Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

EXTENSIBLE MARKUP LANGUAGE CON VISUAL .NET

Trabajo de graduación previo a la obtención del título de

Ingeniero de Sistemas

Autor: Jorge Luis Idrovo
Pedro Peñafiel

Director: Ing. Marco Orellana

Cuenca, Ecuador

2006

TABLA DE CONTENIDO

1. Introducción **1**

1.1. Introducción 1

1.2. HTML e Internet 2

1.3. ¿Qué es XML? 3

1.4. Objetivos y Orígenes 4

2. Características del XML **6**

2.1. Introducción 6

2.2. Estándares abiertos 6

2.3. Características Principales 7

2.4. DTD 9

2.5. Esquemas 10

2.6. Fortalezas y Debilidades del XML 11

2.7. DOM 13

3. Documentos XML en el Web **15**

3.1. Desplegar documentos 15

3.1.1. CSS 15

3.1.2. XSL 16

3.2. Vínculos entre Documentos XML 17

4. Visual Net **23**

4.1. Introducción 23

4.2. Características de Visual Net 26

4.3. Clase System.Xml 27

5. Desarrollo de la Aplicación 34

5.1. Introducción de la Aplicación 34

5.2. Desarrollo de la Aplicación 35

6. Conclusiones y Recomendaciones 43

7. Glosario 44

8. Bibliografía 45

Índice de Anexos

Anexo 1 46

Anexo 2 47

Anexo 3 48

Anexo 4 49

Resumen

Extensible Markup Language (XML) es un formato del texto simple, muy flexible derivado de SGML. Originalmente se lo diseñó para arreglar los desafíos de publicación electrónica de gran potencia, XML también está jugando un papel importante en el intercambio de una gran variedad de datos en el Web y otras aplicaciones.

Los lenguajes o dialectos de XML pueden ser diseñados por cualquiera y pueden procesarse en el software de complacencia. XML también se diseño para ser bastante legible. XML es un subconjunto simplificado de *Standard Generalized Markup Language* (SGML). Su propósito primario es facilitar la distribución de datos por los sistemas de información diferentes, particularmente los sistemas conectados mediante Internet.

XML proporciona una estructura de árbol a la información.

XML no es un reemplazo para HTML, XML y HTML fueron creados con principios diferentes:

XML fue diseñado describir los datos y enfocar en cuales son estos datos, se enfoca en el contenido de dicha información.

HTML fue diseñado para mostrar esos datos y se enfoca en como los datos van a mostrarse, se orientó a la apariencia.

Abstract

Extensible Markup Lenguaje (XML) is a very flexible simple text format derived from SGML. Originally designed to fix large-scale electronic publishing challenges, XML is also playing an important role in the exchange of a great variety of data on the Web and other applications.

The XML languages or dialects can be designed by anyone and can be processed in any software. XML was also designed to be very legible. XML is a subset simplified from *Standard Generalized Markup Language* (SGML). Its primary purpose is to facilitate the distribution of data through the different information systems, particularly the systems connected through the Internet.

XML provides a tree structure to information. XML is not a substitute for HTML. XML and HTML were created with different principles. XML was designed to describe the data to focus on which these data are. It focuses on the content of that information. HTML, on the other hand, was designed to display those data, and it focuses on how the data are going to be show. It was oriented towards appearance.

Capitulo 1

1. Introducción

1.1. Introducción

Muchas personas han oído hablar de XML (Lenguaje de Marcas Extensible, *Extensible Markup Language*), y comúnmente se tiene una idea de que es una especie de HTML avanzado por lo que surgen preguntas como: ¿Qué es XML?, ¿Es XML una nueva versión de HTML?, ¿Para que sirve?, ¿Por qué se creó?.

HTML (*Hypertex Markup Language*) ha alcanzado gran popularidad en los últimos años, pero debemos reconocer que nos hemos encontrado con sus propias limitaciones, y que algunas de estas se han tratado de subsanar con scripts, javascripts, Active X, HTML dinámico, etc; pero en realidad estas herramientas no aportan una solución global a las limitaciones del HTML.

En este proyecto responderemos interrogantes planteadas, y se analizará la utilización de XML como formato estándar para el intercambio de datos y como este lenguaje nos proporciona un formato para describir datos estructurados, facilitando realizar declaraciones más precisas de contenido y permitiendo obtener resultados de búsquedas con más significado. Además, estudiaremos de qué forma XML habilitará una nueva generación de aplicaciones de manipulación y visualización de datos basadas en Web.

1.2. HTML e Internet

Para entender por que se creó XML por parte del W3C, primero debemos aclarar como ha evolucionado el Internet y el lenguaje HTML.

HTML es originariamente un subconjunto del SGML (*Standard Generalized Markup Language*). Al inicio se basaba en una colección de etiquetas las cuales permitían describir documentos de texto y vínculos de hipertexto que permitían desplazarse entre diferentes documentos, siempre con independencia de la máquina. Conociendo las normas de actuación de estas etiquetas y disponiendo de un sencillo editor de textos ASCII, se pueden confeccionar fácilmente documentos HTML.

HTML, no es un lenguaje de programación, es un lenguaje de especificación de contenidos para un tipo específico de documentos SGML. Es decir, mediante HTML podemos especificar, usando un conjunto de marcas, cómo va a representarse la información en un navegador o browser. La facilidad de uso y la particularidad que no es propiedad de nadie, hizo al HTML el sistema idóneo para compartir información en Internet. La expansión de Internet le ha dado una posición de privilegio y ha hecho que la idea inicial se modifique considerablemente.

En principio, la intención de HTML era que las etiquetas fueran capaces de marcar la información de acuerdo con su significado, sin importar cómo se mostraban en la pantalla. Lo importante era el contenido y no la forma, o sea, que era un lenguaje de marcas orientado a describir los contenidos, dejando a cada navegador la tarea de dar el formato del documento según su criterio de interpretar las marcas.

Por diversos motivos, los navegadores fueron añadiendo más etiquetas HTML dirigidas a controlar la presentación, y los usuarios las utilizaron para que sus documentos estuviesen perfectamente formateados, sin permitir diferencias importantes entre visualizadores distintos, por lo que HTML pasó a ser un lenguaje de marcas más dirigido al control de la presentación.

Aunque los estándares visuales y de interfase de usuario son una capa necesaria, no son suficientes para representar y administrar los datos. Hasta hace pocos años, Internet era un simple medio de acceso a texto e imágenes. No había ningún estándar establecido para la búsqueda inteligente, el intercambio de datos, la presentación adaptable ni para la personalización.

Internet debía ir más allá del establecimiento de un estándar de acceso y presentación de información, de forma de poseer un estándar para la comprensión de la información, una forma común de representar los datos para que el software pueda buscar, desplazar, presentar y manipular mejor los datos ocultos en una oscuridad contextual. HTML no puede hacerlo porque es un formato que describe la apariencia que debería tener una página Web.

HTML ofrece amplias facilidades de presentación, no ofrece ninguna forma basada en estándares para administrar los datos. Un estándar de representación de datos ampliaría Internet del mismo modo que el estándar de visualización HTML lo hizo hace pocos años.

Permitirá una gran variedad de nuevos usos, todos basados en una representación estándar para el desplazamiento de datos estructurados por todo la Web tan fácilmente como desplazamos las páginas HTML hoy en día.

1.3. ¿Qué es XML?

XML es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados, el cual conserva todas las

propiedades importantes del antes mencionado SGML. Es decir, XML es un metalenguaje, dado que con él podemos definir nuestro propio lenguaje de presentación y, a diferencia del HTML, que se centra en la representación de la información, XML se centra en la información en si misma. La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea, dependiendo del contenido del documento.

1.4. Objetivos y Orígenes

Un grupo de trabajo del consorcio World Wide Web (W3C) a partir de 1996 empezó a desarrollar XML, con el objetivo de desarrollar protocolos comunes para la evolución de Internet. Se trata de un consorcio de la industria internacional con sedes conjuntas en el Instituto Tecnológico de Massachussets, de Estados Unidos, el Instituto Nacional de Investigación en Informática y Automática europeo y la *Keio University Shonan Fujisawa Campus* de Japón. El W3C tiene como misión la publicación para uso público de protocolos o estándares globales de uso libre. Al comenzar el proyecto, los objetivos planteados por el grupo de desarrollo de XML fueron:

- XML debe ser directamente utilizable sobre Internet.
- XML debe soportar una amplia variedad de aplicaciones.
- XML debe ser compatible con SGML.
- Debe ser fácil la escritura de programas que procesen documentos XML.
- El número de características opcionales en XML debe ser absolutamente mínimo, idealmente cero.

- Los documentos XML deben ser legibles por los usuarios de este lenguaje y razonablemente claros.

- El diseño de XML debe ser formal, conciso y preparado rápidamente.

- XML debería ser simple pero perfectamente formalizado.

- Los documentos XML deben ser fáciles de crear.

- La brevedad en las marcas XML es de mínima importancia.

Capitulo 2

2. Características de XML

2.1. Introducción

XML es un formato basado en texto, diseñado para almacenar y transmitir datos. Un documento XML se compone de elementos XML, cada uno de los cuales consta de una etiqueta de inicio, de una etiqueta de fin y de los datos comprendidos entre ambas etiquetas, siendo de esta manera fáciles de crearlos.

2.2. Estándares abiertos

XML se basa en una tecnología desarrollada a partir de estándares probados y optimizada para la Web. La iniciativa XML consta de un conjunto de estándares relacionados entre sí:

- XML (*Extensible Markup Language*). Es una recomendación, que significa que el estándar es estable y que los desarrolladores de Web y de herramientas pueden adoptarlo plenamente.

- Namespaces. En XML es una recomendación que describe la sintaxis y la compatibilidad de los espacios de nombres para los intérpretes de XML.

- DOM (*Document Object Model*). Es una recomendación que ofrece un estándar para el acceso mediante programación a los datos estructurados (a través de *scripts*), de modo que los desarrolladores puedan interactuar de forma coherente con los datos basados en XML y computarlos.

- XSL (*Extensible Stylesheet Language*). XLS es la cara de presentación del XML. Este debe representar de forma independiente a la plataforma utilizada la información existente en los documentos XML.

- XML Linking Language. Es un lenguaje que ofrece vínculos en XML parecidos a los de HTML, pero más potentes. Los vínculos pueden tener varias direcciones y pueden existir en el nivel de los objetos, no sólo en el nivel de las páginas.

2.3. Características Principales

Extensible

Dentro de XML se pueden definir un conjunto ilimitado de etiquetas. Mientras que en HTML pueden utilizarse para desplegar una palabra en negrita o itálicas o dar cualquier tipo de formato al texto, en XML proporciona un marco de trabajo para etiquetado de datos estructurados. Al irse adoptando las etiquetas XML a lo largo de una intranet de alguna organización y a lo ancho de la Internet, habrá una correspondiente habilidad para buscar y manipular datos sin importar las aplicaciones dentro de las cuales se encuentre.

Representación estructural de los datos.

El XML proporciona una representación estructural de los datos que puede ser ampliamente implementable y fácil de distribuir. El XML es un subconjunto del SGML que está optimizado para su transmisión por Web; al estar definido por el Consorcio de la World Wide Web, asegura que los datos estructurados serán uniformes e independientes de aplicaciones o compañías. Esta interoperabilidad resultante está dando el inicio a una nueva generación de aplicaciones de Web para comercio electrónico.

El lenguaje XML proporciona un estándar de datos que puede codificar el contenido, la semántica y el esquema de una amplia variedad de casos que van desde simples a complejos, por ejemplo XML puede ser utilizado para marcar lo siguiente:

- Un documento ordinario.
- Un registro estructurado, tal como un registro de citas u órdenes de compra.
- Un registro de datos, tal como el resultado de una consulta.
- Metacontenido acerca de un sitio Web, tal como un Formato de Definición de Canal (*Channel Definition Format*, CDF).
- Presentaciones gráficas, tales como la interfase de usuario de una aplicación.

Una vez que los datos estén en el escritorio del cliente, pueden ser manipulados, editados, y presentados de una gran variedad de maneras, sin tener que regresar al servidor. Los servidores se pueden convertir ahora en más escalables, debido a las menores cargas de cálculo y ancho de banda. Además, dado que los datos son intercambiados en el formato XML, pueden ser fácilmente mezclados desde diferentes fuentes.

Los datos son separados de la presentación y el proceso.

La fortaleza de XML es que mantiene la separación entre la interfase de usuario y los datos estructurados. El HTML especifica como visualizar datos en un navegador, en cambio XML define el contenido. XML solo utiliza etiquetas para describir los datos. Para presentar los datos en un navegador XML, este utiliza hojas de estilo tales como el Lenguaje de Estilo Extensible (XSL) y las Hojas de Estilo en Cascada (CSS). El XML separa los datos de la presentación

y el proceso, permitiendo desplegar y procesar los datos tal como se desee, al aplicar diferentes hojas de estilo y aplicaciones.

Esta separación de datos de la presentación permite una integración de datos perfecta de fuentes diversas. Cualquier información puede ser convertida a XML, permitiendo a los datos ser intercambiados en línea tan fácilmente como las páginas de HTML despliegan datos hoy. Los datos codificados en XML pueden ser transmitidos sobre la Web hasta el escritorio. No es necesario retroajustar información en formatos propietarios almacenados en bases de datos o documentos de mainframes y, debido a que se usa el HTTP para transmitir documentos XML sobre la red, no se necesitan cambios para esta función. Los documentos XML son fáciles de crear.

Conversión de los datos XML en autodescriptivos.

Los datos codificados en XML son autodescriptivos, pues las etiquetas descriptivas están entremezcladas con los datos. El formato abierto y flexible utilizado por XML permite su uso en cualquier lugar donde sea necesario intercambiar y transferir información. Dado que el XML es independiente del HTML, se puede insertar código XML en documentos HTML. El W3C ha definido un formato mediante el cual se pueden encapsular en páginas HTML los datos basados en XML. Al incrustar datos XML en una página HTML, se pueden generar varias vistas a partir de los datos entregados, utilizando los datos semánticos que contiene el XML.

2.4. DTD

El DTD (definición del tipo de documento, *Document Type Definition*) proporciona la gramática para una clase de documentos XML. Esta gramática contiene la definición del conjunto de etiquetas que puede contener esa clase de documentos XML, los nombres que pueden utilizarse en los elementos, dónde pueden aparecer y cómo se interrelacionan entre ellos.

Los documentos XML enviados con un DTD se reconocen como "XML válido". En este caso, un intérprete de XML podría comparar los datos entrantes con las normas definidas en el DTD para comprobar que los datos se han estructurado correctamente. Los datos enviados sin un DTD se reconocen como "bien formados". En XML no existen DTDs predefinidos, por lo que es labor del diseñador especificar su propia DTD para cada tipo de documento XML. En la especificación de XML se describe la forma de definir DTDs particularizadas para documentos XML, que pueden ser internas (cuando van incluidas junto al código XML) o externas (si se encuentran en un documento propio).

2.5. Esquemas

Últimamente se está imponiendo otra forma más eficaz de definir la estructura de un documento XML, conocida como esquemas. Un esquema es una especificación formal de las normas de un documento XML, que indica qué elementos se permiten en un documento y en qué combinaciones están permitidas. Los nuevos lenguajes de esquemas, definidos en las propuestas XML-Data (Datos de XML) y DCD (Descripción del contenido del documento) enviadas por el XML-Data Working Group al W3C, proporcionan las mismas funciones que un documento DTD. No obstante, puesto que estos lenguajes de esquema son extensibles, los desarrolladores pueden ampliarlos con información adicional, como los tipos de datos, la herencia y las normas de presentación. Esto hace que los nuevos lenguajes de esquemas sean mucho más potentes que los DTD.

La expresión de esquemas dentro de XML aumenta la potencia del formato XML, pues permite que el software examine determinados datos para comprender su estructura, sin necesitar ninguna descripción previa incorporada de la estructura de los datos. Con un esquema, un autor puede definir exactamente qué nombres de

elementos se permiten en un documento y, dentro de cada elemento, qué subelementos, atributos y relaciones se admiten. El autor puede importar fragmentos de otros esquemas, así como ampliar tipos a través de la herencia. Todo ello permite establecer relaciones complejas entre los elementos sin perder la simplicidad de la estructura de árbol léxico.

2.6. Fortalezas y Debilidades

La meta fundamental del XML es hacer la cooperación y la interoperabilidad más fácil entre módulos que pertenecen a diferentes sistemas, e incluso a diferentes organizaciones. Entonces, ¿por qué el XML se ha convertido en tecnología del alto nivel tan rápidamente? La respuesta es simple: XML es texto simple, pero es también elegante. Un lenguaje de marcas hace fácil identificar (y extraer en un tiempo posterior) segmentos específicos de información con significados especiales. Usar texto simple y mantener el contenido universalmente interpretable son dos condiciones para implementar una tecnología exitosamente entre sistemas heterogéneos. Una cadena del texto se puede transferir y se puede manejar fácilmente en cualquier plataforma destino. Sin embargo, si solo se planea definir un protocolo basado en texto para hacer que los módulos se comuniquen, no es necesario XML. El ASCII, de hecho, ha existido durante mucho tiempo. La diferencia recae en las etiquetas XML que se utilizan para delimitar segmentos de información.

La principal fortaleza del XML es también, actualmente, su debilidad principal. El XML es demasiado genérico para ser utilizado sin definir externamente la sintaxis exacta de un documento y cómo puede localizarse y extraerse cada fragmento de datos intercambiados. Tal estandarización es más simple alcanzar dentro de una sola aplicación corporativa, incluso si atraviesa varias plataformas heterogéneas de hardware y de software. Conseguir el mismo resultado en un contexto

más amplio requiere un enorme esfuerzo para llegar a algunas definiciones estándares.

Desgraciadamente, cuando se trata de integrar sistemas de diferentes organizaciones, especialmente de una manera abierta, el XML muestra algunas limitaciones. El XML trata de la descripción de los datos, pero solamente es útil cuando la gente está de acuerdo en la manera que los datos deben ser descritos. Una factura, por ejemplo, es lógicamente el mismo tipo de documento para cualquier proceso que lo manipule. Pero diferentes procesos pueden convertirla en formatos binarios diferentes. A pesar del formato de almacenamiento físico, el documento necesita ser transmitido y ser recibido en un formato comúnmente reconocido disponible en todas las plataformas. Por supuesto, el XML se puede utilizar como este formato.

Por este motivo, en la actualidad se están definiendo esquemas por grupos sectoriales con similares intereses, de forma que existirán esquemas estándares avalados por asociaciones de empresas y organismos que garanticen que cualquier usuario que las adopte como suyas, trabaje con las mismas etiquetas. Como ejemplos de estos esquemas estándares tenemos: CDF - *Channel Definition Format* (define canales para enviar información periódica a los clientes), CML - *Chemical Markup Language* (define información del sector químico), MathML - *Mathematical Markup Language* (define datos matemáticos), SMIL - *Synchronized Multimedia Integration Language* (define presentaciones de recursos multimedia).

BizTalk

BizTalk es una iniciativa mundial dirigida a crear una base de datos de formatos o esquemas de documentos en XML. Importantes compañías se han unido ya a BizTalk, que ofrece un gran número de esquemas de XML a los cuales las compañías interesadas puedan referirse. Un documento de BizTalk es un documento XML que

proporciona las etiquetas de un cierto vocabulario y sigue las reglas que la organización ha definido para este tipo de documento. Este sitio crecerá hasta convertirse en un portal para localizar, administrar, publicar y obtener información sobre XML, XSL y los modelos de información utilizados en miles de aplicaciones.

Microsoft pretende establecer por medio de BizTalk un marco para el comercio electrónico de empresa a consumidor.

2.7. DOM

Supongamos que usted es un programador de Visual Basic y ha recibido algunos datos en un documento XML. Ahora desea extraer la información del documento XML e integrar esos datos en sus soluciones de Visual Basic. Por supuesto, podría escribir código para analizar el contenido del documento XML, pues no deja de ser un documento de texto. Sin embargo, esta solución no sería muy productiva y desaprovecharía una de las ventajas de XML**:** el ser una forma estructurada de representar datos. Una forma mejor de recuperar información de documentos XML es utilizar un analizador o intérprete de XML. Un intérprete de XML es, a grandes rasgos, un programa que lee un documento XML y permite disponer de sus datos.

En su calidad de programador en Visual Basic, querrá utilizar un intérprete que sea compatible con el DOM (modelo de objeto de documento, *Document Object Model*) de XML. Este define un conjunto estándar de comandos que los intérpretes exponen para facilitarle el acceso al contenido de los documentos XML desde sus programas. Un intérprete de XML que sea compatible con DOM toma los datos de un documento XML y los expone mediante un conjunto de objetos que se pueden programar. Además, un intérprete de XML puede utilizar un DTD o un esquema para determinar si un documento es válido.

DOM para XML es un modelo de objetos que muestra el contenido de un documento XML. La Especificación de nivel 1 del DOM del W3C define actualmente lo que debería mostrar un DOM como propiedades, métodos y eventos. Para utilizar XML DOM, hay que crear una instancia de un intérprete XML.

Métodos y propiedades especificados por DOM:

- Método Load: que permite cargar documentos XML procedentes de un disco local, de la red o de una dirección URL.

- Propiedad Async: determina si el intérprete de XML carga los documentos de manera asincrónica o no.

- Propiedad ReadyState: indica en que estado se encuentra la carga del documento

- Propiedad ValidateOnParse: indicar al intérprete que no valide el documento.

- Propiedad ChildNodes: muestra la lista de nodos XML DOM.

- Propiedad Level: que devuelve el número de nodos secundarios existentes.

- Propiedad HasChildNodes: facilita el recorrido de la jerarquía de nodos para examinar elementos, atributos y valores.

Capitulo 3

3. Documentos XML en el Web

3.1. Desplegar documentos

Los navegadores, primero chequean la sintaxis del código del documento y después presentan la información del documento con un formato determinado. Los documentos HTML utilizan las descripciones de formatos internas del propio navegador, o si existen descripciones CSS (opcionales), utilizan la información de la hoja de estilo para ajustar la presentación en la pantalla. Los documentos XML siempre necesitan normas que describan su presentación. Para esto podemos optar por dos soluciones: las mismas descripciones CSS que se utilizan con HTML y/o las descripciones que se basan en XSL.

CSS describe formatos y presentaciones, pero no sirve para decidir qué tipos de datos deben ser mostrados y cuáles no. Esto es, CSS se utiliza con documentos XML en los casos en los que todo su contenido debe mostrarse sin mayor problema. XSL no solo permite especificar cómo queremos presentar los datos de un documento XML, sino que también sirve para filtrar los datos de acuerdo a ciertas condiciones. Se parece un poco más a un lenguaje de programación.

3.1.1. CSS

Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores. Básicamente, una hoja de estilos es un

conjunto de especificaciones de formato que se aplican a los elementos HTML.

La idea detrás de CSS es que exista una separación entre los contenidos por un lado y la forma de presentarlos por otro.

Los beneficios de usar CSS son dobles. Por un lado, evitamos hacer a los archivos demasiado pesados (excluyendo el largo código requerido para las tablas anidadas y el añadido de características gráficas), y definimos el "estilo visual" de un sitio entero sin necesidad de hacerlo etiqueta por etiqueta, para cada una de las páginas. Por otro, trabajamos con estándares, y separamos hasta cierto punto la estructura (o código) de la presentación, logrando una manera más nítida de trabajar, y lo que es más: en un sencillo documento CSS, es una plantilla gráfica para todo un sitio. Vale decir, que cualquier cambio hecho a un estilo CSS, se reflejará en todos los elementos que sean referidos a éste, automáticamente, con sólo editar un sencillo documento CSS.

3.1.2. XSL

SGML tiene su estándar para representar sus documentos, el DSSSL (*Document Style Semantics and Specification Language*). Como XML es una versión reducida de SGML parecía lógico hacer una versión reducida del DSSSL, llamada XSL (lenguaje de hojas de estilo extensible, *Extensible Stylesheet Language*).

XSL es un lenguaje de hojas de estilos diseñado para su utilización en el Web. XLS debe representar de forma independiente a la plataforma utilizada o al medio de representación la información recogida en los documentos XML.

En cuanto a la inclusión de imágenes en las páginas XML, estas son enlaces, que pueden representarse por alguno de los tipos soportado por las especificaciones XLink y XPointer. XSL, además de permitir la descripción de la presentación física, también posibilita la ejecución de bucles, sentencias del tipo IF...THEN, selecciones por comparación, operaciones lógicas, ordenaciones de datos, utilización de plantillas, etc.

3.2. Vínculos entre documentos XML(XLink/XPointer)

Los enlaces e hipervínculos son importantes para los documentos XML, el W3C ha sacado las especificaciones que las controlan fuera de las descripciones del DTD, creando 2 normas: XLink y XPointer. Estas definen el modo de enlace entre diferentes documentos. Este lenguaje va más allá de los enlaces simples que sólo soporta el HTML. Esta especificación soporta las siguientes características:

- 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)
- Hiperenlaces múltiples (anillos, múltiples ventanas, etc).
- Enlaces agrupados (múltiples orígenes).

XLink y XPointer

XPointer es una extensión de Xpath que permite cargar en un visualizador de documentos XML solo aquellos elementos de un documento que nos interesen.

Su equivalente en HTML es la etiqueta <A NAME="nombre">, con la cual se llama a otro documento.

XPointer es similar, pero más potente. XPointer va a permitir añadir a una dirección del tipo http://www.sitio.com/documento.xml, el complemento #xpointer(expresión), donde expresión es una expresión XPath, con algunas propiedades extra que no contempla el propio XPath.

Uso de XPointer

XPointer es una extensión de XPath. Es imprescindible saber XPath para poder usarlo.

Una expresión XPointer se añade a un URI (*Uniform Resource Identifier*), como puede ser un URL (*Uniform Resource Locator*) o un URN (*Uniform Resource Name*).

La idea es añadir al final del URI lo siguiente:

#xpointer(expresion)

Hay que tener en cuenta que se pueden concatenar expresiones XPointer las cuales se evalúan de izquierda a derecha mientras devuelvan un conjunto vacío de nodos. Ejemplo:

documento.xml#xpointer(/libro/capitulo[@public])xpointer(/libro/capitulo[@num="2"])

En primer lugar se buscaría el conjunto de nodos delimitado por /libro/capitulo, y solo en el caso de que no existiese ninguno, se buscaría a continuación por /libro/capitulo[@num="2"].

Extensiones a XPath

Funciones

Una de las funciones que permiten localizar más rápidamente un elemento dentro de un documento XML es la función id(), la cual solo puede ser usada en aquellos ficheros XML que tengan elementos para los que se ha definido el atributo id y tienen asociado algún DTD que especifique que dicho identificador es único e irrepetible.

XPointer añade a la función id() las funciones here() y origin().

here() se utiliza en expresiones XPointer que apuntan a zonas internas del propio documento en que se utilizan dichas expresiones (normalmente este "apuntar" se hace mediante XLink), y hace referencia al nodo contexto desde el cual se hace la llamada mediante el URI.

En cuanto a origin() se suele utilizar también en expresiones XPointerrelacionada con enlaces de XLink, y refiere al elemento desde el cual se produjo el salto con el XLink.

Puntos y rangos

XPath es una herramienta potente para seleccionar aquellas partes de un elemento XML que están perfectamente etiquetadas, pero no lo es tanto para seleccionar partes del documento XML que están dentro de un nodo texto. Por ejemplo, con XPAth podemos seleccionar perfectamente un nodo como <autor>José Santos Rodríguez</autor>, pero es algo muy diferente el seleccionar la primera palabra del contenido de dicho elemento.

XPointer considera que entre cualesquiera dos caracteres consecutivos de texto de un documento XML, así como entre cada par de elementos también consecutivos, existe un punto. El fragmento de texto que existe entre dos puntos es un rango.

Para usar los puntos en una expresión XPointer, hemos de recurrir a la función point() a la que hay que añadirle un predicado indicando qué punto en concreto deseamos seleccionar.

Rangos

Los rangos son sin duda también esenciales para trabajar con XPointer. Por ejemplo, un usario puede seleccionar con el ratón un trozo de un documento que puede perfectamente ir de la mitad de un párrafo hasta tres capítulos más abajo. Cualquier zona continua de un documento puede ser descrita con un rango. Un rango comienza en un determinado punto y finaliza en otro, cada uno de ellos determinado por una expresión XPath. Si el punto inicial determina no a un punto sino a un conjunto de nodos, entonces el primer punto de dicho conjunto se considera el punto inicial. E igual ocurre con el punto final: si su expresión XPath hace referencia no a un punto sino a un conjunto de nodos, se elegirá como punto final del rango el último punto de dicho conjunto de nodos.

Para especificar un rango añadimos la función /range-to (punto-final) a la expresión XPath del nodo inicial, siendo punto-final la expresión XPath del nodo final.

Funciones para rangos

- XPointer incluye funciones para el tratamiento de rangos. La mayoría de ellas trabajan no con conjuntos de nodos, sino con conjuntos de localizaciones, que en definitiva son conjuntos de nodos, más puntos, más rangos.

- range(*location-set*) devuelve un conjunto de localizaciones porcada localización existente en el argumento. El rango es el mínimo necesario para cubrir la localización entera. Vamos, que convierte localizaciones en rangos.

- range-inside(*location-set*) devuelve un conjunto de localizaciones conteniendo el interior de cada una de las localizaciones que se pasan como parámetro. Si una de las localizaciones es un elemento, devolvería su contenido, pero no el elemento en sí, pero si es un punto o un rango, devuelve dicho punto o rango.

- start-point(*location-set*) devuelve un conjunto de localizaciones que contiene un punto que representa el primer punto de cada una de las localizaciones que se le pasan.

- end-point(*location-set*) devuelve justo un conjunto de localizaciones que contiene posterior a cada una de las localizaciones que se le pasan.

Rangos de cadena

XPointer proporciona alguna capacidad muy básica para la selección de cadenas gracias a la función string-range(). Esta función necesita dos parámetros, en primer lugar un conjunto de nodos en el que buscar, y a continuación una cadena que buscar en ellos. Devuelve un conjunto de nodos que contiene un rango por cada aparición de la cadena buscada. Se pueden agregar dos parámetros opcionales, índice y longitud, para indicar cuántos caracteres pasado el emparejamiento se deben pasar para empezar el rango y cuántos caracteres debe tener éste.

Secuencias de hijos

Una secuencia de hijos es una abreviatura usada en XPointer para recuperar elementos por su posición, más que por su nombre.

En caso de que los elementos tengan identificadores, se pueden usar también estos atajos, de forma que #p1/4 haría referencia al cuarto hijo del elemento identificado como p1.

XLink define la forma en la que los documentos XML deben conectarse entre sí. XPointer describe cómo se puede apuntar a un lugar específico de un determinado documento XML. Resumiendo, XLink determina el documento al que se desea acceder y XPointer marca el lugar exacto en dicho documento. Al contrario de lo que ocurre con HTML, en XML existen dos tipos básicos de hipervínculos: simples y extendidos.

Por desgracia, aún no hay muchas herramientas que soporten XPointer, de hecho es un estándar que aún esta en discusión, pero será una herramienta muy útil cuando este disponible.

Capitulo 4

4. Visual Studio Net

4.1. Introducción a Visual Studio Net

Visual Studio .NET es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones empresariales y de misión crítica, sino que también nos provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. Los componentes principales de la plataforma .NET son:

Un entorno de ejecución de aplicaciones, también llamado "Runtime", que es un componente de software cuya función es la de ejecutar las aplicaciones .NET e interactuar con el sistema operativo ofreciendo sus servicios y recursos.

Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.

Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores y linkers, que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.

Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones

Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones .NET

Por otra parte, .NET representa la evolución COM (Component Object Model), la plataforma de desarrollo de Microsoft anterior a .NET y sobre la cual se basaba el desarrollo de aplicaciones Visual Basic 6 (entre otros tantos lenguajes y versiones).

Algunas de las ventajas e inconvenientes de la plataforma .Net.

A continuación se resumen las ventajas más importantes que proporciona Net Framework:

- Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

- Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL).

- Compilación just-in-time: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.

- Garbage collector: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (garbage collector). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por

que liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente (mediante el método disponse() liberamos el objeto para que el recolector de basura lo elimine de memoria).

- Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.

- Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

¿Todo son ventajas?

Procesos como la recolección de basura de .Net o la administración de código introducen factores de sobrecarga que repercuten en la demanda de más requisitos del sistema.

El código administrado proporciona una mayor velocidad de desarrollo y mayor seguridad de que el código sea bueno. En contrapartida el consumo de recursos durante la ejecución es mucho mayor, aunque con los procesadores actuales esto cada vez es menos inconveniente.

El nivel de administración del código dependerá en gran medida del lenguaje que utilicemos para programar. Por ejemplo, mientras que Visual Basic .Net es un lenguaje totalmente administrado, C Sharp (C #) permite la administración de código de forma manual, siendo por

defecto también un lenguaje administrado. Mientras que C++ es un lenguaje no administrado en el que se tiene un control mucho mayor del uso de la memoria que hace la aplicación.

4.2. Características de Visual Studio Net

A continuación se muestran algunas de las características principales de la plataforma Microsoft .NET:

- Es una plataforma de ejecución intermedia, ya que las aplicaciones .NET no son ejecutadas directamente por el sistema operativo, como ocurre en el modelo tradicional de desarrollo. En su lugar, las aplicaciones .NET están diseñadas para ser ejecutadas contra un componente de software llamado Entorno de Ejecución (muchas veces también conocido como "Runtime", o , "Máquina Virtual"). Este componente es el encargado de manejar el ciclo de vida de cualquier aplicación .NET, iniciándola, deteniéndola, interactuando con el Sistema Operativo y proveyéndole servicios y recursos en tiempo de ejecución.

- La plataforma Microsoft .NET está completamente basada en el paradigma de Orientación a Objetos.

- .NET es multi-lenguaje: esto quiere decir que para poder codificar aplicaciones sobre esta plataforma no necesitamos aprender un único lenguaje específico de programación de alto nivel, sino que se puede elegir de una amplia lista de opciones.

- .NET es una plataforma que permite el desarrollo de aplicaciones empresariales de misión crítica, entendiéndose por esto que permite la creación y ejecución de aplicaciones de porte corporativo que sean críticas para la operación de tipos variados de organizaciones. Si bien también es muy atrayente

para desarrolladores no profesionales, estudiantes y entusiastas, su verdadero poder radica en su capacidad para soportar las aplicaciones más grandes y complejas

- .Net fue diseñado de manera tal de poder proveer un único modelo de programación, uniforme y consistente, para todo tipo de aplicaciones (ya sean de formularios Windows, de consola, aplicaciones Web, aplicaciones móviles, etc.) y para cualquier dispositivo de hardware (PC's, Pocket PC's, Teléfonos Celulares Inteligentes, también llamados "SmartPhones", Tablet PC's, etc.). Esto representa un gran cambio con respecto a las plataformas anteriores a .NET, las cuales tenían modelos de programación, bibliotecas, lenguajes y herramientas distintas según el tipo de aplicación y el dispositivo de hardware.

- Uno de los objetivos de diseño de .NET fue que tenga la posibilidad de interactuar e integrarse fácilmente con aplicaciones desarrolladas en plataformas anteriores, particularmente en COM, ya que aún hoy existen una gran cantidad de aplicaciones desarrolladas sobre esa base.

- .NET no sólo se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, sino también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación. Para esto hace un uso extensivo de numerosos estándares globales que son de uso extensivo en la industria, algunos ejemplos de estos estándares son XML, HTTP, SOAP, WSDL y UDDI.

4.3. Clase System.xml

La Clase System.xml es un conjunto de temas que trata el uso del las clases XML del espacio de nombres System.xml.

Este espacio de nombres tiene un conjunto completo de clases XML para análisis, validación y manipulación de datos XML mediante sistemas de lectura, sistemas de escritura y componentes compatibles con el consorcio World Wide Web (W3C) DOM. También se explican las consultas XPath (XML Path Language) y las transformaciones XSLT (Extensible Stylesheet Language Transformations). La lista siguiente contiene las clases principales del espacio de nombres XML

Clase	Descripción
NameTable	Implementa XmlNameTable de un único subproceso.
XmlAttribute	Representa un atributo. Los valores válidos y predeterminados del atributo se definen en una definición de tipo de documento (DTD) o en un esquema.
XmlAttributeCollection	Representa una colección de atributos a los que se puede obtener acceso por nombre o por índice.
XmlCDataSection	Representa una sección CDATA.
XmlCharacterData	Proporciona métodos de manipulación de texto que son utilizados por varias clases.
XmlComment	Representa el contenido de un comentario XML.
XmlConvert	Codifica y descodifica nombres

	XML y proporciona métodos de conversión entre tipos de Common Language Runtime y tipos de esquemas del lenguaje de definición de esquemas XML (esquemas XSD). Cuando se convierten tipos de datos, los valores devueltos no dependen de la configuración regional.
XmlDataDocument	Permite que los datos estructurados se almacenen, recuperen y manipulen mediante un DataSet relacional.
XmlDeclaration	Representa el nodo de declaración XML <?xml version='1.0' ...?>.
XmlDocument	Representa un documento XML.
XmlDocumentFragment	Representa un objeto pequeño tamaño, que resulta útil para realizar operaciones de inserción de árboles.
XmlDocumentType	Representa la declaración de tipo de documento.
XmlElement	Representa un elemento.
XmlEntity	Representa una declaración de entidad, como <!ENTITY... >.
XmlEntityReference	Representa un nodo de referencia a entidad.
XmlException	Devuelve información detallada sobre la última excepción.

XmlImplementation	Define el contexto para un conjunto de objetos XmlDocument.
XmlLinkedNode	Obtiene el nodo inmediatamente anterior o siguiente a éste.
XmlNamedNodeMap	Representa una colección de nodos a los que se puede tener acceso por nombre o por índice.
XmlNamespaceManager	Resuelve, agrega y quita espacios de nombres en una colección y proporciona la administración del ámbito de estos espacios de nombres.
XmlNameTable	Tabla de objetos en forma de cadena subdividida.
XmlNode	Representa un único nodo del documento XML.
XmlNodeChangedEventArgs	Proporciona datos para los eventos NodeChanged, NodeChanging, NodeInserted, NodeInserting, NodeRemoved y NodeRemoving.
XmlNodeList	Representa una colección ordenada de nodos.
XmlNodeReader	Representa un lector que proporciona acceso rápido, sin almacenamiento en caché y con desplazamiento sólo hacia delante, a los datos XML de un objeto XmlNode.

XmlNotation	Representa una declaración de notación, tal como <!NOTATION... >.
XmlParserContext	Proporciona toda la información de contexto que necesita el objeto XmlReader para analizar un fragmento de XML.
XmlProcessingInstruction	Representa una instrucción de procesamiento que XML define para conservar información específica del procesador en el texto del documento.
XmlQualifiedName	Representa un nombre XML completo.
XmlReader	Representa un lector que proporciona acceso rápido a datos XML, sin almacenamiento en caché y con desplazamiento sólo hacia delante.
XmlReaderSettings	Especifica un conjunto de características compatibles en el objeto **XmlReader** creado mediante el método Create.
XmlResolver	Resuelve los recursos XML externos designados por un identificador de recursos uniforme (URI).
XmlSecureResolver	Ayuda a proteger otra implementación de XmlResolver ajustando el objeto **XmlResolver** y

	restringiendo los recursos a los que tiene acceso el **XmlResolver** subyacente.
XmlSignificantWhitespace	Representa el espacio en blanco entre marcas en un nodo de contenido mixto o espacio en blanco dentro del ámbito xml:space= "preserve". También se hace referencia a esto como espacio en blanco significativo.
XmlText	Representa el contenido de texto de un elemento o atributo.
XmlTextReader	Representa un lector que proporciona acceso rápido a datos XML, sin almacenamiento en caché y con desplazamiento sólo hacia delante.
XmlTextWriter	Representa un sistema de escritura que proporciona un medio rápido, sin almacenamiento en caché y con desplazamiento sólo hacia delante para generar secuencias o archivos con datos XML que satisface las recomendaciones relativas a espacios de nombres en XML y Extensible Markup Language (XML) 1.0 del Consorcio W3C.
XmlUrlResolver	Resuelve los recursos XML externos designados por un identificador de recursos uniforme (URI).

XmlValidatingReader	Representa un lector que proporciona validación de definiciones de tipos de documentos (DTD), de esquemas reducidos de datos XML (esquemas XDR) y del lenguaje de definición de esquemas XML (esquemas XSD).
XmlWhitespace	Representa los espacios en blanco en el contenido del elemento.
XmlWriter	Representa un sistema de escritura que constituye un medio rápido, no almacenado en caché y de sólo avance para generar secuencias o archivos con datos XML.
XmlWriterSettings	Especifica un conjunto de características compatibles en el objeto XmlWriter creado mediante el método System.Xml.XmlWriter.Create.

Capitulo 5

5. Desarrollo de la aplicación

5.1. Introducción de la aplicación

Como resultado de la investigación realizada sobre el tema XML se ha implementado una aplicación principalmente en la herramienta de desarrollo Visual Studio .Net.

La aplicación fue realizada para un fotógrafo profesional, el cual desea publicar su trabajo en la Web, ya que hoy en día acceder a la información que esta disponible en Internet es muy fácil, además a través de esta aplicación puede publicar de una manera sencilla y eficaz todo su trabajo y darlo a conocer a cualquier persona que tenga acceso a Internet.

Para responder con las necesidades planteadas por el fotógrafo hemos realizado una Galería Virtual en la cual se exponen las imágenes de sus distintos trabajos clasificadas por diversos temas.

Para poder lograr esto contamos con una base de datos en Microsoft Access, en esta base de datos se almacena toda la información de las imágenes, los temas, y las diversas técnicas de fotografía. Además el fotógrafo cuenta con un sistema de administración de toda la información la cual le permite ingresar en cualquier momento las imágenes, así como agregar temas y técnicas. De igual manera puede eliminar la información que no necesite.

5.2. Desarrollo de la Aplicación

La aplicación en su mayoría fue desarrollada en Visual Studio .Net, cuenta con una base de datos Microsoft Access y desde Visual Studio se generan documentos XML para interactuar con Macromedia Flash. Como primer paso se creó la base de datos en Microsoft Access, la base de datos tiene el nombre de galeria. En el siguiente diccionario de datos se muestran todas las tablas con sus respectivas relaciones.

Tabla: Fotos

Campo	Cd_foto
Descripción	Contiene el código de la fotografía
Tipo	Auto Numérico
Llave Principal	Si
Llave Foránea	No

Campo	Fot_descripcion
Descripción	En este campo se incluirá información como el tipo de cámara o la velocidad a la que fue tomada la foto
Tipo	Caracter (50)
Llave Principal	No
Llave Foránea	No

Campo	Fot_nombre

Descripción	Contiene el nombre de la foto
Tipo	Caracter (30)
Llave Principal	No
Llave Foránea	No

Campo	Cd_tema
Descripción	Contiene el código del tema de la fotografia
Tipo	Numérico
Llave Principal	No
Llave Foránea	Si

Campo	Cd_tecnica
Descripción	Contiene el codigo de la Técnica de la fotografia
Tipo	Numérico
Llave Principal	No
Llave Foránea	Si

Campo	Ancho
Ancho	Contiene el ancho de la imagen en pixeles
Tipo	Numérico
Llave Principal	No
Llave Foránea	No

Campo	Alto
Descripción	Alto de la imagen en píxeles
Tipo	Numérico
Llave Principal	No
Llave Foránea	No

Campo	x
Descripción	Posición de imagen en el eje de las x
Tipo	Numérico
Llave Principal	No
Llave Foránea	No

Campo	Y
Descripción	Posición de la imagen en el eje de la y
Tipo	Numérico
Llave Principal	No
Llave Foránea	No

Tabla Temas

Campo	Cd_tema
Descripción	Código del tema
Tipo	Auto Numérico
Llave Principal	Si
Llave Foránea	No

Campo	Tem_nombre
Descripción	Nombre del tema
Tipo	Caracter (50)
Llave Principal	No
Llave Foránea	No

Campo	Tem_intro
Descripción	Breve introducción del tema
Tipo	Memo
Llave Principal	No
Llave Foránea	No

Tabla Técnicas

Campo	Cd_tecnica
Descripción	Código de la técnica
Tipo	Auto Numérico
Llave Principal	Si
Llave Foránea	No

Campo	Tec_tipo
Descripción	Código del tema
Tipo	Caracter (50)
Llave Principal	No
Llave Foránea	No

Tabla Usuarios

Campo	Nombre_usuario
Descripción	Nombre del Usuario
Tipo	Caracter (20)
Llave Principal	Si
Llave Foránea	No

Campo	Contraseña
Descripción	Contraseña del administrador
Tipo	Caracter (20)
Llave Principal	No
Llave Foránea	No

Existirán dos tipos de páginas Web: las que serán visitadas por el usuario y las que serán manipuladas por el administrador o fotógrafo para realizar las diferentes modificaciones y actualizaciones de las fotografías.

Sección del Usuario

La página principal que visitará el usuario tiene una animación creada en Macromedia Flash en la cual existen las siguientes opciones:
(Ver anexo 1)

- **Acerca de Mí:** Al ubicarnos sobre esta opción aparece información relacionada con la trayectoria del fotógrafo.

- **Galería:** Al elegir esta opción nos dirigimos hacia una nueva página donde se cargan los temas existentes, estos temas son leídos de un archivo XML el cual es generado en la sección del administrador. Cada vez que el administrador o fotógrafo crea, modifica o elimina un tema en la bases de datos este archivo XML es actualizado., de esta manera los temas de la galería siempre están actualizados. Además en la parte central derecha se cargan las fotos dependiendo del tema que se ha

seleccionado, estas imágenes también son leídas de un archivo XML que de igual manera se genera desde la sección del administrador. Para poder cargar los diferentes archivos XML en la aplicación de flash usamos comandos de Action Script de Macromedia Flash, a través de estos comandos se lee el XML y se lo carga.
(Ver Anexo 2)

- **Contactos**: Esta opción muestra en la pantalla la información necesaria para contactarse con el fotógrafo.

- **Administrador**: Por esta opción el administrador ingresa a realizar los cambios que necesite en la información.

- **XSL**: Esta opción nos lleva a una nueva página donde se pueden ver ejemplos del uso de XSL.

Sección del administrador

Una vez que se ingresó en esta sección se pide un nombre de usuario y contraseña ya que en esta sección solamente puede ingresar el administrador. Para lograr que solamente el administrador ingrese utilizamos la autenticación mediante formularios con la cual creamos una Cookie de control con la cual el administrador puede ingresar a todas las páginas para manipular la información. En todas las páginas del administrador existe un botón a través del cual puede cerrar su cesión cuando termine de realizar su trabajo.
(Ver Anexo 3)

Si no se ha generado esta Cookie toda persona que intente ingresar a cualquier página del administrador no podrá hacerlo y será redirigida a la página Login.aspx donde se le pide el nombre de usuario y contraseña. (Ver Anexo4)

En esta sección del administrador es donde se generan los archivos XML los cuales posteriormente serán utilizados en la galería. Se genera un archivo XML de temas, uno de técnicas y por cada tema ingresado generamos un XML de fotos. Así obtenemos un solo XML por cada tema y se lo lee directamente desde flash, de esta manera no se cargan todas las fotos sino solamente las que se necesitan, solo se cargan las de un tema por cada evento On Press de Macromedia Flash.

Los archivos XML mencionados anteriormente (temas, técnicas, fotos) son generados después que el administrador realiza un ingreso, modificación o eliminación, para lograr generar los XML utilizamos la clase System.xml de Visual Studio .Net la cual nos permite escribir los archivos XML. En nuestro caso leemos la base de datos y de los resultados de esa lectura son generados los archivos.

6. Conclusiones y Recomendaciones

La utilización de XML abre una nueva perspectiva en el tratamiento y recuperación de la información y se ha convertido en una herramienta de trabajo fundamental en los diferentes entornos documentales.

Es deseable que todas las experiencias que surjan, traten siempre de ofrecer compatibilidad con los estándares en los que se basan, evitando propuestas fuera de un estándar establecido, con el fin de conseguir sistemas fiables y que puedan ser utilizados por todos, posibilitando, en definitiva, el buen aprovechamiento de la información en todas sus facetas.

Con la realización de este trabajo recomendamos utilizar XML en las diferentes aplicaciones ya que de esta forma estamos garantizando que nuestra información es compatible con todas las plataformas de desarrollo. Además los datos pueden ser manejados de una manera rápida y sencilla.

7. Glosario

XML *Extensible Markup Language*: Lenguaje de Marcas Extensible

HTML *Hypertex Markup Language*: Lenguaje de Marcas de Hipertexto

Javascripts: Aplicaciones en Lenguaje Java

Scripts: Pequeñas aplicaciones usasdas en la Web

SGML Standard Generalized Markup Language: Lenguaje Estandar Generalizado de Marcas

W3C: Consorcio World Wide Web

Namespaces: Espacios de Nombres

DOM *Document Object Model:* Modelo de Objeto de Documento

XSL (*Extensible Stylesheet Language*): Lenguaje de Hojas de Estilo Extensible para ser usadas en XML

CDF: Esquema que define canales para enviar información periódica a los clientes

DTD (Document Type Definition): Definición del Tipo de Documento

XML-Data: Datos de XML

CML - Chemical Markup Language: Esquema que define información del sector químico

MathML - Mathematical Markup Language: Esquema que define datos matemáticos

SMIL - *Synchronized Multimedia Integration Language*: Esquema que define presentaciones de recursos multimedia

CSS: Hojas de estilo en cascada

DSSSL (Document Style Semantics and Specification Language): Estandard de SGML para representar sus documentos

URL (Uniform Resource Locutor): Localizador Uniforme de Recurso

URI (Uniform Resource Identifier): Identificador Uniforme de Recursos

URN (Uniform Resource Name): Nombre Uniforme de Recurso

XPointer: Define cómo se puede apuntar a un lugar específico de un determinado documento XML

XLink: Define la forma en la que los documentos XML deben conectarse entre sí

8. Bibliografía

Microsoft. Página principal de msdn

- www.microsoft.com/spanish/msdn/default.mspx
- Fecha de Ingreso: Febrero – Marzo 2007

World Wide Web Consortium. Página principal del W3C

- www.w3.org
- Fecha de Ingreso: Febrero – Marzo 2007

O'Reilly Media.

- www.mxl.com
- Fecha de Ingreso: Febrero 2007

Anexo1

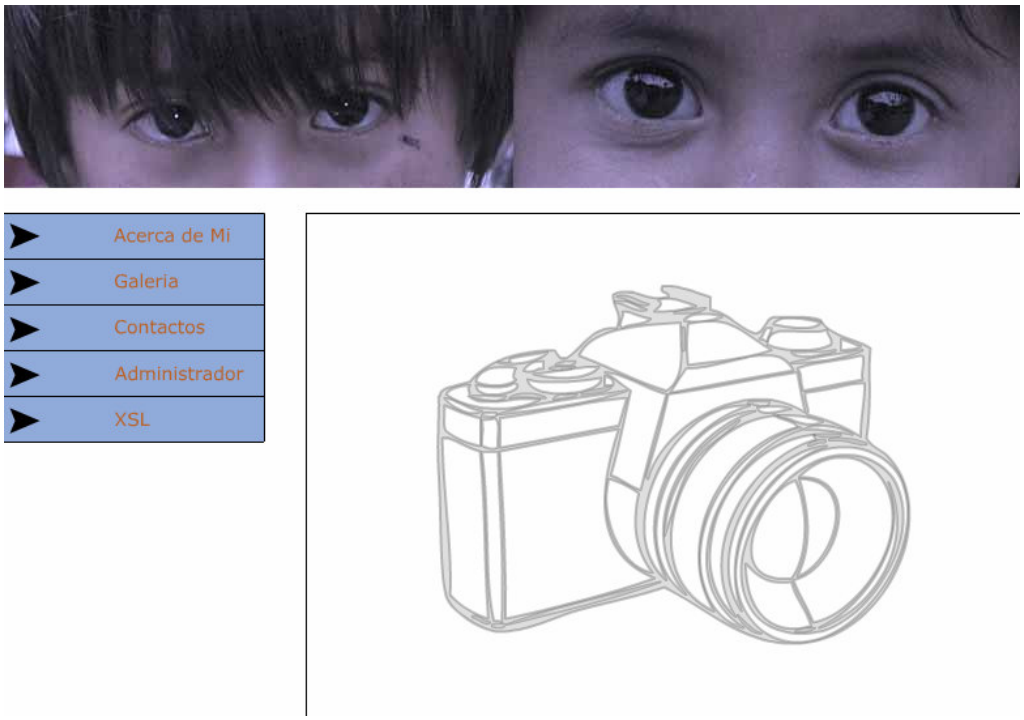

Figura1: Pagina inicial de la Galería

Anexo2

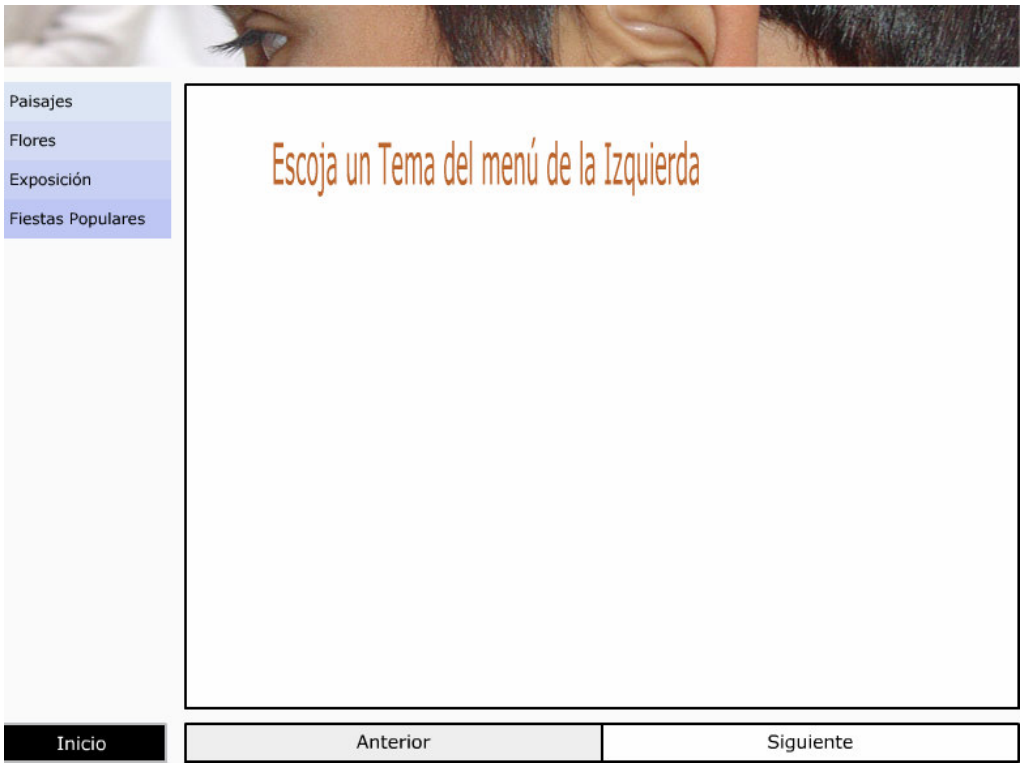

Figura 2: Galería donde se cargan los temas y las imágenes

Anexo3

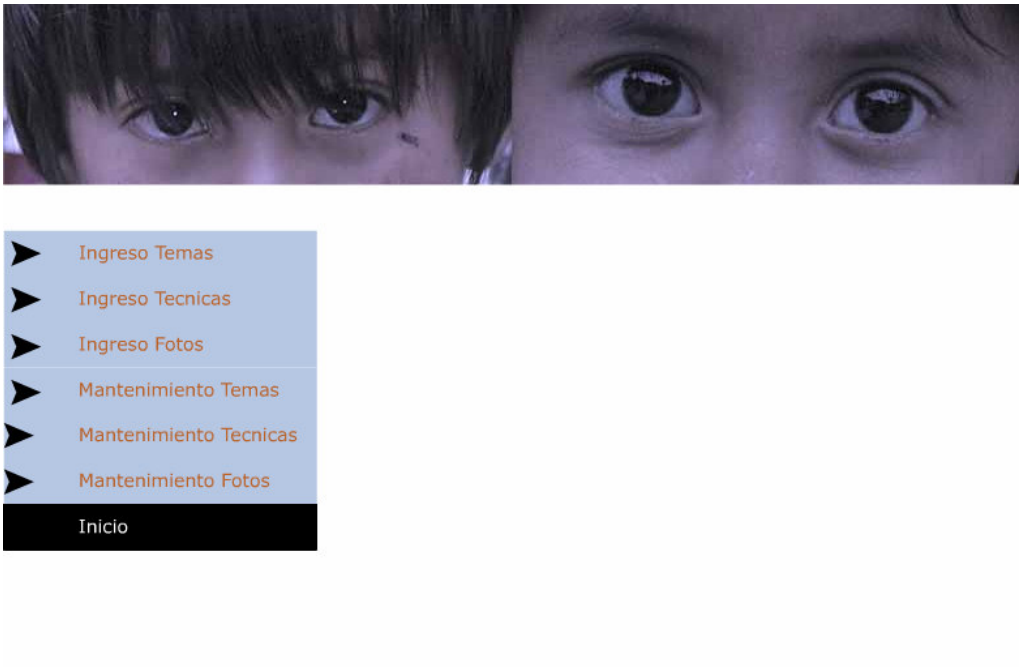

Figura 3: Página inicial del administrador

Anexo 4

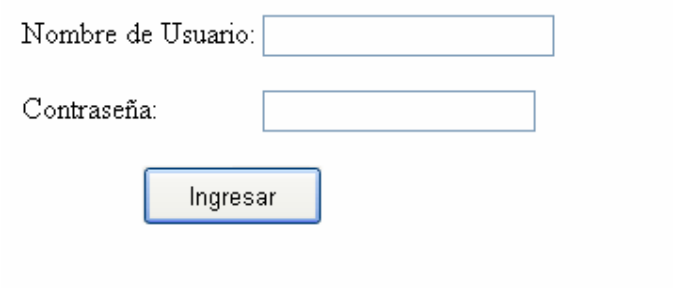

Figura 4: Pagina de Autenticación