Escola Universitària
Politècnica de Mataró

Ingeniería Técnica en Informática de Gestión

APLICACIONES WEB XML/XSLT

SERGIO LÓPEZ GASCÓN
ENRIC SESA NOGUERAS

OTOÑO 2008

RESUMEN

Este proyecto consiste en la programación de una aplicación web, basada en el modelo XML/XSL, para la publicación de críticas culturales en Internet. La plataforma de desarrollo será .NET y se encargará de servir información en XML.

Para la transformación de esta información en HTML para web se usarán plantillas XSLT, capaces de presentar diferentes vistas de presentación dependiendo de la información que llegue.

RESUM

Aquest projecte consisteix en la programació d'una aplicación pràctica web, basada en el model XML/XSL, per a la publicació de critiques culturals en internet. La plataforma de desenvolupament será .NET y s'encarregarà de servir información en format XML.

Per a la transformació d'aquesta informació en HTML per a web s'utilitzaran plantilles XSLT, capaces de presentar diferents vistes de presentació depenent de la informació que arribi

ABSTRACT

This project deals of programming a web application based on the model XML / XSL for the publication of cultural criticism on the Internet. The development platform will be .NET, and will provide information on XML.

For processing this information in HTML for web templates are used XSLT, capable of presenting different presentation of views, depending on the information that arrives.

ÍNDICE

1. INTRODUCCIÓN

Actualmente, la necesidad de publicar y leer información en Internet es innegable. Hoy en día, y más en un futuro, Internet es uno de los más grandes medios de comunicación que disponemos. Y es un medio cada vez más multidispositivo; no sólo podemos acceder a la web mediante el ordenador a través de los navegadores, podemos conectarnos también mediante el móvil, la pda, la televisión, una impresora, un tablet pc…

Pero no sólo es un medio multidispositivo. También es un medio que todo el mundo ha de poder usar, posibilitando en buena medida el acceso a personas con discapacidad.

Que la información llegue correctamente al usuario, independientemente del dispositivo desde el cual acceda o la discapacidad del usuario que la consulte, es una de las responsabilidades de los desarrolladores de aplicaciones web.

Para ello han de tener en cuenta varias especificaciones técnicas y requisitos fundamentales que ya están definidos. En definitiva, han de usar lo que en el entorno web llamamos estándares. Han de pensar una web para todos y para cualquier tipo de dispositivo.

La *W3C* (Consorcio World Wide Web), consorcio por excelencia dedicado a producir consenso en relación a las tecnologías Web, ya define los criterios y recomendaciones a seguir para hacer una web estándar.

Existen profesionales de usabilidad y accesibilidad web que no tienen porqué conocer como está hecha una aplicación web al 100% en todas sus capas para gestionar la información que necesita el usuario. Únicamente centrándose en el desarrollo de la capa más cercana al usuario, la capa de presentación, pueden distribuir los contenidos de la mejor manera posible.

Hoy en día el mejor sistema para cubrir esta necesidad es una aplicación web basada en una estructura de presentación XML/XSL, donde los programadores o desarrolladores de software

generan XML bajo Java, .NET, C++, etc, y los diseñadores de información gestionan la información encapsulada en XML mediante plantillas XSL.

Se propone este proyecto viendo la necesidad que hay actualmente. Muchas empresas o instituciones públicas necesitan dar salida a su información en Internet, cumpliendo como se ha dicho anteriormente, con los estándares web.

Estos organismos casi siempre tienen toda la información que publican de manera estática, mezclando en un mismo documento la información, con la presentación y la semántica de la web. Necesitan de aplicaciones web capaces de resolver estos problemas, ya que por ley, están obligados a dar un servicio cumpliendo con los estándares.

En Internet cada vez más se almacenan datos en XML (Extended Markup Language). Un lenguaje usado como medio de codificación y gestión de contenidos e información para entornos web.

XML se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar también en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Según la especificación de la *W3C,* XML proporciona unas ventajas inmejorables para trabajar con información:

• Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

• El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

• Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones. XML a pesar de sus ventajas, no deja de ser una “hoja con datos”.

XSLT (XML Stylesheets Language for Transformation) es la tecnología que usaremos, ya que es ideal por su respeto a la estandarización de la web y su única finalidad es trabajar con XML para la propagación de su contenido en Internet u otras plataformas.

Según las recomendaciones de la W3C: “XSLT o Transformaciones XSL presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

Las hojas de estilo (aunque el termino de hojas de estilo no se aplica sobre la función directa del XSLT) XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla: unidas al documento fuente a transformar, esas reglas de plantilla alimentan a un procesador de XSLT, el cual realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una página web, directamente en el monitor de un usuario”.

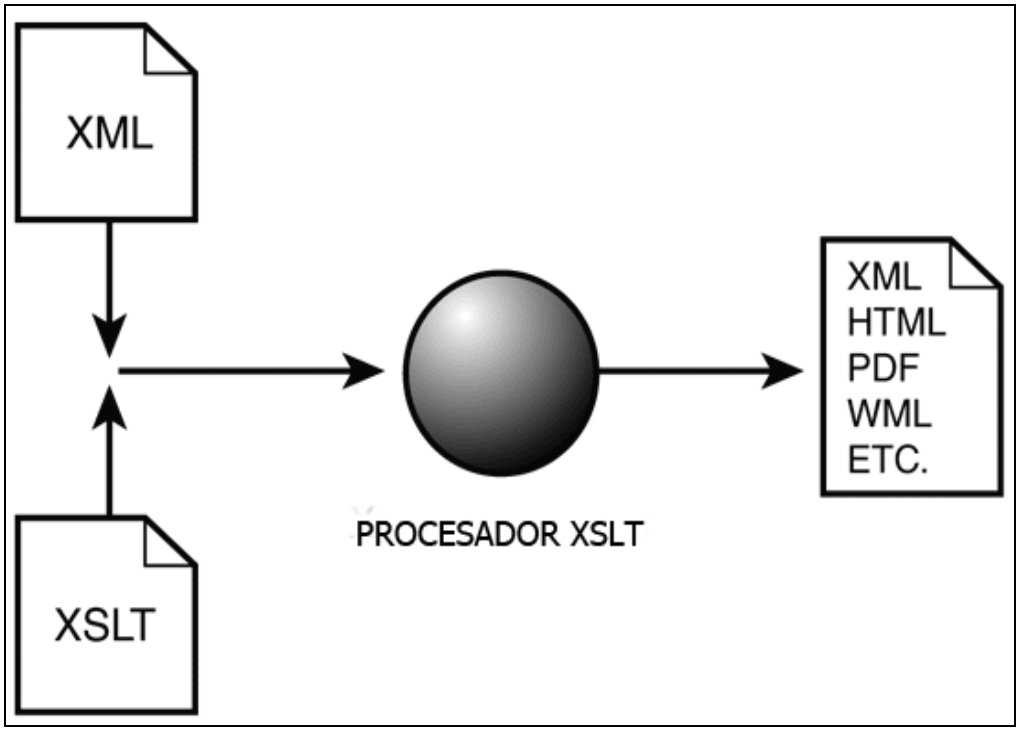

Fig 1. Esquema XSLT

2. OBJETIVO

El objetivo principal de este proyecto es estudiar el modelo de aplicaciones web basadas en la estructura XML/XSL y comprobar las ventajas y desventajas de este tipo de aplicaciones.

Como ejemplo se desarrollará una aplicación web basada en el modelo XML/XSL, para la publicación de críticas culturales en Internet, bajo un portal administrado por los mismos usuarios, donde serán ellos mismos los que compartan valoraciones e información de productos culturales, acercando la aplicación al mundo web 2.0. La temática cultural estará compuesta por cine, literatura y música.

La base del proyecto se realizará sobre tecnología .NET. Tecnología muy común actualmente y que usan la mayoría de empresas para desarrollar sus portales. Se estudiará también las diferentes posibilidades que nos ofrece .NET para realizar una aplicación XML/XSL y cuál es la mejor manera de combinar plantillas XSL sobre .NET para hacer portales webs lo más estándares y accesibles posible.

3. VENTAJAS Y DESVENTAJAS DE LAS APLICACIONES WEB XML/XSL

Para realizar una aplicación web, lo más importante es escoger bien la tecnología que nos asegure que la presentación de los datos será compatible con las normas que rigen actualmente las webs estándar, facilitando al usuario acceder a la información independientemente del medio por el cual acceda o la discapacidad del mismo.
La capa de presentación ha de facilitar un código HTML limpio y estable, y muchas aplicaciones no logran separar eficientemente la lógica del negocio de la generación de la presentación, dando como resultado una aplicación aparentemente bien hecha, pero inaccesible para muchos usuarios.

Las aplicaciones web basadas en el sistema XML/XSL facilitan crear portales webs estándar y accesibles ya que permite separar totalmente la lógica de negocio de la generación de la presentación.

El funcionamiento básico de estas aplicaciones consiste en que la aplicación genera la información que necesita el usuario en formato XML, y mediante una serie de transformaciones con hojas de estilo XSL, se obtiene el código HTML resultante que será lo que verá el usuario final. Lo más común en este tipo de transformaciones es realizar la transformación bajo el estándar HTML ya que un código HTML bien formado ya es multiplataforma de por sí, pero es posible generar otras salidas como WML, PDF o XML a partir de XML dependiendo de las estructuras XSLT que entren en juego y las necesidades del usuario.

Cabe decir que ninguna transformación que le hagamos al documento XML afectará al sistema de información ni afectará a la integridad de los datos o de la lógica de aplicación. Esta separación también nos ayuda en la detección de errores, ya que es muy sencillo separar los errores producidos al mostrar los datos, de los producidos al generarlos. Usando las plantillas adecuadamente, podemos reutilizar partes de la interfaz y aumentar la productividad y la coherencia visual de la interfaz.

El lenguaje utilizado para la generación de la interfaz, XSL, es una recomendación del World Wide Web Consortium (W3C) independiente de plataforma, fabricante y lenguaje de programación. Por ello tiene múltiples aplicaciones que no sólo se limitan a las aplicaciones web en Java o .NET. También implica que podemos encontrar implementaciones de diferentes fabricantes y escoger la que más nos convenga según nuestros requerimientos de precio, soporte, rendimiento... Al mismo tiempo significa que es un lenguaje maduro, muy documentado y probado, haciendo más fácil formar a la gente, o encontrar a la gente ya formada, que con un lenguaje propietario o minoritario.

La separación en dos etapas facilita enormemente la creación de aplicaciones que deben ser accedidas desde múltiples dispositivos. Partiendo del mismo documento XML se pueden generar diferentes interfaces simplemente mediante la aplicación de una XSL diferente. Lo mismo se puede decir a la hora de reutilizar una aplicación con interfaces diferentes por fabricante, idioma, navegador que accede...

La separación en la metodología entre las diferentes fases del desarrollo facilita el trabajo en equipo y la asignación de tareas a personas diferentes, facilitando el encontrar y formar al personal necesario para ayudar en el desarrollo de aplicaciones web.
Por el contrario, existen también ciertas desventajas a tener en cuenta antes de empezar a trabajar con este sistema:

- Problemas de rendimiento debidos al procesamiento con XSL. Dependiendo del compilador para realizar el procesamiento XML/XSL, la cantidad de información que contenga el XML y de la lógica utilizada en el XSL, puede tardar más o menos en obtener la salida esperada. Es conveniente optimizar al máximo la hoja XSL para aumentar el tiempo del proceso, aunque por líneas generales retardo no es excesivamente largo para el usuario.

- Hay que tener en cuenta que una hoja XSL mal escrita puede introducir un retardo considerable o no mostrarse la información, por lo que no conviene descuidar este factor.
- La creación de interfaces con hojas XSL exige unos conocimientos diferentes a la simple edición de páginas HTML. Por otro lado, el cambio de mentalidad que implica XSL, ya que no es HTML pero tampoco es un lenguaje de programación como tal, dificultan el aprendizaje de estas técnicas.

4. TECNOLOGÍA UTILIZADA

Antes de entrar al desarrollo de la aplicación, definiremos la tecnología que usaremos para familiarizarnos antes con ella. Este apartado es importante para el proyecto para saber de que estamos hablando en cada proceso del desarrollo de la aplicación.

Únicamente explicaremos de la tecnología a utilizar lo que realmente haga falta para realizar el proyecto, sin extendernos en esquemas y definiciones que no aporten nada.

Estudiaremos el lenguaje de datos XML y el de transformaciones XSLT, todo siguiendo las definiciones de la *W3C.*

4.1 XML

4.1.1 Introducción

XML, sigla en inglés de *Extensible Markup Language* («lenguaje de marcas extensible»), es un metalenguaje de etiquetas desarrollado por el denominado "*World Wide Web* Consortium".

La base del XML es la introducción de datos estructurados en un fichero de texto mediante una serie de reglas, creando archivos fácilmente generados y leídos por un ordenador, cuya principal utilización consiste en permitir compartir estos datos a todos los niveles, por todas las aplicaciones y soportes.

XML es una tecnología potenciada por un Internet creciente, ya que existen muchos sistemas distintos que tienen que comunicarse entre sí.

De la misma forma que el lenguaje HTML, el XML utiliza "tags" -etiquetas- y atributos para su definición, pero HTML combina la interpretación de los datos con la presentación de estos ,

mientras que el lenguaje XML usa las etiquetas sólo para delimitar datos, y deja su interpretación, a la aplicación que los lee.

4.1.1 Sintaxis

XML se escribe en un documento de texto ASCII, igual que el HTML la cabecera del texto se inicializa con el siguiente comando:

```
<?xml version="1.0"?>
```

En el resto del documento se deben escribir etiquetas como las de HTML.

```
<ETIQ1>...<ETIQ2>...</ETIQ2>...</ETIQ1>
```

Cualquier etiqueta puede tener atributos.

```
<ETIQ atributo1="valor1" atributo2="valor2"...>
```

Los comentarios de XML se escriben igual que los de HTML.

```
<!-- Comentario -->
```

Para definir qué etiquetas y atributos debemos utilizar al escribir en XML tenemos que fijarnos en la manera de guardar la información de una forma estructurada y ordenada.

Reglas básicas del XML:

1 elemento root: Todo el documento XML debe estar contenido en un único elemento, al que llamaremos 'root'.

Elementos anidados: Todos los elementos deben contener elementos completos, es decir, deben contener tanto la etiqueta de apertura como la de cierre. Por tanto se debe respetar el orden inverso al de apertura a la hora de cerrarlos.

{ <elemento1>…<elemento2>...</elemento2>…</elemento1> }

Atributos entre comillas: Todos los atributos de los elementos deben tener su valor entre comillas. Pudiéndose elegir entre comillas simples o dobles.

{ <elemento atributo="valor"> = <elemento atributo='valor'> }

Mayúsculas y minúsculas: Las etiquetas en XML, a diferencia con el HTML, diferencian entre mayúsculas y minúsculas.

Cerrar elementos: Todos los elementos deben cerrarse con la etiqueta correspondiente.

{ <elemento>...</elemento> }

Elementos vacíos: Los elementos que no contengan nada pueden abrir y cerrarse con una única etiqueta

{ <elemento/> }

A continuación se muestra como ejemplo un fichero XML, resultante del proceso de generación de XML de la aplicación web creada para el proyecto.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
   <detallProducte>
    <tipus>pelicula</tipus>
    <Id_producte>2</Id_producte>
    <titol>El increíble Hulk</titol>
    <resumen>En 'El increíble Hulk', el científico Bruce Banner (Edward Norton) recorre el mundo en busca de un antídoto que le permita librarse de su Alter Ego. Perseguido por el ejército y por su propia rabia interna, es incapaz de sacar de su cabeza a Betty Ross (Liv Tyler).</resumen>
    <descripcion>En 'El increíble Hulk', el científico Bruce Banner (Edward Norton) recorre el mundo en busca de un antídoto que le permita librarse de su Alter Ego. Perseguido por el ejército y por su propia rabia interna, es incapaz de sacar de su cabeza a Betty Ross (Liv Tyler).</descripcion>
    <director>Louis Leterrier</director>
    <ano>19/11/2008</ano>
    <genero>Fantástico</genero>
    <img>hulk_grande</img>
    <valoracionMedia>5,5</valoracionMedia>
    <critica>
      <element>
        <usuario>Sergio</usuario>
        <texto>Me parecio una buena peli,espero q hagan la tercera parte!!</texto>
        <fecha>26/11/2008 18:52:38</fecha>
        <valoracion>7</valoracion>
      </element>
      <element>
        <usuario>Isabel</usuario>
        <texto>Esta peli, como todas segundas partes queda sosa de guión y le sobran efectos especiales sin comentar la absurda pelea final.</texto>
        <fecha>14/12/2008 0:15:37</fecha>
        <valoracion>4</valoracion>
      </element>
    </critica>
  </detallProducte>
</Document>
```

4.2 XSLT

4.2.1 Introducción

El XSLT forma parte de la familia de lenguajes del XSL (siglas de *Extensible Stylesheet Language),* basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio.

Desde 1997 varias empresas informáticas como Arbortext, Microsoft e Inso se pusieron a trabajar en una propuesta de XSL (antes llamado "xml-style") que presentaron al W3C y cuyo fin era permitir modificar el aspecto de un documento. Con las hojas de estilo ya se podían lograr algunas mejoras en el aspecto del documento, pero XSL permite otras muchas aplicaciones como múltiples columnas, orden de visualización de los datos de una tabla, múltiples tipos de letra con amplia variedad en los tamaños, etc.

La familia de lenguajes XSL está compuesta por tres tipos:

- XSLT que permite convertir documentos XML de una sintaxis a otra (por ejemplo, de un XML a otro o a un documento HTML).
- XSL-FO (lenguaje de hojas extensibles de formateo de objetos), que permite especificar el formato visual con el cual se quiere presentar un documento XML.
- XPath, o XML Path Language, es una sintaxis (no basada en XML) para acceder o referirse a porciones de un documento XML.

Las hojas de estilo separan la información (almacenada en un documento XML) de su presentación, usando en cada caso las transformaciones que sean necesarias para que el contenido aparezca de la forma más adecuada en el cliente. Se pueden usar diferentes hojas de estilo, o incluso la misma, para presentar la información de diferentes maneras dependiendo de la información o de la composición de los nodos del XML de entrada.

Un hoja XSL puede actuar por si sola como único elemento de transformación y encargarse de dar formato a todo el sitio web, o adjuntarse a una estructura de hojas XSLT que pueden combinarse entre ellas y encargándose de tratar diferentes secciones del portal, siendo referenciadas mediante elementos xsl:include y xsl:import.

Una estructura de hojas XSLT ha de tener una hoja de estilo principal *(principal stylesheet module*) que inicialice la transformación, y a su vez, incorpore el resto de plantillas XSL directa o indirectamente.

Se conoce como incorporación directa cuando la misma hoja de estilos o plantilla llama a otra mediante xsl:include o xsl:import sin que de por medio intervenga ninguna otra plantilla, y se conoce como incorporación indirecta cuando una plantilla puede llamar a otra plantilla que aunque no ha sido incluida previamente mediante xsl:include o xsl:import, si que lo ha sido a través de una plantilla que sí ha sido incluida. Esto resulta útil ya que XSLT no nos permite incluir más de 1 vez la misma plantilla en una estructura XSLT, y con una buena distribución podemos trabajar con todas las plantillas.

Pongamos como ejemplo la plantilla XSL “A”, la plantilla XSL “B”, la plantilla XSL “C”.

En una incorporación directa :

A incluye a B → A conoce a B

A no incluye a C, B no incluye a C → A no conoce a C

En una incorporación indirecta:

A incluye a B → A conoce a B

A no incluye a C, B incluye a C → A conoce a C, a través de B.

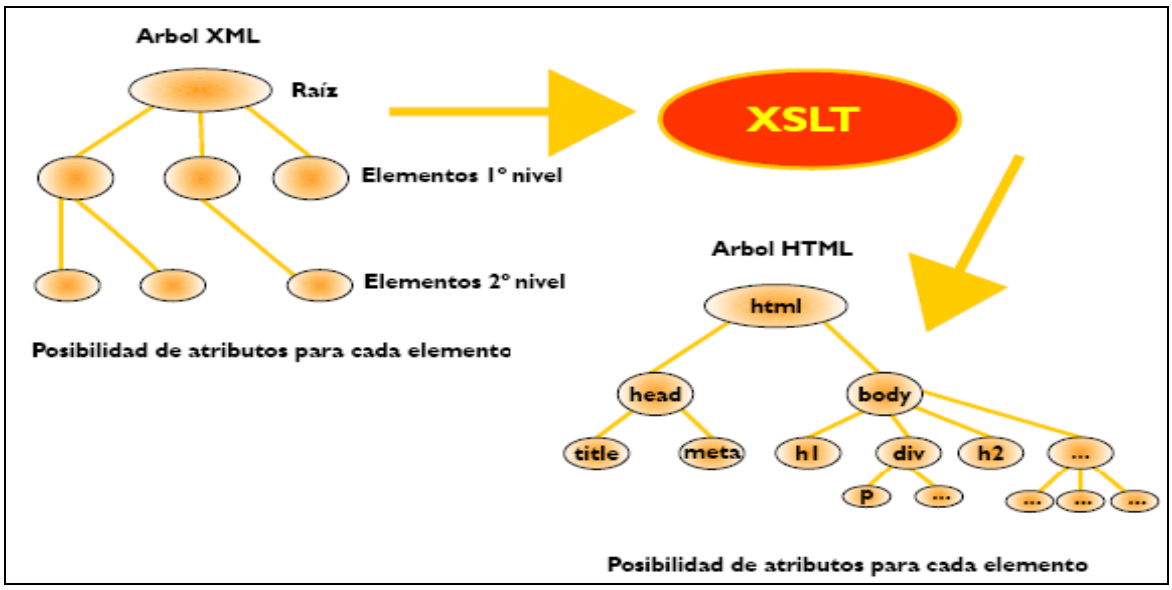

Fig 2. Introducción XSLT

Una aplicación concreta de XLT es, por ejemplo, su utilización para diseñar un portal. Al poder separar el contenido (XML) de la presentación (XLT) es posible utilizar un mismo entramado o esqueleto para diseñar un portal en XML y cambiar la apariencia gráfica utilizando una plantilla u hoja de estilo mediante XLT.

Actualmente hay varias versiones del estándar XSLT: la versión 1.0, que es la que implementan la mayoría de los procesadores, y se denomina "recomendación", es decir, para el consorcio W3, lo equivalente a un estándar, y la versión 2.0 , que, a fecha de hoy, aún se está acabando de desarrollar y validar como estándar.

Algunos procesadores, tales como el Saxon, implementan ya esta última versión. Hay algunas diferencias importantes: el tratamiento uniforme de los árboles, uso de múltiples documentos de salida, y funciones definidas por el usuario que se pueden definir en XSLT, y no sólo en Java u otro lenguaje, como sucedía en estándares anteriores.

4.2.2 Sintaxis

A grandes rasgos se verá la sintaxis de un documento XSLT, ya que en los anexos se describirán todos los métodos y funciones que disponemos a la hora de desarrollar XSLT.

Como se ha visto anteriormente, un documento XML se compone de información estructurada en nodos en forma de árbol. Si se tiene en cuenta que un XSLT ha de recorrer un documento XML entenderemos que la sintaxis básica para el desarrollo de XSLT, son recorridos, bucles, y condicionales que naveguen por la estructura del XML.

La cabecera para todos los documentos será la misma:

```
<xsl:stylesheet version="1.0"
                xmlns="http://www.w3.org/1999/xhtml”
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0"
            encoding="utf-8"
            media-type="text/html"
            doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
            doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
                           indent="yes"/>
```

Iniciamos una hoja de estilos xsl mediante la etiqueta `stylesheet` para todos los documentos que hagamos. Indicamos la versión 1 de la hoja de estilo con el atributo `version` y además iniciamos que seguimos las recomendaciones de la W3C y seguimos su *namespace* (criterio y restricciones de la recomendación sobre la estructura de tipos de elemento y nombres de atrubuto), con `xmlns` y `xmlns:xsl`.

Hemos de indicar también la salida que resultará de la transformación con la etiqueta `output`. En la salida hemos de indicar el método de entrada con el atributo `method`. También hemos de marcar la versión del xslt y la codificación de caracteres que seguirá la plantilla mediante

version y **encoding** respectivamente. Para finalizar la cabecera le decimos el medio de salida después de la transformación con **media-type** y la validación que seguirá con **doctype-public**.

La siguiente etiqueta también es obligatoria. Aquí se indica el nombre del **template** que usaremos. Podemos hacerlo marcándolo con **match** y la secuencia de nodos a la que actuará, o bien dotándola de un nombre por el que luego lo llamaremos si es que nos hace falta con el atributo **name**.

```
<xsl:template match="XMLData/nodes">
<xsl:template name="nodes">
```

Estas últimas instrucciones las usaremos para cerrar el marcado XSLT, también es obligatorio ya que si no nos daría un error al parsear.

```
</xsl:template>
</xsl:stylesheet>
```

4.3 ASP.NET

ASP.NET es una plataforma para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la version 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Uno de los puntos fuertes de esta plataforma es el uso de su modelo de programación *code-behind*. El modelo *code-behind* de ASP.NET alienta a los desarrolladores a construir aplicaciones con la idea de presentación y contenido separados en mente. En teoría, esto permite a un diseñador web, por ejemplo, enfocarse en la creación del diseño con menos posibilidades de alterar el código de programación mientras lo hace. Idea muy cercana a estándares web que pautan las formas de marcar el código HTML, separando contenido, semántica y presentación de la web.

La capa de presentación será el resultado de la transformación de XSL y XML, por tanto no se utilizarán las herramientas visuales de presentación que nos ofrece la plataforma.

5. INTERACCIÓN XML / XSLT

5.1 Procesadores

Hay muchas formas de usar las hojas de estilo. Lo más normal es usarlas dentro de un entorno de publicación.

Un entorno de publicación de XML permite mantener sitios completos basados en XML, y generar páginas en diferentes formatos a partir de ellos.

La mayor utilización de la tecnología XML es para comunicación entre diferentes aplicaciones, de esta forma lo que se necesita en estos casos es aplicar hojas de estilo dentro de una aplicación, o usarlas desde línea de comandos a partir de otra aplicación o otro lenguaje de programación. En ese caso, lo más útil es usar un procesador de hojas de estilo, que habitualmente se encuentra en conjunción con un parser XML, con o sin validación.

De estos, los más usados son los siguientes:

- Xalan. Es una herramienta escrita en Java, y lo único que se necesita para hacerla funcionar es una máquina virtual Java (JVM), tal como la de Microsoft, Sun o IBM. Xalan necesita un parser XML para funcionar.

- Saxon: un procesador bastante rápido, también escrito en Java, con su propio parser incluido, aunque se puede usar uno externo.

A su vez, los procesadores de hojas de estilo se pueden usar como librerías de clases, o desde otros lenguajes.

En definitiva, la funcionalidad de los procesadores es la de convertir la información contenida en un archivo o objeto XML y uno XSL y transformarla en una aplicación concreta ya sea una página Web HTML o un documento estructurado XML de almacenamiento de datos.

En el siguiente esquema queda reflejada la funcionalidad de los procesadores:

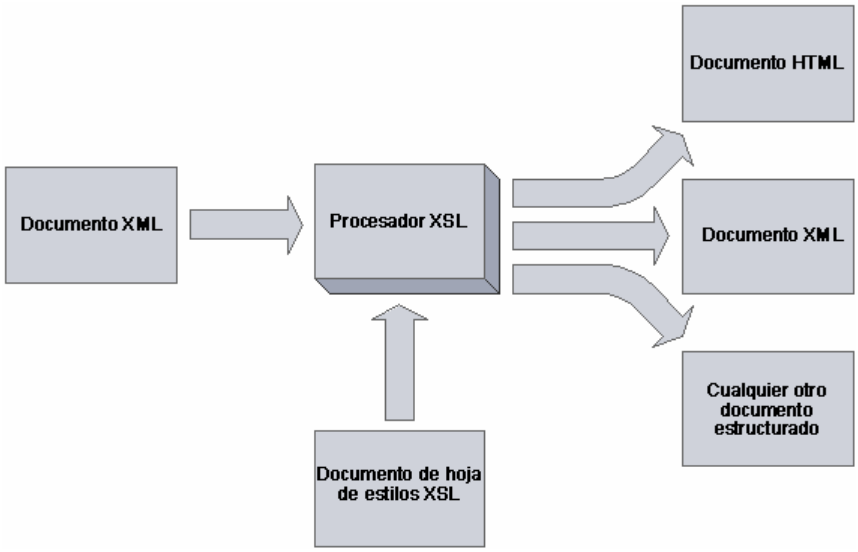

Fig.3. Esquema procesadores XSL

5.2 Ejemplos

El procesamiento de un XML se basa, como se ha informado anteriormente, por una parte en la información contenida en el XML y por otra en la presentación de esta en el XSLT.

XML	XSLT
<?xmlversion="1.0"?> <PROYECTO> <FICHA> <ALUMNO> Sergio López </ALUMNO> <NOMBRE>Proyecto XSLT</NOMBRE> <NOTA>10</NOTA> </FICHA> </PROYECTO>	<?xml version="1.0"?> <xsl:stylesheet version = “1.0”> <xsl: templeate match =’/’> <html><head><title></title></head> <body> **<xsl:valueof** **select=”proyecto/ficha/nombre”/>** </body> </xsl:templeate> </xsl:stylesheet>

Tal como se muestra en el ejemplo, el archivo XML tan solo contiene información útil, ordenada de una forma estructurada en árbol, donde el nodo “proyecto” es el nodo padre, siendo el nodo “ficha” su nodo hijo y el resto de nodos “alumno”, “nombre” y “nota” hijos del nodo “ficha”.

A continuación se muestra un ejemplo más completo extraído de la aplicación. En este caso se trata la problemática de transformar a HTML el detalle de un producto. La información XML es la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <detallProducte>
    <tipus>llibre</tipus>
    <Id_producte>4</Id_producte>
    <titol>Los Pilares de la Tierra</titol>
    <resumen>El gran maestro de la narrativa y el suspense nos transporta a la Edad Media, a un fascinante mundo de reyes, damas, caballeros, pugnas feudales, castillos y ciudades amuralladas.</resumen>
    <descripcion>El gran maestro de la narrativa y el suspense nos transporta a la Edad Media, a un fascinante mundo de reyes, damas, caballeros, pugnas feudales, castillos y ciudades amuralladas. El amor y la muerte se entrecruzan vibrantemente en este magistral tapiz cuyo centro es la construcción de una catedral gótica. La historia se inicia con el ahorcamiento público de un inocente y finaliza con la humillación de un rey. "Los pilares de la tierra" es la obra maestra de Ken Follett y constituye una excepcional evocación de una época de violentas pasiones.</descripcion>
    <autor>Ken Follet</autor>
    <img>pilares_tierra_gran</img>
    <editorial>Plaza y Janés</editorial>
    <coleccion>Éxitos</coleccion>
    <paginas>1135</paginas>
    <genero>Histórica</genero>
    <valoracionMedia>6,5</valoracionMedia>
  </detallProducte>
</Document>
```

A continuación se muestra la plantilla xsl “detall-producte.xsl” que se encarga de procesar la información que viene dada en el XML.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns="http://www.w3.org/1999/xhtml"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="html" version="1.0" encoding="UTF-8" media-
type="text/html" doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN" doctype-
system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" indent="yes"/>


<xsl:template name="detall.producte" match="Document/detallProducte">
   <xsl:if test="Document/detallProducte/tipus = 'llibre'">
       <div id="caixa-intro" class="cantonades llibre">
         <div class="detall-img"><a
href="img/libros/{Document/detallProducte/img}.jpg"><img width="115" height="173"
src="img/libros/{Document/detallProducte/img}.jpg"
alt="{Document/detallProducte/titol}" /></a></div>
              <div class="detall-info">
                <h2><xsl:value-of select="Document/detallProducte/titol" /></h2>
                <p><xsl:value-of select="Document/detallProducte/resumen" /></p>
                <div class="info-adicional">
                  <p><xsl:value-of select="Document/detallProducte/descripcion"
/></p></div>
                <dl>
                  <dt>Autor:</dt>
                  <dd><xsl:value-of select="Document/detallProducte/autor" /></dd>
                  <dt>Editorial:</dt>
                  <dd><xsl:value-of select="Document/detallProducte/editorial"
/></dd>
                  <dt>Colección:</dt>
                  <dd> <xsl:value-of select="Document/detallProducte/coleccion"
/></dd>
                  <dt>Páginas:</dt>
                  <dd><xsl:value-of select="Document/detallProducte/paginas" /></dd>
                   <dt>Género:</dt>
                  <dd><xsl:value-of select="Document/detallProducte/genero" /></dd>
                  <dt>Valoración Media:</dt>
                    <xsl:choose>
                          <xsl:when test="Document/detallProducte/valoracionMedia !=
'NeuN'">
                          <dd><xsl:value-of
select="Document/detallProducte/valoracionMedia" /></dd>
                       </xsl:when>
                       <xsl:otherwise>
                         <dd>--</dd>
                       </xsl:otherwise>
                    </xsl:choose>

                </dl>
                <p id="mas-info"><a href="#">más info..</a></p>
              </div>
            </div>
    </xsl:if>
</xsl:template>
</xsl:stylesheet>
```

En el archivo XSLT, se estructura un conjunto de datos que no tiene que ver con la información hábil sino en la presentación de ésta. De esta forma, dentro del archivo se definen unos “tags” de HTML, para estructurar los datos en un entorno Web, y se establece la ubicación y las propiedades de la información contenida en el XML.

Realizando el procesamiento de los datos el resultado HTML que ya interpreta el navegador web es el siguiente:

```
<div id="caixa-intro" class="cantonades llibre">
  <div class="detall-img">
    <a href="img/libros/pilares_tierra_gran.jpg">
      <img width="115" height="173" src="img/libros/pilares_tierra_gran.jpg"
alt="Los Pilares de la Tierra"/>
    </a>
  </div>
  <div class="detall-info">
    <h2>Los Pilares de la Tierra</h2>
    <p>El gran maestro de la narrativa y el suspense nos transporta a la Edad Media,
a un fascinante mundo de reyes, damas, caballeros, pugnas feudales, castillos y
ciudades amuralladas.</p>
    <div class="info-adicional">
      <p>El gran maestro de la narrativa y el suspense nos transporta a la Edad
Media, a un fascinante mundo de reyes, damas, caballeros, pugnas feudales, castillos
y ciudades amuralladas. El amor y la muerte se entrecruzan vibrantemente en este
magistral tapiz cuyo centro es la construcción de una catedral gótica. La historia
se inicia con el ahorcamiento público de un inocente y finaliza con la humillación
de un rey. "Los pilares de la tierra" es la obra maestra de Ken Follett y constituye
una excepcional evocación de una época de violentas pasiones.</p>
    </div>
    <dl>
      <dt>Autor:</dt><dd>Ken Follet</dd>
      <dt>Editorial:</dt><dd>Plaza y Janés</dd>
      <dt>Colección:</dt>dd>Éxitos</dd>
      <dt>Páginas:</dt><dd>1135</dd>
      <dt>Género:</dt><dd>Histórica</dd>
      <dt>Valoración Media:</dt><dd>6,5</dd>
    </dl>
    <p id="mas-info">
      <a href="#">más info..</a>
    </p>
  </div>
</div>
</div>
```

Fig. 4. Resultado procesamiento XML – XSL

El resultado de la transformación vendría dado de diversos procesos. En la figura 4 vemos como aplicando *templates* XSLT a un árbol XML podemos obtener un modelo de árbol diferente como resultado, pudiendo generar así un fichero con la información únicamente guardada en el XML resultante.

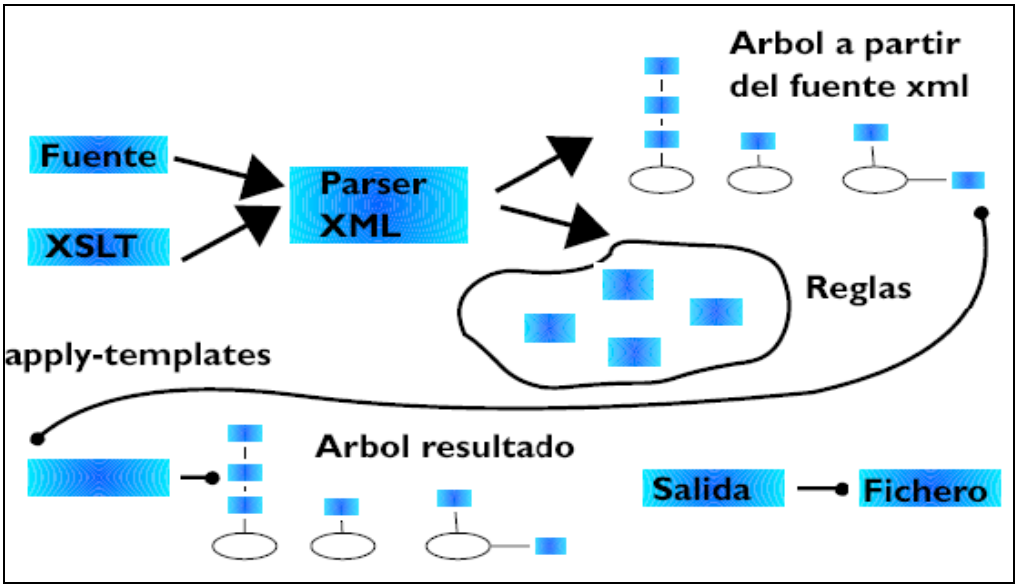

Fig. 5. Transformación XSLT.

Podemos transformar hojas XSL en HTML mediante estos procesadores directamente a través de la consola java, indicándole como parámetros el directorio del compilador, la entrada de información xml, la estructura xsl, y la salida html esperada.

Prácticamente todos los entornos de desarrollo para realizar aplicaciones web ya incorporan de por si sistemas de procesamiento de XSLT como saxon, o sistemas propios. Para la aplicación que se desarrolla en este proyecto el entorno de desarrollo elegido , Visual Studio 2008, ya incorpora estos sistemas de procesamiento de XSLT.

Como se comentaba anteriormente, una de las desventajas es que cualquier etiqueta mal cerrada o mal escrita de XSLT genera un error en el proceso de transformación. Un cúmulo grande de errores puede afectar al rendimiento del servidor en el que se aloja ya que cada petición del usuario inicia un proceso de transformación de XML. Imaginemos un portal con un gran número de usuarios y cada usuario realiza “x” peticiones al servidor, en el que cada petición necesita de una transformación XML/XSL. Si cada una de estas peticiones genera un error a causa de una estructura XSLT mal formada, el servidor puede llegar a caerse al estar ocupado intentando resolver peticiones y generando errores al mismo tiempo.

Para evitar subir estructuras XSLT mal formadas podemos usar servidores previos de pruebas o tratar el procesamiento en nuestra propia máquina local antes de subirlo al servidor.

Un ejemplo de cómo podríamos realizar una transformación en nuestro pc mediante saxon seria a través de la línea de comandos y usando la consola java:

```
java  -jar dir/saxon9.jar -s:source -xsl:stylesheet -o:output
```

dir/saxon9.jar : Directorio del compilador.

-*s:source*: Ruta del archivo xml de información.

-*xsl:stylesheet*: Ruta de la estructura xslt.

-*o:output*: Ruta de archivo html donde se mostrará el resultado de la transformación. Si no se indica se muestra el resultado directamente en la consola como código fuente del html.

El resultado de la transformación nos indicará si existen errores que impidan subir el archivo al servidor.

6. DESARROLLO DE LA APLICACIÓN

6.1. Introducción

La realización de la aplicación se centrará en combinar una estructura XML/XSL para la capa de presentación bajo una plataforma de desarrollo .NET, y crear un portal web basado en una aplicación web XML/XSL.

La temática del portal es cultural, donde los mismos usuarios serán los que compartirán sus valoraciones sobre cine, música y literatura. Actualmente la web se decanta por un sistema llamado red social, donde los contenidos los aportan los mismos usuarios de la web, prescindiendo en gran medida de un administrador del portal que tenga que actualizar la información de la web. La aplicación práctica se acercará más a este modelo de web 2.0 que al tradicional, los usuarios valoraran las aportaciones de otros usuarios.

Las funcionalidades básicas del portal serán:

- Buscar productos culturales.
- Mostrar novedades de productos culturales.
- Valorar productos culturales.

6.2 Requisitos de la aplicación

La aplicación web ha de poder ofrecer al usuario las diferentes funcionalidades:

- Facilitarle la búsqueda de productos culturales.
- Permitirle valorar los productos culturales.
- Ofrecerle las últimas 5 novedades de productos culturales.

Los productos culturales sobre los que trabajará la aplicación serán de tres temáticas diferentes: música, cine, y literatura.

Búsqueda de productos culturales

El usuario ha de poder buscar los productos culturales por separado. Un buscador para productos de cine, otro para productos de música y otros para productos de literatura.

Los productos pueden ser encontrados por medio de su nombre, sin necesidad de que el usuario haya introducido el nombre exacto. Si el usuario no ha introducido el literal del título entero la aplicación ha de poder ofrecerle todos los resultados parecidos al input introducido.

Los productos de literatura también pueden ser encontrados por el nombre de su autor, el año, la colección, la editorial, y el género literario al que pertenece.

Los productos de música pueden ser encontrados a través de otros criterios como el año de publicación del producto, el género musical al que pertenece y el intérprete.

Del mismo modo los productos de cine pueden ser encontrados por el año de publicación, el género al que pertenece, el nombre del director y el nombre de los actores.

El resultado de la búsqueda ha de listar todos los productos encontrados con los criterios de búsqueda introducidos. En todos los productos ha de aparecer como mínimo el título del producto, la valoración media obtenida de las críticas de los usuarios, y una breve descripción del mismo.

A su vez a de permitir al usuario entrar en el detalle de ese producto y visualizar todas las críticas y valoraciones que han hecho los usuarios sobre él.

Valoración de productos culturales

La aplicación ha de permitir al usuario valorar del 1 al 10 los productos culturales que consulta. El usuario puede ser anónimo y el comentario de la crítica también. Lo único obligatorio es la valoración del producto del 1 al 10.

A partir de esta valoración la aplicación ha de calcular la valoración media de todos los usuarios y mostrarla.

Novedades del Portal

Para una mayor sensación de actualización del portal la aplicación ha de poder mostrar en la página principal las 5 últimas novedades de cada temática cultural.

6.3 Estructura de la aplicación

La finalidad de la aplicación es crear un portal web dinámico únicamente a partir de información encapsulada en XML.

Los navegadores webs leen la información mediante código HTML por tanto necesitamos de transformaciones XSLT que nos pasen de un formato XML a HTML.

Tal como se muestra en la figura 6, el esquema de la estructura consta de 2 capas (Cliente , Servidor) conectadas mediante la red de Internet.

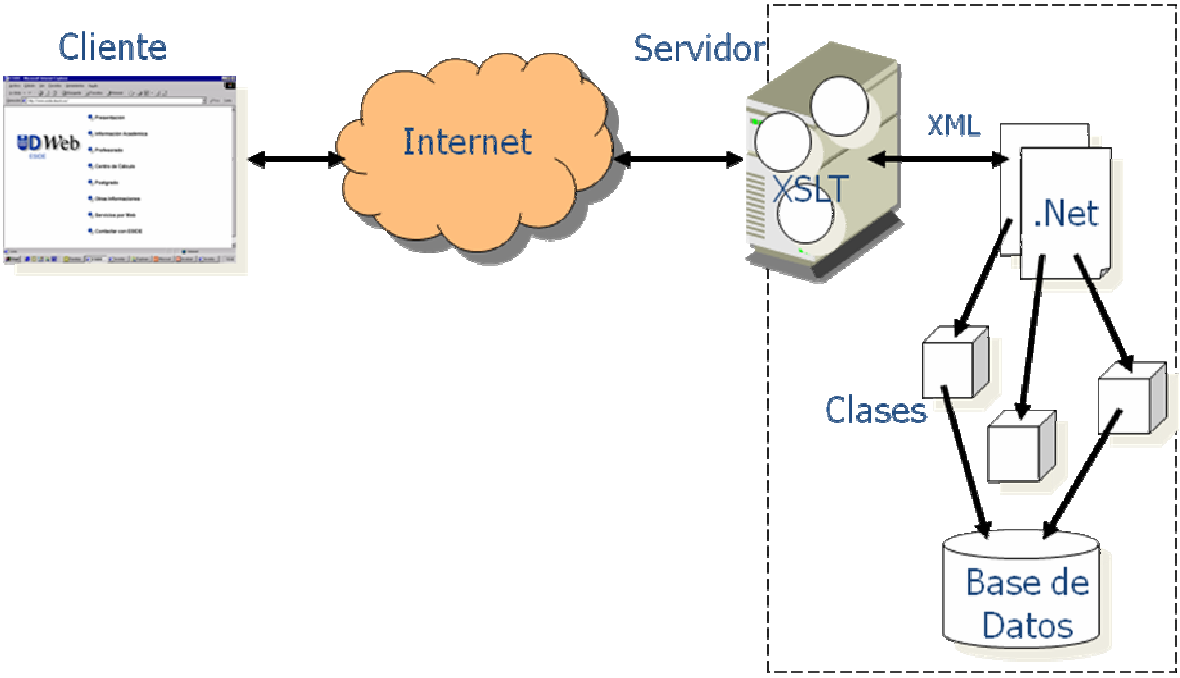

Fig 6. Estructura de la aplicación

En la capa cliente encontramos todos los agentes de usuario capaces de navegar por la red interpretando el código HTML, ya sean navegadores web, PDA, móviles, etc… que solicitan páginas webs a internet.

En la capa de servidor se encuentra la aplicación web XML/XSL bajo la plataforma de desarrollo . NET.

En la capa de persistencia tenemos exclusivamente la base de datos SqlServer y los procedimientos de consulta necesarios. Esta capa se divide por tanto en tablas, vistas, procedimientos, funciones. Usaremos procedimientos para cada consulta o inserción diferente de la Base de Datos.

La capa de negocio se encargará de toda la lógica para generar la información que pida el usuario en formato XML. Compuesta por las clases que hagan falta. Las clases en .NET se dividen mediante el sistema *code-behind* en que cada clase se divide en su parte de diseño y su parte de código. La aplicación solo usará la parte de código ya que la presentación la cargaremos mediante transformaciones a HTML del XML que se genera.

Generaremos cada clase correspondiente a la funcionalidad de la web, como consulta del detalle de un producto, inserción de críticas, página de bienvenida, etc...

6.4. Maqueta HTML

Primero de todo se ha de pensar que es una aplicación web, y como se ha dicho anteriormente, el código ha de estar validado según los estándares web. Así pues, el primer paso a hacer será una maquetación en HTML validado + CSS que posteriormente transformaremos a XSLT.

La finalidad del proyecto será tener una aplicación para la publicación de críticas culturales que vengan dadas en formato XML, así que se diseñará un portal ideado para dar prioridad a la información que venga. Una cabecera con el título (<h1>) del sitio, y dos módulos centrales bastará.

En el módulo o contendor principal se ubicarán las noticias en formato listado (<ul>), y el módulo secundario se usará para un listado de enlaces externos o “banners”.

El módulo central también lo usaremos para el detalle de una noticia si es que el XML así nos lo indica.

Se ha de tener en cuenta que no puede haber etiquetas HTML con código CSS que controle la presentación, ya que se ha de separar semántica (HTML) y presentación (CSS).

Cuando tengamos la estructura del HTML pura, le incluimos la hoja de estilos para darle presentación.

Una vez montado el HTML + CSS pasamos a validarlo. Para validarlo hay varios añadidos a los propios navegadores o a los programas de desarrollo que lo realizan. El más usado en el mundo web es validarlo según la W3C.

En la página http://validator.w3.org/check podemos validar nuestro documento HTML. Aquí nos indicará los errores que existen en el marcado si es que lo hemos hecho mal, o por el

contrario nos informará mediante un mensaje de que el documento validado sigue las recomendaciones de estándares.

6.5 Estructura XSLT

El siguiente paso será realizar una estructura de XSLT. Para eso hemos de analizar que nos ofrece la tecnología XSLT para solucionar nuestro problema.

Una de las ventajas de usar XSLT es la capacidad modular que tiene. Podemos trocear el HTML en pequeños componentes XSLT, que al unificarlos montarán el HTML resultante dependiendo de la información que le llegue.

Así pues dividiremos la estructura XSLT en módulos según las funcionalidades y partes de la web.

Como vemos en la figura 7, el XML genérico es incluido en el archivo index.xsl. Toda la información que llegue a este módulo mediante el XML será extensible también para el resto de módulos, ya que éstos son incluidos en él, y por extensión el XML es visible para ellos.

Se ha pensado una estructura modular ya que es más flexible a la hora de cambiar cosas de la página. Creando nuevos módulos y añadiéndolos a la estructura podemos ampliar la funcionalidad del sitio, según los requisitos de la aplicación.

Estos módulos se pueden incorporar en una carpeta aparte para organizarlos mejor, llamada en el proyecto “componentes”

Todos los archivos externos que no sean información XML, pueden ser llamados por cualquier módulo. En esta aplicación el módulo head.xsl, por ejemplo, llama a los archivos que se harán

cargo del comportamiento y la presentación del portal. Estos archivos con los JS o CSS respectivamente. Son los más comunes en el desarrollo de páginas web.

La estructura XSLT desarrollada también permite la posibilidad de incluir más información XML. Si aparte de la información genérica que reporta el XML, queremos usar la aplicación para otro tipo de web, los XML pueden convivir perfectamente entre ellos. Esta es una de las ventajas de usar esta estructura XSLT

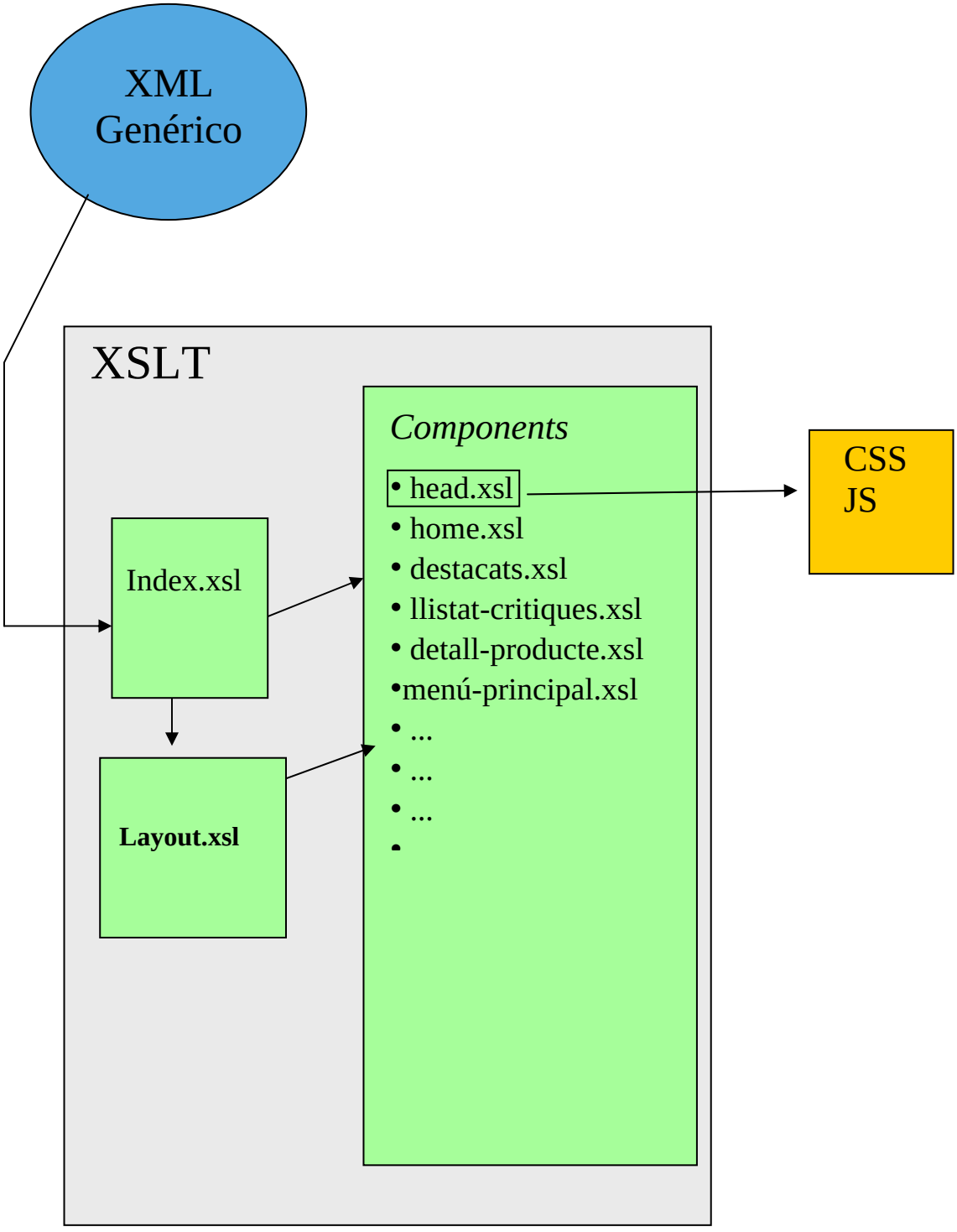

Fig. 7. Estructura XSLT

Correspondencia de módulos xsl y maqueta HTML

1.- menu-principal.xsl

2.- introduccio.xsl

3.- destacats.xsl

4.- titol-web.xsl

5.- detall-producte.xsl

6.- llistat-critiques.xsl

7.- cerca-producte.xsl

8.- resultats-cerca.xsl

9.- copy.xsl

6.6. Modelo BBDD

Para la base de datos se ha realizado un modelo conceptual que consta de 12 tablas, que se centra en una tabla principal de “Producto”.

Los productos culturales pueden ser sólo de tres tipos diferentes, que son “Película”, Disco” y “Libro”.

Cada producto puede tener un número ilimitado de críticas de usuarios, de los que no hace falta un registro de *login* y *password* ya que no es una aplicación web para almacenar la sesión del usuario, ni información indispensable de éste. Además las valoraciones a los productos pueden ser anónimas.

Las críticas se valoran con un número del 1 al 10, entendiendo el número 10 como la valoración más positiva. La valoración media del producto será un campo calculado que no hará falta almacenar.

De cada producto también se almacena el género al cual pertenece y las personas que participan en la creación del producto, ya sean directores, actores, intérpretes o autores.

Una vez validado el modelo y generado el modelo físico, la conversión de los datos hay que hacerla para una base de datos SqlServer, ya que es la que utiliza Microsoft Visual Studio, plataforma con la que se realizará el proyecto. Así pues se genera un script de la base de datos para SqlServer de Microsoft.

Finalizado este proceso, solo queda incorporar el script en la base de datos del proyecto, iniciar el script y obtener el modelo de datos en el proyecto de Microsoft Visual Studio.

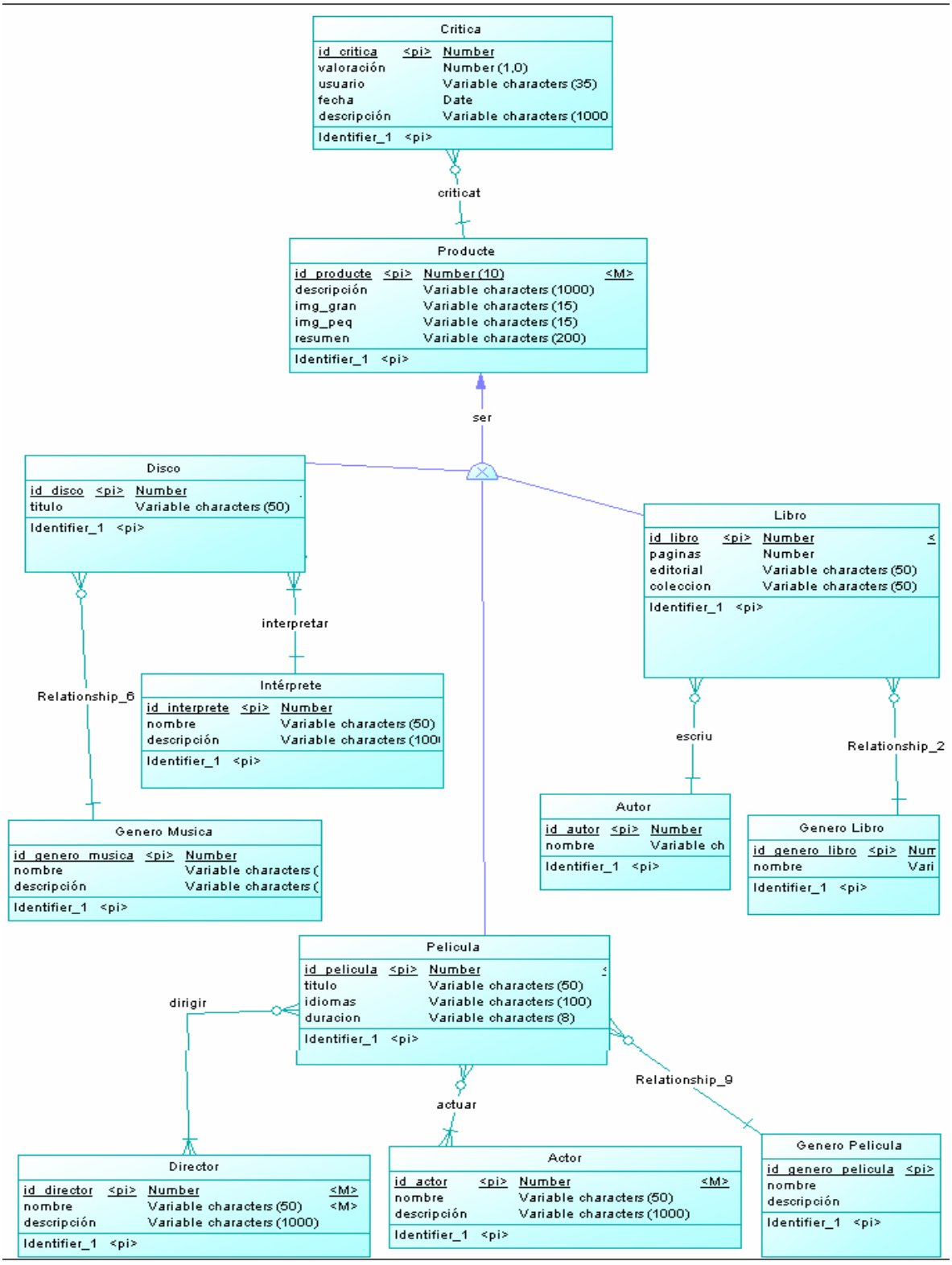

Fig. 9. Modelo conceptual BBDD

6.7. Generación de XML con .NET

Para la generación de XML con la información extraída de la base de datos se ha basado en la idea básica de la concatenación de cadenas de caracteres, Strings, para luego transformarlo como objeto XML para que el parser de XSLT lo procese como tal. .

Buscando información sobre este tipo de tratamientos de XML se ha conseguido transformar una cadena de caracteres a un objeto XML mediante el recurso MSXML2.DOMDocument.

Primero de todo hemos de extraer la información de la base de datos y mantenerla en un contenedor para ir generando el String que luego nos servirá como base del XML.

Un recurso básico para contener información en un objeto es a través de DataSet. Así pues toda la información que se extrae de la base de datos se guarda momentáneamente en un objeto DataSet, que luego recorreremos e iremos formando la cadena de caracteres.

```
Dim StrXMLDestacatsPel As String


StrXMLDestacatsPel = StrXMLDestacatsPel + "<element><Id_producte>" +
CStr(datosBD.Tables(0).Rows(intCont).Item(1)) + "</Id_producte>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<titol>" +
datosBD.Tables(0).Rows(intCont).Item(2) + "</titol>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<img>" +
datosBD.Tables(0).Rows(intCont).Item(7) + "</img>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<text>" +
datosBD.Tables(0).Rows(intCont).Item(3) + "</text>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<info><director>" +
datosBD.Tables(0).Rows(intCont).Item(4) + "</director>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<ano>" +
CStr(datosBD.Tables(0).Rows(intCont).Item(5)) + "</ano>"

StrXMLDestacatsPel = StrXMLDestacatsPel + "<genero>" +
datosBD.Tables(0).Rows(intCont).Item(6) + "</genero></info></element>"
```

La idea, aunque resulte laboriosa por la cantidad de nodos a trabajar, es sencilla y práctica ya que controlamos en todo momento qué información va a parar encapsulada y en que nodos lo hará.

Una vez tenemos un String con toda la información necesaria, pasamos a crear el objeto XML. Podemos usar MSXML2.DOMDocument que ya incorpora Visual Studio para transformar el objeto String a un objeto XML. El proceso es simple y se puede realizar con pocas líneas de código.

```
Dim objXML
Dim strXml As String

objXML = Server.CreateObject("MSXML2.DOMDocument")

strXml = CreacionXMLHome(IODataset)

objXML.loadXml(strXml)
```

Primero creamos el objeto indicando mediante el recurso MSXML2.DOMDocument que será un objeto XML.

Seguidamente creamos el String con el DataSet resultante de la búsqueda en la base de datos de la información.

Finalmente cargamos esta variable String al objetoXML para que la transforme como un objeto XML como tal.

La figura 9 muestra un esquema resumen de cómo la aplicación genera un objeto XML, siguiendo todos los procesos desde que se extrae la información hasta que se convierte en un objeto XML.

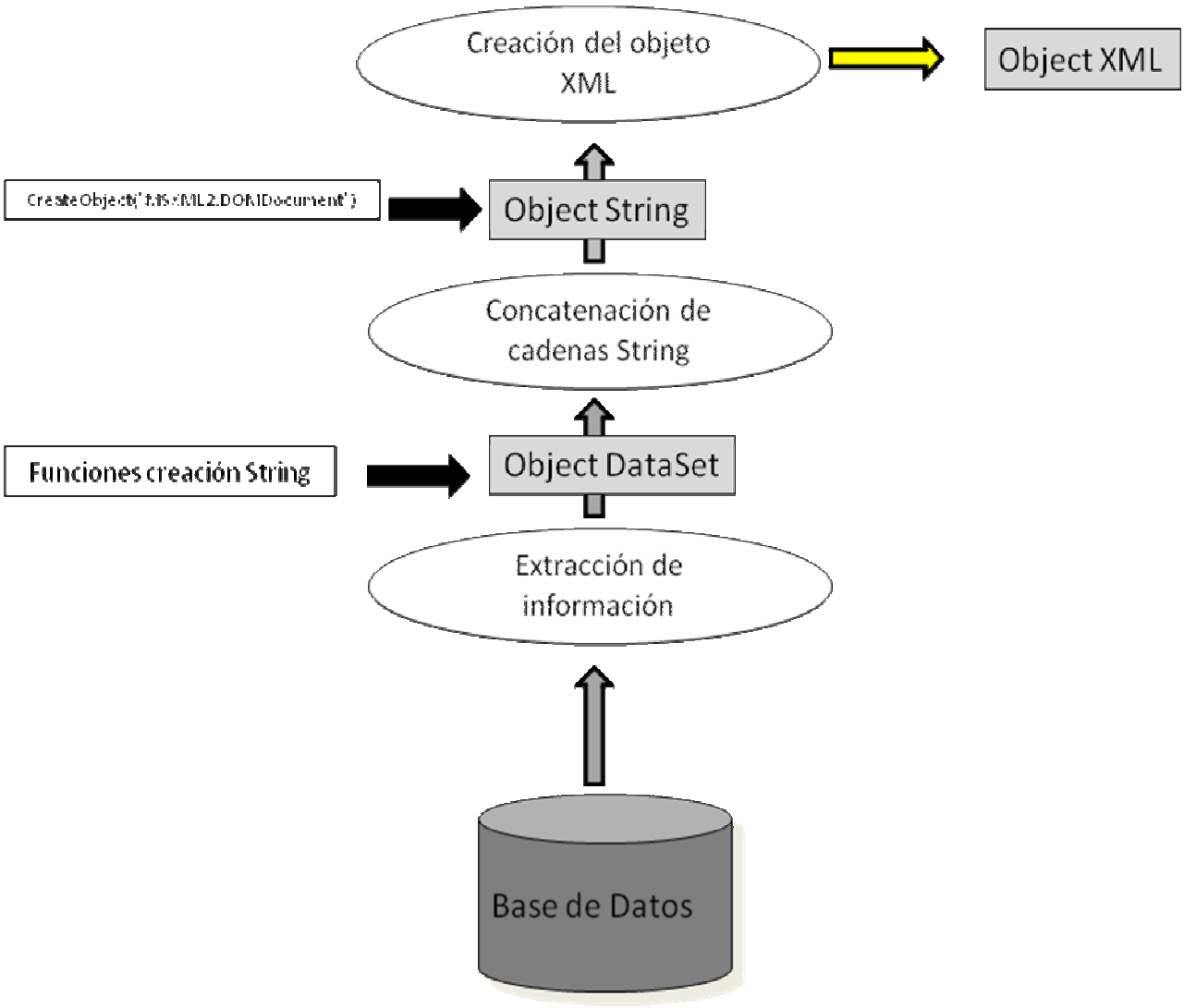

Fig.10 Generación de XML

Otra manera de poder realizar este paso a un objeto XML, es saltarnos el paso de la concatenación de caracteres, y pasar directamente la información del DataSet al objeto final XML.

El inconveniente de hacerlo de esta manera es que para cada nodo hemos de crear un objetoXML diferente, para al final de todo el recorrido del DataSet unificarlos como uno solo, teniendo en cuenta el orden en el que los juntamos los objetos para respetar la estructura de árbol del XML.

Viendo estos inconvenientes se ha preferido la opción de crear un objeto XML a partir de un String y ahorrarse la creación de múltiples nodos que pueden dar más posibilidades de error.

6.8. Procesamiento XSLT y XML con .NET

Una vez tenemos el objeto XML hemos de pasarlo a HTML para que los navegadores web lo entiendan. Para eso hemos de procesar el XML con XSL.

Ya que trabajamos con .NET, podemos usar las funciones de soporte que incorpora el lenguaje para realizar este procesamiento, así no hace falta que instalemos ningún procesador xalan o saxon para que funcione la aplicación, ni indicar en que ruta puede encontrar el compilador de transformación.

Para realizar las pruebas previas al procesamiento y ver que las hojas XSL están bien formadas, se ha instalado en local un procesador saxon, para comprobar que la estructura no tiene errores antes de incorporarla a la aplicación.

Todo el proceso de compilación de las hojas XSL con el XML lo he resumido en una función, que se encarga de devolver el HTML resultante.

```
Function CargarXSL(ByVal XMLDOM2, ByVal strFitxer)

        '------------------------
        'Declaració de variables
        '------------------------
        Dim XSLDOM 'Parser per carregar l'xsl
        Dim strHTMLRes 'HTML resultant del XSL Transform
        '////////////////////////////
        'Instanciem el parser de xml
        '////////////////////////////
        XSLDOM = Server.CreateObject("MSXML2.DOMDocument")

        'carreguem el xsl amb async a false
        XSLDOM.async = False
        XSLDOM.load(strFitxer)

        'Processem l'xml amb l'stylesheet que hem carregat
        strHTMLRes = XMLDOM2.transformNode(XSLDOM)

        '-------------------
        'Alliberar memoria
        '-------------------
        XSLDOM = Nothing

        'retornem l'html en format string
        CargarXSL = strHTMLRes
    End Function
```

Por parámetros pasamos el objeto XML obtenido del proceso de generación del XML y la estructura XSLT creada.

Instanciamos el parser de XML que también hemos usado para generar el XML y cargamos la estructura XSLT en un objeto del tipo MSXML2.DOMDocument.

Sólo queda transformar los nodos en HTML mediante la función *transformNode*

```
strHTMLRes = XMLDOM2.transformNode(XSLDOM)
```

HTML = Objeto XML + Estructura XSLT

Finalmente devolvemos el HTML y el navegador se encarga de interpretarlo.

7. PRESUPUESTO

Para realizar el presupuesto del proyecto se han divido los costes entre los de mano de obra y los materiales.

Para los materiales tendremos en cuenta las licencias de software utilizado. El software es exclusivo para el desarrollo web.

- **Macromedia Dreamweaver CS3**: Software que usaremos para desarrollar tanto el HTML, el XML, y los XSL.
- **Photoshop CS3**: Software que usaremos para el diseño gráfico de la web.

- **Visual Studio 2008:** Software para la programación en .NET.
- **Sybase Power Designer :** Software para la modelización de la base de datos.

Para la adquisición de Macromedia Dreamweaver CS3 lo más económico y rentable es comprarlo directamente en la página web de Adobe. Aquí podemos descargar la última versión por 555,64 €.

De igual forma podemos adquirir Photoshop CS3 por 1.042,84 €.

Visual Studio 2008 cuesta 600€ y obtener la licencia de Sybase Power Designer 1200 €.

Se amortiza el software en 3 años, realizando 5 proyectos al año. Por tanto amortizamos a este proyecto 1/15 parte del coste total del software.

Por tanto el coste total de la amortización del software para este proyecto asciende a un total de 226,56 €

En cuanto a la mano de obra, contabilizaremos las horas dependiendo del rol de profesional usado para la realización del proyecto.

Los roles asumidos son :

- **Maquetador**: Profesional que se encarga de la maquetación HTML de la web cumpliendo con los estándares y el desarrollo de los XSL.
- **Diseñador gráfico**: Profesional que se encarga del diseño web de la página.
- **Programador**: Profesional que se encarga del desarrollo de la programación .Net y el desarrollo de la Base de Datos.
- **Analista**: Profesional que se encarga del análisis de los requisitos de la aplicación y estudia el mejor lenguaje a utilizar

Perfil	**nº Horas**	**Precio Hora**	**Total sin Iva**
Maquetador	55	25 €	1357
Diseñador	20	35 €	700
Programador	115	47 €	5405
Analista	25	55 €	1375
Total	160		8837

En total y sumando el software utilizado más la mano de obra, el coste total del proyecto es de 12.235,48 €:

	Total
Mano de obra	8837
Software	226,56
Total	**9063,56**

8. CONCLUSIÓN

Como se ha comentado en la introducción del proyecto, la necesidad de publicar información en Internet, y que ésta llegue al usuario independientemente de las limitaciones que tenga, es un reto que hoy en día se está consiguiendo.

Al finalizar el proyecto, se puede concluir que se cumplen los objetivos propuestos usando la tecnología XSLT. Se ha conseguido publicar información XML en Internet mediante un formato HTML sin tener que alterar la presentación ni el contenido, cumpliendo con los estándares recomendados por la fundación de la W3C.

La gran ventaja que presenta XSLT es la capacidad de transformar una única fuente de información XML en infinidad de formatos y de adaptar el contenido y la presentación para que se adapte a cualquier medio según la necesidad.

Otra ventaja es que gracias a la utilización de XSLT, permite separar el contenido de la presentación o marcado HTML, pudiéndose modificar aspectos visuales fácilmente sin que los contenidos se vean mezclados en el proceso.

La mayoría de veces el diseño web viene limitado por la plataforma de desarrollo que tiene detrás y las facilidades que ofrece a los desarrolladores cumplir con el diseño gráfico pactado. Usando aplicaciones web XML/XSL se pueden cumplir con estos requisitos ya que la plataforma de desarrollo no tiene porqué dar soporte a la presentación de la aplicación. La estructura XSL se convierte en la maqueta HTML realizada.

La versión de XSLT usada es la primera, ya que es la más estandarizada, aunque tiene varias limitaciones a mejorar. Una de ellas es que no se pueden crear métodos propios, obligándonos a usar exclusivamente los que nos vienen por defecto. Aunque los métodos que se ofrecen son de gran ayuda, siempre hay algún problema que se solucionaría desarrollando métodos propios, como alguna operación matemática o comparación que no se obligue a repetirla siempre que se

quiera usarla. Hay que decir que la evolución de XSLT tiene como unos de los requisitos solventar este inconveniente, y hay procesadores como SAXON que ya implementan la versión 2 de XSLT, dando libertad al programador para que cree sus propias funciones.

XSLT es una tecnología que tiene futuro y que tiene organizaciones fuertes detrás que desarrollan sus aplicaciones con este lenguaje. Muchas instituciones públicas y organismos oficiales como ayuntamientos o empresas privadas, utilizan estas estructuras modulares basadas en XSLT.

Las aplicaciones que sirven XML también pueden ser utilizadas para otros propósitos que no sean servicios web. Podemos compartir información entre plataformas que se comunican a través de información encapsulada en archivos XML. También puede hacerse compatible usar el mismo XML para dar un servicio WAP mediante otra estructura de transformación XSL.

Cabe decir que hoy en día con la misma estructura XSL que transforma a HTML podemos usar el servicio WAP ya que este sistema lo que hace es leer HTML bien formado. Si nuestra estructura XSL transforma a un HTML validado y bien formado ya nos servirá para la lectura de dispositivos móvil.

Gracias también a aplicaciones web XML/XSL podemos separar de forma clara las distintas fases de desarrollo y asignar cada parte a un grupo de trabajo.

Esta forma de separar en fases el desarrollo nos permite simultanear algunas tareas y asignar distintos cometidos a distintos miembros del equipo de desarrollo, sin necesidad de que todos los miembros del equipo sepan de todo.

Así por ejemplo, si implementamos la lógica de negocio en .NET, los programadores encargados de dicha tarea no necesitan saber nada de HTML, ni de aplicaciones web, ni de XSLT. Simplemente deben generar el XML de respuesta a partir de los parámetros de entrada recibidos y del modelo de datos. Al mismo tiempo, el personal encargado de implementar las

hojas XSLT para la transformación no tiene por qué conocer el modelo de datos ni saber programar en .NET, le basta conocer cómo será el XML y el esperado.

9. GLOSARIO DE TÉRMINOS

APLICACIONES WEB XML/XSL: Aplicaciones para internet basadas en servir información en formato XML y mediante una serie de transformaciones XSLT presentan la información adaptada al medio de salida definido. Uno de los medios de salida más comunes es HTML.

ASP.NET: Es una tecnología de scripts que corren en el servidor y pueden ser utilizados para crear aplicaciones dinámicas e interactivas en el Web. Una página ASP.net es una página de HTML que contiene scripts que son procesados por un servidor Web antes de ser enviados al navegador del usuario.

CSS: *Cascading style sheets* **(**hojas de estilo**)** es un formato usado en las páginas web para separar el estilo (la forma en la que se ve una página web) de la estructura (o código), es una característica del Html que da más control a los programadores, diseñadores y usuarios de la web sobre cómo se muestran las páginas web

HTML: (*Hyper Text Mark-up Language* o Lenguaje de Marcas de Hipertexto), e el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

SAXON: Tipo de procesador de hojas XSL escrito en java que permite transformar documentos XML en documentos HTML. Saxon implementa hasta la versión 2.0 de XSLT.

Microsoft Sql Server: Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de

muchos usuarios grandes cantidades de datos de manera simultánea[*cita requerida*], así como de tener unas ventajas que más abajo se describen.

Xalan: Tipo de procesador de hojas XSL escrito en java que permite transformar documentos XML en documentos HTML. Saxon implementa hasta la versión 1.0 de XSLT.

XML: Son las siglas de *Extensible Markup Language*, una especificación/lenguaje de programación desarrollada por el W3C. XML ha sido diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones.

XPath: (*XML Path Language*) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos (*plain text*). XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. XPath fue creado para su uso en el estándar XSLT, en el que se usa para seleccionar y examinar la estructura del documento de entrada de la transformación.

XSL-FO: Un documento XSL-FO es un documento XML en el que se especifica cómo se van a formatear unos datos para presentarlos en pantalla, papel u otros medios. El significado de las siglas XSL-FO es *eXtensible Stylesheet Language Formatting Objects.* Hay que destacar que en el documento XSL-FO figuran tanto los datos como el formato que se les va a aplicar.
La unidad básica de trabajo en un documento XSL-FO es el "Formating Object", unidad básica para presentar (formatear) la información. Estos objetos de formato se refieren a páginas, párrafos, tablas, etc.

XSLT: Son las siglas de *XML Stylesheets Language for Transformation*, un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Las hojas de estilo XSLT - aunque el término de hojas de estilo no

se aplica sobre la función directa del XSLT - realizan la transformación del documento utilizando una o varias reglas de plantilla. Estas reglas de plantilla unidas al documento fuente a transformar alimentan un procesador de XSLT, el que realiza las transformaciones deseadas poniendo el resultado en un archivo de salida, o, como en el caso de una página web, las hace directamente en un dispositivo de presentación tal como el monitor del usuario.

WAP: es un estándar abierto internacional para aplicaciones que utilizan las comunicaciones inalámbricas, por ejemplo acceso a servicios de Internet desde un teléfono móvil.

Se trata de la especificación de un entorno de aplicación y de un conjunto de protocolos de comunicaciones para normalizar el modo en que los dispositivos inalámbricos, se pueden utilizar para acceder a correo electrónico, grupo de noticias y otros.

W3C: Es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (*Uniform Resource Locator*, Localizador Uniforme de Recursos), HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

10. BIBLIOGRAFIA

[1] http://www.microsoft.com/NET *Microsoft NET Framework*, 2008

[2] http://xml.utilitas.org/xslt_mini_como.html Tutorial XSLT, 2002

[3] Jesús Sánchez Allende; Scott Short, *Creación De Servicios Web Xml Para La Plataforma .Net*, McGraw-Hill 2002

[4] Borja Manero; G. Andrew Duthie, *Aprenda Microsoft Asp.net Ya*, McGraw-Hill 2002