UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA – LEON

FACULTAD DE CIENCIAS

DEPARTAMENTO DE COMPUTACION

MONOGRAFÍA.

"GENERACIÓN DE DOCUMENTOS XML BIEN FORMADOS Y VALIDOS".

ELABORADO POR:

Br. MARJORIE DEL SOCORRO PICADO PACHECO.
Br. NUBIA MERCEDES TERCERO CHEVEZ.

TUTORES.

Ing. RICARDO ESPINOZA.
Lic. NESTOR CASTRO.

LEÓN, 26 DE SEPTIEMBRE DEL 2005.

INDICE.

AGRADECIMIENTOS.

Hoy que al fin culminé mi trabajo monográfico quiero dar gracias primeramente a Dios que estuvo conmigo en todo memento regalándome salud, conocimientos, fuerzas para finalizar esta etapa tan importante para mí.

Gracias a Dios Jesucristo, Maria Santísima por permitirme terminar mi tesis. Agradezco profundamente a mi Madre que con su amor, dedicación, comprensión y apoyo económico me ayudó a estudiar y por ende a culminar la realización de mi preparación profesional.

A mi abuelita que a pesar de la distancia siempre estuvo pendiente de mí, brindándome su cariño, comprensión y apoyo económico para estudiar y terminar mi carrera.

Marjorie del Socorro Picado Pacheco.

Al finalizar mi trabajo monográfico quiero darle gracias a Dios y a nuestra Madre Santísima por haberme permitido terminar este trabajo con mucho éxito, también porque me iluminaron a cada momento dándome sabiduría, entusiasmo y fuerzas.

Agradezco también a mi familia que estuvo conmigo a cada momento y me brindaron su apoyo incondicional para realizar este trabajo.

A mis maestros que me brindaron su apoyo, conocimiento y supieron guiarme en este trabajo.

Nubia Mercedes Tercero Chévez.

DEDICATORIA.

Quiero dedicar mi monografía a Dios, Jesucristo y Maria, pues es gracias a ellos que logré finalizar este trabajo ya que en todo momento me regalaron salud, conocimiento y entusiasmo a mi familia y a mí.

Mamá, también quiero dedicar mi tesis a usted ya que ha estado conmigo en todo momento, en las buenas y en las malas, en todas esas noches en yo me desvelé, usted también se desveló conmigo por todas esas cosas y por lo mucho que la quiero, le dedico mi trabajo monográfico.

Otra persona muy importante para mí es mi abuelita que aunque estemos separadas por la distancia siempre hemos estado unidas por el amor, a ella también dedico mi monografía con mucho cariño.

Marjorie del Socorro Picado Pacheco.

Dedico este trabajo monográfico, primeramente a Dios Nuestro Señor Jesucristo y a nuestra Madre Santísima por permitirme llegar hasta esta etapa.

A mi familia que en todo momento de este trabajo sentí el apoyo de ellos, especialmente el de mi madre y mis hermanos que me motivaron y me dieron aliento para culminar este trabajo.

Nubia Mercedes Tercero Chévez.

INTRODUCCIÓN.

Hoy en día por la necesidad existente de intercambiar información electrónica entre dos o más ordenadores, se hace necesario la creación de un lenguaje que permita este intercambio de una manera más sencilla.

La forma más común para realizar dicho intercambio ha resultado ser la Web (Internet). Pero la compatibilidad existente entre las diferentes plataformas provoca el gran problema de no poder compartir la información de una forma sencilla y directa.

La existencia de múltiples soluciones trajo como consecuencia que la información contenida en la red de ordenadores era un almacén con mucho contenido pero sin orden ni estructura. En este momento, la empresa IBM creó GML (Generalized Markup Language o Lenguaje de Marcado Generalizado), para la edición y manipulación de documentos en la red, sin embargo surgieron varios problemas.

Es por ello que las diferentes organizaciones se han dado a la ardua tarea de crear un leguaje que venga a resolver estos problemas, ya que una vez creado un lenguaje que pueda soportar la representación genérica de un documento, sin importar el hecho de que cada aplicación tenga su propio lenguaje y por último el hecho de que un ordenador no tiene la capacidad de comprender por si mismo la información que almacena.

Como solución a estos problemas se introdujo el concepto de marcado, separando dentro de un mismo documento el contenido, de la estructura del mismo.

Una vez definido el marcado en los documentos, lo único que faltaba era conseguir que este marcado fuese universal, es decir, que se realizara bajo

unas normas fijas que le permitieran ser un lenguaje de referencia para todos, por lo que se trabajó en la estandarización del lenguaje.

En la década de los 70 apareció SGML (Lenguaje de Marcado Generalizado Estándar), que se correspondía con una versión de GML, aceptado como estándar por la organización ISO. Este estándar tuvo mucho éxito y aceptación, se realizaron muchos trabajos para su adaptación a las necesidades existentes, pero no llegó a imponerse totalmente por su complejidad y porque requería mucho software para su proceso.

La necesidad de comunicación e interacción seguía existiendo y aumentando a medida que el uso de ordenadores se extendía como herramienta en muchas áreas de la actividad humana, como consecuencia lógica de esta necesidad, surgieron otras iniciativas. En 1989 el Laboratorio Europeo de Física de Partículas (CERN), creó un lenguaje de marcado. El objetivo de dicho lenguaje era el de poder compartir documentos entre sus equipos de trabajo a través de Internet, era una aplicación sencilla y simple de SGML, que se llamó HTML.

El lenguaje HTML nace en 1991 de manos de Tim Bernes-Lee del CERN como un sistema hipertexto con el único objetivo de servir como medio de transmisión de información entre físicos de alta energía como parte de la iniciativa WWW. En 1993 Dan Connelly escribe el primer DTD (Document Type Definition) de SGML describiendo el lenguaje. En 1994 el sistema había tenido tal aceptación que la especificación se había quedado ya obsoleta. Por aquel entonces WWW y Mosaic eran casi sinónimos debido a que el browser Mosaic del NCSA (National Center for Supercomputing Applications) era el más extendido debido a las mejoras que incorporaba. Es entonces cuando nace el HTML 2.0 en un draft realizado también por Dan Connelly. El crecimiento exponencial que comienza a sufrir el sistema lleva a organizar la First International WWW Conference en Mayo de 1994. El principal avance de 2.0 de HTML es la incorporación de los llamados forms, formularios que permiten que el usuario cliente envíe información al servidor y ésta sea recogida y procesada allí.

La aparición de este lenguaje de etiquetas supuso un gran avance en el uso de la red para el intercambio de información debido a su sencillez ya que permitía crear un documento en pocos minutos y requería un software sencillo para su manipulación. La idea inicial de HTML consistía en describir cómo eran los documentos que se deseaba intercambiar y que una herramienta que se llamó navegador, decidiera cómo se debía mostrar dicho documento. Los documentos eran hiperdocumentos ya que contenían capacidades hipertextuales mediante las cuales se podía crear saltos o enlaces entre documentos.

HTML no permite definir de forma estricta la apariencia de una página, aunque una utilización algo desviada hace que se utilice en ocasiones como un lenguaje de presentación. Además, la presentación de la página es muy dependiente del browser (o programa navegador) utilizado: el mismo documento no produce el mismo resultado en la pantalla si se visualiza con un browser en modo línea, Mosaic o Netscape, o sea, HTML se limita a describir la estructura y el contenido de un documento, y no el formato de la página y su apariencia.

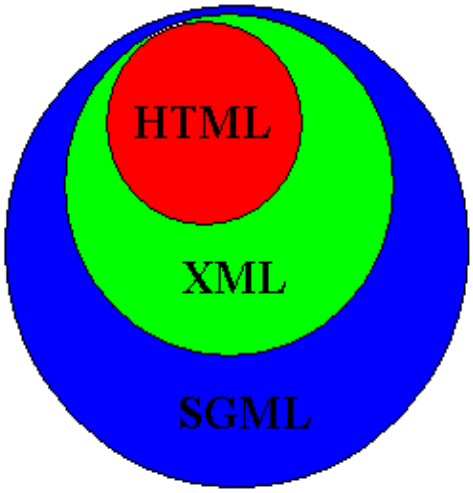
La gran utilidad de HTML hizo que en muy poco tiempo, múltiples organizaciones adoptaran su uso y se inició de esta forma una alocada carrera por trasladar información a este formato y por dotar a esta tecnología de más y más posibilidades, empezando por posibilidades multimedia (imagen, sonido y otros) y por otras que dieran una mayor interactividad a la información mostrada. HTML se expandió de forma tan rápida y desorganizada, guiada por organizaciones que pretendían únicamente fines comerciales que finalmente se utilizó para fines no previstos en sus orígenes, creando así la maraña de información más grande y desorganizada que nunca ha existido recibiendo el nombre de WWW (World Wide Web, Telaraña de ámbito mundial).

Actualmente, se está utilizando HTML como interfaz universal para todo tipo de aplicaciones por lo sencillo que resulta y la obtención automática de millones de clientes capaces de manejar ese interfaz. Debido a esto, HTML se ha quedado pequeño para ciertas aplicaciones, y sobre todo, se ha hecho necesario dotar a la Web de una estructura más semántica que permita

organizar la telaraña. Por todo esto, el Consorcio de la World Wide Web (W3C) intentó introducir SGML en la Web. Eso supondría solucionar los problemas existentes con el HTML, pero debido a la dificultad de esta tarea, el W3C se vio obligado a buscar alguna alternativa que tuviera las ventajas de SGML y la sencillez de HTML.

Finalmente en 1998, el W3C hizo público la normalización del estándar XML (Lenguaje de Marcado eXtensible). XML es la alternativa buscada consiguiendo ser muy potente (no tanto como SGML) y muy sencillo (no tanto como HTML).

Mediante XML se podrá definir el HTML, con lo que se podría considerar los siguientes conjuntos:



Con este trabajo se dará a conocer los elementos necesarios que se utilizan para crear documentos XML bien formados y válidos.

XML no va a sustituir a HTML, éste seguirá siendo el lenguaje que se utilizará para la representación de los datos, pero el intercambio de datos se hará en XML.

ANTECEDENTES.

A pesar de ser un lenguaje nuevo, innovador, atractivo y que está evolucionando con rapidez aún no se han realizado trabajos anteriores por lo cual no podemos mencionar referencias.

JUSTIFICACIÓN.

La creciente expansión que está tomando el estándar XML en la Web, hace pensar a muchos expertos que va a ser utilizado en el futuro por varias empresas y usuarios en el intercambio de información, debido a su sencillez y a las utilidades que ofrece.

Debido a la necesidad de enviar grandes cantidades de información y al crecimiento que está tomando el lenguaje XML hemos decidido desarrollar este tema para dar a conocer aspectos generales de este nuevo lenguaje, los elementos básicos que se necesitan para la creación de documentos y mediante ejemplos, mostrar cómo se generan documentos XML bien formados y válidos.

OBJETIVO GENERAL.

Mostrar mediante ejemplos la generación de documentos XML bien formados y válidos.

OBJETIVOS ESPECÍFICOS.

1. Dar a conocer las principales sentencia del lenguaje XML.
2. Mostrar la importancia de crear documentos XML bien formados.
3. Mostrar la importancia de crear documentos XML válidos, utilizando la técnica de DTD interna y externa.

1. INTRODUCCIÓN AL ESTÁNDAR XML.

1.1 ¿Qué es XML?

XML es el estándar de Extensible Markup Language, es un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructuradas.

1.2 Objetivos de XML:

- Utilizable para transmitir datos sobre Internet.
- Soporta una amplia variedad de aplicaciones.
- Es compatible con SGML.
- Facilita la creación de programas que procesan documentos XML.
- El número de características opcionales en XML debe ser mínima.
- Los documentos XML deben ser legibles por usuarios, razonablemente claros, formales y concisos.
- Los documentos XML deben ser fácilmente creables.
- La brevedad en las marcas XML es de mínima importancia.

1.3 Principales características

- Es una arquitectura más abierta y extensible. No necesita versiones para que pueda funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el momento de utilizar internet/intranet por medio de un validador de documentos (parser).

- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML (los RDF Resource Description

FrameWork), en comparación a los atributos de la etiqueta <META> del HTML.

- Integración de los datos en plataformas diferente. Se podrá hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local o extensa.

- Los datos están compuestos por múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permitirá agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos.

- Gestión y manipulación de los datos desde el propio cliente web.

- Los motores de búsqueda devolverán respuestas más adecuadas y precisas, ya que la codificación del contenido web en XML consigue que la estructura de la información resulte más accesible.

- Se desarrollarán de manera extensible las búsquedas personalizables y subjetivas para robots y agentes inteligentes. También conllevará que los clientes web puedan ser más autónomos para desarrollar tareas que actualmente se ejecutan en el servidor.

- Se permitirá un comportamiento más estable y actualizable de las aplicaciones web, incluyendo enlaces bidireccionales y almacenados de forma externa.

- El concepto de "hipertexto" se desarrollará ampliamente, permitirá denominación independiente de la ubicación, enlaces bidireccionales, enlaces que pueden especificarse y gestionarse desde fuera del documento, hiperenlaces múltiples, enlaces agrupados, atributos para

los enlaces, etc. Creado a través del Lenguaje de enlaces extensible (XLL).

- Se podrá exportar datos a otros formatos de publicación (papel, web, cd - rom, etc.). El documento maestro de la edición electrónica podría ser un documento XML que se integraría en el formato deseado de manera directa

2. REGLAS BÁSICAS PARA CREAR DOCUMENTOS XML.

2.1 XML Básico.

Un documento XML es un documento de marcado, lo cual quiere decir que está escrito en texto simple (sin caracteres especiales no visibles que no sean cambios de líneas o tabuladores) que determinadas partes del mismo no son contenido real sino información estructural. Se denomina “XML Básico” ya que las normas establecidas deben ser válidas para todo documento XML.

2.2 Reglas básicas para documentos.

Las normas establecidas para crear documentos XML se pueden dividir en tres elementos:

2.2.1. Análisis y descripción de los componentes de un documento XML.

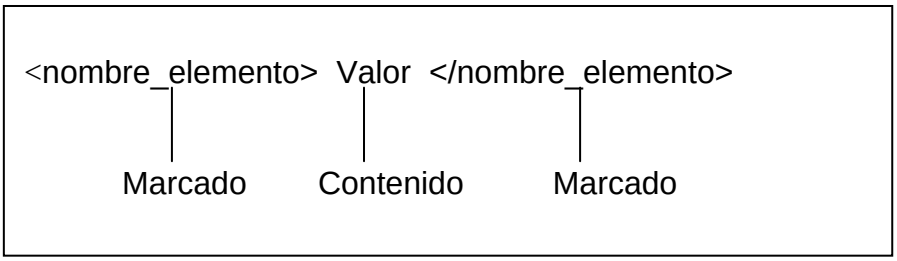
Todo documento XML está formado por cinco componentes:

- Elementos.
- Atributos.
- Valores.
- Separadores.
- Comentarios.

Elemento

Constituyen la base de la estructura de un documento. Se designa elemento al conjunto formado por un cierto marcado y el contenido que delimita, a la vez un elemento consta de una serie de componentes que son:

- Marcado de apertura.
- Contenido.
- Marcado de cierre.



El marcado de apertura se distingue por los signos “<” y “>” y por dentro el nombre del elemento, dicho nombre no podrá contener espacios ni caracteres que indican el marcado de apertura (“>” “<”).

El contenido puede ser cualquier texto, como información importante del documento y se desea que forme parte del elemento. Sin embargo puede ocurrir que exista elemento vació y elemento contraído.
El marcado de cierre es análogo al marcado de apertura y su sintaxis es la siguiente “</”.

Atributos

Los atributos se introducen en el interior de los elementos para agregar información adicional a los mismos, se utilizan para especificar un elemento. Un elemento puede tener los atributos que sean necesarios.

Un atributo se incluye en el marcado de apertura mediante un espacio o más, del nombre del elemento y de otros atributos.

La forma de crear un atributo es el signo "=" y un valor entre comillas, estas comillas pueden ser simples o dobles.
Si el valor de un atributo requiere de comillas de algún tipo y dentro de éste, se desea resaltar otro atributo se deberá utilizar el otro tipo de comillas.

```
<elemento  atributo = " valor" >  Valor  </elemento>
```

Los separadores son caracteres especiales que tienen como propósito separar elementos delimitando su comienzo y final, estos son:

"<" ">" "/". Establecen el límite del marcado. Separan datos de estructura
Comillas de apertura y cierre, simples o dobles, constituyen la separación del contenido de los atributos del resto del marcado.
Espacio, tabulador, cambio de línea, son caracteres no visibles que establecen la separación de elementos contenido en el marcado.

```
<elemento     atributo = "valor" >
        <tabulador>  valor </tabulador>
</elemento>
```

Comentarios.

Tienen como objetivo incluir información al documento que no forma parte del contenido ni del marcado.
Un comentario no necesita marcado de cierre, es un elemento auto contenido especial.

```
< ! - - Este es un comentario - - >
```

2.2.2. Normas generales para documentos:

La estructura de un documento XML debe ser estricta y predecible de tal manera que permita su tratamiento automático.

Elemento raíz:

El elemento raíz debe estar presente en todo documento XML, este elemento contiene a todos los demás, solo puede haber un elemento raíz en todo el documento.

Marcado de cierre:

Es de carácter obligatorio que en todos los elementos se incluya el marcado de cierre, si no es así los elementos deberán ser de tipo vacío con sintaxis abreviada.

Elementos anidados:

Los elementos anidados consisten en elementos que están dentro de otro o dentro del elemento raíz en todo documento XML.

```
<elemento  atributo = “valor”>
        <elemento1> Valor
                <elemento2>  Valor  </elemento2>
                <elemento3>  Valor
                 </elemento3>
         </elemento1>
</elemento>
```

Mayúsculas y minúsculas:

El lenguaje XML es sensible a mayúscula y minúsculas, es decir no es lo mismo escribir “atributo” que “Atributo”. Para evitar estas confusiones es aconsejable establecer normas y cumplirlas.

Atributos:

Los nombres de los atributos deben ser diferentes si se encuentran en el mismo elemento; pero pueden ser iguales si están en distintos elementos.

Uso de las comillas:

Las comillas se utilizan para indicar los valores de los atributos, pueden ser simples o dobles. No se permite valores de los atributos sin comillas.

Separadores y entidades:

Para utilizar los separadores “<”, “>”,”&”, comillas dobles y simples dentro del valor de un elemento o atributo, se deberá usar una sintaxis especial llamada entidades predefinidas, y el usuario puede definir más. Estas cinco entidades predefinidas sustituyen a los separadores, las cuales son:

&amp equivale a: &
&lt equivale a: <
&gt equivale a: >
&quot equivale a: “
&apos equivale a:’

2.2.3. Estructura general:

Un documento XML consta de tres partes:

- Cabecera
- Declaración de la estructura que debe cumplir el documento.
- Elemento raíz.

Cabecera:

Las cabeceras se utilizan para establecer parámetros globales al documento que permite su uso a nivel global en la red.
La cabecera y sus componentes son opcionales. Si los componentes aparecen son los siguientes.

< ?xml versión = “1.0” encoding = “UTF - 8” >

El primero establece la versión de XML que se va a utilizar. Por el momento se utilizará la versión 1.0.
El segundo componente indica el juego de caracteres que se utiliza en el resto del documento, en este caso se utilizará el juego de caracteres Latín. Los posibles valores son: UTF – 8,UTF – 16,EUC – JP,ISO – 10646 –UCS2,ISO – 8859-1 y muchos más
Lo mínimo que puede aparecer en la cabecera es la versión de XML para asegurar que el documento seguirá siendo válido durante largo tiempo, aunque aparezcan nuevas versiones.

Declaración de la estructura que debe cumplir el documento.

Otro elemento es la declaración que se utilizará para establecer cual es la estructura que debe seguir un documento. Esta declaración es opcional, si no se especifica quiere decir que solo es correcto ante las reglas generales de XML y al formato que se está describiendo.

Si ésta declaración aparece puede tener tres tipos diferentes de contenido, pero siempre se ajustará al siguiente formato.

```
<! DOCTYPE nombre_elemento_raiz estructura>
```

El marcado de este componente también es especial, utilizando "<!" como elemento de comienzo del marcado. Este componente tiene por nombre DOCTYPE y va seguido de un nombre del elemento (definido por el usuario) que se creará siguiendo las normas definidas para estos nombres. Este nombre de elemento es el nombre del elemento raíz que debe tener el documento para ser considerado correcto.

Los tres tipos de definiciones DOCTYPE, se diferencian por el contenido que se encuentra en la estructura. Estos tres tipos son:

Definición de estructura interna: este es el caso más sencillo en el cual las definiciones que marcarán la estructura del documento se realizan de forma local en el mismo elemento DOCTYPE.

Definición de estructura externa: en este caso, las definiciones se almacenan en un fichero externo, para poder usar este tipo se necesita incluir en la cabecera la declaración standalone = "yes". En el elemento DOCTYPE se deberá referirse al fichero que contienen las definiciones.

Definición de estructura externa e interna: este es una combinación de las anteriores en las que se tiene referencia a un fichero externo de definiciones y además definiciones internas a continuación. Es importante tener presente que las definiciones internas se imponen a las externas.

Los nombres XML y DOCTYPE usado en la cabecera y declaración de estructura respectivamente son fijos y debe ir en minúscula la primera y el segundo en mayúsculas obligatoriamente.

Elemento raíz.

Es el que contiene toda la información y estructura del documento, incluyendo además otros elementos, atributos y texto creando de esta manera el árbol del documento.

Además de estos componentes, dentro del elemento raíz o de un elemento contenido en él, puede presentarse un último componente de los documentos, éste es la sección CDATA, ésta sirve para incorporar texto en un elemento que no se desea que sea comprobado ni procesado como XML, es un fragmento con otro formato.

3. DIFERENCIAS ENTRE XML Y HTML.

Al hablar de XML se piensa que es igual a HTML, pero más avanzado, debido a esto es necesario mencionar las diferencias que existen entre ambos lenguajes, las cuales son:

- Los elementos que componen a XML pueden dar información sobre lo que contienen, no necesariamente sobre la presentación, como ocurre en HTML.

- XML es un metalenguaje, HTML es simplemente un lenguaje.

- HTML no indica lo que está representando, XML describe el contenido de lo que etiqueta.

- HTML permite la comunicación entre las personas, XML va a permitir la comunicación entre las máquinas.

- HTML es un tipo de documento SGML que se utiliza en la Web

- XML no es un documento SGML, sino que es una versión abreviada de SGML mejorada para su utilización en Internet. Esto significa que con él

se podrá definir documentos propios (el usuario podrá definir sus propias etiquetas) y, por tanto, ya no se dependerá de un único e inflexible tipo de documento HTML.

- XML no trabaja con etiquetas predefinidas, HTML sí.

- La sintaxis es estricta. Ya no vale dejar de entrecomillar los atributos o utilizar las mayúsculas y minúsculas sin ningún control. La especificación XML determina claramente una serie de reglas que especifican cuando un documento está bien formado.

- La utilización de una DTD. En HTML, a pesar de ser una aplicación SGML, no era obligatorio utilizarlas y aunque para trabajar con XML tampoco será necesario, sí que será recomendable. Posiblemente no acompañen al documento XML en su distribución, pero resultan muy útiles en la elaboración y validación de los documentos.

- Los elementos vacíos. Son los elementos del tipo <img>, <hr>, etc. de HTML, en los que no existe etiqueta final al no tener contenido. Ahora, en XML, la propia etiqueta de inicio llevará una contrabarra al final que los identificará.

- Con HTML se pueden hacer accesos a información comparativa en diferentes tiendas por ejemplo, pero nada más. Con XML el usuario podrá ordenar los datos o actualizarlos en tiempo real o realizar un pedido.

- HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que XML es un subconjunto de SGML especializado en la gestión de información para la Web. En la práctica XML contiene a HTML aunque no en su totalidad.

Es importante saber que XML no es un reemplazo para HTML. En el desarrollo futuro de la web es más probable que XML sea utilizado para describir los datos, mientras que el HTML será utilizado para ajustar formato y para exhibir los mismos datos. XML es una plataforma, un software y una herramienta independiente del hardware para transmitir la información.

XML puede separar datos de HTML con XML. Cuando el HTML se utiliza para exhibir datos, los datos se almacenan dentro del HTML, con XML, los datos se pueden almacenar en archivos separados de XML.

Con XML, los datos se pueden intercambiar entre los sistemas incompatibles. En el mundo real, los sistemas informáticos y las bases de datos contienen datos en formatos incompatibles. Unos de los desafíos más desperdiciadores de tiempo para los diseñadores ha sido el intercambiar datos entre diferentes sistemas sobre el Internet, el convertir los datos a XML puede reducir grandemente esta complejidad y crear datos que puedan ser leídos por diversos tipos de aplicaciones.

4. ESTRUCTURA DEL XML.

El metalenguaje XML consta de cuatro especificaciones indicadas abajo, sin embargo el propio XML sienta las bases sintácticas y el alcance de su implementación, estas especificaciones son:

DTD (Document Type Definition): Definición del tipo de documento. Es, en general, un archivo/s que encierra una definición formal de un tipo de documento y, a la vez, especifica la estructura lógica de cada documento. Define tanto los elementos de una página como sus atributos. El DTD del XML es opcional, en tareas sencillas no es necesario construir una DTD, en este caso se trataría de un documento "bien formado"(well-formed), por otra parte si lleva DTD será un documento "validado" (valid).

XSL (eXtensible Stylesheet Language): Define o implementa el lenguaje de estilo de los documentos escritos para XML.

Permite modificar el aspecto de un documento. Se pueden lograr múltiples columnas, texto girado, orden de visualización de los datos de una tabla, múltiples tipos de letra con amplia variedad en los tamaños. Este estándar está basado en el lenguaje de semántica y especificación de estilo de documento, conocido como DSSSL (Document Style Semantics and Specification Language, ISO/IEC 10179) y por otro lado, se considera más potente que las hojas de estilo en cascada (CSS, Cascading Style Sheets), usado en un principio con el lenguaje DHTML. Se espera que el CSS sea usado para visualizar simples estructuras de documentos XML (actualmente se ha conseguido mayor integración en XML con el protocolo CSS2 (Cascading Style Sheets, level 2) ofreciendo nuevas formas de composición y una más rápida visualización y, por otra parte, XSL pueda ser utilizado donde se requiera más potencia de diseño como documentos XML que encierran datos estructurados (tablas, organigramas, etc.).

XLL (eXtensible Linking Language): Define el modo de enlace entre diferentes enlaces. Se considera que es un subconjunto de HyTime (Hipermedia/Timed-based structuring Language o Lenguaje de estructuración hipermedia/basado en el tiempo, ISO 10744) y sigue algunas especificaciones del TEI (Text Encoding Initiative o Iniciativa de codificación de texto). Desde marzo de 1998 el W3C trabajó en los enlaces y direccionamientos del XML. Provisionalmente se le renombró como Xlink y a partir de junio se le denomina XLL. Este lenguaje de enlaces extensible tiene dos importantes componentes: Xlink y el Xpointer. Va más allá de los enlaces simples que sólo soporta el HTML. Se podrá implementar con enlaces extendidos.

Jon Bosak establece los siguientes mecanismos hipertextuales que soportará esta especificación:

- Denominación independiente de la ubicación.
- Enlaces que pueden ser también bidirecccionales.
- Enlaces que pueden especificarse y gestionarse desde fuera del documento a los que se apliquen (Esto permitirá crear en un entorno

intranet/extranet un banco de datos de enlaces en los que se puede gestionar y actualizar automáticamente. No habrá más errores del tipo "404 Not Found").

- Hiperenlaces múltiples (anillos, múltiples ventanas, etc.).
- Enlaces agrupados (múltiples orígenes).
- Transclusión (el documento destino al que apunta el enlace aparece como parte integrante del documento orígen del enlace).
- Se pueden aplicar atributos a los enlaces (tipos de enlaces).

XUA (XML User Agent): Estandarización de navegadores XML. Todavía está en proceso de creación de borradores de trabajo. Se aplicará a los navegadores para que compartan todas las especificaciones XML.

5. VALIDEZ DE UN DOCUMENTO.

Una vez que se ha establecido las características básicas de un documento se debe plantear una cuestión fundamental: ¿A qué se refiere que un documento sea correcto, que cumple las normas generales o las específicas de sus reglas de estructura? La respuesta a esta interrogante es que se debe distinguir entre dos conceptos, los cuales son: “documentos bien formados” y “documentos válidos”.

- **Documentos bien formados:** es un documento que cumple con las normas de la sintaxis general de XML.
- **Documentos válidos**: es un documento bien formado que es conforme a una definición de estructura ya sea interna, externa o ambas.

Es necesario establecer definiciones estructurales que permitan validar documentos para compartirlos en la red, y que puedan ser usados por los usuarios especializados. Para validar un documento se pueden utilizar varios métodos uno de ellos son las **DTDs;** el otro es la utilización de **Esquema.**

Una DTD (Declaración de Tipo de Documento) es la gramática del documento XML, ya que dicho documento tiene que cumplir con la declaración de los elementos, atributos y entidades declaradas en la propia DTD.

Es importante recordar que el uso de la DTD es opcional, es decir se podría utilizar un esquema para validar un documento o simplemente no usar DTD ni Esquema, por lo que el documento no sería valido aunque pueda ser bien formado.

Clasificación de las DTD:

1. Documento basado en una DTD externa.
2. Documento basado en una DTD interna.

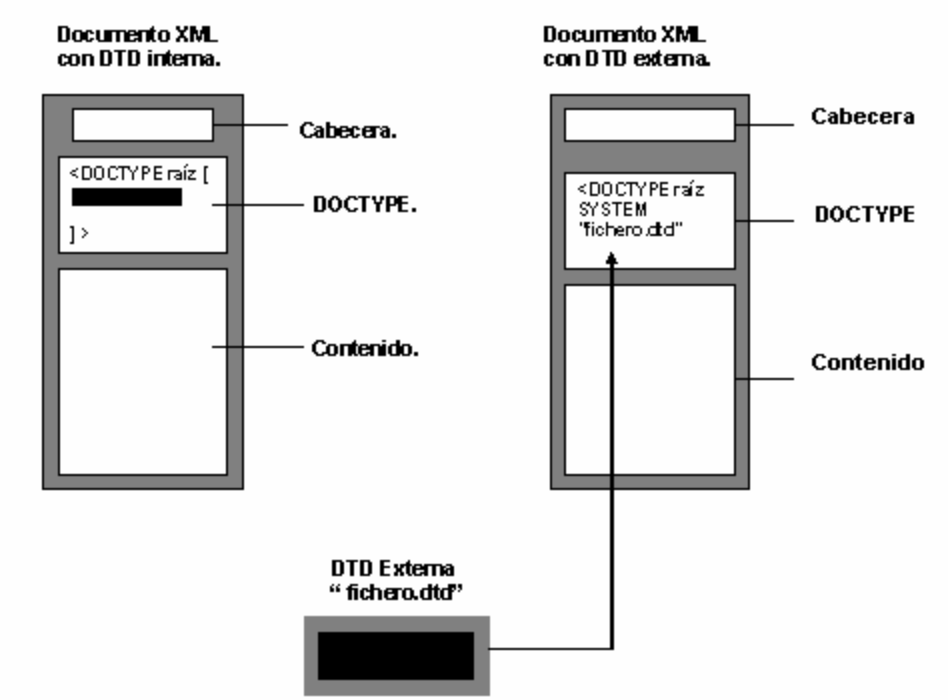
Las ventajas de utilizar DTD externa son las siguientes:

- Permite crear múltiples documentos de un mismo tipo, basándolos en la misma DTD.
- Permite utilizar una DTD ya existente, con el consiguiente ahorro de tiempo y el aumento de compatibilidad con otros autores.

Por otro lado, las ventajas de utilizar una DTD interna son:

- Permite que un documento contenga su propia validación.
- Es útil si se pretende crear un único documento para esa DTD.

En general casi nunca se usarán documentos con DTD interna puesto que la utilización de XML para el intercambio de información hace recomendable el uso de DTD externa.



Los dos tipos de DTD permiten declarar los siguientes tipos de componentes:

- Elementos y sus modelos de contenidos (vacíos, contiene subelemento, mixtos etc.).
- Atributos de cada elemento.
- Entidades para hacer referencia a elementos previamente definidos.
- Instrucciones de procesamiento para aplicaciones.

A pesar que existen dos tipos, solamente se definirá una sintaxis puesto que en ambos casos se debe usar la misma. La idea que se debe manejar es que una DTD externa no es más que un fichero que contiene lo que se colocaría entre los corchetes de una definición de DTD interna.

Para crear DTDs es necesario declarar los elementos que se van a utilizar. La declaración de un elemento empieza con " <**! ELEMENT"** seguido del nombre que se le va a dar en la DTD y por el identificador genérico del tipo que se desea asignar al elemento que se declara. La sintaxis es la siguiente:

```
<!ELEMENT nombre tipo>
```

Donde:

nombre: es el nombre del elemento que se va a declarar.

tipo: es uno de los distinto tipos de elementos que existen.

6. TIPOS DE ELEMENTOS DE UNA DTD.

Los tipos de elementos que se pueden combinar para crear DTDs son los siguientes.

6.1. EMPTY

Este tipo de elemento no tiene contenido. Se suele utilizar conjuntamente con atributos aunque no es necesario. Si no tiene atributos, el uso de este tipo de elemento en un documento servirá para indicar algo si aparece.

Sintaxis.

```
<!ELEMENT nombre EMPTY>
```

Donde:

nombre: es el nombre que se le va a dar al elemento.

EMPTY: indica que el elemento va a ser de tipo vació es decir, sin contenido.

6.2. ANY

Este elemento acepta cualquier tipo de contenido.

Sintaxis.

```
<!ELEMENT nombreElemento ANY>
```

Donde:

nombreElemento: es el nombre que se le va a dar al elemento.

ANY: indica que el tipo de elemento es sin restricciones. El uso de este elemento debe evitarse ya que rompe con unas de las normas más características o generales que tiene el estándar XML, la definición exacta del contenido del elemento, o lo que es lo mismo, el control sobre el contenido del documento.

6.3. Mezcla (MIXED)

Este tipo de elemento puede contener tanto datos de caracteres como elementos secundarios. Todo ello se puede mezclar de la forma que se desee.

Sintaxis.

```
<!ELEMENT nombreElemento (#PCDATA | elemento)*>
```

Donde:

nombreElemento: es el nombre del elemento.

#PCDATA: indica que este elemento será de solo texto.

Hay que prestar atención al carácter " * " ya que en este caso es obligatorio su ubicación. Además, "#PCDATA" tiene que definirse en primer lugar y siempre antes que cualquier elemento secundario.

6.4. Secuencia (sequence of)

Este tipo de elemento sólo contiene otros elementos, que deben declararse después de la declaración del elemento de tipo secuencia.

Sintaxis.

```
<! ELEMENT nombreElemento (subelemento1, subelemento2, subelemento3,...)
indicador>
```

Donde:

nombreElemento: es el nombre con el que se va a identificar al elemento.

"(" ")": rodean el conjunto de elementos que forman la secuencia.

Las ",": indica la secuencia y el orden en el que se han de definir los subelementos.

subelementox: es cada uno de los elementos "hijos" que va a contener "nombreElemento".

indicador: hace referencia al indicador de frecuencia, es decir, el número de veces que se podrá utilizar en un documento XML.

6.5. #PCDATA

Este tipo de elemento solo puede contener texto. Se suele utilizar para elementos que no contengan elementos hijos.

Sintaxis.

```
<!ELEMENT nombreElemento (#PCDATA)>
```

Donde:

nombreElemento: es el nombre que se le va a dar al elemento.

#PCDATA: indica que el contenido de este elemento únicamente podrá ser texto.

6.6. Selección (choice of)

Este tipo de elemento sirve para establecer una lista de subelementos, pero sólo permite que dicho elemento contenga uno de los elementos declarados en la lista.

Sintaxis.

```
<!ELEMENT nombreElemento (subelemento1 | subelemento2 |
subelemento3,…) indicador>
```

Donde:

nombreElemento: es el nombre que va a servir para identificar a este tipo de elemento.

El símbolo " | ": indica elección de un elemento hijo de entre el conjunto de subelementos.

subelementox: son cada uno de los elementos hijo del elemento.

indicador: es un indicador de frecuencia, el cual permite definir el número de veces que podrá aparecer "nombreElemento" en un documento XML.

7. INDICADORES DE FRECUENCIA.

Los elementos pueden ir acompañados de indicadores de frecuencia, es decir, indicadores del número de veces que se deben utilizar en un documento XML.

Tipos de indicadores de frecuencia:

Indicador de frecuencia	**Significado**
Por defecto (default)	Sin indicador de frecuencia, el estándar XML entiende que dicho elemento aparecerá una única vez en el documento XML y es obligatorio.
?	Indica que este elemento únicamente podrá aparecer una vez o ninguna, es por tanto opcional.
+	El elemento es obligatorio y debe aparecer una vez como mínimo, no hay restricciones en cuanto al número máximo de veces que puede aparecer.
*	Indica que el elemento es opcional, ya que no puede aparecer o aparecer un número indeterminado de veces.

8. DECLARACIONES DE ATRIBUTOS.

Los atributos permiten agregar información a un elemento. La diferencia entre atributos y elementos son principalmente las siguientes:

- Un atributo no puede contener subatributos ni subcomponentes en general.
- Los atributos solo pueden ser usados una vez para un elemento, y pueden aparecer en cualquier orden indistintamente de cómo se declararan en la DTD.

Al momento de crear un atributo se debe tener en cuenta lo siguiente:

- Los atributos no pueden contener textos complejos.
- El texto contenido en los elementos no se puede restringir ni en tamaño ni en contenido.
- Los atributos tienen tipos especiales con valores muy restringidos y características especiales que algunas ocasiones los hacen ser la única elección.
- Un elemento, se puede modificar, un atributo no.

La declaración de un atributo es:

```
<!ATTLIST elemento atributo tipo presencia valorPorDefecto>
```

Donde:

elemento: indica el nombre del elemento al que está asociado el atributo que se declara.

atributo: es el nombre que se le da al atributo.

tipo: indica el tipo de atributo.

presencia: es el indicador de presencia, es decir, es el que determina si el atributo es obligatorio, opcional o tiene valor por defecto.

valorPorDefecto: es el valor que va a tomar el atributo por defecto, es decir, en caso de que no se asigne ningún valor al mismo en el documento.

9. VALORES POR DEFECTO.

El apartado “valorPorDefecto” puede tener los siguientes valores.

Tipo de valor	**Significado.**
#REQUIRED	No tiene valor por defecto, es obligatorio especificarlo.
#IMPLIED	Se puede omitir, es decir, puede que no tome un valor. Es opcional. Si no se le da a un atributo un valor de forma explícita, se le puede dar un valor cuando se utilice en el documento.
#FIXED valor	Indica que si el atributo toma un valor fijo y el cual no se podrá cambiar, es decir, siempre que se inicialice un atributo aparecerá con este valor.
Valor	Se le da este valor al atributo por defecto en caso de que no se le asigne a la hora de usarlo en el documento.

10. TIPOS DE ATRIBUTOS.

Algunos de los tipos de atributos que se pueden utilizar para crear DTD son:

10.1. CDATA:

Son aquellos atributos que vienen especificados por el tipo CDATA. Permiten introducir cadenas de texto. Sintaxis:

```
<! ATTLIST nombreElemento nombreAtributo CDATA presencia
valorPorDefecto>
```

Donde:

nombreElemento: es el nombre del elemento al cual pertenece el atributo.

nombreAtributo: es el nombre que se le va a dar al atributo.

CDATA: es el identificativo del atributo. Indica que el atributo declarado va a contener una cadena de texto.

presencia: indica si el atributo es obligatorio, opcional, o tiene un valor fijo.

valorPorDefecto: permite especificar el valor obligatorio o el que tomaría en caso de no incluirse en la declaración del documento.

10.2. ID:

Permite identificar a un elemento. En un documento sólo puede haber un atributo del tipo ID con un cierto valor, todo los atributos ID de un documento tienen que tener diferentes valores. El valor de un atributo ID debe comenzar por una letra o guión de subrayado, seguido de letras, números, guiones de subrayado o puntos.

Sintaxis.

```
<!ATTLIST nombreElemento nombreAtributo ID presencia
valorPorDefecto>
```

Donde:

nombreElemento: es el nombre del elemento al cual pertenece el atributo.

nombreAtributo: es el nombre que se le va a dar al atributo.

presencia: indica si el atributo es obligatorio, opcional, o tiene un valor fijo.

valorPorDefecto: esta opción es válida en caso de que en “presencia” se ponga “#FIXED”.

10.3. ENTITY:

Hace referencia a un objeto que no debe ser analizado sintácticamente según las reglas de XML.

Sintaxis:

```
<! ATTLIST nombreElemento nombreAtributo ENTITY presencia
valorPorDefecto>
```

Donde:

nombreElemento: es el nombre del elemento al cual pertenece el atributo.

nombreAtributo: es el nombre que se le va a dar al atributo.

presencia: indica si el atributo es obligatorio, opcional, o tiene un valor fijo.

valorPorDefecto: esta opción es válida en caso de que en “presencia” se ponga “#FIXED”.

11. COMO TRABAJAR CON XML.

Para trabajar con XML hacen falta una serie de herramientas, que se debe tener instaladas en la máquina.
La funcionalidad de estas herramientas se detalla a continuación:

11.1. Editores XML.

Un editor XML es una aplicación que ofrece facilidades para crear y editar documentos XML.
XML es un texto basado en lenguaje de marca. Una gran cosa de XML es que los archivos pueden ser creados y corregidos usando un simple editor de texto como Bloc de Notas. Sin embargo, cuando se inicia a trabajar con XML enseguida se encontrará que es mejor corregir los documentos XML usando un editor profesional de XML.

Muchos diseñadores de la web usan Bloc de Notas para corregir documentos HTML y XML por que éste se incluye con los Sistemas Operativos más utilizados y es simple de usar. Pero si se usa Bloc de Notas para corregir documentos XML pronto se encontrarán problemas, por que Bloc de notas no sabe que se está escribiendo documentos XML así que no podrá ser capaz de corregir los documentos.

Hoy en día XML es una tecnología importante, y los diseñadores de proyectos utilizan tecnologías basadas en XML como:

- Schema XML para definir estructuras y tipos de datos XML.
- XSLT para transformar datos XML.
- SOAP para intercambiar datos entre aplicaciones.

- WSDL para describir servicios web.
- RDF para describir recursos web.
- XPath y XQuery para acceso a datos XML.
- SMIL para definir gráficos.

Para poder escribir los documentos XML sin error se necesita un editor inteligente de XML.
Los editores profesionales de XML ayudan a escribir documentos XML sin error, validar documentos contra una DTD o Schema. Un editor XML debería ser capaz de:

- Agregar automáticamente las etiquetas de cierre.
- Forzar a escribir documentos XML válidos.
- Verificar los documentos XML contra una DTD.
- Verificar los documentos XML contra un Schema.
- Proporcionar color a los códigos de sintaxis de los documentos XML.

Para empezar a editar documentos XML es recomendable utilizar un simple editor de texto y, una vez familiarizados con la sintaxis y características del lenguaje, se pueden utilizar editores específicos.)

Se pueden diferenciar dos tipos de editores XML:

- Los que representan el fichero XML en forma de árbol
- Los que presentan el documento XML en su formato original

Los del primer tipo presentan la estructura del documento XML en forma de árbol y permiten construir el documento trabajando sobre este árbol y formularios adicionales.

Algunos de este tipo son:

- **XML Notepad**. Es un editor de XML desarrollado por Microsoft. Para su utilización es necesario tener instalado, como mínimo, la versión 4.01 del Explorer, aunque sólo puede ser aprovechado en su totalidad con la versión 5.

- **Visual XML**. Es un editor de XML escrito en Java con JFC (Swing). Su autor es Pierre Morel.

Los del segundo tipo son editores normales de ficheros de texto, pero con facilidades de edición enfocadas al XML.

Algunos de estos son:

- **XED**, es un editor de XML desarrollado por Henry Thompson. Permite garantizar que el autor no va a escribir documentos que no estén bien formados y puede leer la DTD para sugerir la introducción de elementos válidos.

- **PSGML para Emacs**. Es un modo superior de Emacs para trabajar con SGML que se ha modificado para soportar XML. Lee la DTD, puede utilizar un analizador externo para validar documentos, realiza coloración de sintaxis y otras muchas cosas.

En ambos tipos hay que diferenciar los que trabajan contra una DTD y, por lo tanto, validan el contenido de lo que se escribe, y los que simplemente aseguran que el documento XML es bien formado, es decir, sintácticamente correcto respecto de las especificaciones del XML.

La elección de uno u otro dependerá del tipo de documento que se desee escribir. Si se desea escribir un documento XML (tipo la información que se introduce en una base de datos) es conveniente utilizar el del primer tipo. En

cambio, si se quiere escribir un documento con gran cantidad de texto (por ejemplo un manual) vendrá mejor uno del segundo tipo. En cualquiera de los dos casos es recomendable que permita trabajar sobre una DTD.

Cabe resaltar que una DTD se define como la estructura de un documento XML, es decir, los elementos que formarán ese tipo de documento y cómo están relacionados. A diferencia que en SGML, en XML no es obligatorio crearlas, aunque es recomendable por que nos facilitará la validación de documentos.
Algunos ejemplos de editores DTD son el TDTD para Emacs o el EZDTD que también trabaja con SGML.

11.2. Parser para XML.

Un parser o procesador de XML es la herramienta principal de cualquier aplicación XML. Mediante este parser no sólo se podrá comprobar si los documentos están bien formados o válidos, sino que también se permitirá incorporar a las aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML.

De acuerdo con su función, los procesadores XML se dividen en dos tipos: validadores y no-validadores. En común tienen, que ambos deben informar de las violaciones de las restricciones de documentos bien formados dadas en su especificación, además, los procesadores validadores, deben informar de la violación de las restricciones expresadas por las declaraciones de la DTD.

Actualmente hay muchos y para todos los lenguajes y plataformas: Java, C, Phyton, Visual Basic, Perl, Tcl, Delphi, etc., aunque los parsers en Java son mayoría, debido a que el Java y el XML se complementan muy bien. El XML contribuye con datos independientes de la plataforma (documentos y datos portables). Java contribuye con el procesamiento independiente de la plataforma (soluciones de software portátiles orientadas a objetos).

Ya están disponibles varios de estos parsers XML:

PARSER	VERSION	PLATAFORMA
CMarkup	7.3	C++, Visual C++ (MFC)
Disk DOM	1	Java
DTDParser	1.21	Java (version para Window, Unix, Linux.)
DTDParser	2.0	Java
Expat	1.95.8	C
FastParser	1.6.9.1	Java
Fusion XML DOM MicroParser	1.1	C
Fusion XML SAX MicroParser	1.1	C
Fxp	2.0	CML
GNOME XML Library	2.6.1.9	Unix/ Linux Window
HXML	0.2	Haskell
JAXP	1.2	Java
Lark	1.0	Java
Larval	8	Java
LibXMLingh	0.1.1	C++
MXP1: XML Pull parser 3 rd Edición (XPP3)	1.1.3.4	Java, C++
Oracle XML parser para C	9.2.0.4	Linux, Solaris 2.6 y NT 4/ Service Pack 3
Oracle XML parser para Java	9.2.0.4	JDK 1.1.x o superior, Windows, Unix.
Parsifal	0.9.1	Windows, Unix (C)
PXP	1.1	O Caml
RXP	1.4.0	Unix, Win32
SP	1.3.4	C++
TclXML	3.0	Tcl
Tony	0.9	Objective Caml
Xerces_C++	2.6.0	Win32, Linux, Unix
X_Fetch Performer	2.1	Java
XJParser	1.0	Java, todos los distribuidos por Windows 95, 98, NT y Solaris.
X erces2_ J	2.6.2	Java
XML para Java	4.3	Linux, para toda plataforma java.
XML para wxWindows	0.6.0	wxWindows (C++)

11.3. Navegadores para XML.

Es primordial saber qué ofrece el futuro y qué se podrá obtener de XML como lenguaje de publicación en Internet. Para ello hace falta conocer de qué forma soportarán los navegadores el nuevo lenguaje.

El problema surge del hecho de que el navegador debe incluir de algún modo un parser para XML y un motor que acepte las hojas de estilo (XSL o CSS).
La mayoría de los navegadores del Internet ofrecen ayuda a XML. Sin embargo, no todos los navegadores tienen apoyo total para XML, XML Namespaces, y XSLT.

El Internet Explorer ofrece la siguiente ayuda a XML:

1. La visión del documento XML.
2. Apoyo total para los estándares DTD de la W3C.
3. XML ajustado en el HTML como islas de datos.
4. Uniendo datos de XML a los elementos del HTML.
5. Transformar y exhibir XML con XSL.
6. Exhibir XML con CSS.
7. Acceso para XML DOM.

El Internet Explorer: Internet Explorer 6 posee ayuda completa para XML, incluyendo la ayuda para Namespaces, las hojas de estilo en el CSS así como XSLT 1.0

El Internet Explorer 5: El Internet Explorer 5 también tiene ayuda de XML, pero la pieza de XSL no es compatible con la recomendación del funcionario W3C XSL.

Firefox 1.0.2: el navegador Firefox tiene ayuda para XML y XSLT (y XML + CSS).

Mozilla 1.8: Mozilla incluye Expat para XML que analiza y tiene ayuda para exhibir XML + CSS. Mozilla también tiene cierta ayuda para Namespaces.

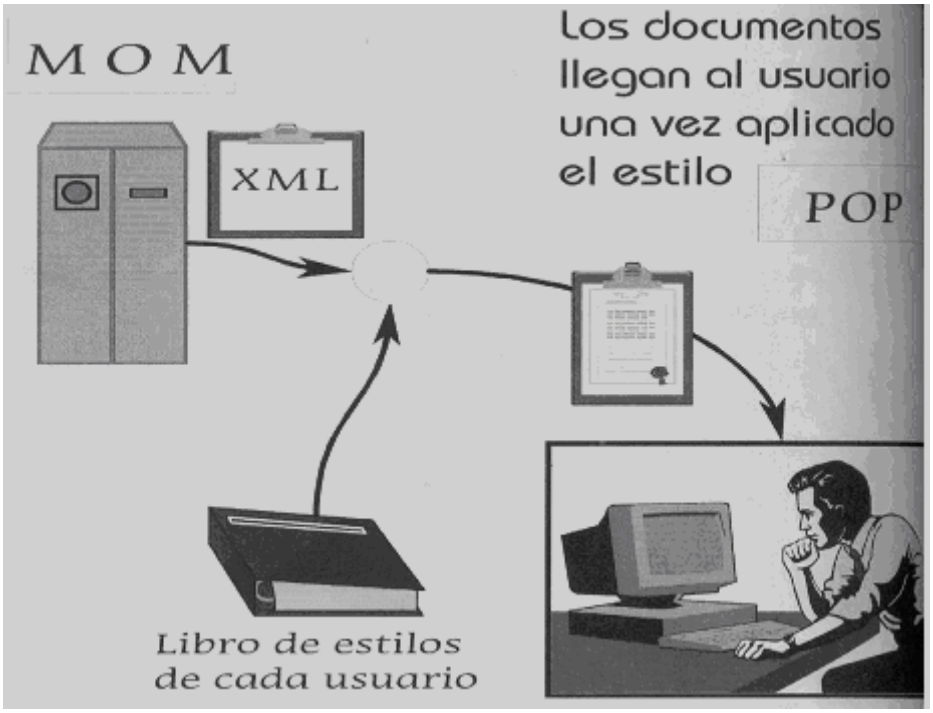
Mozilla 1.8 está disponible con una implementación de XSLT.

Ópera 8: Ópera soporta XML y XML + CSS

Netscape 6: incluyó en el Communicator 5.x soporte para metadatos, pero a través de un estándar sobre el que también esta trabajando el W3 Consortium, el RDF (Resource Description Framework), que no es sino una implementación de XML (un subconjunto).

12. RELACIÓN ENTRE LAS FORMAS DE EDICIÓN.

Existen dos conceptos para la creación de un documento XML. Estas son: Middleware Orientado al Mensaje (MOM) y Edición Orientada a la Presentación (POP).



Middleware Orientada al Mensaje.

Middleware Orientada al mensaje consiste en crear documentos de intercambio entre aplicaciones, es decir la representación interna de información (MOM). Con los documentos se crea una capa intermedia la cual interactúa entre sistemas que no están próximos del usuario. Lo más importante de esta capa es la manera en como se va a representar semánticamente la información.

Edición Orientada a la Presentación.

La edición orientada a la presentación crea documentos, cuyo principal objetivo consiste en el aspecto visual.

Actualmente este tipo de edición es muy compleja, pero ahora es más importante debido al crecimiento de las actividades informáticas. Usualmente los documentos se visualizan sobre varias plataformas con diferentes características, es decir si se crea una aplicación en un determinado dispositivo, ésta misma aplicación no se visualizará de la misma manera en otro dispositivo puesto que se debe tomar en cuenta ancho de banda de recepción ya que esto varía de acuerdo al tamaño y capacidad del dispositivo donde se visualice la aplicación.

La forma más adecuada para visualizar la información es haciendo uso de XSL.

El consorcio de la W3C ha establecido la definición de XSL (Extensible Stylesheet Language) versión 1.0.

El significado del enfoque es que el documento XML que desee ser mostrado, no será modificado en modo alguno, se combinará con la hoja de estilo y se creará un nuevo documento. La hoja de estilo establecerá el contenido de éste

segundo documento, entendiendo que será construido a partir del contenido del primero y de las normas de estilo de la hoja. XSL permite expresar la intención de que un documento se muestre de una forma u otra.

XSL en sí está formado por dos recomendaciones W3C, la ya mencionada y una más que define un lenguaje llamado XSLT versión 1.0. Esta segunda recomendación tiene una versión 1.1 que fue abandonada y una versión 2.0 que está en estado borrador.

Hay que tener claro que XSL es una declaración de las transformaciones conceptuales a realizar, no de cómo hacerlas lo cual queda a la discreción de los creadores de herramientas. La tarea de la transformación se divide en dos partes según XSL, la transformación del árbol del documento y el formateo.

Transformación del árbol del documento: está pensada para transformar el documento XML en otro documento XML aunque se realizan transformaciones a muchos otros formatos como: HTML, PDF, texto, MS-Word y casi cualquier otro que se puede plantear.

Esta transformación se realiza mediante XSLT el cual es un lenguaje específico para realizar esta tarea. La transformación hecha con XSLT puede dar otro documento XML. Si es así, sobre el documento resultante se podría volver a aplicar una nueva transformación por lo que se convierte en una especie de ciclo.

Formateo del documento: en esta segunda fase, un programa “formateador” aplica una serie de objetos de formato sobre el documento para realizar su mapeado sobre algún tipo de dispositivo (físico o lógico) de presentación. Esta fase aporta grandes posibilidades al momento de representar los documentos en distintos dispositivos siempre que se creen los objetos de formateo adecuado y se apliquen al documento.

Surge la idea de generar con estas hojas de estilo copias para teléfonos wap (WML), para navegadores web (HTML) para visores específicos, para pantallas táctiles para dispositivos de braille y para sintetizadores de voz etc.

Este lenguaje de visualización se llama XSL – FO

12.1. XSL

El esquema de trabajo en XSL es el de aplicar primero la transformación XSLT y luego los objetos de formateo XSL – FO.

12.2. XSLT

Es un lenguaje declarativo, esto quiere decir que el usuario no indica como quiere que se realicen los procesos intermedios hasta obtener el documento resultado, sino que indica el contenido que desea para el documento final en función del contenido del documento origen.

XSLT se basa en el uso de plantillas. Una plantilla es una estructura que define una expresión que selecciona unos elementos del documento origen a los que se aplica y el formato de salida como resultado en el documento de salida. Una transformación XSLT consiste en una serie de plantillas que se van aplicando y así van transformando el árbol del documento hasta que ninguna plantilla más es aplicable.

Transformar el árbol del documento no implica modificación alguna sobre el documento original, quiere decir que se crea otro árbol con transformaciones respecto al original.

12.3. XSL - FO

El proceso de formateo se utiliza para convertir el resultado de la transformación XSL en un documento preparado para el lector u oyente. Este proceso consiste en crear un nuevo árbol, un árbol que describa las áreas

geométricas que deben configurar la representación del documento. El nombre de este árbol es árbol de área. La visualización final, queda en manos del dispositivo final que transforme el árbol en el que está incluido el contenido a mostrar y todas las características de tamaño, color, posición, orientación, etc.

12.4. CSS

Las hojas de estilo para HTML son las CSS. Para XML se tiene XSL, XSLT y XSLT – FO pero este componente aún no está extendido en las aplicaciones, como CSS sí lo está y XML es compatible con HTML, CSS se usará temporalmente como hoja de estilo para XML.
La función de estas hojas es establecer la forma en que se debe presentar un documento en cierto dispositivo de salida.

La primera recomendación W3C de hojas de estilo CSS, llamada Cascading Style Sheets, level 1 o CSS1 establece unas hojas de estilo pensadas para HTML como primer tentativa de separar la semántica de la presentación (idea base de XML).

Existe una segunda versión llamada Cascading Style Sheets level 2 o CSS2 que realiza una revisión afondo de la tecnología y está basada en CSS1.

En CSS2 se tiene la posibilidad de crear hojas para todo tipo de documento estructurado, para nuestro estudio nos interesa HTML y XML.

La forma de trabajar con las hojas es parecida, pero en lo que difieren son las características de cada lenguaje.

Los aspectos más relevantes son:

- En HTML existe un conjunto de elementos de marcado limitado, conocido y controlable.

- En XML el marcado es variable, extensible, creado por el usuario, la hoja de estilo es necesaria para decir al dispositivo como tiene que mostrar los elementos por que el dispositivo no conoce nada sobre ellos.

- En HTML la etiquete de marcado especial para incluir hojas de estilo, es style.

- En XML la inclusión de hojas de estilo se realiza siempre como elementos externos al documento, invocado mediante instrucciones de procesamiento (PI).

13. XML NAMESPACE.

XML Namespaces proporcionan un método para evitar conflictos del nombre del elemento. El nombre está en conflicto puesto que los nombres del elemento en XML no se predefinen, un conflicto conocido ocurrirá cuando dos diversos documentos utilicen los mismos nombres del elemento.
En este ejemplo los dos documentos consisten en un cuadro de pintura, ambos tienen el mismo nombre de elemento, pero con diferente contenido y definición.

Ejemplo:

```
<cuadro>
   <altura>  15 cm </altura>
   <ancho> 20 cm <ancho>
 </cuadro>


<cuadro>
  <color> azul marino </color>
```

<imagen> puesta de sol <imagen>
</cuadro>

Si estos dos documentos de XML fueran agregados juntos, habría un conflicto del nombre del elemento porque, ambos documentos contienen un elemento con diversos contenidos y definición.

Una manera para solucionar estos conflictos puede ser la utilización de prefijos.

<m:cuadro>
<m:altura> 15 cm </m:altura>
<m:ancho> 20 cm <m:ancho>
</m:cuadro>

<n:cuadro>
<n:color> azul marino </n:color>
<n:imagen> puesta de sol </n:imagen>
</n:cuadro>

Ahora no habrá conflicto conocido porque los dos documentos utilizan un diverso nombre para su elemento. Usando un prefijo, se ha creado dos diversos tipos de elementos.

Utilizando Namespace.

<m:cuadro xmlns:m= “http://www.lawebdelprogramador.com/cuadro”>
<m:altura> 15 cm </m:altura>
<m:ancho> 20 cm <m:ancho>
</m:cuadro>

```
<n:cuadro xmlns:n= “http://www.altova.com/pintura”>
  <n:color> azul marino </n:color>

 <n:imagen> puesta de sol </n:imagen>
  </n:cuadro>
```

En vez de usar solamente prefijos, se ha agregado una cualidad de los xmlns a la etiqueta para dar al prefijo un asociado conocido cualificado con un namespace. La cualidad del namespace de XML (xmlns) se pone en la etiqueta del comienzo de un elemento y tiene la sintaxis siguiente:

xmlns:namespace - prefix="namespaceURI"

14. TECNOLOGÍAS RELACIONADAS A XML.

XHTML (HTML extensible) permite manejar páginas XHTML como documentos XML en aplicaciones que hacen uso del estándar.

XML DOM (modelo del objeto del documento de XML) define una manera estándar para tener acceso y manipular documentos de XML.

XSL (lenguaje extensible de la hoja del estilo) - XSL consiste en tres porciones: XSLT - un lenguaje para transformar los documentos de XML, XPath - un lenguaje para navegar en documentos de XML, y XSL-FO - un lenguaje para los documentos del formato XML.

XSLT (transformaciones de XSL) se utiliza para transformar documentos de XML en otros formatos de XML, como XHTML.

XPath es un lenguaje para navegar en documentos de XML.

XSL-FO: es un lenguaje basado en XML del margen de beneficio que describe el formato de los datos de XML para que la salida se realice en diferentes medios.

XLink: es un lenguaje para crear hyperlinks en documentos de XML.

XPointer (lenguaje del indicador de XML) permite que los hyperlinks de XLink señalen a partes más específicas en el documento de XML.

DTD (Document type definition) se utiliza para definir los elementos legales en un documento de XML.

XSD (esquema de XML) es un alternativa XML-basado a DTDs.

XForms (formulario de XML) utiliza XML para definir datos de formularios.

XQuery (lenguaje de consulta de XML) se diseña para hacer consultas a datos de XML.

SOAP (protocolo simple del acceso del objeto) es un protocolo XML basado en el intercambio de las aplicaciones sobre el HTTP.

RSS (syndication realmente simple) es un formato para las noticias y el contenido.

WAP (protocolo de uso sin hilos) fue diseñado para demostrar el contenido del Internet en clientes sin hilos, como los teléfonos móviles.

SMIL (lenguaje sincronizado de la integración de las Multimedia) es un lenguaje para describir presentaciones multimedias.

SVG (gráficos del vector de Scalable) define gráficos en formato de XML.

15. APLICACIONES BASADAS EN XML.

XML puede tener tres posibles roles:

a) De contenedor de información

Cuando XML actúa de contenedor, ignora por completo la información que contiene. Esta información irá normalmente almacenada en un elemento del tipo CDATA, estos elementos no son analizados por el parser de XML.

Suponer que se tiene dos aplicaciones distintas en un proyecto, se puede establecer una gramática XML para encapsular los mensajes XML (añadiendo una cabecera al documento), y tener un programa que procese las cabeceras de los mensajes y las direcciones a la aplicación correspondiente.
También puede servir para definir un flujo de datos en un workflow.

b) Para definir el contenido de los mensajes.

Para que se puedan intercambiar información dos aplicaciones y puedan procesarla automáticamente.

c) Para describir el contenido de los mensajes.

Si se utiliza con este rol, se puede describir el esquema de contenidos de los mensajes. Se está definiendo recursos, se utiliza como metadatos. Se puede facilitar el acceso a la información a los agentes de software.

Estos son algunos roles y posibles aplicaciones del XML, pero pueden haber otros más.

La idea del XML es tener la información (definida semánticamente: XML), la

estructura de la información (DTD/Schemas), formato (XSL) y procesos (Java, VB, etc.), todo ello separado.
Es un lenguaje entendible por las máquinas.

Estas son algunas aplicaciones del XML y tecnologías asociadas.

Descargar trabajo en el Servidor.

Por medio del Modelo de Objetos de Documentos (DOM), se puede evitar trabajo al servidor, espera al cliente y no saturar tanto la red.

Por ejemplo, una tienda en Internet que vende una serie de productos, el cliente al conectarse, obtiene el catálogo de productos, y va seleccionando ítems, pero cada vez que selecciona uno, no se manda al servidor nada, sino que por medio del DOM se trata esa selección creando un nuevo nodo del árbol (en el cliente), una vez que el usuario termina, puede ver su cesta de la compra y verificar su pedido para mandarlo al servidor.
Una vez le llegue, éste responderá al cliente del estado de su pedido. En todo el proceso únicamente hay una primera petición al servidor para bajarse la lista de productos y una segunda donde se le envía el pedido. Este proceso se puede aplicar a las tiendas on-line, a los bancos, etc.

Soporte a Clientes.

Gracias a los enlaces extendidos (XLL), el cliente si tiene algún tipo de problema, pinchando sobre un enlace de este tipo le puede salir una lista con la persona concreta de contacto, partes de manuales concretos donde resolver esa duda, departamentos determinados, etc.

Personalización de la WEB.

Gracias al XSL/XSLT, se puede transformar y dar el formato que se desea a un mismo documento XML, por lo tanto si se sabe por medio de los login,

quién es el usario/cliente que accede se puede personalizar la vista de la información.

El cliente/usuario de esta forma se sentirá más a gusto, considerará que esa información es suya y que es reconocido en la Web. Muy útil para los Sitios de Comercio Electrónico.

Independencia del dispositivo de acceso a la información.

Otra aplicación de las XSL/XSLT, si se dispone de un servidor WAP y otro WEB, se puede saber en función del protocolo, desde donde se pide la información y en función del dispositivo, sacarle la misma información de una forma u otra.

Gestión de la información/Conocimiento.

En una empresa donde se maneje infinidad de información, normalmente el usuario al buscarla no está interesado en leerse toda, incluso cuando se abre un documento solamente le interesa los tres últimos párrafos pero para encontrarlos, debe leerse todo el documento con el correspondiente tiempo invertido en ello, pues bien: si se etiqueta la información y a cada usuario se le proporciona una serie de etiquetas de interés, se podría resaltar la información que le es interesante, frente a la que no es relevante. Las búsquedas serían mucho más rápidas y no se perdería tanto tiempo. Con XML se pueden ocurrir muchas más aplicaciones para una mejor gestión de la información.

Buscador WEB.

Si se dispone de un Sitio donde toda la información se encuentre etiquetada en documentos XML, las búsquedas serían mucho más efectivas, ya que se conjuga la potencia de la búsqueda indexada junto la búsqueda semántica.

Intercambio de información.

Si se contratara a una empresa un servicio de noticias, y facilitan la estructura de los datos que se va a recibir (DTD/Schema), se sabrá en todo momento que tipos de documentos XML se está recibiendo, y se podrá tratar de la forma que se desee.

Ofrecer mecanismos más versátiles de mostrar datos.

Actualmente, bajo el nombre de DOM (Document Object Model) se está desarrollando una API que sea soportada por todos los procesadores de XML y HTML. La idea detrás de esta API es que se pueda representar (a través de javascripts o JavaApplets) documentos XML en los navegadores Web, pero de una forma más sofisticada que los documentos HTML, ya que XML no solo proporciona una sintaxis, sino también una semántica.

Buscadores inteligentes.

Debido a que la información en los documentos XML está etiquetada por su significado de forma precisa, podemos localizarla de forma mucho más clara que en documentos HTML. Con DTDs estandarizados para distintas aplicaciones (librerías, tiendas de deporte, catálogos de componentes,…) podríamos programar buscadores Web que recuperasen información sobre un producto de cualquier website en el mundo sabiendo que todos tendrán el mismo formato de datos (gracias al DTD), aunque no tengan necesariamente la misma representación gráfica (gracias al XML/XSL).

Intercambio de información entre sistemas heterogéneos.

El fundamento es el mismo que para los buscadores inteligentes. Debido a que el DTD proporciona un formato estándar para representar la información de un tema específico, puede usarse para simplificar el intercambio de información entre distintas fuentes (actualmente existen ya dos DTD estandarizados uno para fabricantes de chips y otro para industrias químicas, llamado CML).

16. UTILIDADES DE XML.

Se especificarán algunas de las implementaciones que se han desarrollado con XML. Suponer que se establezca alguna implementación propia para Bibliotecas y Centros de Documentación; y con ello se pueda extraer la información desde la misma red (no por métodos de infranet).

CDF (Channel Definition Format): Los canales creado por Microsoft en el explorador IE4 con tecnología push.

RDF (Resource Description Framework): Esquema de descripción de recursos. Una de las aplicaciones más importantes que permitirá describir los datos de cada documento y definir las relaciones que hay entre los datos XML. Tratará de los metadatos (metadata). Se les podría considerar como "los META del XML". Muchas compañías en Internet se están adhiriendo a esta aplicación. RDF posee las siguientes virtudes:

Mejores motores de búsqueda. Se han adherido a esta especificación Yahoo!, Altavista, Excite, Lycos, WebCrawler, Amazon, etc.

La capacidad de describir los contenidos y sus relaciones en una biblioteca digital o sede Web. Permitirá el acceso a una parte concreta del documento y se facilitará el intercambio de los datos.

Se pueden calificar los contenidos para establecer la protección infantil y de la propia intimidad, desarrollado a través de las marcas (tags) de PICS (Platform for Internet Content Selection).

Establece los derechos de propiedad intelectual en las propias páginas Web.

OSD (Open Software Description Format): Formato abierto de descripción de software. Desarrollo de software en múltiples plataformas. Describe el reparto de software a través de la Red. Las etiquetas XML con las que está descrito definen los componentes, la versión, que es la plataforma en la que ha sido creado, la relación con otros componentes, etc. Esto hará que se simplifique el

proceso de instalación para el usuario y permitir también un fácil uso de las actualizaciones.

CML (Chemical Markup Language): Lenguaje de marcas para química. Describe, entre otras fórmulas, las estructuras moleculares y cristalinas, los análisis de espectros y otros objetos de interés para los químicos.

MathML (Mathematical Markup Language): Lenguaje de marcas para matemáticas. Apto para codificar signos matemáticos, símbolos científicos, etc. El MathML es un lenguaje de bajo nivel que tiene en cuenta la comunicación máquina a máquina de datos estructurados como información de bases de datos. El lenguaje MathML utiliza dos series de códigos progresivos: el primero presenta los signos matemáticos en series crecientes, y el segundo transmite el significado semántico de las expresiones matemáticas, lo que posibilita la codificación de símbolos y signos tanto matemáticos como científicos.

EDI (Electronic Document Interchange): Intercambio electrónico de datos. Lleva un tiempo en proceso de estudio, pero finalmente dará un espaldarazo con el estándar XML. InfoVía Plus lo integra en uno de sus servicios (InfoEDI).

OFX (Open Financial Exchange): Intercambio financiero abierto. Marco de trabajo enunciado actualmente en SGML, aunque Microsoft dice que pronto será enunciado en XML. A través del software de gestión financiera (Quicken, Microsoft Money, etc.) se podrá conectar con el banco para gestionar las cuentas y extraer información de la cuenta bancaria. Esto se hará a través de unos protocolos seguros.

TEI (Text Encoding Initiative): Iniciativa que partió de diversas asociaciones profesionales en los campos de humanidades. Trata de establecer etiquetas que propicien la descripción de textos científicos y literarios. En principio solo está disponible para SGML(6).

17. RELACIÓN ENTRE XML Y LAS BASES DE DATOS.

Con el incremento del uso de XML, se está sobrepasando los objetivos inicialmente propuestos. Con XML se pretendía dotar a la Web de mayor estructura creando documentos orientados a la semántica y no a la representación y esto se está logrando. Al mismo tiempo, XML ha superado los límites de la Web y se ha convertido en el lenguaje de representación de datos en el intercambio de información B2B, en el lenguaje en que se comunican los programas. Hoy en día, se está planteando la utilidad o necesidad de hacer que XML llegue más lejos y no sólo se use en el intercambio sino también en el almacenamiento de información de los sistemas

Cabe resaltar que se puede considerar un documento XML como una base de datos en sentido estricto. Un conjunto de documentos XML se podría considerar una base de datos también, pero más extensa. Más aún una Web formada por documentos XML se podría considerar una inmensa base da datos.
Si un documento XML es una base de datos tenemos el lenguaje de consulta, XPath. Pero si tenemos varios documentos o la Web, es posible que XPath no sea capaz de cubrir todas las necesidades de consulta.

De los esfuerzos realizados han surgido diversas propuestas de lenguajes como: XQL, XML - QL, XQuery, X-Query, Quilt y otros.

La idea en que se basan todos estos esfuerzos son:

- XML como base de almacenamiento de información.
- XPath como lenguaje para direccionar nodos.
- Extensiones para realizar las consultas con más flexibilidad y comodidad
- El resultado debe ser un documento XML bien formado.

Es necesario mencionar que SQL es el lenguaje por excelencia para acceso a bases de datos.

Las herramientas anteriormente descritas permiten el acceso a bases de datos en documentos XML, sin embargo que pasaría si se deseara construir una de estas herramientas, o que dichas herramientas no sean lo suficientemente flexibles o potentes. Para solucionar estas debilidades se introducen los APIs DOM y SAX.
Para procesar documentos XML, la mayoría de las herramientas XML trabajan con los APIs SAX o DOM.

SAX (API simple para XML) y DOM (modelo del objeto del documento) eran ambos diseñados para permitir que los programadores tengan acceso a su información sin tener que escribir un analizador de sintaxis en su lenguaje de programación de la opción. Manteniendo la información formato de XML 1,0, y usando su programa de SAX o de DOM APIs está libre utilizar cualquier analizador de sintaxis deseado. Esto puede suceder porque los programas de escritura del analizador de sintaxis deben poner el SAX y el DOM en ejecución APIs usando su lenguaje de programación preferido. SAX y DOM APIs están ambos disponibles para los lenguajes múltiples (Java, C, el Perl, Python, etc.).
Así que, SAX y DOM fueron creados para responder al mismo propósito, que le está dando el acceso a la información salvada en documentos de XML usando cualquier lenguaje de programación (y un analizador de sintaxis para ese lenguaje). Sin embargo, los dos llevan acercamientos muy diversos cuando se accede a su información

SAX es un API para XML basado en eventos. Con él, el analizador SAX reporta los eventos como el inicio o final de los elementos de la aplicación y mientras pasa a través del documento. Como el analizador reporta los eventos mientras visita las diferentes partes del documento, no tiene que construir una estructura interna. Esto reduce los recursos de sistema necesarios, lo que hace a este analizador atractivo para grandes documentos.

El API DOM, sigue una construcción estilo árbol. Los elementos tienen relaciones padre-hijo con otros elementos. Con este API, el analizador

construye una estrucura interna por la que una aplicación puede navegar. DOM permite que una aplicación tenga acceso aleatorio al documento estructurado como un árbol, y el coste es el incremento de la memoria utilizada.

Para trabajar con cualquiera de los dos APIS se necesita conseguir una implementación de los mismos en el lenguaje que se vaya a utilizar. El lenguaje java es idóneo para trabajar con XML, por sus características y por la rápida implementación que se hace en él de las últimas versiones.

Para implementar el APIS SAX para bases de datos, se necesita implementar un analizador que opere sobre una fuente de datos JDBC, iterar sobre cada fila y columna y generar los eventos apropiados mientras se itera. La especificación SAX proporciona la clase “org.xml.sax.ImputSource”. Para representar una base de datos se necesita una forma especializada que pueda representar una tabla de una base de datos por lo tanto se implementa JDBCImportSource que extiende la clase org.xml.sax.ImputSource.

Para construir un árbol DOM para una tabla de base de datos se podría iterar sobre las filas y columnas y construir nodos para un árbol mientras se visita, o se puede emplear otra librería, como la implementación JAXP de Sun. Para implementar el API DOM utilizando dicho aproximación sólo se necesita una reutilización adecuada del analizador SAX para bases de datos expuestos anteriormente (JDBCImputSource).
La clase JDBCDOMParser implementa el API DOM para base de datos.

Para la mayoría de las situaciones, el API SAX para bases de datos es más eficiente que el API DOM. Sin embargo, algunas aplicaciones necesitan acceso aleatorio a documentos XML y por lo tanto requieren la estructura en forma de árbol que ofrece el API DOM para bases de datos.

Con estos APIs, podemos evitar el coste asociado de convertir una base de datos en documentos XML y el coste de mantenerlos sincronizados.

18. EJEMPLOS PRÁCTICOS DE XML.

Luego de haber explicado todo lo necesario para elaborar documentos XML se continuó con la generación de dichos documentos que en base al Tema Objetivos y Justificación son los siguientes:

- Documento XML Bien Formado.
- Documento XML haciendo uso de una DTD externa.
- Documento XML haciendo uso de una DTD interna.

El trabajo consistió en la generación de documentos XML los cuales tratan sobre la venta de una computadora

Para la elaboración de dichos documento se utilizó el software Altova XMLSpy v2005, el cual es uno de los más importantes programas para trabajar con XML, ya que es muy completo e incluye muchas de las tecnologías de las que consta el estándar XML. Entre las funciones de las que consta cabe destacar que permite trabajar con DTD, Esquemas, XSLT, XSL-FO, etc. Además ha sido un producto que se ha mantenido al día pese a la cantidad de cambios que han sufrido las recomendaciones.

Por otro lado no sólo permite trabajar con cada tecnología independiente sino que permite interrelacionar varias de estas. Por ejemplo XML-Spy permite crear un Esquema a partir de una DTD, o permite crear un documento XML a partir de una DTD o esquema, aplicar una hoja xslt a un documento. Además es fácil de usar, proporciona sintaxis a color, etiquetas completas automáticamente, chequeo automático de los documentos (si están bien formados), cambio fácil entre la vista del texto y la vista del grid, importación y exportación de bases de datos

La edición de todos estos componentes se realiza sobre el mismo marcado de ventanas, sobre las que van cambiando los botones con las acciones disponibles.

Entre las posibles vistas que se puede utilizar en XML-Spy figuran:

- La vista código fuente
- La vista navegador.
- La vista árbol.
- La vista esquema (sólo para esquemas).

Se puede observar los paneles para gestión de proyectos donde se crean carpetas para agrupar los tipos de documentos, y los paneles con las listas de ayuda.

Además son destacables los menús y botones donde se puede apreciar la amplia gama de opciones disponibles.

Cabe destacar que esta aplicación esta disponible en Internet y es gratuita, para descargarlo se debe visitar la página de Internet **www.altova.com** . Su instalación es sencilla, solo se debe dar doble click al fichero .exe y seguir las instrucciones que brinda el software. Una vez instalado se muestra una pantalla en la cual se debe elegir la opción instalar el software por 30 días ya que la opción de instalar la aplicación permanente no está disponible para los países centroamericanos.
Hecho esto Altova solicita el nombre de usuario, compañía y correo electrónico para enviar al usuario el Keycode, éste se utiliza para activar el software.

Específicamente, el software de Altova se puede utilizar en las plataformas de Windows y Linux. Sin embargo para Linux, Altova no brinda garantía con respecto a usar su software en esta plataforma.

Existen muchos otros Software que se pueden utilizar para la creación de documentos XML.

Algunos de estos son:

- Peter's XML Editor.
- XML Authority y XML Instante.
- Microsoft XML Notepad.
- SoftQuad XMetal.
- Stylusstudio.

18.1. Documento XML Bien Formado.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Este es un documento Bien Formado-->
<Venta_Computadora>
        <Características>
                <Precio>  $ 700 </Precio>
                <Marca>  Pentium </Marca>
                <Monitor>  Pantalla plana de 15 plg. </Monitor>
                <Procesador>  2 GHZ </Procesador>
                <Memoria>  256 MB  </Memoria>
                <Disco_Duro>  40 GB  </Disco_Duro>
                <Quemador>  Incluido  </Quemador>
                <Impresora>  Canon S200x  </Impresora>
                <Scáner>  Visionner  </Scáner>
                <Teclado>  PS/2  </Teclado>
                <Mouse>  PS/2  </Mouse>
                <Tarjeta_Red>  SOYNTEC Nexoos 200  </Tarjeta_Red>
                <Parlantes>  Incluidos  </Parlantes>
                <Puertos_USB>  4  </Puertos_USB>
        </Características>
</Venta_Computadora>
```

Vista del Documento XML Bien Formado desde Altova.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!-- Este es un documento Bien Formado-->

<Venta_Computadora>
- <Características>
  <Precio>$ 700</Precio>
  <Marca>Pentium</Marca>
  <Monitor>Pantalla plana de 15 plg.</Monitor>
  <Procesador>2 GHZ</Procesador>
  <Memoria>256 MB</Memoria>
  <Disco_Duro>40 GB</Disco_Duro>
  <Quemador>Incluido</Quemador>
  <Impresora>Canon S200x</Impresora>
  <Scáner>Visionner</Scáner>
  <Teclado>PS/2</Teclado>
  <Mouse>PS/2</Mouse>
  <Tarjeta_Red> SOYNTEC Nexoos 200 </Tarjeta_Red>
  <Parlantes>Incluidos</Parlantes>
  <Puertos_USB>4</Puertos_USB>
  </Características>
  </Venta_Computadora>
```

La creación del Documento XML Bien Formado es muy sencillo de realizar y de la manera en que se elaboró fue así:

- Se utilizó el software Altova XMLSpy v2005, se creó un nuevo documento, XMLSpy v2005 ofrece múltiples opciones para crear diferente tipos de documentos, pero se debe escoger la opción que aparece por defecto que es: xml Markup Language Extensible la cual permite crear documentos XML.

- Luego XMLSpy v2005 pregunta si desea crear el documento XML con DTD O Schema, en este caso como lo que se quería era crear un documento XML Bien Formado lo que se hizo fue cerrar esta ventana y automáticamente el software proporcionó la hoja de trabajo con la cabecera y todos sus componentes.

- Se continuó editando el documento, el cual consta de 14 elementos cada uno con su correspondiente contenido.

Los elementos que se utilizaron para crear el documento XML Bien Formado corresponden a las características necesarias para la venta de una computadora. Estos elementos son:

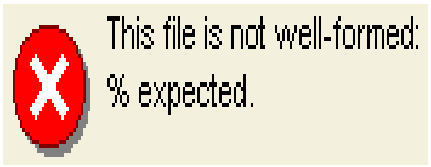
<Precio> <Procesador> <Quemador> <Teclado> <Parlantes>
<Marca> <Memoria> <Impresora> <Mouse> <Puertos_USB>
<Monitor <Disco_Duro> <Scáner> <Tarjeta_Red>

<Venta_Computadora>: es la raíz del documento.

<Características>: es un elemento sin contenido.

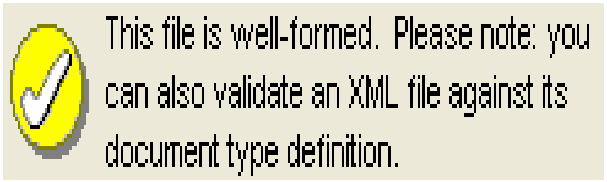
- Una vez que se editó el documento se guardó con la extensión **.xml,** luego se procedió a verificar si el documento estaba Bien Formado es decir si cumplía con las normas sintácticas generales de XML.

Si un documento no está bien formado XMLSpy v2005 muestra el icono



con la descripción del error.

En cambio si el documento está bien formado el software muestra el icono



y a al vez comunica que el documento está bien formado.

- En el browser se visualiza el documento tal como se editó ya que una de las ventajas de XML es que permite ver el contenido del documento con sus elementos; mediante el grid se visualiza el documento en forma de árbol.

18.2. Documento XML haciendo uso de DTD externa.

DTD (archivo externo).

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

<!-- Ejemplo del archivo DTD -->

<!ELEMENT Venta_Computadora (Características*)>

<!ELEMENT Características (idPrecio, Marca, Monitor, Procesador, Memoria,
Disco_Duro, Quemador, Impresora, Scáner, Teclado, Mouse, Tarjeta_Red,
Parlantes, Puertos_USB)>

<!ELEMENT idPrecio EMPTY>

<!ATTLIST idPrecio

ID #REQUIRED
>
<!ELEMENT Marca (#PCDATA)>

<!ELEMENT Monitor (#PCDATA)>

<!ELEMENT Procesador (#PCDATA)>

<!ELEMENT Memoria (#PCDATA)>

<!ELEMENT Disco_Duro (#PCDATA)>

<!ELEMENT Quemador (#PCDATA)>

<!ELEMENT Impresora (#PCDATA)>

<!ELEMENT Scáner (#PCDATA)>

<!ELEMENT Teclado (#PCDATA)>

<!ELEMENT Mouse (#PCDATA)>

<!ELEMENT Tarjeta_Red (#PCDATA)>

<!ELEMENT Parlantes (#PCDATA)>

<!ELEMENT Puertos_USB (#PCDATA)>
```

Documento XML.

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

<!-- Ejemplo del documento XML utilizando el archivo DTD-->

<!DOCTYPE Venta_Computadora SYSTEM "C:\Documents and Settings\Nubia
Mercedes Terce\Mis documentos\Venta.dtd">

<Venta_Computadora>

        <Características>

                <idPrecio idprecio="C11760"/>

                <Marca>  Pentium  </Marca>

                <Monitor>  Pantalla plana de 15 pulg.  </Monitor>

                <Procesador>  2.4 GHZ  </Procesador>

                <Memoria>  256 MB  </Memoria>

                <Disco_Duro>  40 GB  </Disco_Duro>

                <Quemador>  Incluido  </Quemador>

                <Impresora>  Canon S200x  </Impresora>

                <Scáner>  Visionner  </Scáner>

                <Teclado>  PS/2  </Teclado>

                <Mouse>  PS/2  </Mouse>

                <Tarjeta_Red>  SOYNTEC Nexoos 200  </Tarjeta_Red>

                <Parlantes>  2  </Parlantes>

                <Puertos_USB>  4  </Puertos_USB>

        </Características>

</Venta_Computadora>
```

Vista del Documento XML utilizando DTD externa.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!-- Ejemplo del documento XML  utilizando el archivo DTD -->

  <!DOCTYPE Venta_Computadora (View Source for full doctype...)>
- <Venta_Computadora>
- <Características>
  <idPrecio idprecio="C11760" />
  <Marca>Pentium</Marca>
  <Monitor>Pantalla plana de 15 pulg.</Monitor>
  <Procesador>2.4 GHZ</Procesador>
  <Memoria>256 MB</Memoria>
  <Disco_Duro>40 GB</Disco_Duro>
  <Quemador>Incluído</Quemador>
  <Impresora>Canon S200x</Impresora>
  <Scáner>Visionner</Scáner>
  <Teclado>PS/2</Teclado>
  <Mouse>PS/2</Mouse>
  <Tarjeta_Red> SOYNTEC Nexoos 200 </Tarjeta_Red>
  <Parlantes>2</Parlantes>
  <Puertos_USB>4</Puertos_USB>
    </Características>
    </Venta_Computadora>
```

Para la generación del Documento XML haciendo uso de una DTD externa el procedimiento que se realizó fue un poco diferente ya que lo que se pretendía era crear un archivo externo que contenga las declaraciones de los elementos y atributos que se van a utilizar.

Lo que se hizo fue:

- Primeramente se creó un nuevo documento, se escogió la opción por defecto que es xml Markup Language Extensible, se seleccionó crear el documento utilizando DTD; en este punto XMLSpy v2005 solicita abrir el archivo con la extensión .dtd (este archivo que solicita, es el archivo externo), pero no se abrió ningún archivo por que hasta el momento no se había creado, ya que primero se crea el archivo de DTD interna ,para lo cual lo que se hizo fue cerrar esta ventana y de igual manera el software automáticamente muestra la hoja de trabajo con la cabecera.

- Luego de la cabecera se colocó un comentario.

- Se declaró de tipo ELEMENT “Venta_Computadora”, el cual contiene uno o muchos elementos “Características”. Cada elemento Características tiene los siguientes elementos: un “idPrecio”, una “Marca”, un “Monitor”, un “Procesador”, una “Memoria”, un “Disco_Duro” , un “Quemador”, una “Impresora”, un “Scáner”, un “Teclado”, un “Mouse”, una “Tarjeta_Red”, un “Parlantes”, un “Puertos_USB”.

- idPrecio se declaró de tipo ELEMENT, EMPTY indica que es un elemento sin contenido, a la vez idPrecio se declaró también como atributo. Los demás elementos se declararon de tipo ELEMENT, y se les agregó el elemento #PCDATA para indicar que el elemento va contener solo texto.

- Cuando se terminó de editar el documento se guardó con la extensión **.dtd** por que este archivo es el que se utilizará para hacer referencia al momento de crear el documento XML.

- De igual manera que el documento anterior se procedió a verificar que el archivo estuviera Bien Formado, el software lo analizó e indicó que estaba Bien Formado. Una vez comprobado esto, se dió la orden de validar el documento. Si el documento XML no es válido XMLSpy v2005 envía el icono de error y además comunica el mensaje de error. De lo contrario si el documento es válido el software muestra el icono

indicando que el documento es válido.

A partir del documento creado con DTD se pudo elaborar el documento XML.

- Para elaborar este archivo se creó un nuevo documento, se seleccionó la opción de crear el documento XML con DTD, se muestra una ventana solicitando abrir el archivo DTD . Una vez seleccionado el nombre del archivo DTD, el software automáticamente carga en la hoja de trabajo la cabecera, la ruta donde se encuentra el archivo creado y la raíz del documento que en este caso es Venta_Computadora.

- Hecho esto se abrió el elemento Características y automáticamente se cargaron todos los elementos declarados en la DTD . A continuación se agrega el contenido de cada elemento.

- Una vez que se terminó el documento se guardó con la extensión **.xml** y se continuó a verificar que el documento estuviera Bien Formado y Válido.

18.3. Ejemplo de un documento XML agregando registros.

DTD (archivo externo).

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

<!-- Ejemplo de como se pueden agregar registros -->

<!ELEMENT Venta_Computadora (características*)>

<!ELEMENT características (precio, marca, monitor, procesador)>

<!ELEMENT precio (#PCDATA)>

<!ELEMENT marca (#PCDATA)>

<!ELEMENT monitor (#PCDATA)>

<!ELEMENT procesador (#PCDATA)>
```

Documento XML.

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

<! DOCTYPE Venta_Computadora SYSTEM "C:\Documents and
Settings\Nubia Mercedes Terce\Mis documentos\TESIS\Registro.dtd">

<Venta_Computadora>

<características>

        <precio> $ 700 </precio>

        <marca> Pentium </marca>

        <monitor> Pantalla plana de 15 pulgs </monitor>

        <procesador> 2.4 GHZ </procesador>

</características>

<características>

        <precio> $ 600 </precio>

        <marca> Intel Celeron </marca>

        <monitor> 15 pulgs </monitor>

        <procesador> 2.4 GHZ </procesador>

</características>

</Venta_Computadora>
```

Vista del documento agregando registros desde Altova.

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

<!DOCTYPE Venta_Computadora *(View Source for full doctype...)*>

- <Venta_Computadora>

- <características>

<precio>**$ 700**</precio>

<marca>**Pentium**</marca>

<monitor>**Pantalla plana de 15 pulgs**</monitor>

<procesador>**2.4 GHZ**</procesador>

</características>

- <características>

<precio>**$ 600**</precio>

<marca>**Intel Celeron**</marca>

<monitor>**15 pulgs**</monitor>

<procesador>**2.4 GHZ**</procesador>

</características>

</Venta_Computadora>

Para la creación del documento XML agregando registros, se realizó el mismo procedimiento que se llevó a cabo para crear el documento XML haciendo uso de DTD Externa.

Se declaró un elemento “Venta_Computadora” que contiene un elemento “características”. El elemento “características” tiene los siguientes elementos: <precio>, <marca>, <monitor>, <procesador>.

Para agregar registros al documento se utilizó el indicador de frecuencia asterisco (*), el cual permite ir agregando registros al documento. De esta manera cuando se desee agregar un registro más, se mostrará el elemento “características” con sus campos, en este momento se podrá introducir la información correspondiente a cada campo.

18.4. Documento XML haciendo uso de una DTD interna.

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

<!-- Documento XML con DTD Interna-->

<!DOCTYPE Venta_Computadora [<!ELEMENT Venta_Computadora (idPrecio,
Marca, Monitor, Procesador, Memoria, Disco_Duro, Quemador, Impresora,
Scáner, Teclado, Mouse, Tarjeta_Red, Parlantes, Puertos_USB)>

<!ELEMENT idPrecio (#PCDATA)>

<!ELEMENT Marca (#PCDATA)>

<!ELEMENT Monitor (#PCDATA)>

<!ELEMENT Procesador (#PCDATA)>

<!ELEMENT Memoria (#PCDATA)>

<!ELEMENT Disco_Duro (#PCDATA)>

<!ELEMENT Quemador (#PCDATA)>

<!ELEMENT Impresora (#PCDATA)>

<!ELEMENT Scáner (#PCDATA)>

<!ELEMENT Teclado (#PCDATA)>

<!ELEMENT Mouse (#PCDATA)>

<!ELEMENT Tarjeta_Red (#PCDATA)>

<!ELEMENT Parlantes (#PCDATA)>

<!ELEMENT Puertos_USB (#PCDATA)>
]>

<Venta_Computadora>

 <idPrecio>  C11760  </idPrecio>

 <Marca>  Pentium  </Marca>

 <Monitor>  Pantalla plana de 15 pulg.  </Monitor>

 <Procesador>  2.4 GHZ  </Procesador>
```

<Memoria> 256 MB </Memoria>

<Disco_Duro> 40GB </Disco_Duro>

<Quemador> Incluído </Quemador>

<Impresora> Canon S200x </Impresora>

<Scáner> Visionner </Scáner>

<Teclado> PS/2 </Teclado>

<Mouse> PS/2 </Mouse>

<Tarjeta_Red> SOYNTEC Nexoos 200 </Tarjeta_Red>

<Parlantes> 2 </Parlantes>

<Puertos_USB> 4 </Puertos_USB>

</Venta_Computadora>

Vista del Documento XML utilizando DTD interna desde Altova.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Documento XML con DTD Interna  -->
-
<!DOCTYPE Venta_Computadora (View Source for full doctype...)>
- <Venta_Computadora>
<idPrecio>C11760</idPrecio>
<Marca>Pentium</Marca>
<Monitor>Pantalla plana de 15 pulg.</Monitor>
<Procesador>2.4 GHZ</Procesador>
<Memoria>256 MB</Memoria>
<Disco_Duro>40GB</Disco_Duro>
<Quemador>Incluído</Quemador>
<Impresora>Canon S200x</Impresora>
<Scáner>Visionner</Scáner>
<Teclado>PS/2</Teclado>
<Mouse>PS/2</Mouse>
<Tarjeta_Red> SOYNTEC Nexoos 200  </Tarjeta_Red>
<Parlantes>2</Parlantes>
<Puertos_USB>4</Puertos_USB>
</Venta_Computadora>
```

En el documento XML haciendo uso de una DTD interna de igual manera:

- Se creó un nuevo documento, se escogió la opción por defecto, se seleccionó crear el documento con DTD pero no se abrió ningún archivo, si no que se canceló la ventana. Este procedimiento se realizó así puesto que la DTD va incorporada en el mismo documento.

- Se cargó la cabecera en la hoja de trabajo y se empezó a declarar los elementos y atributos que se utilizaron. Terminado esto se abrió la raíz del documento (Venta_Computadora) y se introdujo el contenido de los elementos en el mismo documento. Esto permite que el documento tenga su propia validación.

- El archivo se guardó con la extensión **.xml** y al igual que en los documentos anteriores se verificó que estuviera Bien Formado y Válido.

- Una vez comprobado se observaron los resultados en el navegador

CONCLUSIONES.

En su calidad de estándar para expresar datos estructurados, XML ofrece muchas ventajas a las organizaciones, desarrolladores de software, sitios Web y usuarios finales. Las oportunidades aumentarán cuantos más formatos de datos de mercado vertical se creen para mercados claves, como el mercado de búsqueda avanzada en bases de datos, banca en línea, médico, legal, comercio electrónico, etc. Cuando los sitios ofrezcan datos, en lugar de limitarse a las vistas de datos, las oportunidades serán extraordinarias.

Hoy en día, los servicios de atención al cliente están migrando de centros de llamadas y sedes físicas a sitios Web, por lo que se beneficiarán de la sólida funcionalidad de XML. Dado que la mayoría de estas aplicaciones empresariales implican la manipulación o la transferencia de datos y registros de base de datos, como pedidos de compra, facturas, información del cliente, citas, mapas, etc., XML va a revolucionar las posibilidades de los usuarios finales en Internet, permitiendo la implementación de un amplio abanico de aplicaciones empresariales. Además, la información que ya esté almacenada en sitios Web, tanto si está guardada en documentos como en bases de datos, puede marcarse con vocabularios basados en XML y orientados a intranets. Dichos vocabularios también ayudan a las pequeñas y medianas empresas que necesiten intercambiar información entre clientes y proveedores.

Debido a la importancia de XML es que surgió la idea de realizar este trabajo en el cual se abordaron los aspectos generales para la creación de documentos XML Bien Formados y Válidos implementando DTD interna y Externa. A la vez se generaron ejemplos en los cuales se demuestra la forma en que estos documentos deben ser creados, cumpliendo así los objetivos que se plantearon en el trabajo.

RECOMENDACIONES.

En este trabajo se desarrolló una parte del Lenguaje de Marca Extensible, se dejan los cimientos de este nuevo lenguaje y esperamos que sirva de base para futuras investigaciones, ya que XML es un lenguaje amplio y sería de gran utilidad que en los próximos proyectos que se realice sobre este tema se aprovechen al máximo la cantidad de aplicaciones que posee y que futuros investigadores las aborden para que de esta manera se permita conocer más del lenguaje XML.

GLOSARIO.

API: Interfaz de Programación de Aplicaciones.

B2B: Intercambio de Información entre Empresas.

CSS: Hojas de Estilo en Cascada.

DOM: Modelo de Objeto de Documento.

DTD: Definición de Tipos de Documento.

GML: Lenguaje de Marcado Extensible.

HTML: Lenguaje de Marcado Hipertexto.

ISO: Organización Internacional de Estándares.

MOM: Middleware Orientada al Mensaje.

Middleware: Capa de Software intermedia que sirve para aislar, homogeneizar e intercambiar otras capas de software.

PI: Instrucción de Procesamiento.

POP: Publicación Orientada a la Presentación.

SAX: Api Simple para XML.

SGML: Lenguaje de Marcado Generalizado Estándar.

W3C: Consorcio de la World Wide Web.

XLL: Lenguaje Extensible de enlace.

XML: Lenguaje de Marcado Extensible.

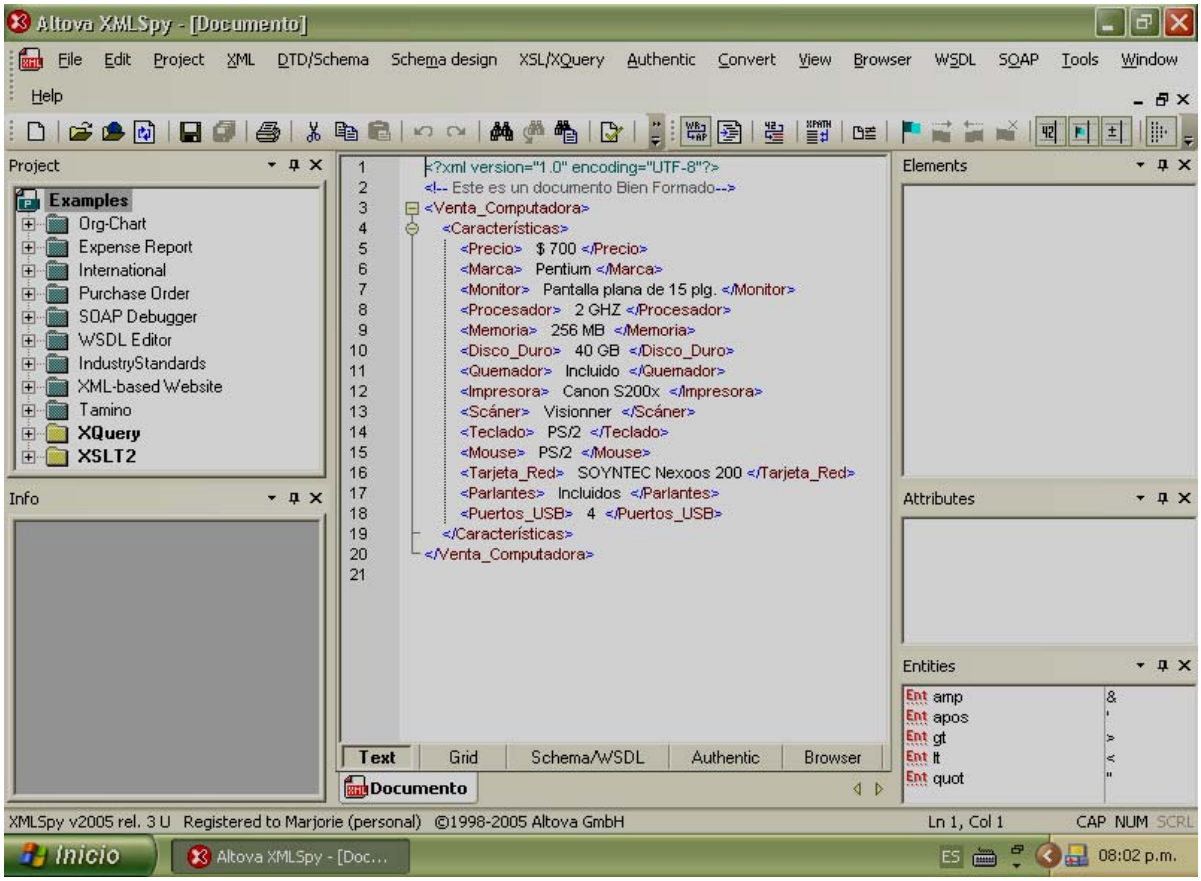
XSL: Lenguaje de Hojas de Estilo Extensible.

XUA: Agentes de usuario para XML.

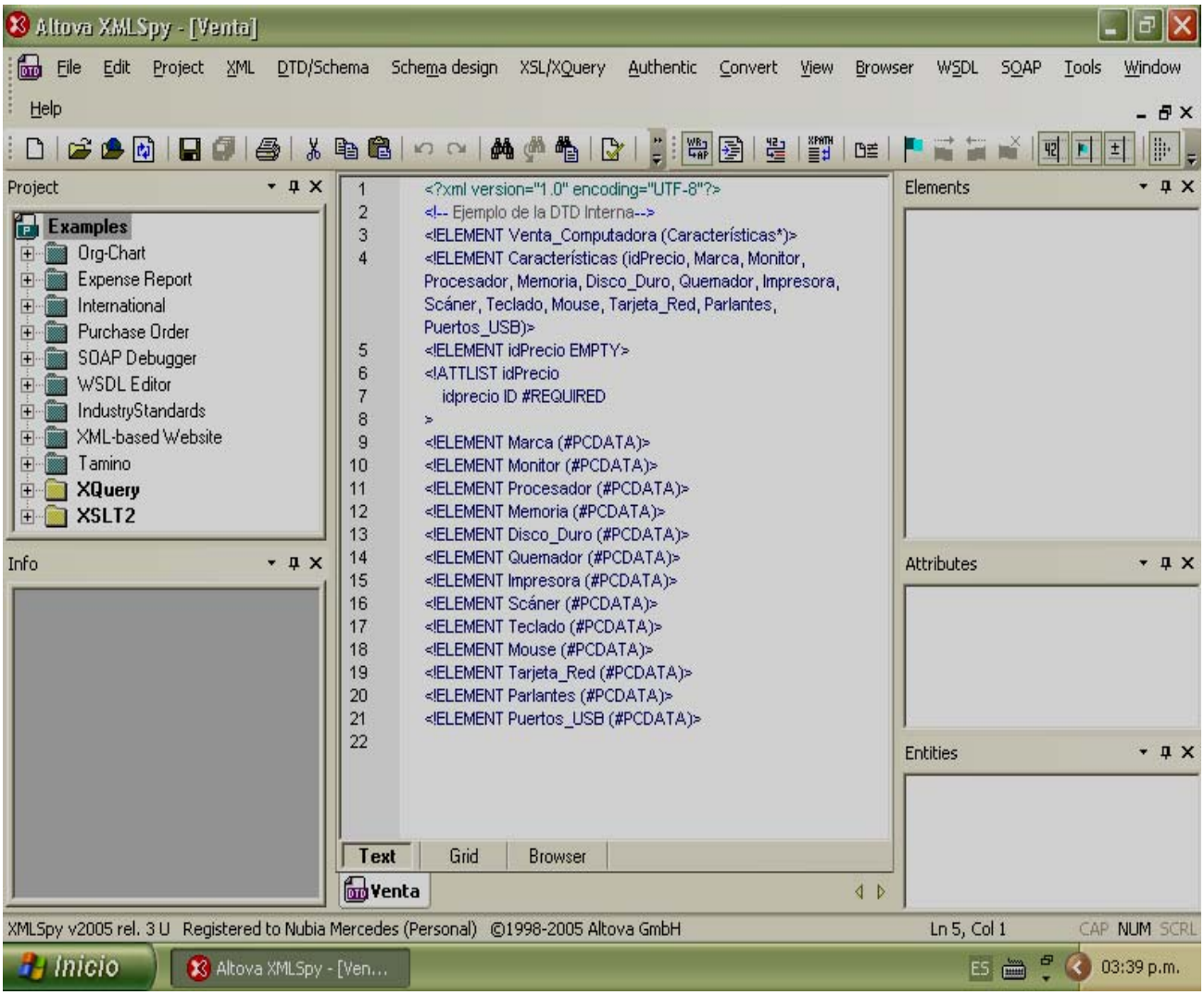
ANEXOS.

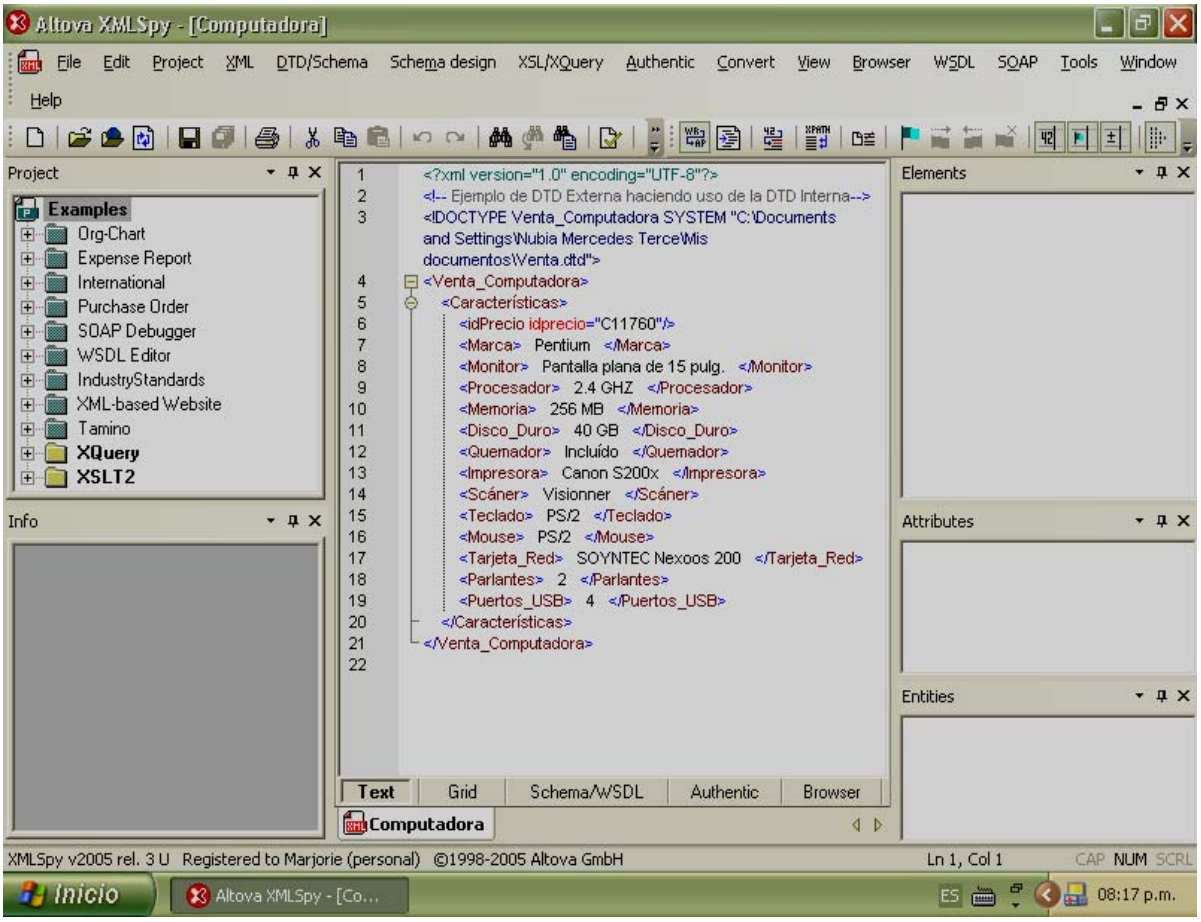
Presentación del software Altova XML Spy v2005.

Documento XML bien formado.

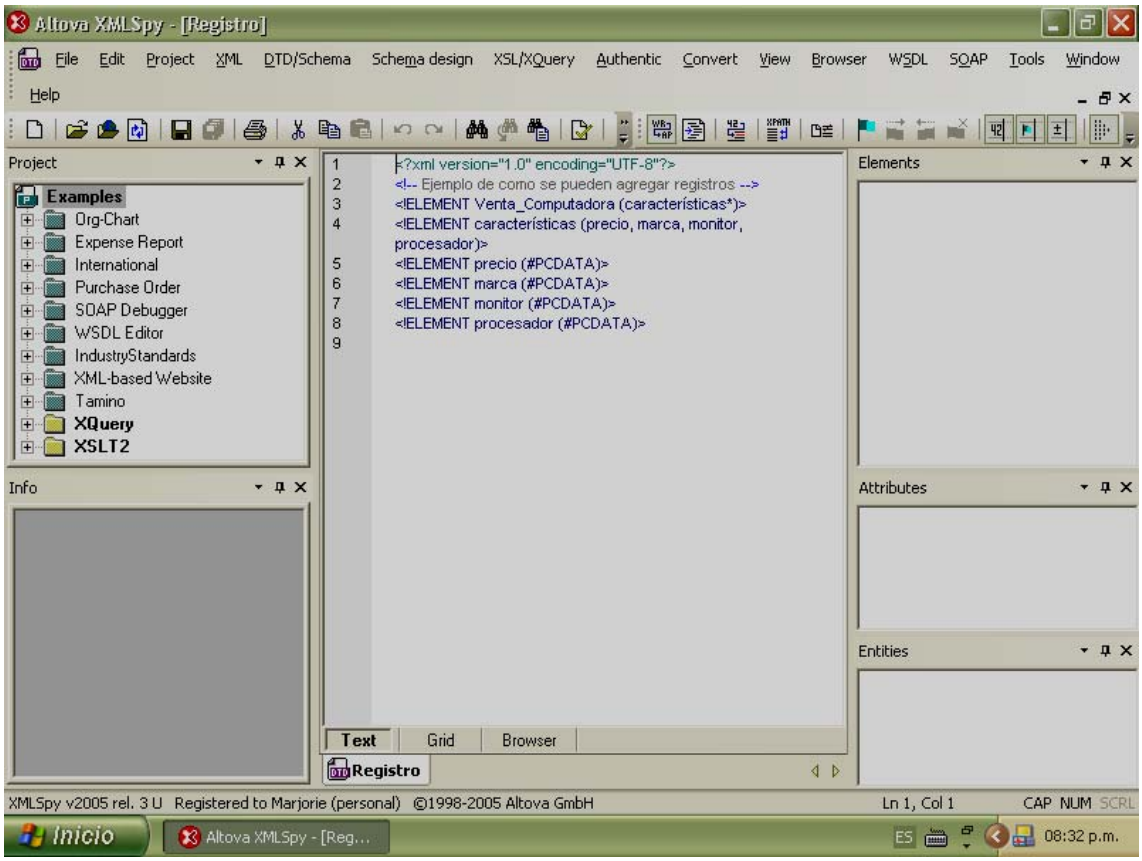


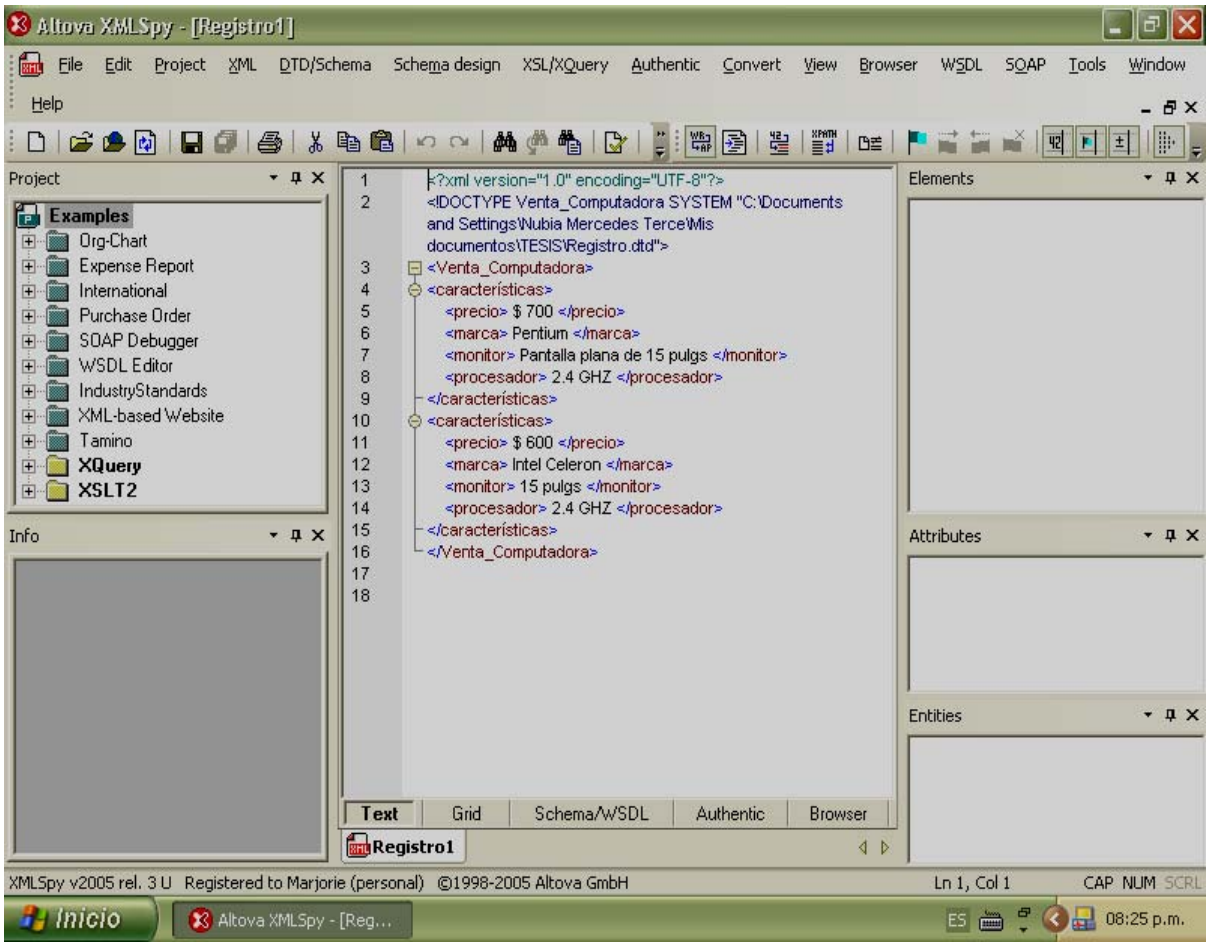
Documento XML con DTD externa.



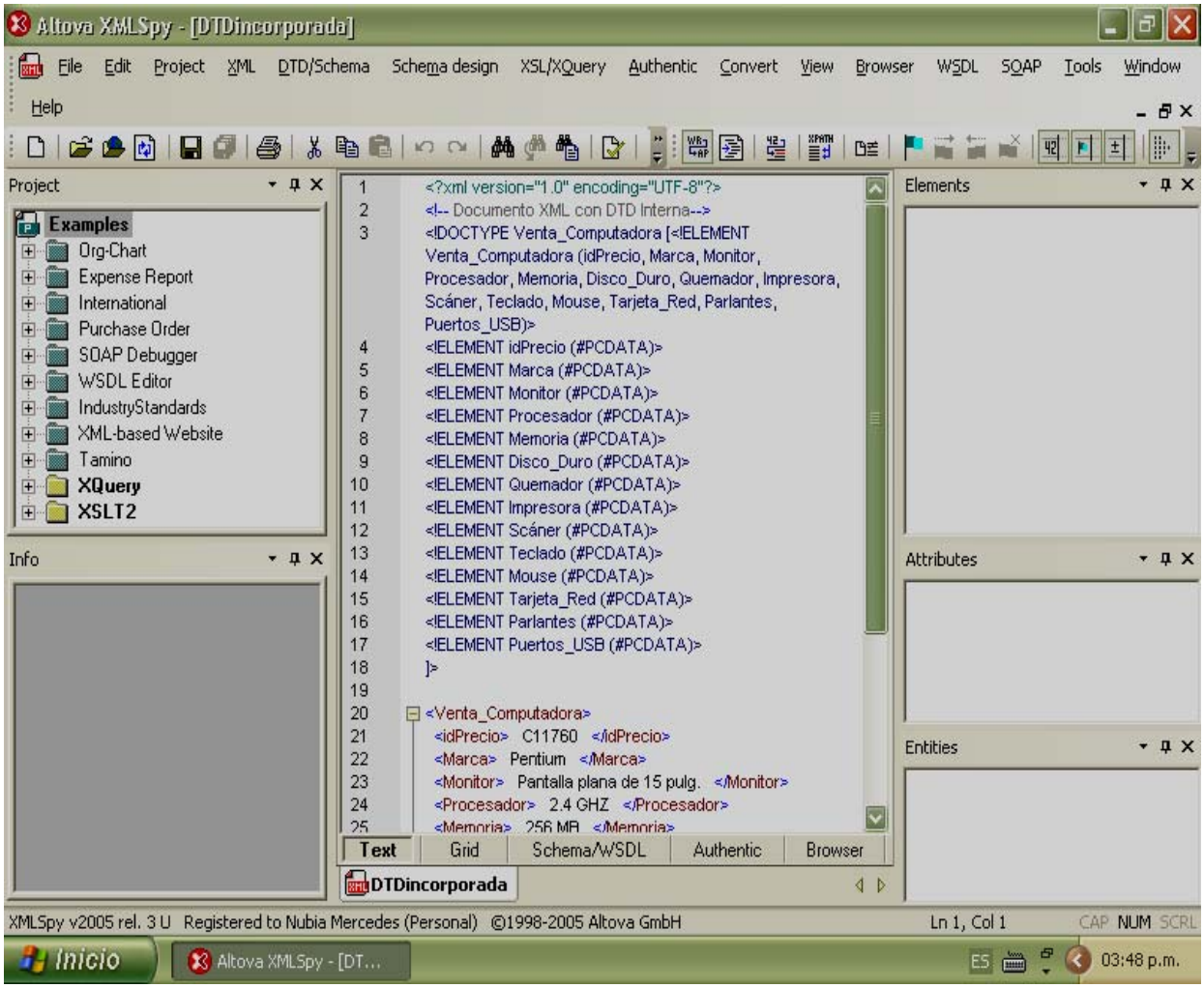

Altova XMLSpy - [Computadora]
File Edit Project XML DTD/Schema Schema design XSL/XQuery Authentic Convert View Browser WSDL SOAP Tools Window
Help
Project
Examples
Org-Chart
Expense Report
International
Purchase Order
SOAP Debugger
WSDL Editor
IndustryStandards
XML-based Website
Tamino
XQuery
XSLT2
Info
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- Ejemplo de DTD Externa haciendo uso de la DTD Interna-->
3 <!DOCTYPE Venta_Computadora SYSTEM "C:\Documents and Settings\Nubia Mercedes Tercero\Mis documentos\Venta.dtd">
4 <Venta_Computadora>
5 <Características>
6 <idPrecio idprecio="C11760"/>
7 <Marca> Pentium </Marca>
8 <Monitor> Pantalla plana de 15 pulg. </Monitor>
9 <Procesador> 2.4 GHZ </Procesador>
10 <Memoria> 256 MB </Memoria>
11 <Disco_Duro> 40 GB </Disco_Duro>
12 <Quemador> Incluído </Quemador>
13 <Impresora> Canon S200x </Impresora>
14 <Scáner> Visionner </Scáner>
15 <Teclado> PS/2 </Teclado>
16 <Mouse> PS/2 </Mouse>
17 <Tarjeta_Red> SOYNTEC Nexoos 200 </Tarjeta_Red>
18 <Parlantes> 2 </Parlantes>
19 <Puertos_USB> 4 </Puertos_USB>
20 </Características>
21 </Venta_Computadora>
22
Elements
Attributes
Entities
Text Grid Schema/WSDL Authentic Browser
Computadora
XMLSpy v2005 rel. 3 U Registered to Marjorie (personal) ©1998-2005 Altova GmbH
Ln 1, Col 1
CAP NUM SCRL
Inicio
Altova XMLSpy - [Co...
ES
08:17 p.m.

Documento XML agregando registros.




Altova XMLSpy - [Registro1]
File Edit Project XML DTD/Schema Schema design XSL/XQuery Authentic Convert View Browser WSDL SOAP Tools Window
Help
Project
Examples
Org-Chart
Expense Report
International
Purchase Order
SOAP Debugger
WSDL Editor
IndustryStandards
XML-based Website
Tamino
XQuery
XSLT2
Info
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE Venta_Computadora SYSTEM "C:\Documents and Settings\Nubia Mercedes Terce\Mis documentos\TESIS\Registro.dtd">
3 <Venta_Computadora>
4 <características>
5 <precio> $ 700 </precio>
6 <marca> Pentium </marca>
7 <monitor> Pantalla plana de 15 pulgs </monitor>
8 <procesador> 2.4 GHZ </procesador>
9 </características>
10 <características>
11 <precio> $ 600 </precio>
12 <marca> Intel Celeron </marca>
13 <monitor> 15 pulgs </monitor>
14 <procesador> 2.4 GHZ </procesador>
15 </características>
16 </Venta_Computadora>
17
18
Text Grid Schema/WSDL Authentic Browser
Registro1
Elements
Attributes
Entities
XMLSpy v2005 rel. 3 U Registered to Marjorie (personal) ©1998-2005 Altova GmbH
Ln 1, Col 1
CAP NUM SCRL
Inicio
Altova XMLSpy - [Reg...
ES
08:25 p.m.

Documento XML con DTD interna.



BIBLIOGRAFÍA

- Estándar XML yTecnologías asociadas. Gutierrez Martínez, José María. Palacios Escribano, Fernando. Gutierrez de Mesa, José Antonio.Editorial Danypress.

- Internet:
- www.w3schools.com
- www.lawebdelprogramdor.com
- www.altova.com
- www.google.com