
 </STEP>

<INPUT TYPE="NONE" NEXT="#init"/>
</STEP>
</DIALOG>
```