INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
SECCIÓN DE GRADUADOS

CLASIFICADOR DE DOCUMENTOS HTML NO ESTRUCTURADOS UTILIZANDO META DATOS

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS CON
ESPECIALIDAD EN COMPUTACIÓN

P R E S E N T A

ING. FELIPE DÁVALOS RODRÍGUEZ

DIRECTOR

M. en C. JOSÉ DE JESÚS EMILIO SOSA IGLESIAS

CODIRECTOR

M. en C. JOSÉ RAFAEL CEN ZUBIETA

MÉXICO D.F. NOVIEMBRE 2002

LISTA DE FIGURAS Y TABLAS

Clasificador de Documentos HTML No Estructurados con Meta Datos

RESUMEN

En este trabajo de tesis se diseña y construye una herramienta de software en plataforma Microsoft Windows, para adicionar Meta Datos a documentos de formato HTML(Hyper Text Markup Language). Se utilizan los estándares internacionales para la inclusión de los nombres de Meta Datos típicos según la propuesta de Dublin-Core. La herramienta adicionalmente permite la construcción de Temáticas jerárquicas o Taxonomías para cualquier dominio de aplicación y la clasificación de documentos remotos accesibles por un URL (Universal Resource Locator), dentro de una temática particular de la jerarquía, siendo posible también, exportar las jerarquías construidas junto con las referencias a los documentos clasificados dentro de ellas, en un formato HTML, para ser publicadas en un servidor Web y ser utilizadas desde los navegadores como sistema de navegación jerárquico a través de la temática correspondiente, permitiendo accesar directamente desde el navegador los documentos clasificados, teniendo esto aplicabilidad directa en la construcción de portales con navegación por temas.

La metodología utilizada para el análisis, diseño y construcción de esta herramienta fué la metodología “Objectory Process de Rational Rose”, con la cual se logró desarrollar la herramienta de una manera sistemática, partiendo del análisis del dominio del problema, para la definición de requerimientos, siguiendo con el análisis y diseño orientado a Objetos, en función de un modelo basado en Componentes de Software, utilizando casos de uso como unidades ejecutables construidas por objetos, para posteriormente implementar la aplicación utilizando el lenguaje de programación "C++" siguiendo una metodología de pruebas iterativas a lo largo del desarrollo para garantizar una alta calidad en el desarrollo del software.

La herramienta fue construida en 3 módulos:
El primer módulo, "Generación de Meta Datos" permite agregar Meta Datos genéricos y de clasificación de acuerdo a una taxonomía especifica que pueda ser utilizada por los robots de indexamiento del Web tales como “Altavista”, “Google” por ejemplo. El segundo módulo "Clasificador de Documentos Remotos" permite crear y cargar taxonomías de clasificación jerárquicas compatibles con el formato RDM (Resource Descriptor Message), donde es posible clasificar documentos los cuales se pueden accesar a través de un URL. Y finalmente el tercer módulo "Navegación Web" permite navegar por los documentos del WWW (World Wide Web) utilizando el componente de Software ActiveX de Internet Explorer.

Meta Data Generation and Classification Tool for unstructured HTML Documents

ABSTRACT

In this Thesis work the main goal is to help document managers to classify and to structure web documents for the Web, so a Microsoft Windows software tool it is designed and developed to add standard Meta Data names to unstructured HTML(Hyper Text Markup Language) documents. International standard was used for the inclusion of typical Meta Data names as the Dublin-Core proposal. In addition, the tool can build hierarchical Taxonomies for any application domain, also a document manager can interactively to classify remote documents using the URL (Universal Resource Locator) to put documents under a classification name in the taxonomy. It is also possible to export hierarchies of classified documents in HTML format to be published in Web servers to access documents through direct taxonomy browsing. Having direct application to the building of Portals for user browsing through a navigational taxonomy.

The Objectory Process methodology from Rational Rose was used in the building of this tool for the analysis, design, implementation and test. Using this methodology, a systematic development process was possible, starting with the analysis domain to define requirements, then an Object Oriented analysis and design based on Software Components was done and the use cases model was applied too. Finally the implementation was done using the C++ programming language, applying a methodology of iterative testing through the development cycle to guarantee high quality in the developed software.

The tool was built in 3 modules: The first module “Metadata Generator” allows to generate generic and classification metadata against a specific taxonomy, so this Meta Data can be useful by web crawlers like Altavista, Google or some others to build indexes for search machines. The second module “Remote document classification” can load and build hierarchic taxonomies compatible with the RDM (Resource Descriptor Message) format and to classify documents by reference to URLs and finally the “Browser Module” which allow navigating through WWW (World Wide Web) documents using the Internet Explorer Activex component.

INTRODUCCIÓN

La mayor parte de la información hoy en día es recopilada del WWW (World Wide Web) a través de la Internet y las herramientas de análisis usadas son potencialmente útiles, pero por lo general la información adolece de una adecuada estructura para ser analizada y clasificada en todo su potencial. Los requerimientos de rapidez, confiabilidad y costos de acceso bajos, son las características relevantes que deben tomarse en cuenta para su análisis. El propósito es mantener información confiable, actualizada y orientada a las necesidades de las áreas de aplicación e investigación tecnológica. Con esta información se puede responder a los planes y acciones basados en el mejor conocimiento de lo que ha sido o podría ocurrir en una determinada área del conocimiento. La búsqueda de información puede incluir actividades diversas para la recolección de información, tales como búsqueda, seguimiento, reconocimiento y exploración, además de actividades de análisis y de evaluación. Actualmente las tecnologías modernas en este campo se enfocan en aplicar nuevos conceptos como son el de Inteligencia Tecnológica y la Administración del Conocimiento, esto obliga a tener mejores procedimientos y herramientas para estructurar adecuadamente la información que se genera y es publicada en Internet.

Toda esta información que se encuentra en formato de Web, puede ser clasificada adecuadamente a través de los encabezados de las páginas HTML por medio de los Meta Datos como el de Clasificación y el de Palabras clave principalmente, así como de los convencionales de título, autor, descripción y generador, entre otros. Con esta información los robots de búsqueda clasifican la referencia a estos documentos y almacenan índices en su Base de Datos para su posterior consulta.

El objetivo principal del presente trabajo de tesis, es desarrollar una aplicación que funcione bajo un sistema operativo de “Microsoft Windows” como herramienta de software, para generar en forma sencilla y automática los Meta Datos que se requieren en los encabezados de los documentos del Web y estructurar adecuadamente los documentos generados por las áreas de investigación, así como también poder recuperar referencias de documentos encontrados en la Internet y clasificarlos de acuerdo a una Taxonomía propia en el dominio de una aplicación, de tal forma que estas referencias clasificadas en un árbol taxonómico queden ubicadas a través de un archivo de interfaz en la base de datos del servidor donde esté instalado un Robot de búsqueda.

Para el desarrollo de esta herramienta de “software” se aplicó la metodología propuesta por Rational Rose “OBJECTORY PROCESS” que permite documentar los requerimientos en forma de casos de uso, logrando un entendimiento común entre el dominio del problema y las capacidades de desarrollo, describiendo cómo crear una arquitectura robusta y estable basada en la administración de requerimientos usando el paradigma de desarrollo por componentes. Esta metodología provee lineamientos para la implementación del sistema, y describe cómo planear, implementar y ejecutar pruebas para verificar que los requerimientos del usuario se cumplan.

Con este objetivo el presente trabajo de tesis se desarrolló en 5 capítulos, los cuales se detallan a continuación:

En el *Capítulo 1* se describe la problemática de los documentos del Web codificados en HTML que están orientados a la presentación de la información más no al contenido, por lo tanto no contienen información estructurada, así también se describe el funcionamiento de los Robot´s de

búsqueda del Web y de la búsqueda de Meta Datos en los documentos que consultan estos Robot´s y que forman parte del encabezado de los documentos HTML. Se explica la importancia de la taxonomía para la clasificación de documentos así como las herramientas de clasificación semi-automáticas y automáticas utilizadas para la estructuración adecuada de los documentos Web. Se resalta la importancia de los nuevos conceptos en el manejo de la información como lo es la administración del conocimiento e inteligencia tecnológica y por último los objetivos generales y particulares de este trabajo de tesis.

En el *capítulo 2* Se describen los requerimientos y necesidades que deben cumplir el desarrollo de la aplicación en función de la metodología propuesta por “Rational Software Corporation” identificada como “Objectory Process” definiendo los diferentes casos de uso para la aplicación a desarrollar así como el modelo y estructura de estos casos de uso.

En el *capítulo 3*. Se realiza el Análisis y el Diseño de la aplicación basándose en las especificaciones planteadas en el capítulo 2, con lo cual se construirá el modelo de diseño basado en los Casos de Uso, sirviendo como una abstracción para la creación del código fuente. Aquí también se describen a detalle los Casos de Uso como unidades ejecutables en función de clases y sus objetos.

En el *capítulo 4*. Se define la organización del código en términos de una implementación utilizando el lenguaje orientado a objetos C++, se implementan las clases de los objetos diseñados en términos de componentes. Se realiza la integración de las implementaciones individuales por subsistema en una sola aplicación.

En el *capítulo 5*. Se definen un conjunto de pruebas para validar el clasificador de documentos, la integración de todos los componentes de la aplicación, y comprobar que todos los requerimientos de diseño han sido implementados e identificar posibles defectos de la aplicación.

En el *capítulo 6*. Se dan las conclusiones obtenidas, los beneficios y limitaciones de la herramienta de software desarrollada, así como los posibles trabajos futuros.

CAPÍTULO 1

PROBLEMÁTICA DE LOS DOCUMENTOS HTML NO ESTRUCTURADOS

El objetivo esencial de este trabajo es tener una herramienta de software amigable para la generación automática de Meta Datos, facilitando y contribuyendo a la solución de la problemática presentada por los documentos en HTML no estructurados y lograr una mejor clasificación por los Robots de Búsqueda. Particularmente, en este capítulo se describe la problemática generada por los documentos no estructurados revisando la tecnología de Web, de los formatos HTML y de los Robots de Búsqueda para resaltar la importancia de un clasificador de documentos para el Web en el proceso de administración del conocimiento en los sistemas de inteligencia tecnológica.

1.1 El Web y el formato del documento HTML

Hoy en día la tecnología a través de redes de computadoras ha abierto un mundo de posibilidades, con el uso de la tecnología WWW (World Wide Web). Desde el punto de vista técnico, Web es un sistema gráfico de información de hipertexto, distribuido, global, interactivo, dinámico e independiente de la plataforma y que funciona en INTERNET con protocolos TCP/IP **[TCP/IP] [IntroTCP/IP].** El hipertexto permite leer y navegar a través del texto y por la información gráfica de una manera no lineal, es decir, se puede ir de un punto a otro según el interés que requiera el usuario enlazando diferentes páginas a través de vínculos en un mismo documento ó en diferentes documentos, así como en un mismo servidor ó de otro servidor conectado a la Internet.

El Web además de texto, también permite incorporar gráficos, sonidos, videos, multimedia y aplicaciones incrustadas, sin importar la plataforma de hardware y software de la computadora con la que esté trabajando. Unicamente se requiere tener un navegador de Internet y a través de éste, se puede conectar a cualquier servidor Web en el mundo y poder consultar la información que en estos sitios esté publicada.

El diseño de las páginas para poder ser publicadas en un servidor de Web, se realiza a través de código HTML (Hyper Text Markup Language) **[HTMLesp] [HTML] [ISO8879]**. Este lenguaje permite crear documentos para el Web a través de una estructura que es descrita con tres etiquetas básicas:

<HTML> es la primera etiqueta que todo documento HTML debe tener. Todos los componentes de una página deben estar contenidos dentro de la etiqueta de apertura <html> y la de cierre </html>.

<HEAD> es una etiqueta medular que permite estructurar los documentos HTML definiendo el encabezado del documento, que contiene información acerca del mismo y que proporciona información importante al usuario y a las máquinas ó Robots de Búsqueda, tales como Autor, Identificador, Clasificación, Derechos de copia, Descripción, Software de Creación, Generador, Palabras clave, entre otras. Con estos datos es posible hacer una clasificación adecuada de la información que se encuentra en la Internet y facilitar la recopilación de sus referencias (direcciones URL) para la consulta de documentos HTML bien estructurados. Para esto se requiere contar con herramientas que faciliten la estructuración de los documentos HTML, es decir poder crear la información que es puesta en la etiqueta de HTML <HEAD> de manera sencilla y rápida.

<BODY> es la tercera etiqueta de la estructura básica de un documento HTML y es donde es puesta toda la información que se quiere publicar en la página para el Web.

1.2 Los Robots de Búsqueda del Web y los Meta Datos

Los Robot de Búsqueda están programados para identificar referencias (URL´s) de fuentes sobre la Intranet ó Internet y almacenarlas en su base de datos, como descriptores de recursos (Resource Descriptor). El Robot de Búsqueda es un programa que actúa como un agente que revisa documentos de la red generando referencias hacia las fuentes de información encontradas y de acuerdo a una clasificación por temas que almacena en su Base de Datos. El trabajo del robot está basado en dos tipos de filtros, el “filtro enumerador” el cual realiza búsquedas de referencias usando mecanismos y protocolos de red probando cada referencia para validar si puede ser enumerada y si es así, analizar y almacenar esa referencia. El otro tipo de filtro es el denominado “filtro generador” el cual prueba cada referencia para validarla y crear un descriptor de referencia (RD) y guardarlo en la base de datos del Robot de Búsqueda **[RDCompass]**.

Una vez que es leído un documento mediante su referencia ó URL, el robot examina el contenido del documento y extrae los hipervínculos para verificar contra sus filtros mediante procesos concurrentes. Si el documento fuente pasa la prueba de enumeración, el robot lo identifica y continúa extrayendo posteriores referencias URL, si el documento pasa la prueba de generación, el robot genera un descriptor de recurso (RD) para éste y lo coloca en la base de datos del Robot de Búsqueda como se muestra en la figura 1.1.

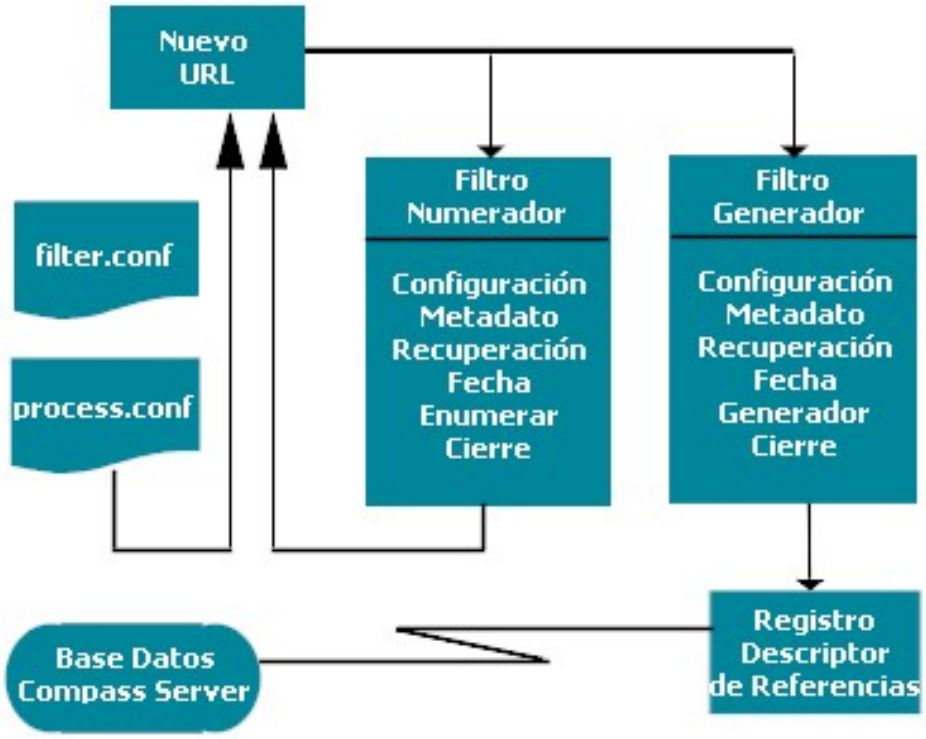


Figura 1.1.- Extracción y enumeración de páginas por un Robot de Búsqueda

Por otro lado, la estructura de codificación en documentos HTML contiene etiquetas que definen la operación de este código, como ya se mencionó en la sección 1.1 la etiqueta HEAD define el encabezado del documento, que contiene información acerca del mismo y que proporciona información del usuario y datos para las máquinas de búsqueda.

Netscape propuso ante el Consorcio del WWW (W3C) la inclusión de Meta Datos a los documentos utilizando la etiqueta Meta. Esta etiqueta se define dentro de la etiqueta <HEAD> y proporciona información de los Meta Datos. Esta etiqueta especifica información acerca del documento, y es usada para identificar propiedades tales como Autor, Título, Descripción, Clasificación, Fecha de vencimiento, Lista de palabras clave, entre otras, y

les asigna valores a estas propiedades. La especificación Meta no define exactamente un conjunto estándar de propiedades, además no tiene efecto sobre la apariencia del documento publicado en el Web sino que es utilizada para ser usada por otros programas como son los Robots de Búsqueda ó Navegadores de Web. Estas propiedades se definen con un atributo llamado NAME que especifica el tipo de Meta Dato, este atributo se usa para transportar la información de la página a los Robots de búsqueda, por ejemplo el valor de una etiqueta de tipo Meta Name=”DESCRIPTION” resumirá el contenido de la página y además podría adicionar una lista de conceptos importantes Meta Name =”KEYWORDS” ó palabras clave del documento. Otro atributo de Meta es CONTENT que especifica el valor de la propiedad. Se pueden configurar los siguientes atributos Name: autor, identificador, clasificación, derechos de copia, descripción, software de creación, generador y palabras clave, entre otras de acuerdo a la descripción de Meta Datos de los elementos del conjunto de "Dublin Core" **[DublinC99]**.

Los Meta Datos permiten estructurar adecuadamente los documentos para el Web y facilitan su clasificación por los Robots de Búsqueda, de aquí la importancia de contar con una herramientas para generar estos de forma sencilla de tal manera que los investigadores ó administradores de un sistema de inteligencia tecnológica puedan contribuir en la solución de la problemática de los documentos no estructurados.

1.3 Taxonomía y beneficios de la clasificación de documentos

Hoy en día con la aplicación del concepto “Administración del Conocimiento” que es el proceso de identificar y capturar la pericia colectiva de los empleados de una compañía cualquiera que sea el lugar donde resida (bases de datos, papel ó experiencia de las personas) y distribuirla hacia cualquier lugar donde ayude a producir los mejores resultados. Tiene por

finalidad capturar, administrar, clasificar y estudiar el conocimiento generado en la organización. El conocimiento se le define como una combinación de información, contextos y experiencias, además establece que la forma de más bajo nivel de conocimiento son los datos. Los datos no tienen significado "per se", pero si son ordenados, clasificados, analizados e interpretados se convierten en información. La información se caracteriza por tener substancia y propósito. Sin embargo, la información no tiene significado, cuando no es combinada con un contexto y una experiencia que en su conjunto llega a ser conocimiento.

En este sentido el Meta Dato toma importancia en una mejor administración del conocimiento con el concepto de "Clasificación", ya que éste permite ordenar las referencias a los documentos encontrados en la Internet de manera adecuada por temas y de acuerdo al contexto en que se utiliza. La técnica más utilizada por la mayoría de los Robots de Búsqueda está basada en el uso de una Taxonomía, donde están configurados los temas y subtemas del dominio ó de un área de conocimiento específica, a través de un árbol jerárquico. Con esta estructuración de temas a través de una taxonomía, se clasifican y almacenan las referencias de los documentos encontrados en el Web, en las Bases de Datos de los Robots de Búsqueda. Existe una propuesta de estandarización para almacenar esta taxonomía en formato RDM **[RDM].** De acuerdo a esto el Meta Dato de clasificación contenido en los documentos HTML, es leído por el Robot de Búsqueda a través de sus filtros, donde es interpretado su contenido, éste es el tema ó temas que clasifican al documento junto con toda su ascendencia, es decir toda la rama de nodos hasta el tema correspondiente, a continuación se muestra un ejemplo de un Meta Dato de clasificación.

```
<meta
name="Classification"
content="Hydroprocessing:Hydrotreatment:Catalysts
&Additives:Catalysts:Catalysts Delivery System:Catalysts Hoppers">
```

Este contenido es procesado por el Robot de Búsqueda de acuerdo a las directrices que tiene programadas y que corresponden a las taxonomias cargadas, si el contenido del Meta Dato de clasificación coincide con alguna de estas taxonomias, el Robot de Búsqueda genera un descriptor de recurso de la referencia de este documento y almacena este índice en su base de datos. Por esta razón el Meta Dato de clasificación tiene vital importancia en la estructuración de documentos que se publican en el Web.

A través de una interfaz de programación de aplicaciones se puede tener acceso a los índices de las bases de datos de los robots, utilizando mensajes de descripción de recursos y algún lenguaje de programación tal como C++ ó Java para interpretar, modificar, ó crear taxonomías en formato RDM.

Los mensajes de descriptores de recursos (RDM) representan un formato de mensaje que pueden usar los programas para intercambiar referencias a documentos a través de una red, por ejemplo: un proceso (Un cliente ó un agente) envía una solicitud de mensaje RDM a otro proceso (Un servidor) el cual procesa la solicitud enviando un mensaje de respuesta RDM, similar al modelo HTTP/1.0 solicitud/respuesta.

El RDM es un mecanismo para encontrar y recuperar Meta Datos acerca de los documentos fuente de la Intranet/Internet, conocido como Descriptor de Recursos. Éste consiste de una lista de parejas atributo valor (por Ej. Autor = Felipe Dávalos, Titulo = “Generador de Meta Datos”) que es asociada con una referencia URL. A un documento en particular.

1.4 Clasificación automática y semi-automática de documentos

Como ya se ha indicado la importancia de estructurar adecuadamente los documentos Web, es momento para mencionar que hay dos tipos de clasificación, la clasificación semi-automática y la clasificación automática.

La semi-automática permite generar un Meta Dato de clasificación a partir de la captura manual de las rutas de temas y subtemas específicos en el área de conocimiento a través de cajas de edición de texto, para posteriormente generar en forma automática el código correspondiente a la sintaxis del Meta Dato e insertarlo en el documento HTML al que se desea estructurar.

La automática permite generar la referencia al documento directamente en un archivo de taxonomía compatible con la base de datos del robot indexador. Estas dos formas de operación serán consideradas para la funcionalidad y operación del clasificador desarrollado para este trabajo de tesis.

También es momento para mencionar que existen desarrolladas varias herramientas de software comercial para la generación de Meta Datos pero que cada una tiene sus ventajas y desventajas una con respecto a la otra. Una de estas herramientas es la desarrollada por “HiSoftware, Inc.” Denominada “TagGen^{TM}” en su versión 4.5P [TagGen], ésta aplicación es para plataforma Windows 95/98/ME/NT/2000, cuenta con la posibilidad de generar los Meta Datos de Palabras Clave (KeyWords) en forma automática y generar en forma semi-automática los Meta Datos de descripción, clasificación, generador, lenguaje, robot, evaluación, derechos de copia, distribuidor, autor y algunos otros. En la Figura 1.2 se muestra la ventana principal de esta herramienta.

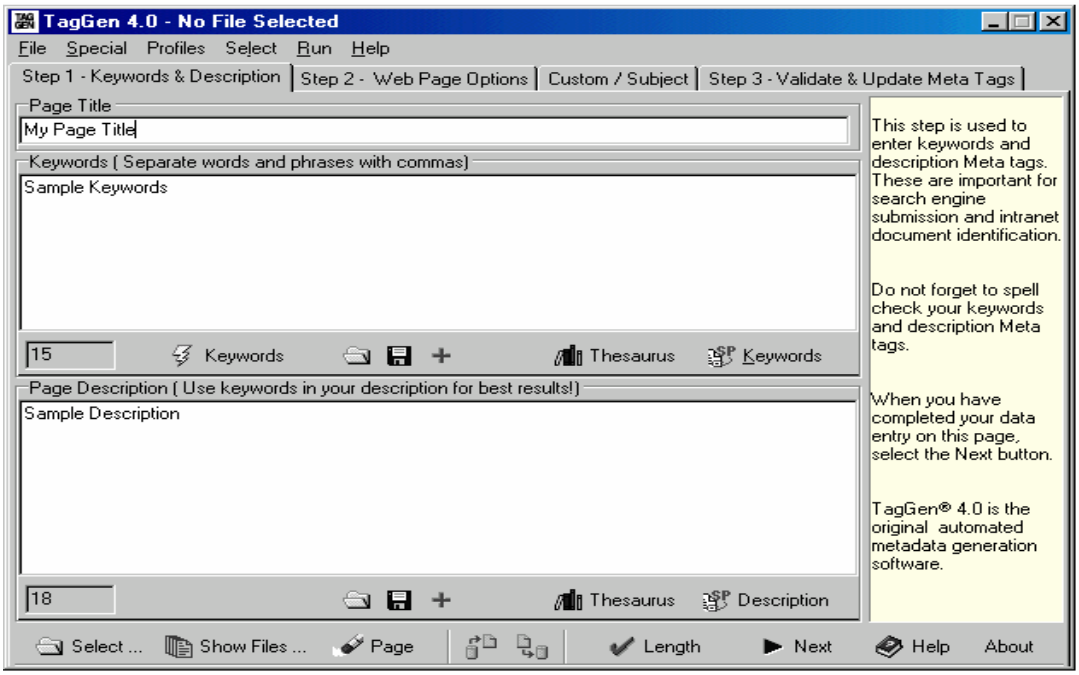


Figura 1.2 Ventana principal de “TagGen”

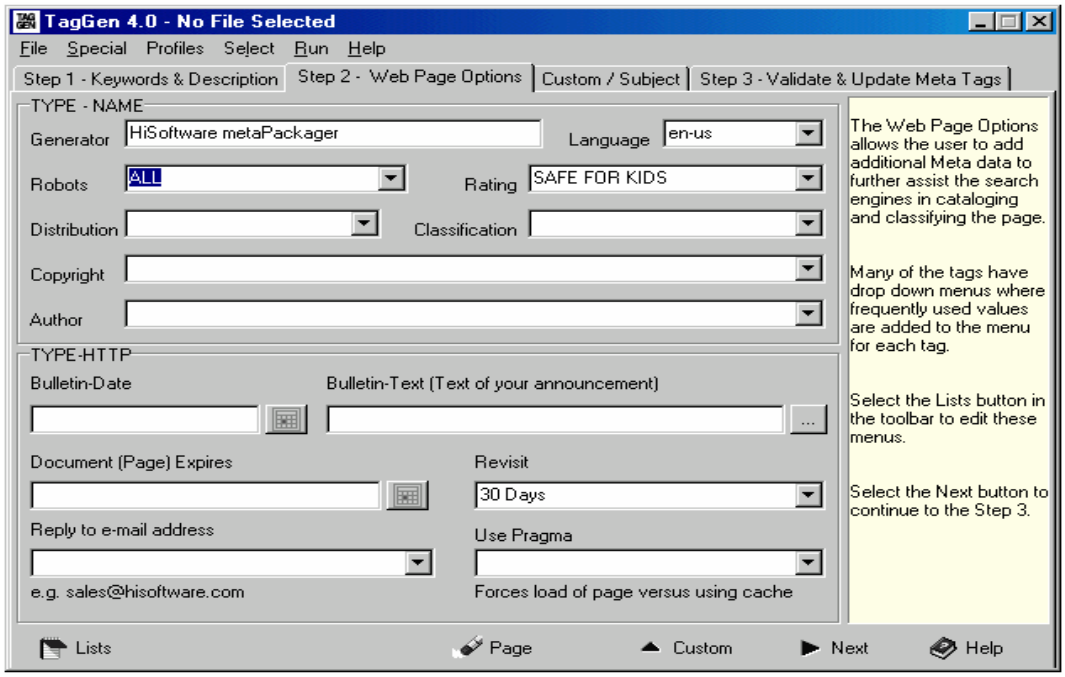
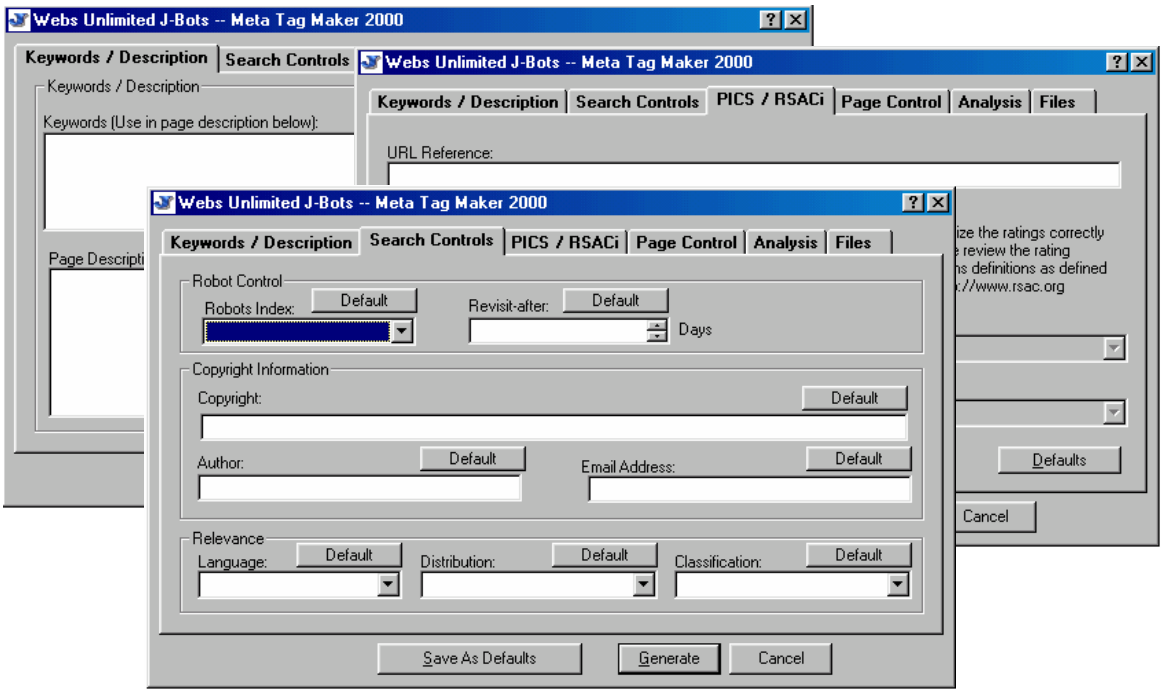


Figura 1.3 Ventana generación semi-automática de Meta Datos con “TagGen”

En la Figura 1.3 “Ventana generación semi-automática de Meta Datos con TagGen” se muestra los diferentes campos de captura y de configuración para la generación de los diferentes Meta Datos.

Otra herramienta es la desarrollada por “WebSunLimited subsidiary of IVR System Corp.” Denominada “Meta Tag Maker 2000” [TagMaker], aplicación que corre en plataforma Windows 95/98/2000, y ésta herramienta está enfocada a los documentos HTML generados por “FrontPage” de “MicroSoft” para generar Meta Datos en forma semi-automática, es decir tiene una caja de dialogo con 6 carpetas, ver Figura 1.4: “Keywords and Description” para generar el Meta de Palabras Clave y el de descripción; “Search Controls” para generar los Metas de autor, robot, clasificación, lenguaje entre otros que definen la interacción con los Robots de Búsqueda; “PICS / RSACi” para generar Meta Dato que restringe el acceso a los navegadores de páginas que no se desean desplegar; “Page Control” para generar el Meta Dato de Expiración y el Meta Dato de “refresh”; “Analysis” que revisa los Meta Datos generados; y “File” para el manejo de los archivos generados y los documentos HTML. En la Figura 1.4 se muestran las diferentes ventanas de esta herramienta.



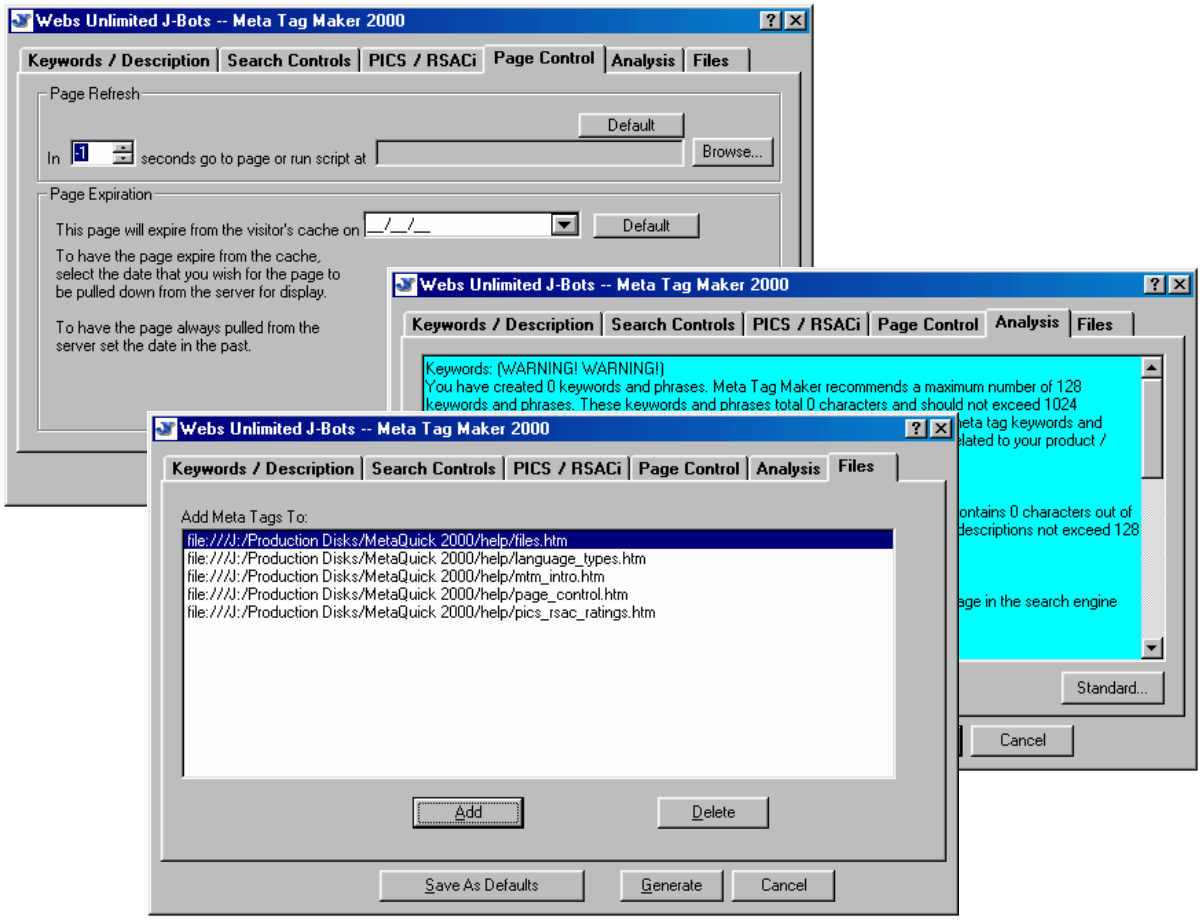


Figura 1.4 Ventanas de la caja de diálogo de MetaTag Maker2000

Hay una gran cantidad de generadores de Meta Datos incluso en línea a través de Internet, solo se mencionan algunos más como referencia, tales como; *MetaWizard, Profesional Meta Tag Generator, WebPromote's Meta-Tag Generator, Meta Tag Builder, SiteUp's Meta-Tag Generator, Fireball Meta-Tag-Generator*, entre otros. [TagVs]

Todos estos generadores de Meta Datos trabajan sobre el documento Web al que se desea adicionar, no todos tienen la capacidad de generar el Meta de Clasificación y además los que tienen esta posibilidad no puede crear el Meta a partir de un árbol de Taxonomía. Las opciones más avanzadas en estos generadores son el que permite crear el Meta de

palabras clave (MetaKeyword) al realizar una búsqueda sobre el documento en HTML de las palabras que contengan una cierta repetición, así también la opción de poder analizar los Meta Datos generados para poder realizar correcciones antes de adicionarlos al código HTML.

El clasificador desarrollado para este trabajo de tesis tiene la opción de generar los Meta Datos que se cubren en la mayoría de estos generadores comerciales, adicionando la capacidad de generar automáticamente el Meta de Clasificación a partir de un archivo en formato RDM que contiene la taxonomía con los temas requeridos para efectuar la clasificación del documento de acuerdo al área de conocimiento. Así también tiene la facilidad de poder visualizar el documento dentro de la misma aplicación interpretando código HTML desplegándolo como en un navegador ó directamente en formato de código HTML, permitiendo al usuario después de generar los Meta Datos, poder editar dicho código con las herramientas que ofrece cualquier editor de texto. Otra característica que se le ha dado a esta herramienta, es poder abrir, editar, adicionar y manipular el árbol taxonómico que corresponde al archivo RDM generado y cargado en la base de datos de un robot indexado. De todos los documentos visualizados en el propio navegador de la herramienta, es posible recuperar las referencias correspondientes y el usuario puede clasificarlas según la taxonomía abierta, así mismo el usuario tiene la opción de adicionar nuevos temas a la taxonomía ó generar una nueva taxonomía. La herramienta desarrollada permite almacenar en un archivo la taxonomía con todas las referencias clasificadas para tener la posibilidad de recuperarla en el momento que el usuario lo requiera y navegar por los documentos a los que hace referencia.

1.5 Importancia de un Clasificador de Documentos

Como ya se ha mencionado, los documentos en formato para su publicación en el Web requieren de una adecuada estructura para su óptima explotación al difundirse por la Internet, así también se ha dicho que el código HTML tiene la sintaxis a través de la etiqueta Meta para la completa estructuración y que la necesidad de aplicar los conceptos como la administración de conocimiento y las tecnologías en sistemas de inteligencia tecnológica, un clasificador de documentos para el Web toma una importancia significativa, ya que es una herramienta que permite generar los datos suficientes para que las referencias a estos documentos que se publican en el Web estén bien estructurados y de esta manera queden ubicados en las bases de datos de las máquinas ó Robots de Búsqueda que se encuentran conectados a la red de redes, la Internet. Al garantizar que los documentos que contienen conocimiento y que están bien estructurados a través de un clasificador, se facilitan las tareas de administración del conocimiento y se contribuye sustancialmente a cubrir y cumplir con las expectativas y objetivos de los sistemas de inteligencia tecnológica para lograr una mejor administración del conocimiento en centros y áreas de investigación y desarrollo que generan información y publican en la Internet.

1.6 Inteligencia Tecnológica

En 1997, el Instituto Mexicano del Petróleo se dio la tarea de aplicar el concepto de Inteligencia Tecnológica, con el firme propósito de aumentar el conocimiento del estado del arte y de las tendencias en ciencia y tecnología, así como potenciar el aprendizaje y la aplicación del conocimiento entre sus investigadores, especialistas y personal encargado de la toma de decisiones. El término Inteligencia Tecnológica (IT) denota una serie de procesos y prácticas que una empresa lleva a cabo sistemáticamente para mantener

alerta a los cambios tecnológicos y científicos en su medio ambiente, así como de los competidores y de las oportunidades comerciales.

El IMP aplicó este concepto para PEMEX-REFINACION desarrollando un Sistema Integral de Inteligencia Tecnológica (SIIT) que puso en operación en octubre de 1999 con los servicios de: Cartera de Proyectos, Difusión de Información Tecnológica, Red de Expertos y Especialistas, Robot de Búsqueda, Sistema Experto de Búsqueda y Buzón de Mejores Prácticas utilizando la tecnología de Internet estructurado en un sistema distribuido de información. Véase la **Figura 1.5** "Arquitectura del Sistema Integral de Inteligencia Tecnológica SIIT del IMP-PEMEX-REFINACION".

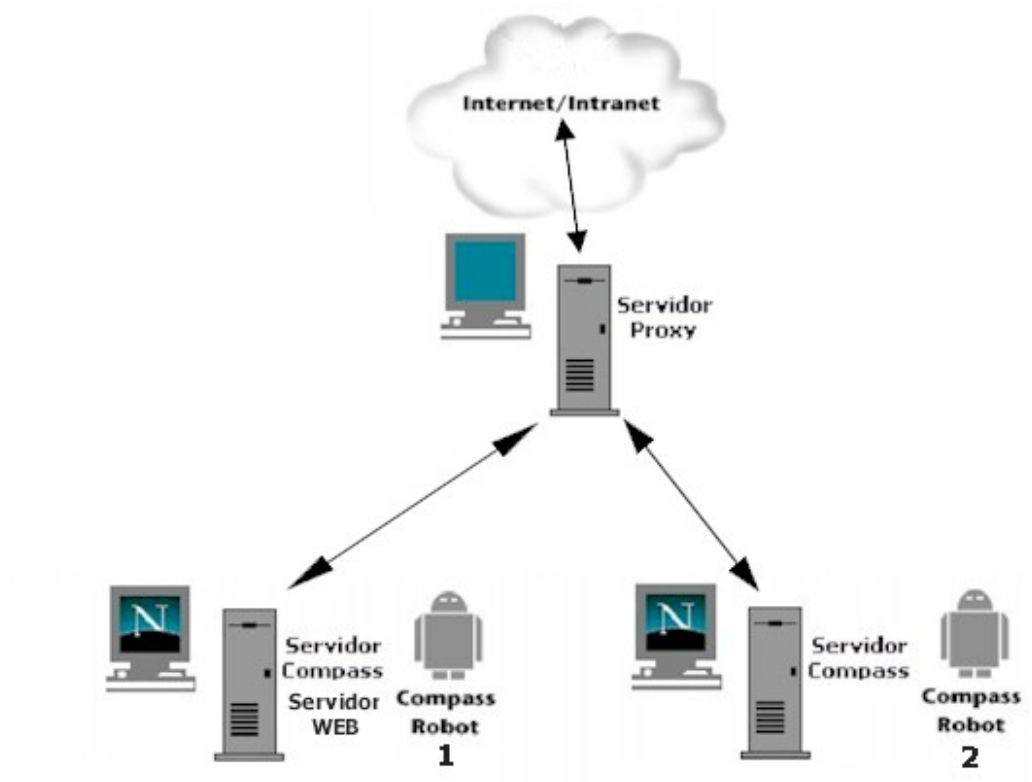


Figura 1.5 Arquitectura del Sistema Integral de Inteligencia Tecnológica SIIT del IMP-PEMEX-REFINACION

Como se mencionó anteriormente, uno de los servicios que proporciona el Sistema Integral de Inteligencia Tecnológica (SIIT) es el Robot de Búsqueda, que está basado en el uso de un robot de clasificación e indexación “Compass” de Netscape

1.7 Objetivos de la Tesis

En virtud de que el software comercialmente disponible para la clasificación de documentos no cuenta con una opción para clasificar por temas de acuerdo con una jerarquía de temas propios de cada dominio de aplicación, genera la necesidad de desarrollar una herramienta de software propietaria para codificar automáticamente los Meta Datos que se requieren en los encabezados (head) de los documentos en HTML en función de una taxonomía propia del dominio de aplicación.

Con este objetivo en mente la idea es construir una aplicación de software hecho a la medida para garantizar principalmente la adecuada clasificación de los documentos publicados en el Web por los Robots de Búsqueda en el proceso de Administración del Conocimiento en la aplicación de Sistemas de Inteligencia Tecnológica a través de una jerarquía de clasificación, así mismo proporcionar en la misma herramienta la posibilidad de construir taxonomías para cualquier domino de aplicación, y la clasificación de documentos remotos accesibles por un URL (Universal Resource Locator), dentro de una temática particular de la jerarquía, siendo posible también, exportar las jerarquías construidas junto con las referencias a los documentos clasificados dentro de ellas, en un formato HTML, para ser publicadas en un servidor Web y ser utilizadas desde los navegadores como sistema de navegación jerárquico a través de la temática correspondiente, permitiendo accesar directamente desde el navegador los documentos clasificados, teniendo esto aplicabilidad directa en la construcción de portales

con navegación por temas. Para el desarrollo de esta herramienta de software, se aplicó la metodología propuesta por Rational Rose “Objectory Process” **[ObjMan]** que permite documentar el análisis y diseño de la aplicación durante todo el ciclo de vida del desarrollo. El lenguaje de programación elegido fué C++ y el ambiente de desarrollo fue la biblioteca de componentes visuales de “C++ Builder de Borland”.

Resumen

En este capitulo se resaltó la problemática que se tiene con los documentos HTML no estructurados y que a través del conjunto de Meta Datos codificados en el encabezado del documento HTML se logra la estructuración adecuada, así mismo se enfatizó la importancia de la clasificación a través de una taxonomía en el proceso de búsqueda que realizan los Robot de Búsqueda y la aportación valiosa del Meta Dato de clasificación en la estructuración de los documentos HTML que se publican en el Web. Se mencionaron las diferentes herramientas que existen para la generación de Meta Datos con sus características y limitaciones y la importancia del clasificador de documentos HTML en la administración del conocimiento y los sistemas de inteligencia tecnológica. Por último se plantearon los objetivos para el desarrollo de esta herramienta de software objeto de este trabajo de tesis. En el siguiente capítulo se definen los requerimientos que deben cubrirse en el desarrollo de esta herramienta.

CAPÍTULO 2

DEFINICIÓN DE REQUERIMIENTOS CLASIFICADOR DE DOCUMENTOS

El propósito de este capítulo es definir los requerimientos de la herramienta de software a desarrollar a través de la metodología propuesta de “Objectory Process”. Aquí se describe el dominio de las necesidades para tener un entendimiento completo de los requerimientos y limitaciones y proporcionar una base de planeación técnica, a través del desarrollo de un modelo de casos de uso que describe lo que la herramienta deberá hacer.

2.1 Descripción de los requerimientos y Modelo del Dominio.

La definición de los requerimientos se describe en vocabulario común usando los términos más comunes en el dominio del problema. Es decir hacer las descripciones textuales del sistema, especialmente en términos de casos de uso. De esta manera, se mantienen las descripciones textuales constantes respecto a lo que se espera que debe hacer la herramienta a desarrollar. A continuación se hace una descripción textual de los requerimientos de la herramienta de software a desarrollar.

Para estructurar un documento HTML es necesario generar los Meta Datos de clasificación, palabras clave y/o generales tales como título, autor, fecha de edición, descripción, generador y lenguaje. El trabajo de esta tesis mediante el desarrollo de una herramienta de programación, permite al usuario abrir el documento en formato HTML, donde serán incrustados los Meta Datos automáticamente por esta herramienta, además permite editar el código del documento y puede desplegarlo ya sea en código HTML fuente ó desplegar la presentación del documento de forma normal.

Para el caso del Meta Dato de clasificación la herramienta permite seleccionar un archivo Taxonomy.rdm generado por el Robot de Búsqueda, donde se encuentra el árbol de temas que se desea clasificar, el archivo es abierto y leído para desplegarse en una ventana. A través ésta, el usuario puede seleccionar hasta diez diferentes tópicos que requiera adicionar al Meta Dato, los cuales serán desplegados en una lista, así mismo poder eliminar algún tópico seleccionado ó limpiar la ventana completamente.

Una vez seleccionados los tópicos requeridos, el usuario puede generar e insertar de manera automática el Meta Dato de clasificación en el código del documento HTML seleccionado. La herramienta detecta si el documento HTML abierto ya contiene Meta Datos y permitirle al usuario seleccionar la opción de eliminarlos sustituyéndolos con los Meta Datos generados por el programa ó adicionar estos a los ya existentes. Así también, tiene la posibilidad de generar este Meta Dato en otro documento HTML que el usuario requiera así como de cargar otra nueva taxonomía.

Para la generación de identificadores Meta de palabras clave (KEYWORDS) la aplicación busca en el texto del documento HTML, las palabras que se repitan con una frecuencia de repetición configurada por el usuario. Las palabras encontradas se deben desplegar en una ventana, de donde el usuario puede hacer una selección múltiple de las palabras clave que requiera para generar el Meta correspondiente. También tiene la opción de modificar el número de repeticiones para la búsqueda así como de poder visualizar el documento HTML con el identificador Meta generado para una posible edición adicional por el usuario.

Esta aplicación también, cuenta con la opción de generar los Meta Datos convencionales como son título, autor, descripción, generador, fecha de creación y lenguaje.

Otra función importante de la aplicación es poder brindar la capacidad de navegar por la INTRANET/INTERNET y seleccionar referencias (URL´s) de documentos que el usuario requiera conservar, pudiendo clasificarlo de acuerdo al árbol de Taxonomía usado por el robot y guardar la clasificación en un archivo que pueda recuperarse cuando el usuario requiera navegar por los documentos clasificados. La aplicación también permite editar la taxonomía abierta al adicionar nuevos temas ó eliminarlos, teniendo la posibilidad de generar una nueva taxonomía compatible con las de los Robots de Búsqueda.

El modelo del dominio obtenido del análisis de la descripción realizada se muestra en la Figura 2.1. donde se indican las entidades reconocidas del dominio del problema.

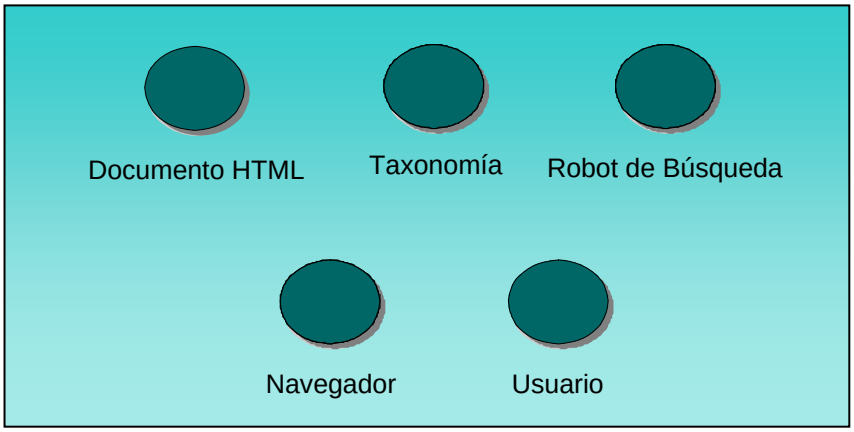


Figura 2.1 Modelo del Dominio

El modelo de dominio está representado por las entidades que resultaron de los términos comunes encontrados en la descripción en vocabulario natural de los requerimientos.

2.2 Modelo de Casos de Uso

Del análisis de los requerimientos y de la definición de las entidades principales de la aplicación se determinaron que los actores son:

- Especialistas que clasifican documentos HTML según una taxonomía
  Servidor de búsqueda (Robot de Búsqueda)
  Usuario de Consultas de INTERNET/INTRANET

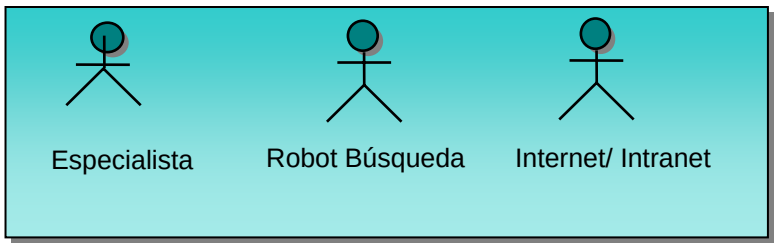


Figura 2.2 Modelo de Actores

Y los Casos de Uso de esta herramienta se describen a continuación:

Generación de Meta Datos de archivos locales

Abre y lee el documento HTML de interés. Abre y lee el archivo del árbol taxonómico para el Meta Dato de clasificación. Detecta si el documento HTML leído contiene Meta Datos. Genera y codifica en el documento HTML seleccionado el tipo de Meta Datos requeridos (Genéricos, Clasificación, Palabras clave). Sustituye ó adiciona los Meta Datos generados por la aplicación.
Edita el documento seleccionado en código HTML. Guarda en disco el documento HTML codificado con los nuevos meta datos generados, para ser enviado al Servidor de Documentos.

Navegación por documentos

La Herramienta permite al usuario navegar por los documentos clasificados previamente a través de INTRANET/INTERNET.

Clasificación de documentos remotos

Abre un árbol taxonómico compatible con los Robots de Búsqueda. Selecciona un documento de interés a través de la navegación de documentos y coloca su referencia en el tópico correspondiente del árbol taxonómico abierto, posteriormente tiene la opción de poder generar un archivo de salida con esta información, que el usuario puede cargar posteriormente para navegar por los documentos clasificados. Adiciona ó elimina temas del árbol taxonómico abierto. Genera nuevas taxonomías compatibles con el Robots de Búsqueda.

2.3 Estructura del Modelo de Casos de Uso

Para obtener el modelo de casos de uso es necesario hacer una descripción detallada del flujo de eventos de los casos de uso propuestos y estructurar el modelo, determinando las relaciones de asociación entre actores y casos de uso, las relaciones de <uso> y <extendidas> entre casos de uso y las relaciones de generalización entre actores.

2.3.1 Caso de uso generación de Meta Datos en archivos locales

Este caso de uso se activa cuando el usuario abre el documento HTML en donde se desea generar los Meta Datos. Una vez abierto este archivo, el usuario seleccionará el tipo de Meta Dato que desea generar.

Si el usuario selecciona la opción de Meta Dato genérico, el programa desplegará un formulario con los campos de texto requeridos para capturar los Meta Datos de título, autor, descripción, generador, fecha de creación y lenguaje. Una vez capturados estos datos el usuario podrá solicitar que el programa genere el código correspondiente en el documento HTML abierto.

Posteriormente se desplegará el código HTML generado con los Meta Datos creados, donde el especialista podrá realizar cambios con las herramientas convencionales de una ventana de edición de texto (copiar, pegar y borrar) si así lo desea ó podrá guardar en el mismo documento HTML abierto. En este momento el especialista tendrá la opción de abrir otros documentos HTML a donde podrá insertar los mismos Meta Datos generados. Si el usuario selecciona la opción de generar el Meta Dato de clasificación, el programa permite abrir un archivo taxonómico (*.rdm) para que posteriormente el usuario pueda abrir y cargar un nuevo documento HTML de interés ó utilizar el mismo documento abierto, donde se insertará el Meta Dato de clasificación.

Una vez abierto el documento HTML el especialista podrá expandir el árbol taxonómico y seleccionar hasta diez tópicos de la taxonomía. El especialista podrá deshabilitar tópicos y volver a habilitarlos hasta que esté conforme con los tópicos requeridos. Una vez terminada la selección el especialista podrá activar el botón de generación de Meta Dato de clasificación.

Si el programa detecta Meta Datos de clasificación en el documento HTML abierto, entonces le solicitará al especialista la opción de eliminarlos para insertar el Meta Dato recientemente generado ó adicionar el Meta Dato reciente al ya existente. Posteriormente se desplegará el código HTML generado con el Meta Dato de clasificación creado, donde el especialista podrá realizar cambios con las herramientas convencionales de una ventana de edición de texto (copiar, pegar y borrar) si así lo desea ó podrá guardar en el mismo documento HTML abierto. En este momento el especialista tendrá la opción de abrir otro documento HTML a donde podrá insertar el mismo Meta Dato de clasificación generado ó abrir una nueva taxonomía.

Si el usuario selecciona la opción de generar el Meta Dato de palabras clave, el programa permitirá cargar y abrir un nuevo documento HTML ó en el mismo documento abierto donde se insertará el Meta Dato a generar. El especialista podrá seleccionar la frecuencia de repetición en la búsqueda de las palabras en el texto del documento, posteriormente activará el botón para buscar palabras clave, las que se desplegarán en una lista en donde el especialista podrá seleccionar las palabras clave que requiera. Una vez seleccionadas las palabras de la lista el especialista podrá activar el botón para generar el Meta Dato de palabras clave.

Si el programa detectó Meta Dato de palabras clave en el documento HTML abierto, éste le pedirá al especialista que seleccione la opción de eliminar estos e insertar el Meta Dato generado ó adicionar el Meta Dato generado al ya existente. Posteriormente se desplegará el código HTML con el Meta Dato de palabras clave creado, donde el especialista podrá realizar cambios con las herramientas convencionales de una ventana de edición de texto (copiar, pegar y borrar) si así lo desea ó podrá guardar en el mismo documento HTML abierto, así también el especialista podrá cambiar la frecuencia de repetición en la búsqueda de palabras clave hasta que se obtengan las requeridas a juicio del especialista.

2.3.2 Caso de uso navegación por documentos

Este caso de uso se activará cuando el especialista active la opción de navegar, abriéndose una ventana con un navegador compatible con los componentes Activex de Internet Explorer **[Activex]**. Desde aquí el especialista podrá navegar por la INTERNET ó INTRANET. La referencia URL del documento en línea, es almacenada para su clasificación en el caso de uso clasificador.

2.3.3 Caso de uso clasificación de documentos remotos

Este caso de uso se activará cuando el especialista active la opción de Clasificador remoto. El especialista podrá accionar el botón de abrir taxonomía para cargar un archivo “rdm”, el cual podrá expandir y clasificar la referencia URL del documento en línea al activar el botón de adicionar referencia URL al tema seleccionado. El usuario podrá adicionar las referencias URL que desee y podrá generar un archivo de salida con todas las referencias clasificadas en el árbol taxonómico, este archivo podrá ser guardado para poder recuperarlo posteriormente. El usuario podrá abrir este archivo, exportarlo a formato HTML y navegar por las referencias clasificadas. Este caso de uso permitirá adicionar nuevos temas al árbol taxonómico ó eliminarlos, así como generar una nueva taxonomía en el mismo formato de un archivo “rdm” compatible con algunos Robots de Búsqueda **[RDCompass]**.

De esta descripción detallada del flujo de eventos de los casos de uso propuestos se obtiene el siguiente modelo de casos de uso.

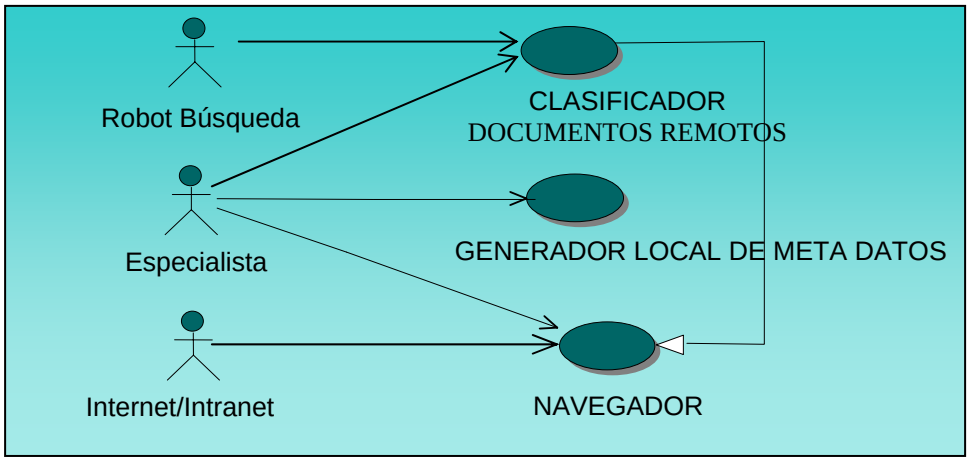


Figura 2.3 Modelo de Casos de Uso

Este modelo de casos de uso es un modelo de la funcionalidad esperada del clasificador de documentos y sirve como un hilo unificador durante todo el ciclo de vida del desarrollo.

Resumen

En este capítulo a partir del dominio de requerimientos, se definió de manera detallada la funcionalidad que debe cumplir la herramienta de software a desarrollar a través de la definición de un modelo de casos de uso, el cual se utiliza en las siguientes etapas de análisis, diseño, implementación y pruebas, así de esta manera se tiene una visión completa de lo que el clasificador de documentos debe hacer. En el siguiente capítulo se tomarán como base los casos de uso definidos para realizar el análisis y diseño orientado a objetos, ya que este modelo de casos de uso se traduce en un modelo de diseño que representa el conjunto de clases y objetos que instancia la funcionalidad definida.

CAPÍTULO 3

ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS DEL CLASIFICADOR DE DOCUMENTOS

En este capítulo se propone un modelo de diseño y una “Vista Lógica” de la arquitectura de la aplicación, que permita el desarrollo de código estable que represente una abstracción completa y basada en los resultados de la captura de requerimientos (el modelo de casos de uso), para que de esta manera se tenga una representación de la arquitectura en términos de clases y sus objetos.

3.1 Análisis y Diseño de Arquitectura

El propósito del análisis y diseño es describir como deben realizarse todos los requerimientos planteados, construir un diseño que resista cambios en los requerimientos, desarrollar una arquitectura de software estable y desarrollar especificaciones que sirvan como entrada para las etapas de implementación y pruebas. Del análisis y diseño se obtiene un modelo de diseño que sirve como una abstracción del código fuente, donde se describe como se realiza cada caso de uso en términos de clases y sus objetos.

La actividad de análisis y diseño esta centrada sobre la noción de una arquitectura. En la figura 3.1 se muestra una vista de la arquitectura del clasificador de documentos (Meta Generador) que muestra el entorno de aplicación de ésta herramienta.

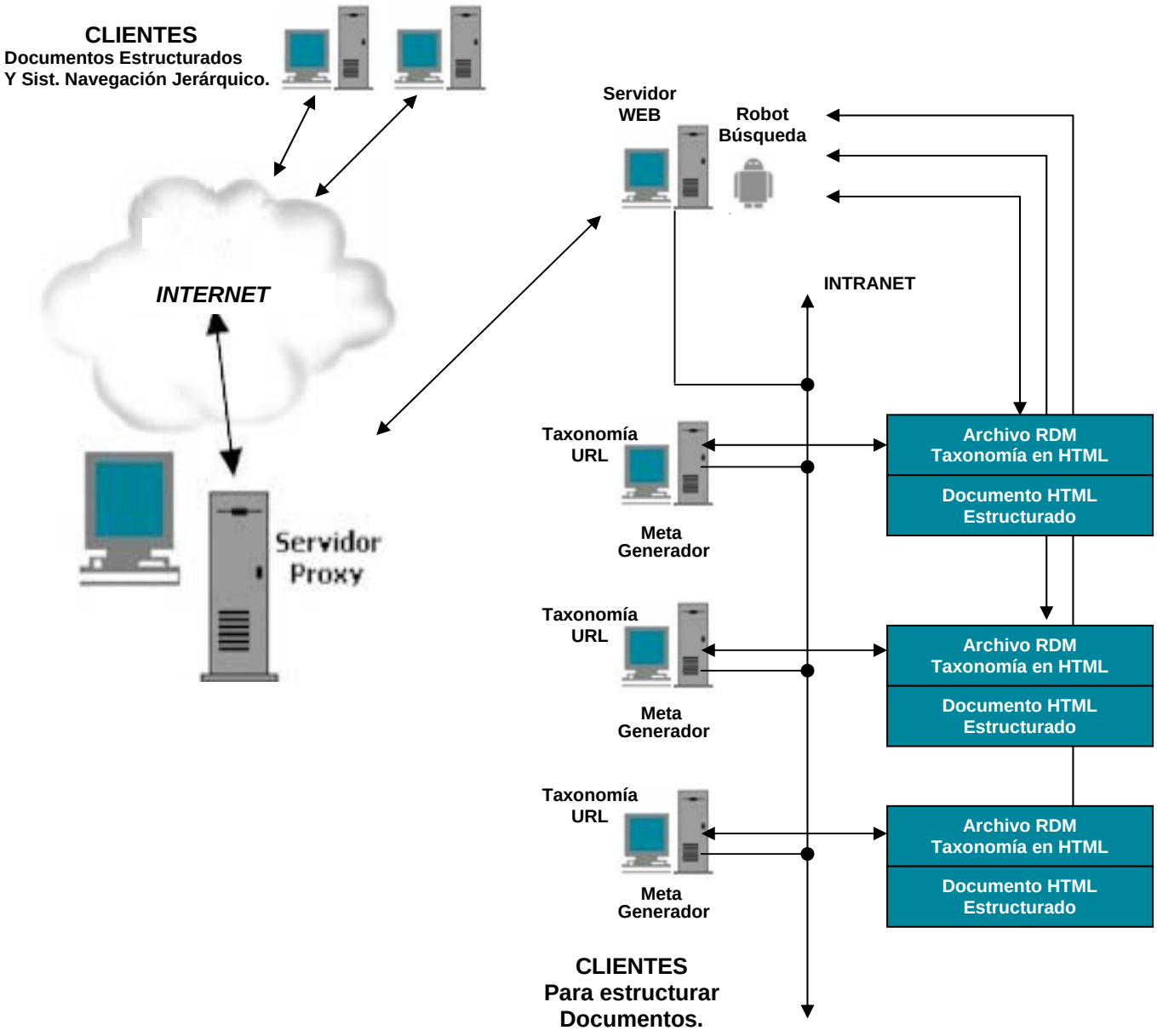


Figura 3.1 Arquitectura del Meta Generador

El desarrollo de esta aplicación está basado en un lenguaje orientado a componentes sobre la base de una biblioteca de componentes visuales (Visual Component Library- VCL) de “Borland”, Estos componentes pueden manipularse en tiempo de diseño y cuya clase principal es “TObject” con la jerarquía mostrada en la Figura 3.2

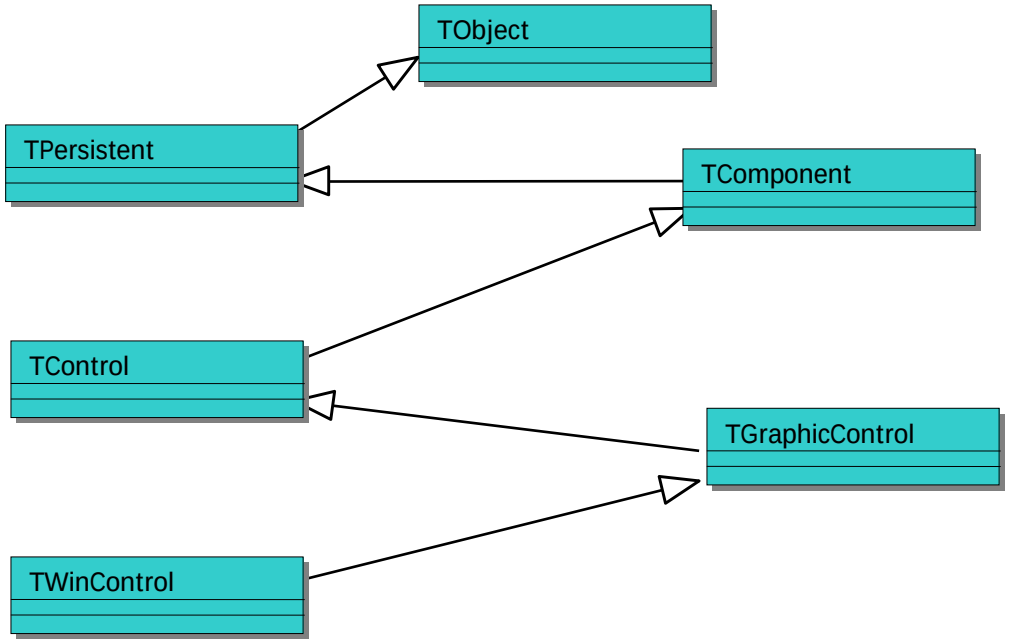


Figura 3.2 Jerarquía de la Biblioteca de Componentes Visuales VCL de Borland

El modelo de diseño está organizado por niveles, el nivel superior que es el nivel de aplicación, seguido de un nivel de especificaciones de negocio (si se tiene como objetivo), un nivel intermedio de interfaz gráfica de usuario y un último nivel de “software” del sistema.

El proceso de análisis de arquitectura se centra en los niveles de aplicación y de negocio, mientras que el proceso de diseño de la arquitectura se centra en los niveles de interfaz de usuario y de software del sistema. Véase Figura 3. 3

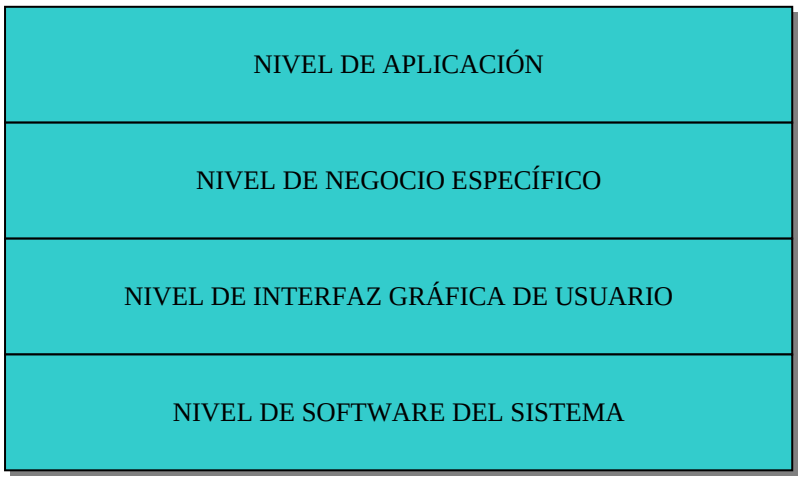


Figura 3.3 Niveles del modelo de diseño.

Para el nivel de la aplicación el modelo de diseño preliminar es como el mostrado en la Figura 3.4, donde se muestra la dependencia de los objetos principales de la aplicación con los componentes de la biblioteca VCL.

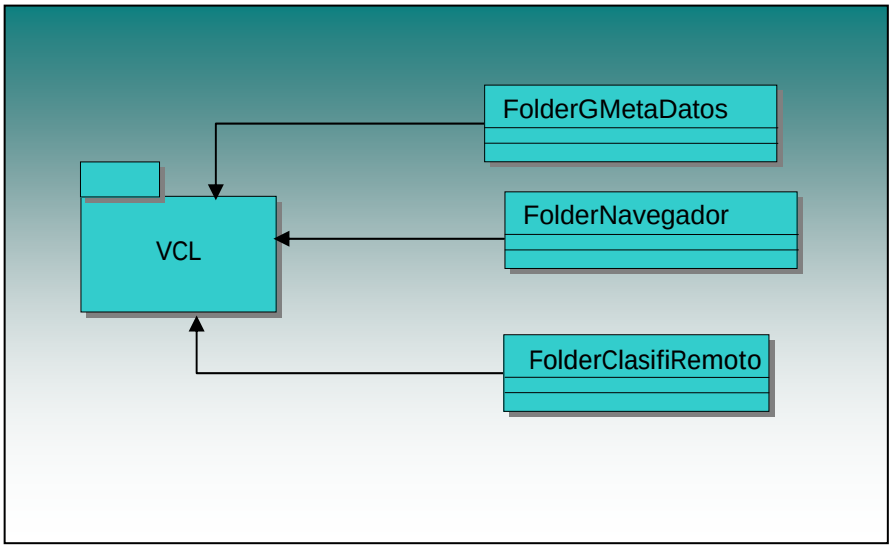


Figura 3.4 Modelo de diseño preliminar

Un aspecto en el diseño de la arquitectura es la definición y selección de los mecanismos arquitecturales, que son usados para dar vida a los objetos de la aplicación. Un mecanismo en el modelo de diseño es una clase, ó un

conjunto de clases, ó un patrón que constituye una solución común a un problema común. Algunos ejemplos de estos mecanismos son; la persistencia, intercambio de información, características de seguridad, redundancia, reporte de errores, conversión de formatos, interfaces de usuario gráficas, administración de transacciones y administración del sistema. En la Figura 3.5 se muestra los mecanismos de análisis considerados para el desarrollo de la aplicación a desarrollar en esta tesis.

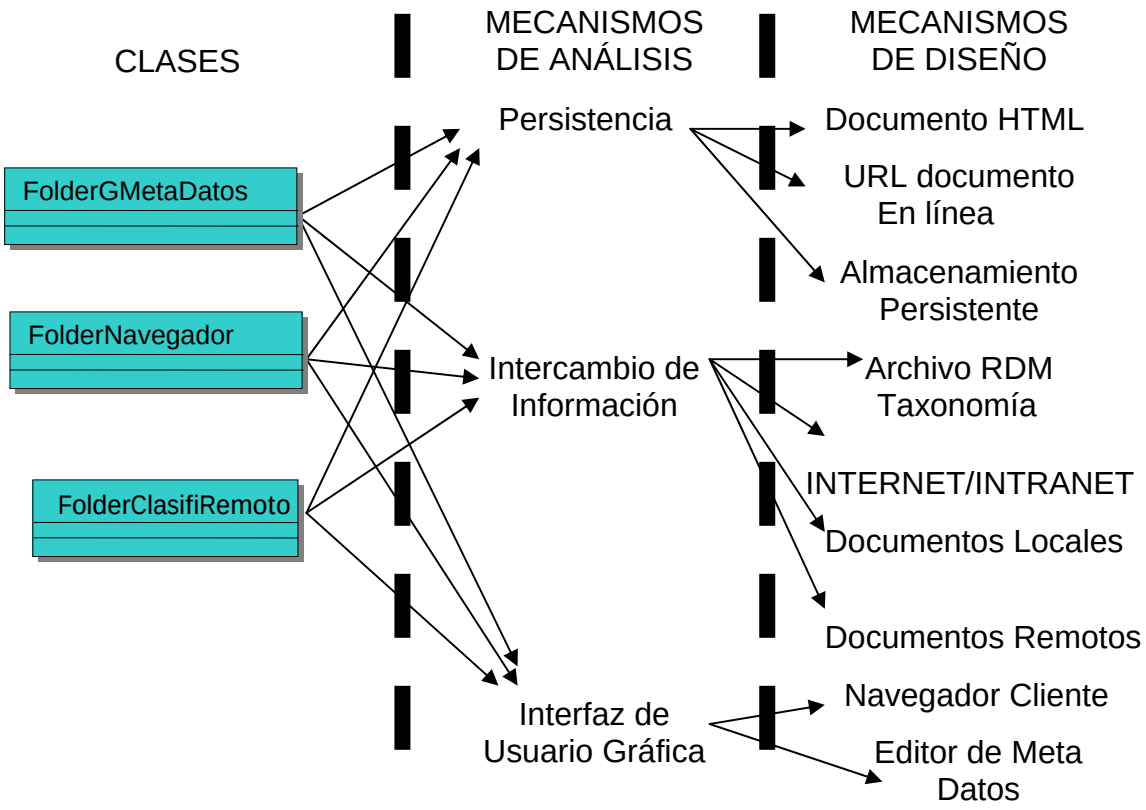


Figura 3.5 Mecanismos de análisis considerados en la aplicación a desarrollar

Considerando que un buen modelo de arquitectura debe cumplir con los requerimientos del sistema para que pueda soportar cambios en el proceso de implementación y permitir un mantenimiento claro y sencillo, ya que la arquitectura es un modelo adaptado al ambiente de implementación y sirve como una abstracción del código fuente. El modelo de arquitectura propuesto

para este trabajo de tesis es el mostrado en la Figura 3.6. Este modelo muestra un diagrama de jerarquía de paquetes junto con las clases ó realización de los casos de uso definidos en los requerimientos del capítulo anterior y de los servicios como una abstracción del modelo de implementación y su código fuente.

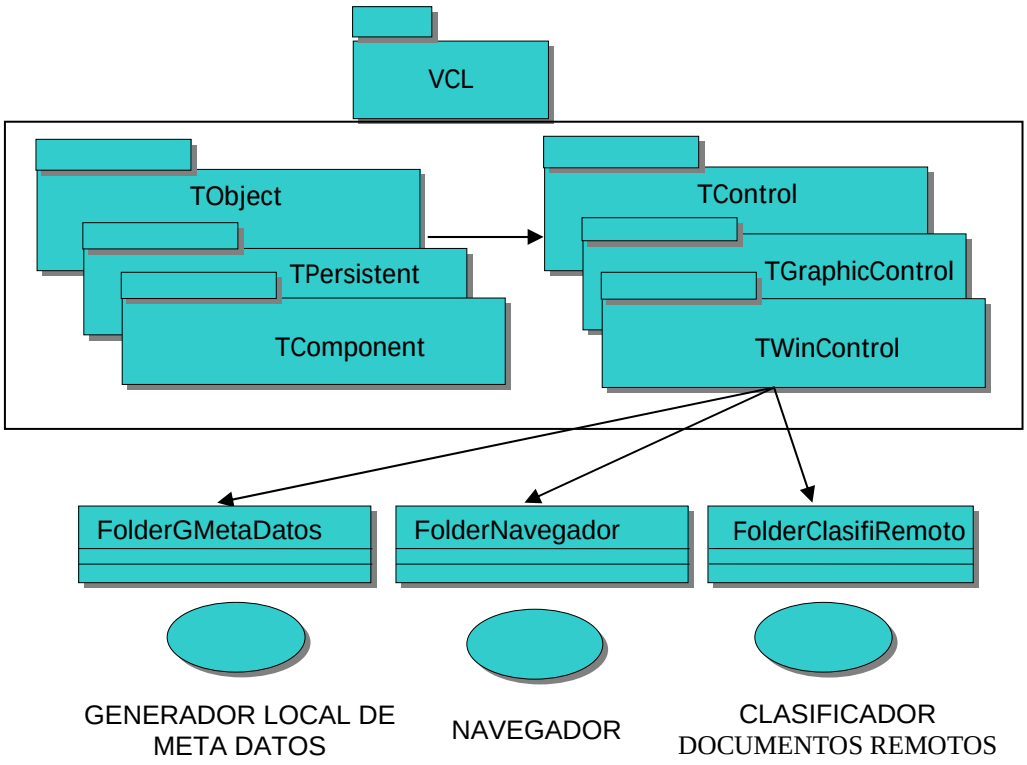


Figura 3.6 Modelo de diseño de la aplicación desarrollada en esta tesis

El diseño de este modelo debe ser orientado al modelo de implementación dependiendo de cómo se estructuran las clases y paquetes principales que componen el desarrollo del código de implementación.

3.2 Análisis de Clases de Objetos

El análisis de Clases de Objetos está basado en la realización de los casos de uso, que describe como estos casos de uso se ejecutan dentro del

modelo de diseño en términos de objetos de colaboración. Para cada realización de caso de uso hay uno ó más diagramas de clases que contiene las clases y objetos que participan en la ejecución del caso de uso. En la Figura 3.7 se muestra la relación entre los requerimientos y el proceso de análisis y diseño de cada caso de uso.

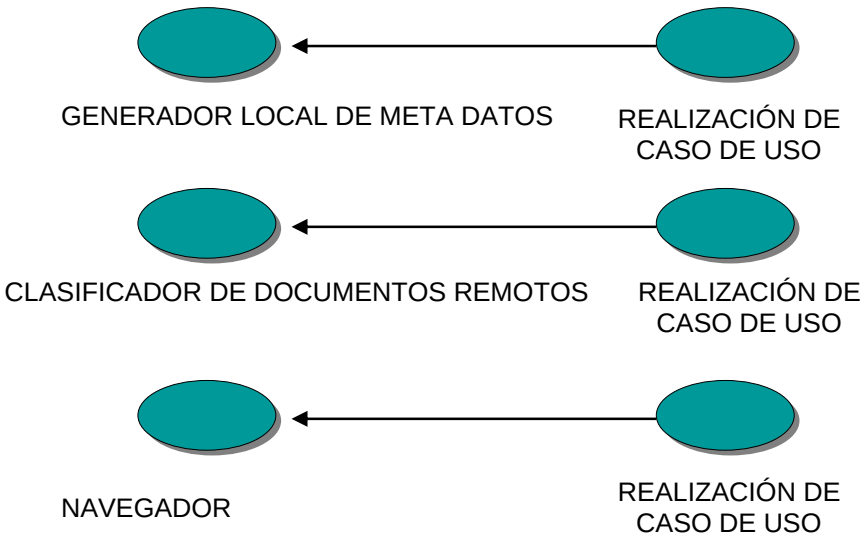


Figura 3.7 Realización de casos de uso

La descripción de este análisis es la base fundamental para realizar el diseño de clases de objetos a través de los diagramas de interacción: el diagrama de secuencias y el diagrama de colaboración. El análisis de esta sección se basa en la identificación y definición de clases para cada realización de los casos de uso.

El análisis de clases y objetos es realizado con la identificación y definición de las responsabilidades, relaciones y atributos de cada clase, basándose en los requerimientos de cada caso de uso. A continuación se describen los roles de participación de cada clase definida, que permiten el diseño del flujo de eventos de la realización de cada caso de uso. Las

responsabilidades de cada clase se definen considerando los siguientes conceptos:

El rol de la clase en el sistema describe cuando sus objetos son creados, usados y eliminados.

Las acciones que los objetos pueden ejecutar.

El conocimiento que los objetos de la clase mantienen y comunican a otros objetos.

Y como los objetos de la clase reaccionan a acciones inesperadas ó a eventos de entrada erróneos de las clases de interfaz de usuario.

3.2.1 Realización de Caso de Uso “Generador Local de Meta Dato”

El diagrama de clases definido para la realización de este caso de uso es mostrado en la Figura 3.8, donde están indicadas las clases de control, las clases de interfaz de usuario y las clases de entidad participantes.

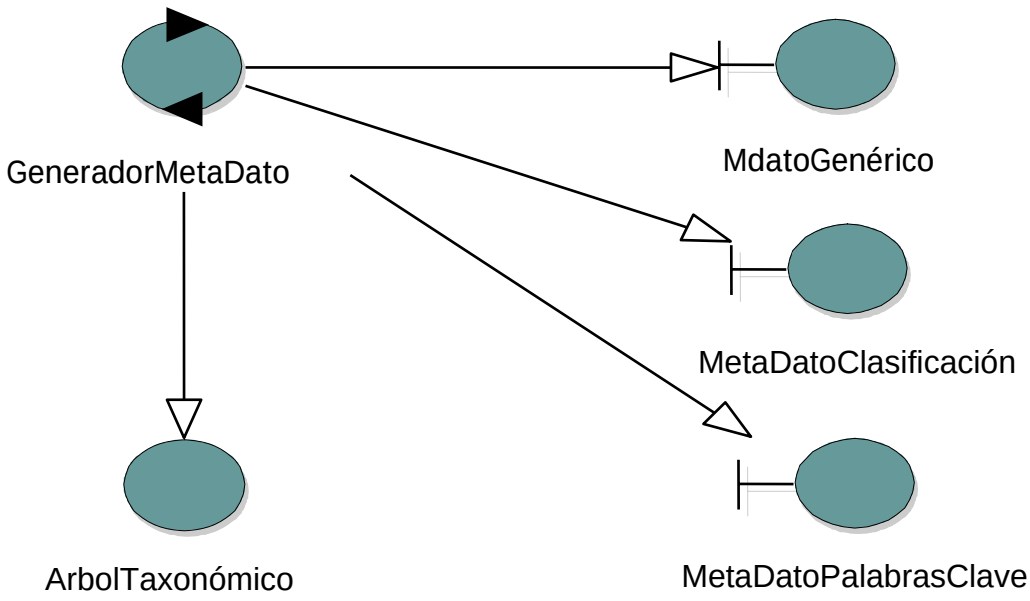


Figura 3.8 Diagrama de clases de realización del caso de uso “Generador Local de Meta Dato”

A continuación son descritos los roles de cada clase contenida en el diagrama de clases de Realización del Caso de Uso “Generador Local de Meta Dato”.

Rol de la Clase “GeneradorMetaDato”

Un objeto de la clase “GeneradorMetaDato” es creado cuando el especialista, a través de la clase de interfaz de usuario correspondiente, selecciona un tipo de Meta Dato a generar. El objeto toma el tipo de meta dato y realiza las tareas pertinentes para generar el Meta Dato seleccionado. Este objeto determina si es el caso, cargar un archivo tipo RDM (árbol de taxonomía) y controlar las interacciones correspondientes con éste. El objeto es eliminado una vez generado el Meta Dato solicitado.

Rol de la Clase “MdatoGenérico”

Un objeto de la clase “MdatoGenérico” es creado cuando el usuario selecciona alguno de los Meta Datos genéricos (título, autor, descripción, fecha de edición, generador y lenguaje). El objeto permitirá abrir un documento HTML, desplegar el formulario de captura para estos Meta Datos y enviar los datos capturados a la clase “GeneradorMetaDato”. Este objeto permitirá al especialista editar el código del documento HTML donde los Meta Datos son colocados. El objeto es eliminado cuando el especialista selecciona otro tipo de Meta Dato, si selecciona otro documento HTML el objeto persistirá.

Rol de la Clase “MetaDatoClasificación”

Un objeto de la clase “MetaDatoClasificación” es creado cuando el usuario selecciona el Meta Dato de clasificación. El objeto permitirá abrir un

documento HTML, desplegar la información recibida de la clase "ArbolTaxonómico" para que el especialista seleccione los tópicos que requiera. El objeto envía los datos capturados a la clase "GeneradorMetaDato". Este objeto permitirá al especialista editar el código del documento HTML donde los Meta Datos son colocados El objeto es eliminado cuando el especialista selecciona otro tipo de Meta Dato, si selecciona otro documento HTML u otra taxonomía el objeto persistirá.

Rol de la Clase "MetaDatoPalabrasClave".

Un objeto de la clase "MetaDatoPalabrasClave" es creado cuando el usuario selecciona el Meta Dato de palabras clave. El objeto permitirá abrir un documento HTML y desplegar una ventana donde el usuario podrá indicar la frecuencia de repetición que requiere para la búsqueda de palabras clave en el Documento HTML abierto. El objeto envía los datos capturados a la clase "GeneradorMetaDato". El objeto desplegará la lista de palabras clave encontradas y contendrá las palabras clave que el usuario seleccione. Este objeto permitirá al especialista editar el código del documento HTML donde los Meta Datos son colocados. El objeto es eliminado cuando el especialista selecciona otro tipo de Meta Dato, si selecciona otro documento HTML el objeto persistirá.

Rol de la Clase "ArbolTaxonómico"

Un objeto de la clase "ArbolTaxonómico" es creado cuando un objeto de la clase "MetaDatoClasificación" solicita abrir un archivo tipo RDM (taxonomía). El objeto despliega el árbol taxonómico contenido en el archivo abierto. El objeto es eliminado cuando la clase "MetaDatoClasificación" solicita una nueva taxonomía.

3.2.2 Realización de Caso de Uso “Navegador”

El diagrama de clases definido para la realización de este caso de uso es mostrado en la Figura 3.9, donde están indicadas las clases de control, las clases de interfaz de usuario y las clases de entidad participantes.

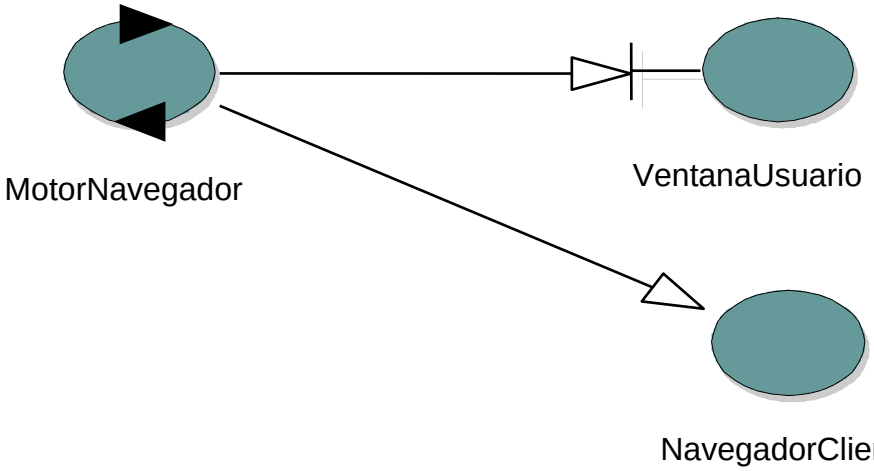


Figura 3.9 Diagrama de clases de realización de caso de uso “Navegador”

A continuación son descritos los roles de cada clase contenida en el diagrama de clases de Realización del Caso de Uso “Navegador”.

Rol de la Clase “MotorNavegador”

Un objeto de la clase “MotorNavegador” es creado cuando el especialista solicita una ventana de navegación. El objeto toma el control del navegador del cliente. El objeto es eliminado cuando el usuario cierra la ventana de navegación.

Rol de la Clase “VentanaUsuario”

Un objeto de la clase “VentanaUsuario” es creado cuando el especialista solicita una ventana de navegación y la clase “MotorNavegador” habilita el

servicio. El objeto despliega la información recibida en una ventana de la aplicación. El objeto es eliminado cuando el usuario cierra la ventana de navegación.

Rol de la Clase “NavegadorCliente”

Un objeto de la clase “NavegadorCliente” es creado cuando un objeto de la clase “MotorNavegador” solicita el servicio de navegación. El objeto es el navegador del cliente. El objeto es eliminado cuando el usuario cierra la ventana de navegación.

3.2.3 Realización de Caso de Uso “Clasificador de Documentos Remotos”

El diagrama de clases definido para la realización de este caso de uso es mostrado en la Figura 3.10, donde están indicadas las clases de control, las clases de interfaz de usuario y las clases de entidad participantes.

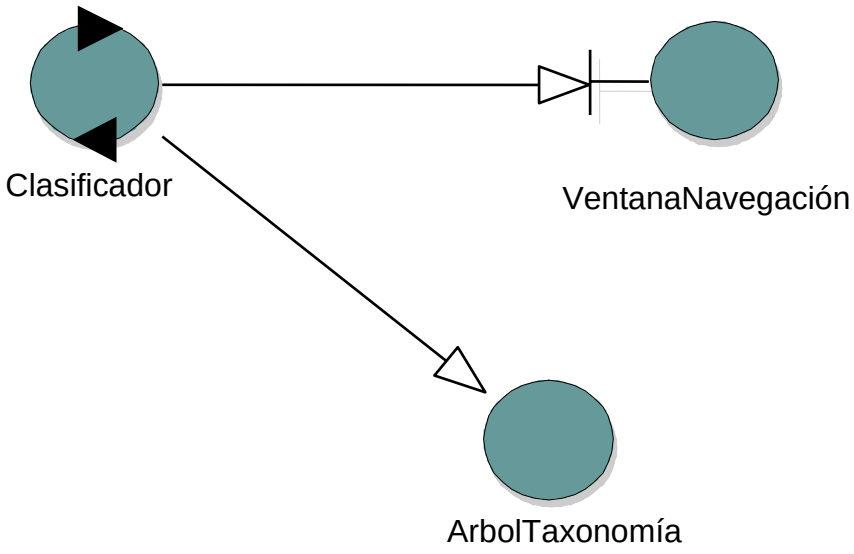


Figura 3.10 Diagrama de clases de realización de caso de uso “Clasificador de Documentos Remotos”

A continuación son descritos los roles de cada clase contenida en el diagrama de clases de Realización del Caso de Uso “Clasificador de Documentos Remotos”.

Rol de la Clase “Clasificador”

El objeto de la clase “Clasificador” es creado cuando un objeto de la clase “VentanaNavegación” solicita registrar alguna referencia de URL en el clasificador remoto. El objeto toma todo el control de enlace entre la referencia de URL y los objetos creados en la clase “VentanaNavegación” es decir con la ventana de navegación de la aplicación. El objeto realiza el proceso de clasificación de acuerdo a la información proporcionada por los objetos de la clase “ArbolTaxonomía” y genera el archivo de salida con las referencias clasificadas. El objeto toma las funciones de abrir archivo de referencias clasificadas y solicita instancias a la clase “VentanaNavegación” para efectuar la consulta de referencia URL. El objeto realiza las tareas de adición de nuevo tema ó eliminación de tema al árbol taxonómico así como la generación del archivo de taxonomía “RDM” compatible con el Robot de Búsqueda. El objeto es eliminado cuando se solicita el registro de una nueva referencia de URL ó el usuario cierra la ventana de navegación.

Rol de la Clase “VentanaNavegación”

El objeto de la clase “VentanaNavegación” es creado cuando el especialista solicita una ventana de navegación y la clase “MotorNavegador” habilita el servicio. El objeto despliega la información recibida en el navegador del cliente, en una ventana de la aplicación. El objeto es eliminado cuando el usuario cierra la ventana de navegación.

Rol de la Clase “ArbolTaxonomía”

Un objeto de la clase “ArbolTaxonomía” es creado cuando un objeto de la clase “Clasificador” solicita abrir un archivo tipo RDM (taxonomía). El objeto envía la información del árbol de taxonomía a la clase “Clasificador”. El objeto es eliminado cuando la clase “Clasificador” solicita una nueva taxonomía.

3.3 Diagramas de Secuencias

Los diagramas de secuencias clarifican como los casos de uso son procesados en función de objetos e interacciones entre estos, y definiendo los requerimientos de las operaciones en que los objetos deben soportarse.

Para cada realización de caso de uso se incorporan los mecanismos de diseño revisados en la sección de análisis y diseño de arquitectura de este capítulo a través de la interacción de objetos descritos en los diagramas de secuencias para cada caso de uso definido y mostrados en las Figuras 3.11, 3.12 y 3.13.

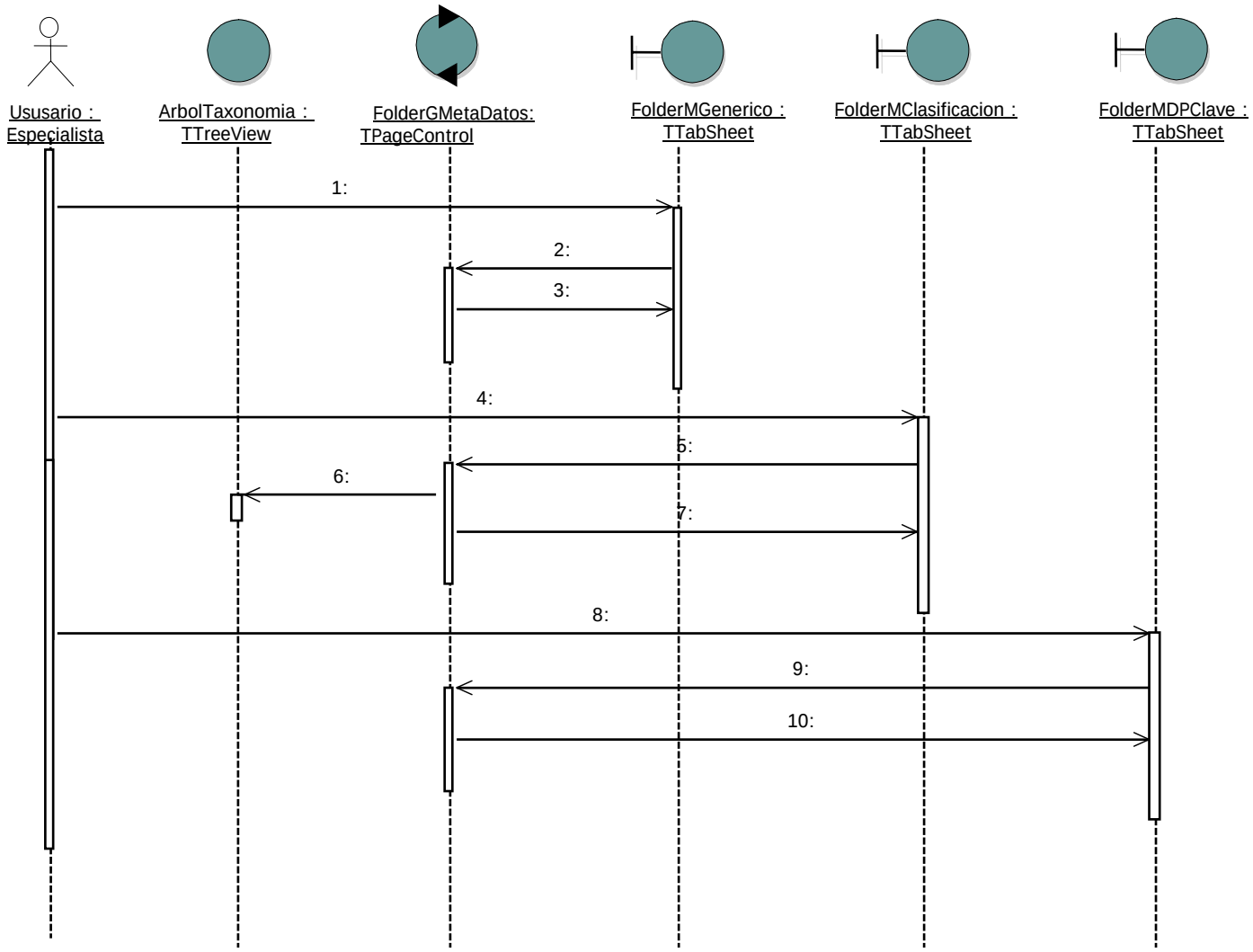


Figura 3.11 Diagrama de secuencias para la realización del caso de uso "Generador Local de Meta Dato"

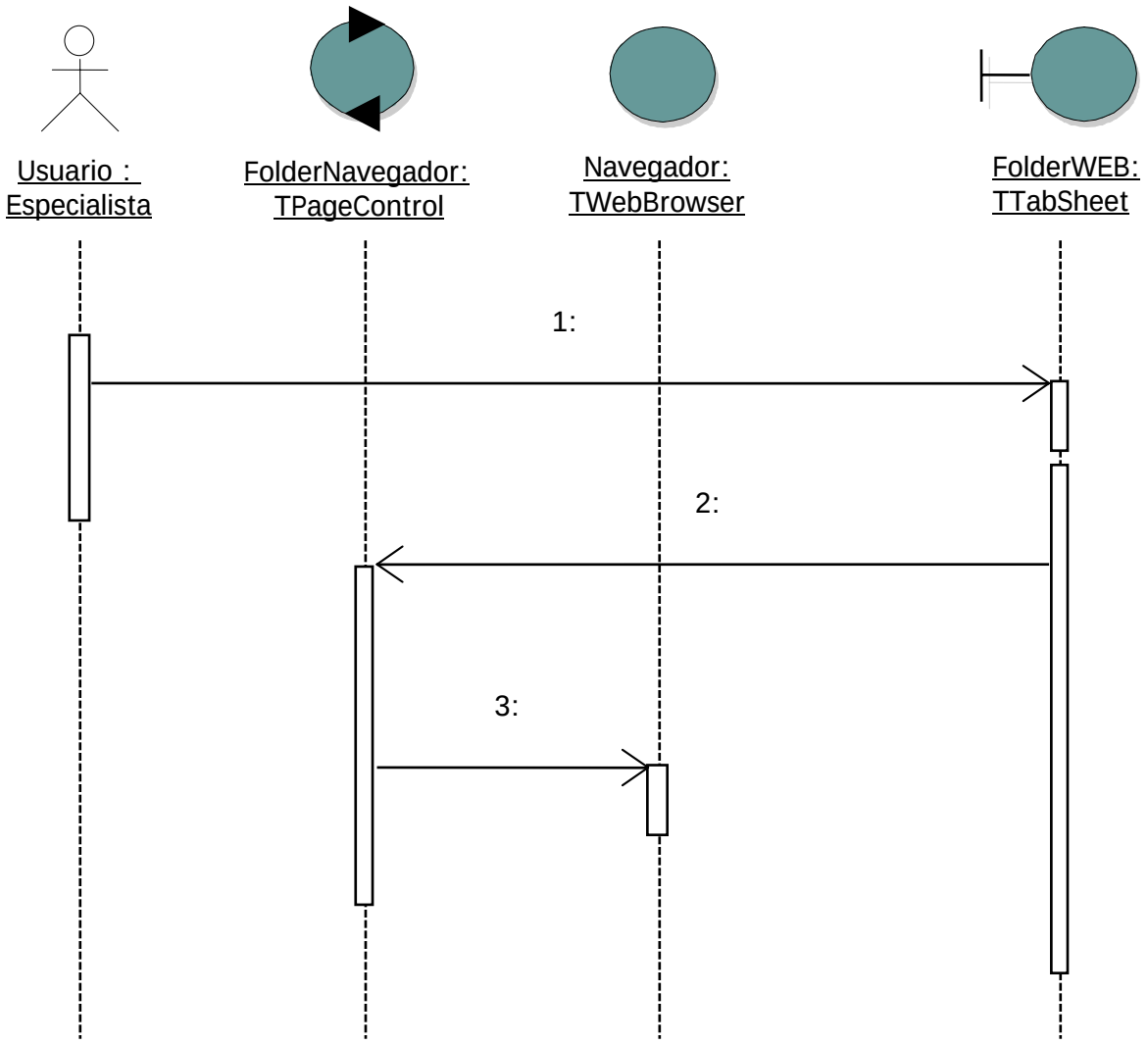


Figura 3.12 Diagrama de secuencias para la realización del caso de uso “Navegador”

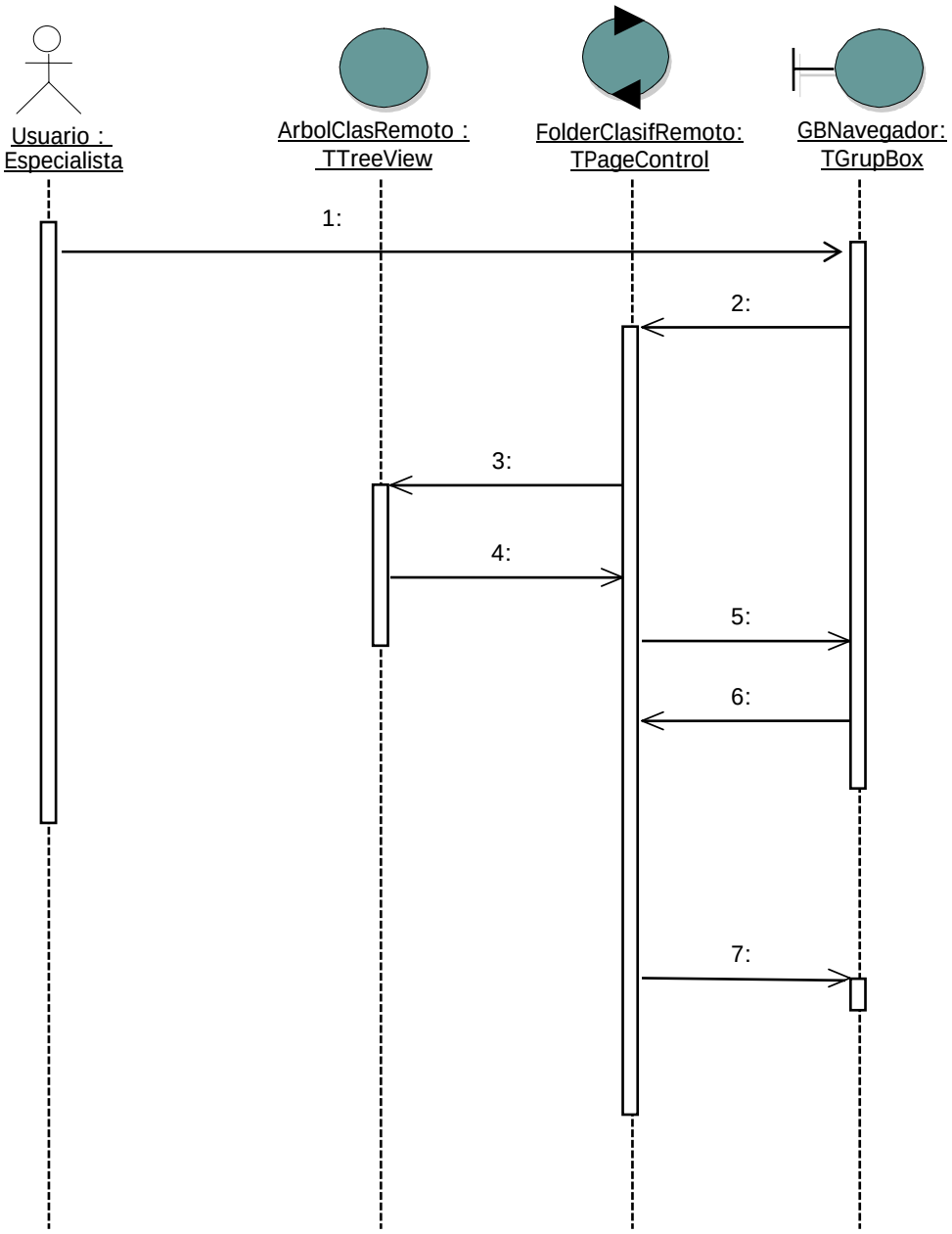


Figura 3.13 Diagrama de secuencias para la realización del caso de uso "Clasificador de Documentos Remotos"

De igual manera y a diferencia de los diagramas de secuencias, están los diagramas de colaboración que muestran las relaciones entre los objetos y que son de importancia en la etapa de diseño, porque clarifican los roles de los objetos en sus flujos de ejecución y determinan las interfaces y responsabilidades de las clases.

Las Figuras 3.14, 3.15 y 3.16 muestran los diagramas de colaboración para cada caso de uso. De esta forma el diseño de la aplicación a desarrollar en este trabajo de tesis es definido totalmente.

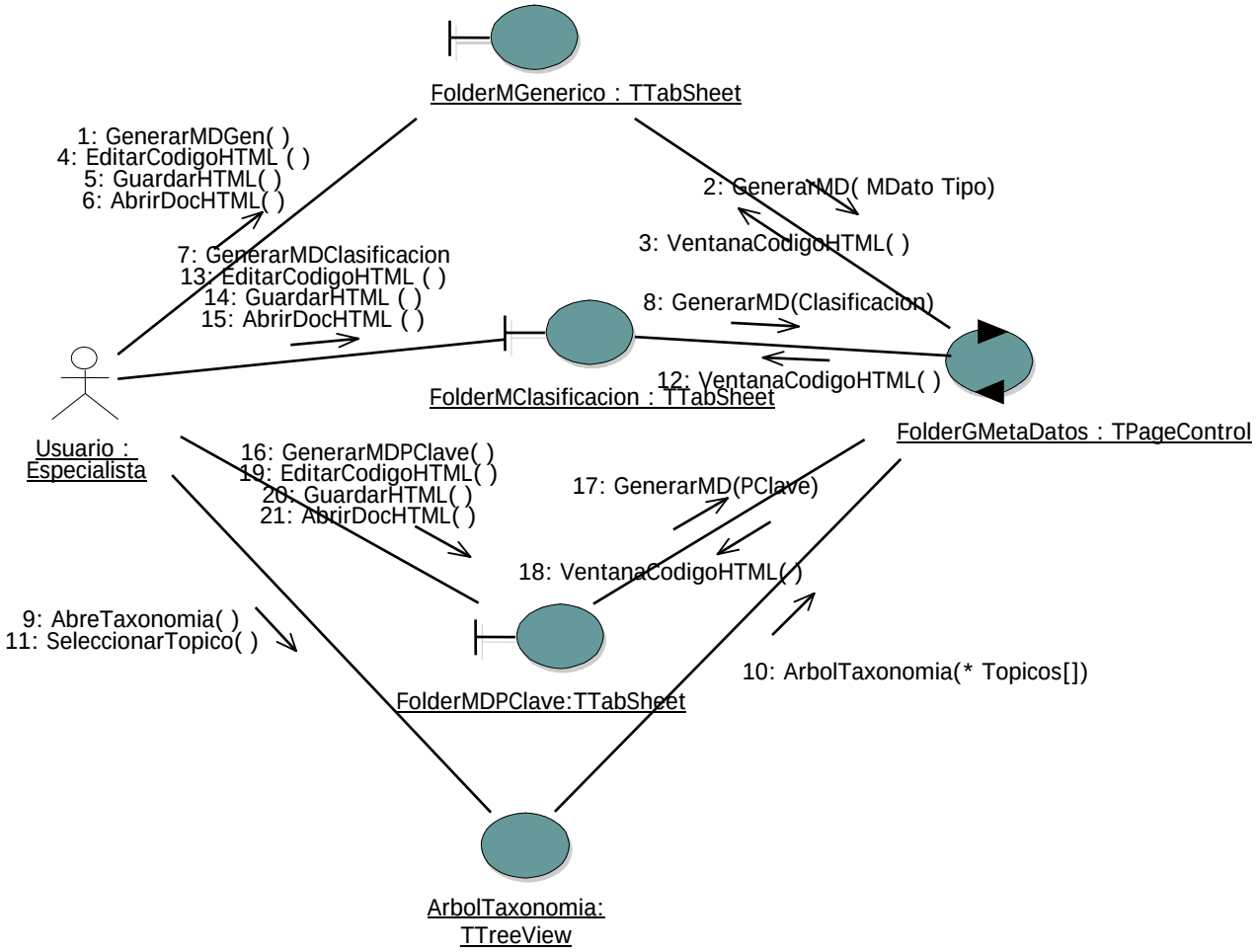


Figura 3.14 Diagrama de colaboración para la realización del caso de uso "Generador Local de Meta Dato"

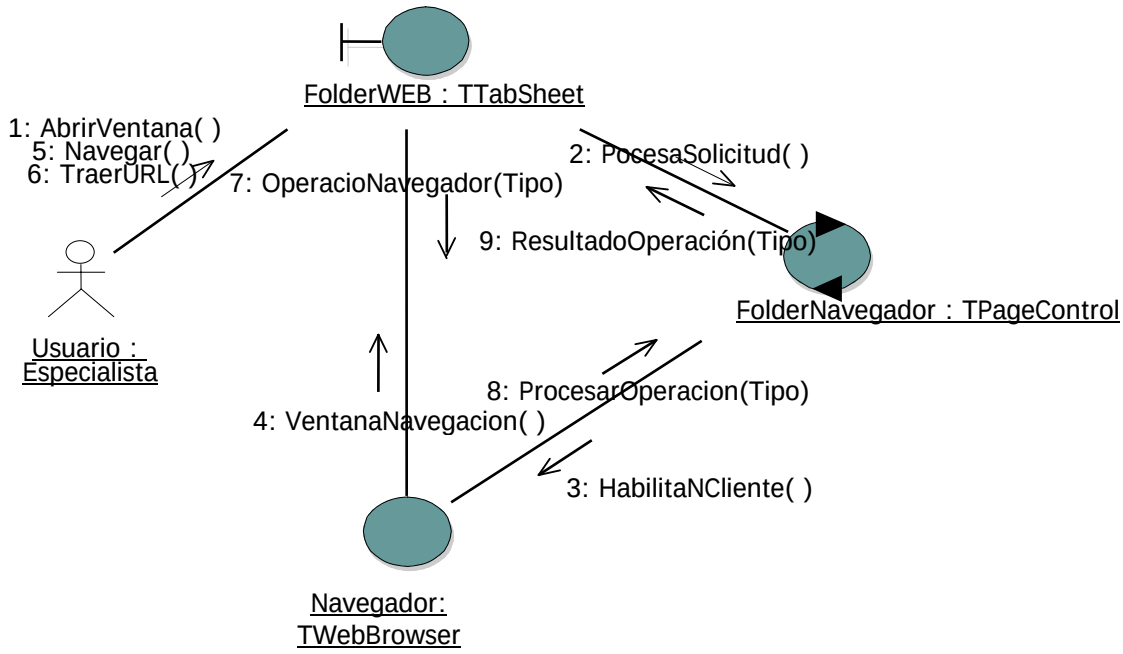


Figura 3.15 Diagrama de colaboración para la realización del caso de uso "Navegador"

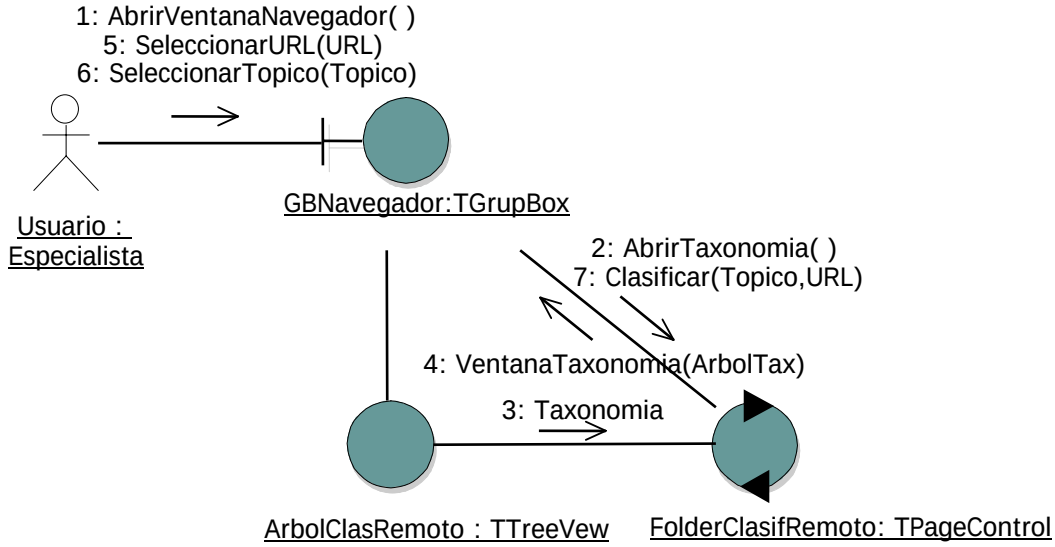


Figura 3.16 Diagrama de colaboración para la realización del caso de uso "Clasificador de Documentos Remotos"

Resumen

En este capítulo se realizó el análisis y diseño donde se definió un modelo de diseño describiendo la realización de cada caso de uso en términos de clase y sus objetos, también se diseñaron los diagramas de secuencias y de colaboración para cada realización de caso de uso, se identifican y describen las responsabilidades, relaciones y atributos de cada clase basándose en los requerimientos del modelo de casos de uso. En el siguiente capitulo se lleva a cabo la implementación de este modelo de diseño en términos de código.

CAPÍTULO 4
IMPLEMENTACIÓN DEL CLASIFICADOR DE DOCUMENTOS

En este capítulo se realiza la etapa de implementación del código en función a los requerimientos definidos en el modelo de casos de uso y en el modelo de diseño, el propósito en este capítulo es definir la organización del código en términos de subsistemas de implementación organizados en niveles, la implementación de clases de objetos en términos de componentes, probar los componentes desarrollados en unidades e integrar los componentes en un sistema ejecutable. La herramienta de desarrollo utilizada en la implementación es el entorno de programación de "C++ Builder de Borland".

4.1 Entorno de desarrollo

La tecnología actual en materia de entornos de desarrollo es muy amplia dentro del cambiante entorno de la industria del software actual, por esta razón no es muy fácil elegir alguna, sin embargo de acuerdo al tipo de aplicación es posible delimitar de acuerdo a las necesidad técnicas del desarrollo. La primera consideración planteada por parte de los usuarios de este desarrollo es tener una aplicación para Windows y orientada a objetos, considerando esto se reviso la tecnología disponible que ofrece los siguientes productos: La biblioteca de objetos de Windows (OWL), las clases fundamentales de Microsoft (MFC) y la biblioteca de componentes visuales de Borland (VCL). Muchos programas desarrollados para Windows están escritos en lenguaje C pero requiere del manejo de un conjunto muy amplio de funciones (API de Windows), los desarrolladores de Borland pensaron que había una manera más fácil de trabajar, de hecho la revolución en este

sentido se inició en varios frentes, pero sin duda en la programación en lenguaje C++ Borland fue uno de los lideres en este campo, quien propuso su biblioteca de componentes visuales (VCL). De aquí la necesidad de seleccionar entornos de desarrollo con algún compilador de C++, pero algunos solo ofrecen desarrollo de aplicaciones con MFC de Microsoft ó OWL de Borland para la programación orientada a componentes, hoy en día estas bibliotecas dejan de usurase con mayor rapidez ya que la programación orientada a componentes toma cada vez más fuerza. En 1995 Borland lanzo su primer producto de entorno de desarrollo orientado a componentes “Delphi” basado en la utilización de la biblioteca de componentes visuales (VCL) y posteriormente lanzo el de “C++ Builder” que ofrece un ambiente de desarrollo avanzado y orientado a componentes. Por esta razón la herramienta de desarrollo seleccionada para la etapa de implementación de este trabajo de tesis fue “C++ Builder de Borland” con su VCL.

4.2 Organización de subsistemas del Clasificador de Documentos

La organización de subsistemas se describe a través del modelo de implementación (ver Figura 4.1) que muestra el conjunto de componentes contenidos en cada subsistema. La identificación de niveles en el modelo de implementación es sobre la base del modelo de diseño desarrollado en él capítulo anterior, que sirve como entrada para mapear uno a uno el modelo de implementación.

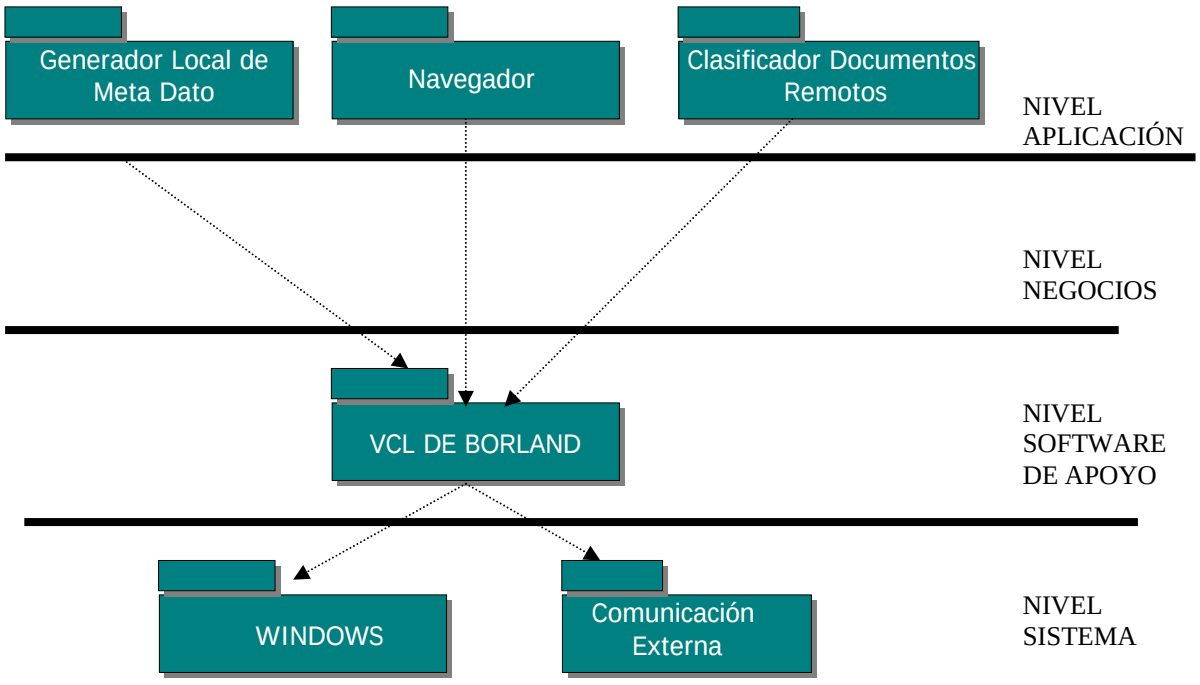


Figura 4.1 Modelo de implementación.

4.3 Implementación de Clases

La implementación de las clases se realiza a través de la biblioteca de componentes visuales en el ambiente de desarrollo “Builder C++ de Borland”. La implementación se basó en la clase TForm que es el componente contenedor utilizado para aplicaciones de Windows, esta clase instancia un objeto Forma que es donde se colocan los componentes de las clases de objetos que se requieran para la aplicación Windows.

La aplicación que se desarrolló en este trabajo de tesis se compone de cuatro clases de objetos TForm, que conforman el programa principal del Meta Generador. La figura 4.2 muestra la implementación del programa principal de la aplicación.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("MetaGenerador.res");
USEFORM("Entrada.cpp", FrmEntrada);
USEFORM("GLocalMetaDato.cpp", FrmGenMD);
USEFORM("Acerca_De.cpp", AcercaDe);
USEFORM("BarraAvance.cpp", BusquedaPC);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
   try
   {
      Application->Initialize();
      Application->CreateForm(__classid(TFrmEntrada), &FrmEntrada);
      Application->CreateForm(__classid(TFrmGenMD), &FrmGenMD);
      Application->CreateForm(__classid(TAcercaDe), &AcercaDe);
      Application->CreateForm(__classid(TBusquedaPC), &BusquedaPC);
      Application->Run();
   }
   catch (Exception &exception)
   {
      Application->ShowException(&exception);
   }
   return 0;
}
//---------------------------------------------------------------------------
```

Figura 4.2 Programa principal del Meta Generador

La clase “TFrmEntrada” implementa la ventana de entrada a la aplicación, la clase “TAcercaDe” implementa la ventana de diálogo para los créditos del desarrollo, la clase “TBusquedaPC” implementa una ventana auxiliar de diálogo para indicar el avance en la búsqueda de las palabras

clave entre otros datos y por último la clase principal de la aplicación “TFrmGenMD” que implementa la ventana principal del Meta Generador.

Los objetos del nivel de aplicación son instanciados a través de esta clase “TFrmGenMD” que contiene los diferentes componentes utilizados y creados en la forma instanciada por esta clase. A continuación se muestra la implementación de las clases de objetos del nivel de aplicación por caso de uso.

4.3.1 Caso de Uso “Generador Local de Meta Datos”

Las clases principales para este caso de uso se derivan de la Clase TwinControl, que contiene los componentes que instancian los objetos que interactúan y colaboran para dar la funcionalidad de este caso de uso. La implementación de este caso de uso se basa en la clase “FoldersGMetaDatos” que se deriva de la clase “TPageControl” que despliega múltiples páginas traslapadas las cuales son objetos instanciados de la clase “TTabSheet”, estas clases son:

FolderMGenerico:TtabSheet
FolderMClasificacion:TTabSheet
FolderMDPClave:TtabSheet

Los componentes que se utilizan para dar la funcionalidad de este caso de uso están creados a través de tres clases derivadas de la clase “TPanel” que opera como contenedor de las clases que instancian a todos los objetos requeridos para la funcionalidad diseñada, éstas se indican en la figura 4.3 conjuntamente con las clases que instancian todos los objeto utilizados.

PanelMDGenerico:TPanel
- GBoxGenericos:TGrupBox
  - ETitulo:TEdit
  - EAutor:TEdit
  - EGenerador:TEdit
  - ComboLenguaje:TEdit
  - EDescripcion:TEdit
- BotonAbrirHTMLmg:TButton
- BotonCrearMGen:TButton
- BotonEditarHTMLFinal:TButton
- BotonGuardarHTMLpg:TButton
- HTMLMGenericos:TWebBrowser
- MemoMetasGEnc:TMemo
- MemoGenericos:Tmemo

PanelMDClasificacion:TPanel
- ArbolTaxonomia:TTreeView
- HTML:TWebBrowser
- ListaTopicos:TListBox
- MemoHTMLS:TMemo
- MemoMetas:TMemo
- BotonAbrirHTML:TButton
- BotonCrearMDC:TButton
- BotonEditarHTML:TButton
- BotonGuardarHTMLclas:Tbutton

PanelMDPclave:TPanel
- BotonAbrirHTMLpc:TButton
- BotonGenerarMPC:TButton
- BotonEditar:Tbutton
- BotonGuardar:TButton
- BotonBuscarPC:TButton
- MaskERepeticion:TMaskEdit
- ListaPCEncontradas:TListBox
- HTMLpc:TWebBrowser
- MemoMPCenc:TMemo
- MemoMetasPC:TMemo

Figura 4.3 Clases que implementan el caso de uso “Generador Local de Meta Datos”

En la figura 4.4 se muestra el diagrama de jerarquía de clases para la implementación de este caso de uso.

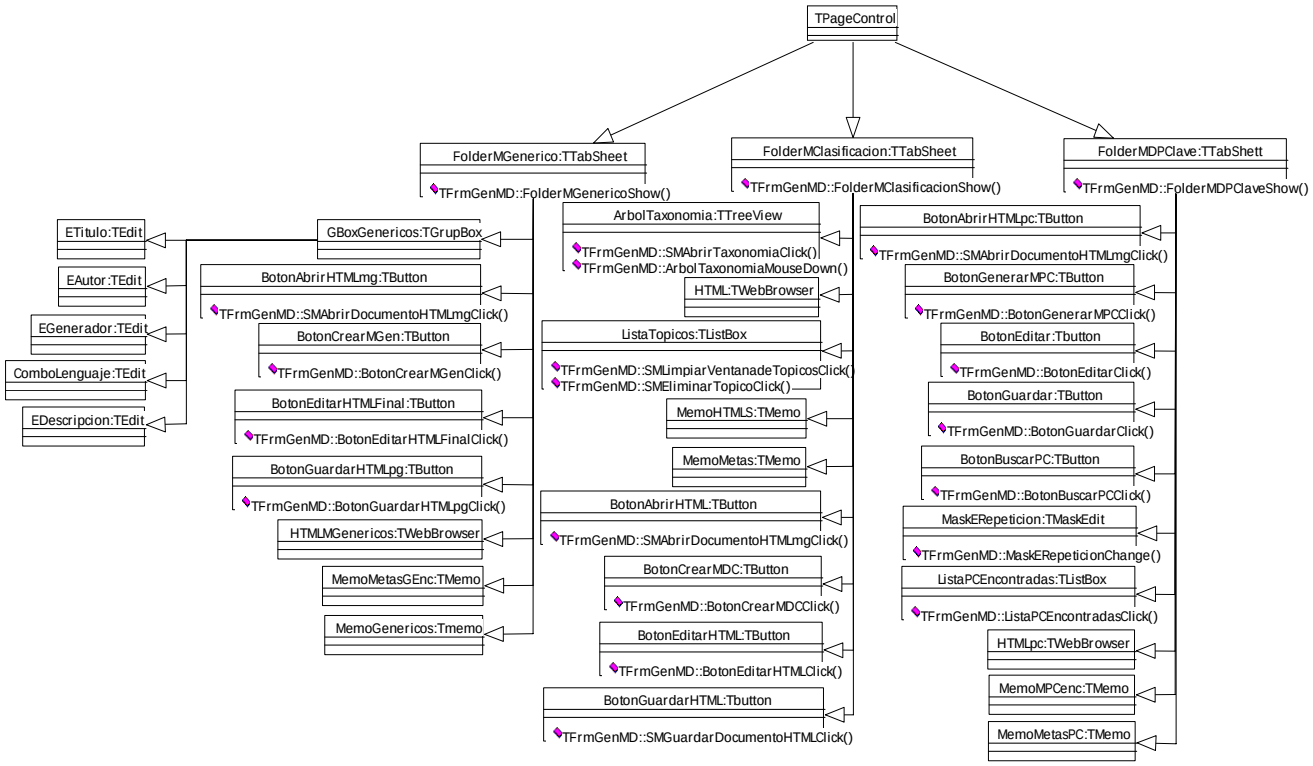


Figura 4.4 Diagrama de jerarquía de clases para la implementación de caso de uso “Generador Local de Meta Datos”

4.3.2 Caso de Uso “Navegador”

De igual manera las clases principales para este caso de uso se derivan de la Clase TwinControl, que contiene los componentes que instancian los objetos que interactúan y colaboran para dar la funcionalidad de éste caso de uso. La implementación de éste caso de uso se basa en la clase “FolderNavegador” que se deriva de la clase “TPageControl” que despliega una página traslapada “FolderWEB” la cual es un objeto heredado de la clase “TTabSheet”, esta clase es:

FolderWEB:TtabSheet

Los componentes que se utilizan para dar la funcionalidad de este caso de uso están creados a través de dos clases derivadas de la clase

"TComboBox" y de la clase "TwebBrowser" que instancian a todos los objetos requeridos para la funcionalidad diseñada, éstas se indican a continuación:

URLComboBox:TComboBox
Navegador:TWebBrowser

En la figura 4.5 se muestra el diagrama de jerarquía de clases para la implementación de este caso de uso.

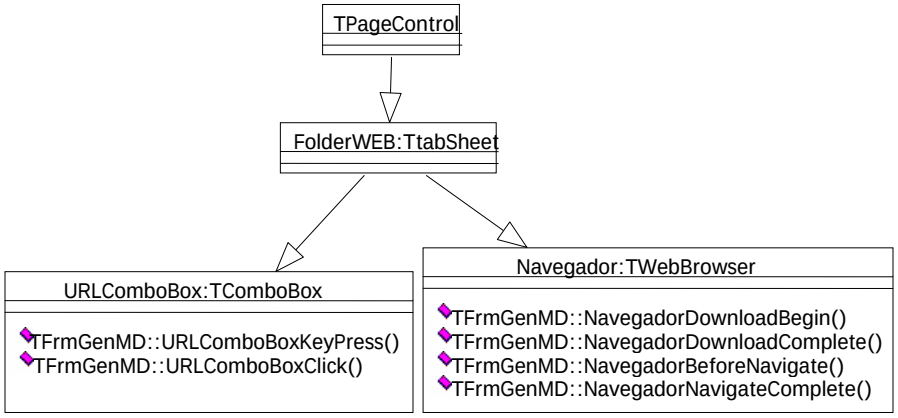


Figura 4.5 Diagrama de jerarquía de clases para la implementación de caso de uso "Navegador"

4.3.3 Caso de Uso "Clasificador de Documentos Remotos"

Del mismo modo las clases principales para éste caso de uso se derivan de la Clase TwinControl, que como ya se mencionó contiene los componentes que instancian los objetos que interactúan y colaboran para dar la funcionalidad a éste caso de uso. La implementación del caso de uso se basa en la clase "FolderClasifRemoto" que se deriva de la clase "TPageControl" que despliega una página traslapada "ClasificadorRemoto" "la cual es un objeto heredado de la clase "TTabSheet".

ClasificadorRemoto:TTabSheet

Los componentes que se utilizan para dar la funcionalidad de este caso de uso están creados a través de cuatro clases que instancian a todos los objetos requeridos para la funcionalidad diseñada, la primera clase deriva de la clase “TTreeView” que instancia un objeto ventana que despliega una lista jerárquica de temas, una segunda clase que opera como contenedor y que deriva de la clase “TPanel” y dos clases más que también operan como contenedores y que derivan de la clase “TGrupBox”. Éstas clases se muestran en la figura 4.6 conjuntamente con las clases que instancian a todos los objetos requeridos:

ArbolClasRemoto:TTreeView
PanelClasRemoto:TPanel
GBEdicionATax:TGrupBox
BotonAddNodo:TButton
EditURLdeNvg:TEdit
BotonElimNodo:TButton
BotonAddTema:TButton
BotonExpanderTax:TButton
BotonContraerTax:Tbutton

GBNavegador:TGrupBox
BotonAbrirHTMLNavdor:TButton
EditURLaNvg:TEdit
BotonExpTax:Tbutton

GBRespaldoTax:TGrupBox
BotonGuardarTaxD:TButton
BotonAbrirTaxD:TButton
BotonGenerarTax:TButton

Figura 4.6 Clases que implementan el caso de uso “Clasificador de Documentos Remotos”

En la figura 4.7 se muestra el diagrama de jerarquía de clases para la implementación de este caso de uso.

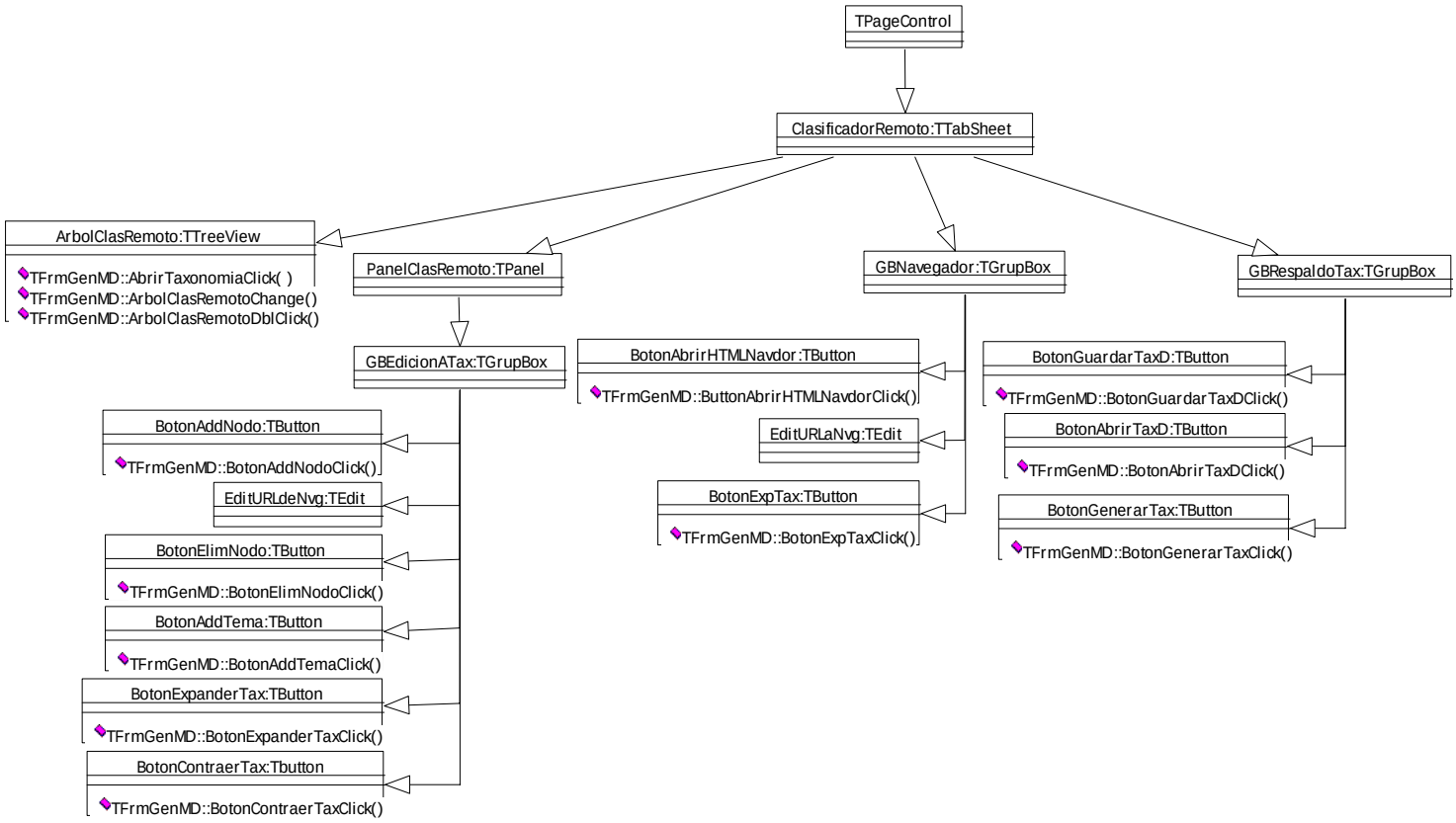


Figura 4.7 Diagrama de jerarquía de clases para la implementación de caso de uso “Clasificador de Documentos Remotos”

4.4 Integración de Funciones Miembro

Las funciones miembro son los métodos implementados para lograr la interacción y colaboración de todos los objetos instanciados en tiempo de ejecución y de ésta manera garantizar la funcionalidad diseñada para cada caso de uso. La parte más fuerte de la implementación está precisamente en el diseño y codificación de los métodos que conforman las funciones miembro de las clases implementadas.

Como se mencionó en el sección 4.3 de éste capítulo, la clase principal del Meta Generador es la clase “TFrmGenMD”, por esta razón los métodos implementados son funciones miembro de esta clase, y en estas

implementaciones se instancian todos los objetos de las clases de objetos diseñados para cada caso de uso.

4.4.1 Caso de Uso “Generador Local de Meta Datos”

Las funciones miembro que se implementan para el Generador Local de Meta Datos se indican a continuación:

```
void TFrmGenMD::SMAbrirDocumentoHTMLmgClick()
void TFrmGenMD::BotonCrearMGenClick()
void TFrmGenMD::BotonEditarHTMLFinalClick()
void TFrmGenMD::BotonGuardarHTMLpgClick()
void TFrmGenMD::SMAbrirTaxonomiaClick()
void TFrmGenMD::BotonCrearMDCClick()
void TFrmGenMD::BotonEditarHTMLClick()
void TFrmGenMD::SMGuardarDocumentoHTMLClick()
void TFrmGenMD::BotonBuscarPCClick()
void TFrmGenMD::BotonGenerarMPCClick()
void TFrmGenMD::BotonEditarClick()
void TFrmGenMD::BotonGuardarClick()
```

La función miembro “SMAbrirDocumentoHTMLmgClick()” abre el documento HTML de trabajo, y lo despliega en los tres objetos instanciados de la clase “TWebBrowser” para el Meta Dato genérico, clasificación y palabras clave, así como también para el objeto instanciado de la misma clase “Navegador:TWebBrowser” correspondiente al caso de uso “Navegador”. Después de abrir el documento HTML se copia el archivo a tres objetos instanciados de la clase “TMemo” donde será listado para su posterior tratamiento, en esta parte se determina si el documento abierto contiene Meta Datos genéricos, Meta Dato de clasificación y Meta Dato de palabras clave, si es el caso se listan los Meta Datos encontrados en tres objetos instanciados también de la clase “TMemo”.

La función “SMAbrirTaxonomiaClick()” abre un archivo tipo RDM el cual es leído, decodificado y transportado al objeto “ArbolTaxonomia” instanciado de la clase “TTreeView” donde es desplegado en forma de árbol de temas y manipulado de acuerdo a las propiedades y métodos heredados de la clase.

Las funciones “BotonCrearMGenClick()”, “BotonCrearMDCClick()” y “BotonGenerarMPCClick()” generan los Meta Datos genéricos, de clasificación y de palabras clave respectivamente, verificando los Meta Datos encontrados en el documento HTML leído para eliminarlos ó adicionar a éstos, los Meta Datos creados por el usuario. Estas funciones son invocadas en el evento “OnClick” de los objetos “BotonCrearMGen”, “BotonCrearMDC” y “BotonGenerarMPC” instanciados de la clase “TButton” respectivamente.

Otra función importante en la implementación de este caso de uso, es “BotonBuscarPCClick()” que realiza la búsqueda de palabras clave en el documento HTML leído y que es invocada del evento “OnClick” del objeto “BotonBuscarPC”. Esta función también instancia un objeto identificado como “BusquedaPC” de la clase “TForm”, que despliega una ventana para mostrar el avance y el tiempo estimado de la búsqueda. Después de terminada la búsqueda, las palabras clave encontradas se listan en el objeto “ListaPCEncontradas” instanciado de la clase “TListBox”.

Para mayor detalle de la implementación de estas funciones refiera al anexo “A” donde está el código completo de implementación.

4.4.2 Caso de Uso “Navegador”

Las funciones miembro que se implementan para el Navegador se indican a continuación:

```
void TFrmGenMD::URLComboBoxClick()
void TFrmGenMD::URLComboBoxKeyPress()
void TFrmGenMD::NavegadorBeforeNavigate()
void TFrmGenMD::NavegadorNavigateComplete()
```

La función “URLComboBoxClick()” abre el documento HTML que esté indicado en la línea de texto seleccionada, dicha función es invocada por el evento “OnClick” del objeto “URLComboBox” instanciado de la clase “TComboBox”.

La función “URLComboBoxKeyPress(char &Key)” abre el documento HTML que esté indicado por la dirección de URL escrita en la línea de texto después de oprimir la tecla de “Enter”, además de adicionar ésta dirección de URL a la lista de la caja de selección del objeto “URLComboBox”. Esta función es invocada por el evento “OnKeyPress” del objeto “URLComboBox” instanciado de la clase “TComboBox”, como se observa esta función recibe un parámetro para la tecla accionada por el usuario.

Las funciones que muestran el momento de conexión y cuando el documento se termina de cargar son “NavegadorBeforeNavigate()” y “NavegadorNavigateComplete()”, éstas son invocadas por los eventos “OnBeforeNavigate” y “OnNavigateComplete” respectivamente, del objeto “Navegador” instanciado de la clase “TWebBrowser”. Esta clase importa un componenete “ActiveX” del “Internet Explorer” de “MicroSoft “, de esta manera el navegador hereda las propiedades y métodos del componente “Internet Explorer”.

Para mayor detalle de la implementación de éstas funciones refiera al anexo “A” donde está el código completo de implementación.

4.4.3 Caso de Uso “Clasificador de Documentos Remotos”

Las funciones miembro que se implementan para el Clasificador de Documentos Remotos se indican a continuación:

```
void TFrmGenMD::AbrirTaxonomaClick()
void TFrmGenMD::BotonAbrirTaxDClick()
void TFrmGenMD::BotonExpanderTaxClick()
void TFrmGenMD::BotonContraerTaxClick()
void TFrmGenMD::BotonAddNodoClick()
void TFrmGenMD::BotonElimNodoClick()
void TFrmGenMD::BotonAddTemaClick()
void TFrmGenMD::ButtonAbrirHTMLNavdorClick()
void TFrmGenMD::BotonExpTaxClick()
void TFrmGenMD::BotonAbrirTaxDClick()
void TFrmGenMD::BotonGuardarTaxDClick()
void TFrmGenMD::BotonGenerarTaxClick()
```

La función “AbrirTaxonomaClick()” abre un archivo tipo RDM el cual es leído, decodificado y transportado al objeto “ArbolClasRemoto” instanciado de la clase “TTreeView” donde es desplegado en forma de árbol de temas y manipulado de acuerdo a las propiedades y métodos heradados de la clase.

La función “ArbolClasRemotoDblClick()” invocada por el objeto “ArbolClasRemoto”, resultado del evento “OnDblClick” para abrir en la ventana del objeto “Navegador” el documento cuya referencia “URL” es apuntada por el cursor de algún nodo del árbol taxonómico.

Las funciones “BotonExpanderTaxClick()” y “BotonContraerTaxClick()” son invocadas en el evento “OnClick” de los objetos “BotonExpanderTax” y “BotonContraerTax” instanciados de la clase “TButton”, ejecutando la expansión y contracción del árbol taxonómico respectivamente a través de

“ArbolClasRemoto->FullExpand()” y “ArbolClasRemoto->FullCollapse()”, métodos heredados de la clase “TTreeView”.

La función “ButtonAbrirHTMLNavdorClick()” es invocada en el evento “OnClick” del objeto “BotonAbrirHTMLNavdor” instanciado de la clase “TButton”, para abrir un documento cuya referencia URL está indicada en la caja de texto correspondiente ó como parte de la taxonomía, a través del objeto instanciado “Navegador”. Esta función permite consultar cualquier documento HTML referenciado a través de su dirección “URL” desde el árbol taxonómico cargado en el objeto instanciado “ArbolClasRemoto”.

La función “BotonExpTaxClick()” es invocada en el evento “OnClick” del objeto “BotonExpTax” instanciado de la clase “TButton”, que permite exportar el árbol taxonómico con las referencias “URL” clasificadas al objeto instanciado “Navegador” y desplegar el árbol taxonómico en formato de documento HTML para poder consultar cualquier documento HTML referenciado a través de su dirección “URL” desde el mismo navegador a diferencia de la función anteriormente descrita.

La función “BotonGenerarTaxClick()” junto con las funciones “BotonAddNodoClick()”, “BotonElimNodoClick()” y “BotonAddTemaClick()” permiten editar taxonomías y crear nuevas taxonomías generándolas en formato RDM.

Para mayor detalle de la implementación de estas funciones refiera al anexo “A” donde está el código completo de implementación.

Resumen

En este capítulo se realizó la codificación de las clases y sus métodos para dar la funcionalidad a cada caso de uso, utilizando como entorno de desarrollo el IDE “C++ Builder de Borland”. En el siguiente capítulo se desarrollan los casos de prueba para cada unidad de implementación aplicando la metodología de “Objectory Process”.

CAPÍTULO 5
PRUEBAS Y RESULTADOS DEL CLASIFICADOR DE DOCUMENTOS

En este capítulo se realizan las pruebas para verificar la interacción entre objetos, verificar la propia integración de todos los componentes del software desarrollado, verificar que todos los requerimientos han sido correctamente implementados e identificar posibles defectos. Las pruebas del software pueden significar el 30 ó 50% del costo total de desarrollo y en la mayoría de los casos, esta actividad es llevada acabo después del desarrollo, situación que se da por dos razones principales. Primero, las pruebas de software son extremadamente difíciles, ya que las diferentes opciones que pueden resultar en un software dado no son cuantificables. Segundo las pruebas son hechas sin una clara metodología. Para resolver esto, se aplicó la metodología propuesta en “Objectory Process” de “Rational” que indica implementar un ciclo de vida para las pruebas y que debe corresponder con el ciclo de vida del desarrollo del software. En este capítulo se muestra la definición de las pruebas de acuerdo a esta metodología, la ejecución de las mismas y la validación de resultados.

5.1 Definición y Ejecución de Pruebas

En el ciclo de vida del desarrollo del software, éste se va refinando en diferentes iteraciones, en este sentido el ciclo de vida de las pruebas también puede tener un enfoque iterativo donde el resultado esperado de las nuevas construcciones sea el objetivo de cada prueba. La figura 5.1 muestra el ciclo de vida de las pruebas.

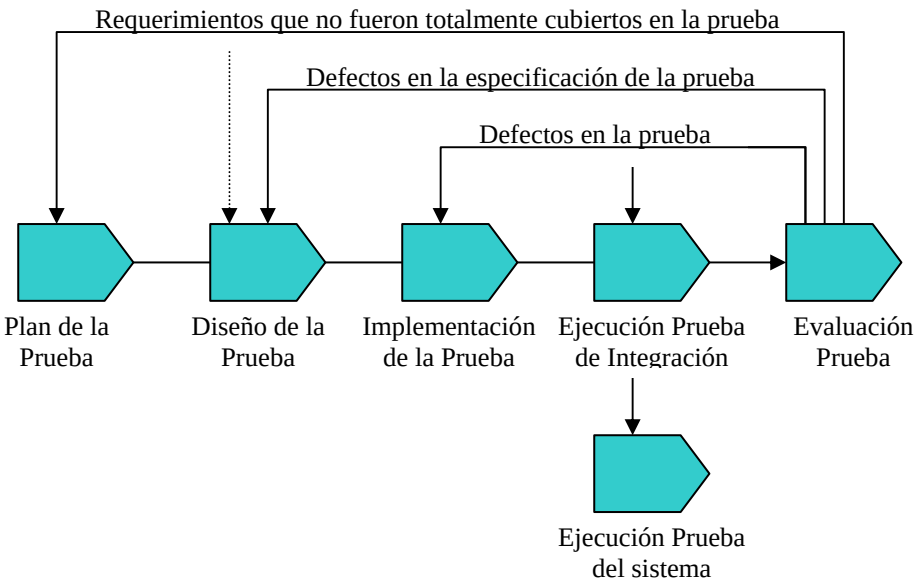


Figura 5.1 Ciclo de vida de prueba

Las pruebas que se definieron para el desarrollo del software del Meta Generador en función a esta metodología, se describen a continuación en el orden en que fueron desarrollados los diferentes componentes del software y de acuerdo a cada caso de uso.

5.1.1 Caso de Uso “Generador Local de Meta Dato”

Para el caso de uso “Generador Local de Meta Dato” en el módulo de Meta Dato Genérico como primer paso se define un plan de prueba donde se describe la estrategia para verificar todos y cada uno de los meta datos genéricos a generar; título, autor, generador, fecha de creación, lenguaje y descripción. En la tabla 5.1 se resumen, las acciones a tomar en el plan de la prueba, estos son la estrategia, los recursos a utilizar y el programa de actividades a desarrollar para validar los resultados arrojados por el programa Meta Generador para éste módulo.

Prueba del módulo Meta Dato Genérico

PLAN DE LA PRUEBA	
Introducción:	Generar un Meta Dato Genérico en un documento HTML
Estrategia	Seleccionar un Meta Dato genérico de cada tipo y generar el Meta en un documento HTML ejemplo, considerando: El documento HTML no contiene Meta Datos Genéricos El documento HTML contiene Meta Datos Genéricos y se adicionan los generados El documento HTML contiene Meta Dato Genérico y se eliminan para insertar los generados Verificando el resultado obtenido en cda caso.
Recursos	Documento HTML
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Generador de Meta Datos” Seleccionar la carpeta de “Meta Dato Genérico” Alimentar el dato para cada tipo de Meta Dato Genérico Generar el Meta Dato Verificar los resultados obtenidos

Tabla 5.1 Plan de la prueba del módulo “Meta Dato Genérico”

Una vez establecido el plan se procede al segundo paso que es diseñar la prueba, que consiste en definir casos de prueba y describir sus procedimientos a desarrollar para ejecutar dichos casos de prueba. En la tabla 5.2 se muestra el diseño de la prueba para este módulo, donde es indicado los diferentes casos de prueba definidos con los cuales se verificará el resultado obtenido después de la ejecución de la prueba.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Meta Dato Titulo	Capturar un titulo Generar Meta Dato Verificar resultado
Meta Dato Autor	Capturar autor Generar Meta Dato Verificar resultado
Meta Dato Generador	Capturar generador Generar Meta Dato Verificar resultado

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Meta Dato Fecha de creación	Capturar fecha de creación Generar Meta Dato Verificar resultado
Meta Dato Lenguaje (para diferentes opciones)	Capturar lenguaje Generar Meta Dato Verificar resultado Repetir con otro lenguaje
Meta Dato Descripción	Capturar descripción Generar Meta Dato Verificar resultado
Crear nuevo Meta Dato	Seleccionar un documento HTML Capturar Meta Datos Genéricos Generar Meta Dato Verificar resultados
Adicionar Meta Dato Genérico	Seleccionar un documento HTML Capturar Meta Datos Genéricos Seleccionar opción para adicionar Generar Meta Dato Verificar resultados
Insertar Meta Dato Genérico	Seleccionar un documento HTML Capturar Meta Datos Genéricos Seleccionar opción para insertar Generar Meta Dato Verificar resultados
Editar código HTML	Habilitar la opción de edición Realizar cambios en el código HTML Verificar resultados
Guardar HTML modificado	Seleccionar un documento HTML Capturar Meta Datos Genéricos Generar Meta Dato Guardar Meta Dato generado Verificar resultados

Tabla 5.2 Diseño de la prueba para el módulo “Meta Dato Genérico”

El siguiente paso después del diseño de la prueba es la implementación y ejecución de la prueba. Durante el proceso de refinamiento del software se repiten las actividades de la prueba diseñada contabilizando los errores y en su caso el éxito de la prueba, en distintas ocasiones denominadas iteraciones las cuales deben cubrir el ciclo completo del desarrollo, el ciclo de vida del desarrollo de este trabajo de tesis se dio con 4 iteraciones. En la

tabla 5.3 se resumen el conteo de fallas y éxitos en la ejecución de los casos de prueba diseñados para el módulo del caso de uso en cuestión durante el ciclo de vida del desarrollo de la aplicación.

EJECUCIÓN DE PRUEBAS								
	Iteración 1		Iteración 2		Iteración 3		Iteración 4	
CASO DE PRUEBA	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito
Meta Dato Título		Éxito		Éxito		Éxito		Éxito
Meta Dato Autor		Éxito		Éxito		Éxito		Éxito
Meta Dato Generador	1			Éxito		Éxito		Éxito
Meta Dato Fecha de creación	1			Éxito		Éxito		Éxito
Meta Dato Lenguaje (para diferentes opciones)	2		1			Éxito		Éxito
Meta Dato Descripción		Éxito		Éxito		Éxito		Éxito
Crear nuevo Meta Dato	2		1			Éxito		Éxito
Adicionar Meta Dato Genérico	3		2		1			Éxito
Insertar Meta Dato Genérico	1			Éxito		Éxito		Éxito
Editar código HTML		Éxito		Éxito		Éxito		Éxito
Guardar HTML modificado	2		1			Éxito		Éxito

Tabla 5.3 Resultados de la ejecución de las pruebas diseñadas para el módulo Meta Dato Genérico, indicando el número de defectos encontrados por caso de prueba e iteración

Por último se realiza la verificación de resultados con respecto a la funcionalidad diseñada para el caso de uso, para de esta manera retroalimentar y continuar con el ciclo de vida de las pruebas. En la tabla 5.4 se presentan los resultados del Meta Generador a la prueba del módulo que genera el Meta Dato genérico en el momento en que se alcanza el éxito de todos los casos de prueba del plan de pruebas diseñado. Para lograr esto se cargó un documento HTML de prueba y se capturaron los campos referentes

a los meta datos de: título, autor, generador, fecha de creación, lenguaje y descripción, posteriormente siguiendo la estrategia planteada en los casos de prueba se ejecutaron los pasos para probar este módulo. En la imagen de la pantalla de salida se puede observar el área de captura de Meta Datos, la ventana del documento HTML abierto y la ventana donde esta el código HTML generado para los Meta Datos capturados.

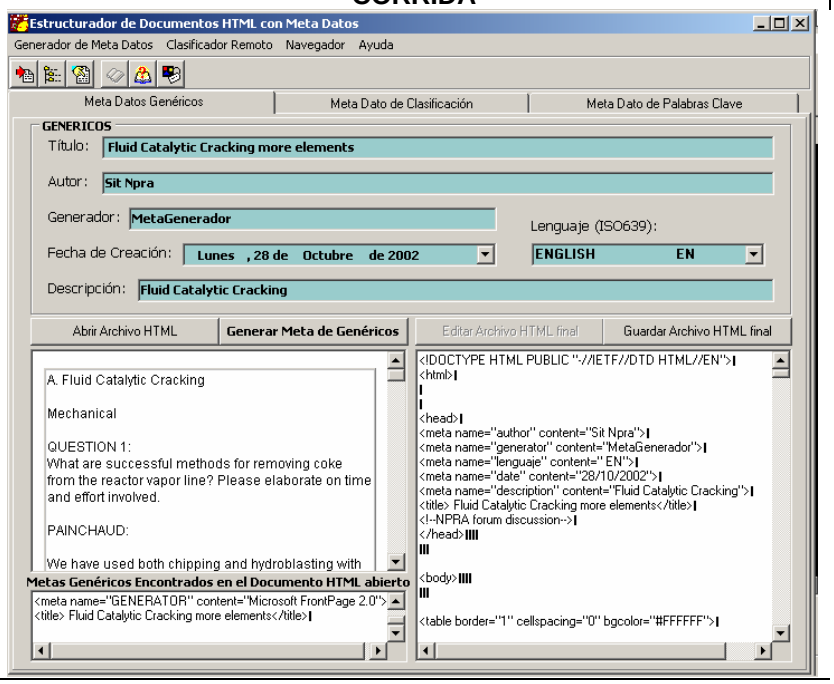
RESULTADOS DE LA PRUEBA DEL MÓDULO META DATO GENÉRICO	
Caso de Prueba	**CORRIDA**
Meta Dato Título Meta Dato Autor Meta Dato Generador Meta Dato Fecha de creación Meta Dato Lenguaje (para diferentes opciones) Meta Dato Descripción Crear nuevo Meta Dato Adicionar Meta Dato Genérico Insertar Meta Dato Genérico Editar código HTML Guardar HTML modificado	

Tabla 5.4 Salida del Meta Generador a la prueba del módulo de generación de Meta Datos genéricos

Aplicando los mismos pasos de la metodología en esta fase de pruebas para el módulo que genera el Meta Dato de clasificación, se diseño un plan de prueba que permita verificar la funcionalidad al cargar una taxonomía, seleccionar tópicos del árbol taxonómico y generar el Meta Dato de clasificación correspondiente. En la tabla 5.5 se resume, las acciones a tomar en el plan de la prueba, estos son la estrategia, los recursos a utilizar y

el programa de actividades a desarrollar para validar los resultados arrojados por el programa Meta Generador para éste módulo.

Prueba del módulo Meta Dato de Clasificación.

PLAN DE LA PRUEBA	
Introducción:	Generar un Meta Dato de Clasificación en un documento HTML
Estrategia	Seleccionar una taxonomía de prueba y generar el Meta Dato en un documento HTML ejemplo, considerando: El documento HTML no contiene Meta Dato de Clasificación El documento HTML contiene Meta Dato de Clasificación y se adicionan los generados El documento HTML contiene Meta Dato de Clasificación y se eliminan para insertar los generados Verificando el resultado obtenido en cada caso.
Recursos	Archivo de taxonomía tipo RDM y documento HTML
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Generador de Meta Datos” Seleccionar la carpeta de “Meta Dato de Clasificación” Seleccionar una taxonomía y abrir el archivo RDM correspondiente Abrir un documento HTML de prueba Expandir el árbol taxonómico y seleccionar tópicos Generar el Meta Dato de Clasificación Verificar los resultados obtenidos

Tabla 5.5 Plan de la prueba del módulo “Meta Dato de Clasificación”

De igual manera se diseño la prueba correspondiente al módulo Meta Dato de Clasificación” donde se definieron 9 casos de prueba para verificar la funcionalidad de este módulo. En la tabla 5.6 se resumen los casos de prueba definidos para este módulo, los cuales permitirán verificar el resultado obtenido después de la ejecución de la prueba.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Taxonomía	Capturar un título Generar Meta Dato Verificar resultado
Documento HTML	Seleccionar documento HTML Abrir documento Verificar resultado
Lista de Tópicos	Seleccionar tópicos del árbol de taxonomía Limpiar lista de tópicos Eliminar tópico de la lista Verificar resultado
Meta Dato de Clasificación	Expandir árbol de taxonomía y seleccionar tópicos Generar Meta Dato de clasificación Verificar resultado
Crear nuevo Meta Dato	Seleccionar un nuevo documento HTML Seleccionar una nueva taxonomía Expandir árbol de taxonomía y seleccionar tópicos Generar Meta Dato Verificar resultados
Adicionar Meta Dato de Clasificación	Seleccionar Taxonomía Seleccionar un documento HTML Expandir árbol de taxonomía y seleccionar tópicos Seleccionar opción para adicionar Generar Meta Dato Verificar resultados
Insertar Meta Dato de Clasificación	Seleccionar Taxonomía Seleccionar un documento HTML Expandir árbol de taxonomía y seleccionar tópicos Seleccionar opción para insertar Generar Meta Dato Verificar resultados
Editar código HTML	Habilitar la opción de edición Realizar cambios en el código HTML Verificar resultados
Guardar HTML modificado	Seleccionar Taxonomía Seleccionar un documento HTML Seleccionar Tópicos Generar Meta Dato de clasificación Guardar Meta Dato generado Verificar resultados

Tabla 5.6 Diseño de la prueba para el módulo “Meta Dato de Clasificación”

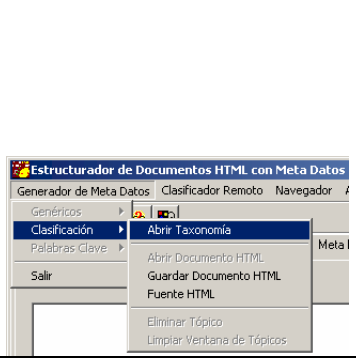
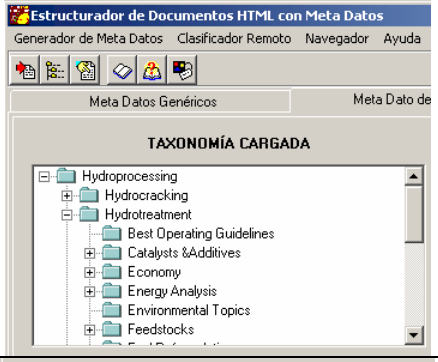
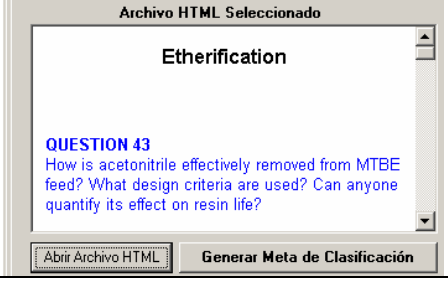
Como ya se ha mencionado una vez diseñada la prueba se procede a la implementación y ejecución de ésta, en la figura 5.7 se resumen las fallas y éxitos obtenidos en la ejecución de cada caso de prueba diseñado para este módulo.

EJECUCIÓN DE PRUEBAS								
	Iteración 1		**Iteración 2**		**Iteración 3**		**Iteración 4**	
CASO DE PRUEBA	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**
Taxonomía	4		2		1			Éxito
Documento HTML		Éxito		Éxito		Éxito		Éxito
Lista de Tópicos	1			Éxito		Éxito		Éxito
Meta Dato de Clasificación	2		2			Éxito		Éxito
Crear nuevo Meta Dato		Éxito						
Adicionar Meta Dato de Clasificación	1			Éxito		Éxito		Éxito
Insertar Meta Dato de Clasificación	1			Éxito		Éxito		Éxito
Editar código HTML		Éxito		Éxito		Éxito		Éxito
Guardar HTML modificado		Éxito		Éxito		Éxito		Éxito

Tabla 5.7 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo Meta Dato de Clasificación, indicando el número de defectos encontrados por caso de prueba e iteración

En la tabla 5.8 se muestra las pantallas de salida del Meta Generador para cada caso de prueba del módulo que genera el Meta Dato de clasificación una vez alcanzada la operación satisfactoria del módulo. De igual manera se cargó un documento HTML y una taxonomía de prueba para posteriormente y siguiendo la estrategia planteada en los casos de prueba diseñados se ejecutaron los pasos para probar este módulo. Para el caso de prueba “Taxonomía” se verificó la funcionalidad del módulo al abrir el archivo tipo “rdm” de una taxonomía de ejemplo, decodificar el formato del mismo e instanciar un objeto árbol que muestra taxonomía abierta. Para el caso de prueba “Documento HTML” se verifico la funcionalidad de abrir un documento HTML de prueba y desplegarlo en la ventana correspondiente y

de esta manera probar el objeto “ActiveX” instanciado para este fin. En el caso de prueba “Lista de Tópicos” se verifico la funcionalidad al seleccionar algún tema del árbol taxonómico y analizando la información generada en la ventana correspondiente que el tema no estuviera seleccionado más de una vez, así como las opciones de eliminar tópico de la lista y limpiar lista. Para el caso de prueba “Meta Dato de Clasificación” se verifico la funcionalidad para la detección de este tipo de Meta Dato en el documento html abierto, el cual es desplegado en una ventana si éste es detectado en el documento por el Meta Generador. Por último se verificó la funcionalidad en la generación del Meta Dato de clasificación en donde después de seleccionar los tópicos requeridos del árbol taxonómico, se probó la generación del Meta Dato adicionando el Meta Dato existente o eliminado el existente e insertando el nuevo Meta Dato de clasificación, el cual puede comprobarse en la ventana donde es desplegado el código HTML generado para este Meta Dato.

RESULTADOS DE LA PRUEBA DEL MÓDULO META DATO DE CLASIFICACIÓN	
Caso de Prueba	**CORRIDA**
Taxonomía	 
Documento HTML	 

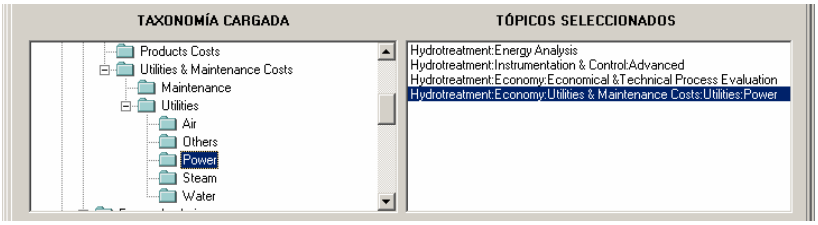
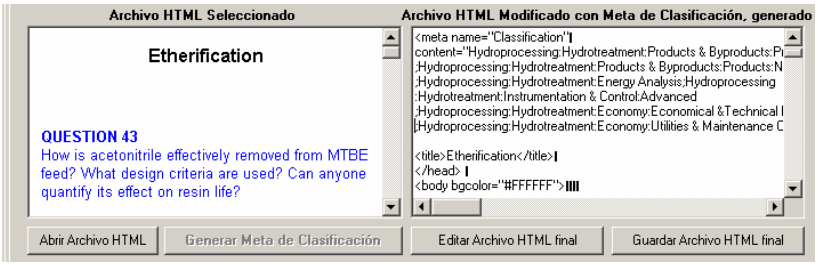
RESULTADOS DE LA PRUEBA DEL MÓDULO META DATO DE CLASIFICACIÓN	
Caso de Prueba	**CORRIDA**
Lista de Tópicos	
Meta Dato de Clasificación	
Crear nuevo Meta Dato Adicionar Meta Dato de Clasificación Insertar Meta Dato de Clasificación	

Tabla. 5.8 Salida del Meta Generador a la prueba del módulo de generación de Meta Datos de Clasificación

Continuando con la metodología del ciclo de la prueba ahora para el siguiente módulo desarrollado, en la tabla 5.9 se muestra el plan de la prueba del módulo “Meta Dato de Palabras Clave” diseñado para verificar su funcionalidad de acuerdo a los requerimientos definidos para este caso de uso.

Prueba del módulo Meta Dato de Palabras Clave

PLAN DE LA PRUEBA	
Introducción:	Generar un Meta Dato de Palabras Clave en un documento HTML
Estrategia	Generar el Meta Dato de Palabras Clave en un documento HTML ejemplo, considerando: Con frecuencia de repetición >= 0 Con diferentes frecuencias de repetición El documento HTML no contiene Meta Dato de Palabras Clave El documento HTML contiene Meta Dato de Palabras Clave y se adicionan los generados El documento HTML contiene Meta Dato de Palabras Clave y se eliminan para insertar los generados Verificando el resultado obtenido en cada caso.
Recursos	Documento HTML
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Generador de Meta Datos” Seleccionar la carpeta de “Meta Dato de Palabras Clave Abrir un documento HTML de prueba Capturar la frecuencia de repetición deseada Realizar búsqueda de palabras de acuerdo a frecuencia de repetición Generar el Meta Dato de Palabras Clave Verificar los resultados obtenidos

Tabla 5.9 Plan de la prueba del módulo “Meta Dato de Palabras Clave”

De igual forma una vez definido el plan se diseña la prueba para este módulo, determinando ocho casos de prueba que verificarán desde el cargado del documento HTML hasta la generación del Meta Dato de palabras clave, probando la funcionalidad del algoritmo de búsqueda programado a diferentes valores de frecuencia de repetición. Así mismo se probó como en los demás módulos, la detección de Meta Datos en el documento HTML cargado y las operaciones de edición y guardado del documento HTML de salida. En la tabla 5.10 se muestra el diseño de esta prueba.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Documento HTML	Seleccionar documento HTML Abrir documento Verificar resultado
Frecuencia de repetición	Definir frecuencia de repetición (diferentes casos) Ejecutar búsqueda Verificar resultado
Meta Dato de Palabras Clave	Definir frecuencia de repetición Ejecutar búsqueda Generar Meta Dato de Palabras Clave Verificar resultado
Crear nuevo Meta Dato	Seleccionar un nuevo documento HTML Definir frecuencia de repetición Ejecutar búsqueda Generar Meta Dato de Palabras Clave Verificar resultados
Adicionar Meta Dato de Palabras Clave	Seleccionar un nuevo documento HTML Definir frecuencia de repetición Ejecutar búsqueda Seleccionar opción para adicionar Generar Meta Dato de Palabras Clave Verificar resultados
Insertar Meta Dato de Palabras Clave	Seleccionar un nuevo documento HTML Definir frecuencia de repetición Ejecutar búsqueda Seleccionar opción para insertar Generar Meta Dato de Palabras Clave Verificar resultados
Editar código HTML	Habilitar la opción de edición Realizar cambios en el código HTML Verificar resultados
Guardar HTML modificado	Seleccionar un nuevo documento HTML Definir frecuencia de repetición Ejecutar búsqueda Generar Meta Dato de Palabras Clave Guardar Meta Dato generado Verificar resultados

Tabla 5.10 Diseño de la prueba para el módulo “Meta Dato de Palabras Clave”

En la tabla 5.11 se resumen las fallas y éxitos obtenidos en la ejecución de los casos de prueba definidos en el diseño de la prueba de este módulo, siguiendo con el paso indicado en la metodología aplicada de implementación y ejecución de prueba.

EJECUCIÓN DE PRUEBAS								
	Iteración 1		Iteración 2		Iteración 3		Iteración 4	
CASO DE PRUEBA	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito
Documento HTML		Éxito		Éxito		Éxito		
Frecuencia de repetición	1		1			Éxito		
Meta Dato de Palabras Clave	2		1			Éxito		
Crear nuevo Meta Dato		Éxito		Éxito		Éxito		
Adicionar Meta Dato de Palabras Clave	1			Éxito		Éxito		
Insertar Meta Dato de Palabras Clave		Éxito		Éxito		Éxito		
Editar código HTML		Éxito		Éxito		Éxito		
Guardar HTML modificado		Éxito		Éxito		Éxito		

Tabla 5.11 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo Meta Dato de Palabras Clave, indicando el número de defectos encontrados por caso de prueba e iteración

En la tabla 5.12 se muestra las pantallas de salida del Meta Generador para cada caso de prueba del módulo que genera el Meta Dato de palabras clave una vez alcanzado la operación satisfactoria del módulo. De igual modo que en los procesos de prueba anteriores se selecciona un documento HTML de ejemplo para ejecutar los casos de prueba diseñados para este módulo. En el caso de prueba “Documento HTML” se verificó la misma funcionalidad del caso de prueba anterior ya que operan de la misma manera. Para el caso de prueba “Frecuencia de repetición” se verificó la funcionalidad del algoritmo de búsqueda utilizado de acuerdo a la frecuencia de repetición deseada, es decir se tomo un valor deseado para la frecuencia de repetición y una vez realizada la búsqueda se corroboró la lista generada contra el documento HTML abierto, lo que mostró que las palabras encontradas tenían una frecuencia de repetición de igual o mayor al valor indicado, en este caso de prueba también fué verificada la funcionalidad del indicador de avance de la búsqueda. Por último se verificó la funcionalidad en la generación del Meta Dato de palabras clave, revisando la selección

múltiple de la lista de palabras encontradas y la generación del Meta Dato cuando se adicionan al documento HTML abierto o cuando se elimina y se inserta el nuevo Meta Dato creado. El código HTML generado para este Meta Dato se observa en el área de texto correspondiente.

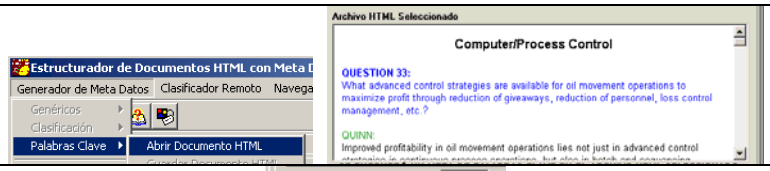
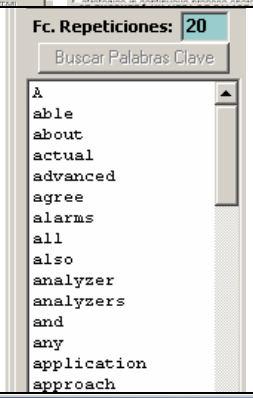
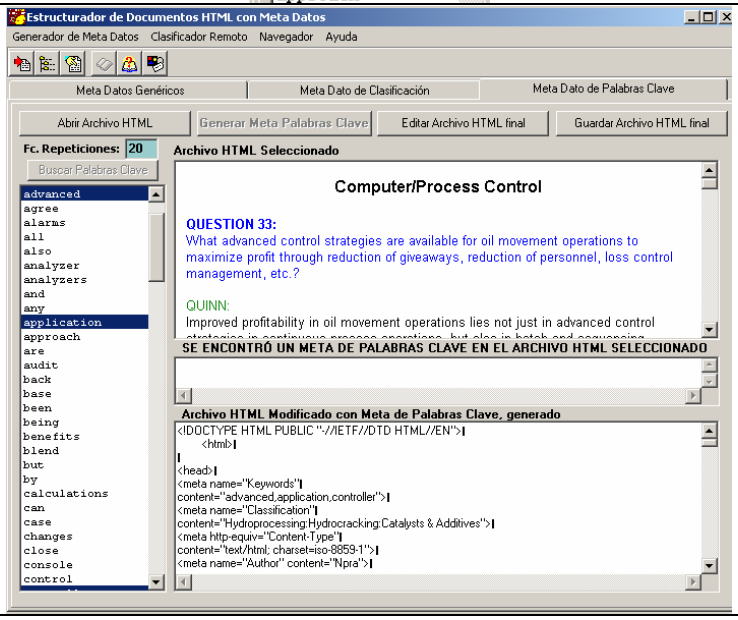
RESULTADOS DE LA PRUEBA DEL MÓDULO META DATO DE PALABRAS CLAVE	
Caso de Prueba	**CORRIDA**
Documento HTML	
Frecuencia de repetición	
Crear nuevo Meta Dato Adicionar Meta Dato de Palabras Clave Insertar Meta Dato de Palabras Clave	

Tabla 5.12 Salida del Meta Generador a la prueba del módulo de generación de Meta Datos de Palabras Clave

Con esto se termino con la verificación de la funcionalidad esperada del módulo generador local de Meta Datos de acuerdo a los requerimientos definidos en el modelo de casos de uso, para de esta manera proseguir a la validación de los resultados a través del análisis de los defectos encontrados en el proceso de pruebas. De esta manera se ha cubierto el ciclo de vida de pruebas diseñadas para verificar la funcionalidad del caso de uso “Generador Local de Meta Datos” y alcanzar los objetivos planteados por la metodología en esta fase de pruebas.

5.1.2 Caso de Uso “Clasificador de Documentos Remotos”

Del mismo modo que en el caso de uso anterior, se aplicará la metodología para definir el plan, diseño, implementación y ejecución de pruebas para verificar la funcionalidad de los módulos que componen al caso de uso “Clasificador de Documentos Remotos”.

En la tabla 5.13 se indican la estrategia, recursos y programa de actividades del plan de prueba diseñado para el módulo “Edición árbol de Taxonomía”.

Prueba del módulo Edición árbol de Taxonomía

PLAN DE LA PRUEBA	
Introducción:	Abrir una taxonomía de clasificación y ejecutar las opciones de edición
Estrategia	Seleccionar una taxonomía de prueba y realizar las funciones de edición, considerando: Adicionar tema al árbol taxonómico Eliminar tema ó documento del árbol taxonómico Expandir y contraer árbol taxonómico Adicionar referencia URL de documento HTML al árbol taxonómico Verificando el resultado obtenido en cada caso.

PLAN DE LA PRUEBA	
Recursos	Archivo de Taxonomía y referencias URL
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Clasificador Remoto” Abrir un documento WEB en el navegador Cargar una taxonomía Expandir el árbol taxonómico y seleccionar nodo Ejecutar funciones de edición Verificar los resultados obtenidos

Tabla 5.13 Plan de la prueba del módulo “Edición árbol de Taxonomía”

La tabla 5.14 muestra el diseño de la prueba para el módulo “Edición árbol de Taxonomía”, donde están descritos los procedimientos de prueba para cinco casos de prueba.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Taxonomía	Seleccionar taxonomía de prueba Cargar y abrir taxonomía de prueba Verificar resultado
Adicionar referencia URL	Abrir navegador Navegar y seleccionar documento WEB de interés Expandir taxonomía y seleccionar tema Adicionar referencia URL del documento WEB en línea al tema seleccionado Verificar resultado
Adicionar tema al árbol taxonómico	Expandir taxonomía y seleccionar nodo Adicionar nuevo tema Verificar resultado
Eliminar tema ó documento del árbol taxonómico	Expandir taxonomía y seleccionar nodo o referencia Eliminar tema ó referencia Verificar resultado
Expandir y contraer árbol taxonómico	Expandir árbol taxonómico Contraer árbol taxonómico Verificar resultado

Tabla 5.14 Diseño de la prueba para el módulo “Edición árbol de Taxonomía”

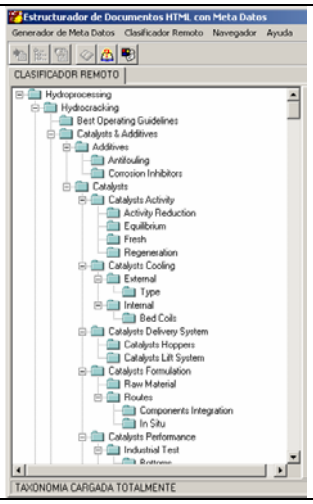
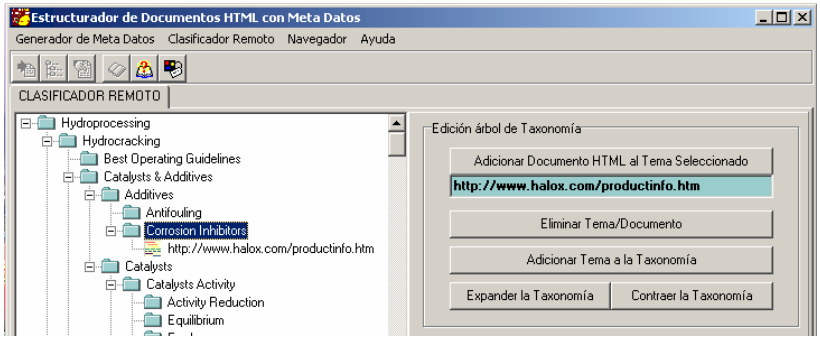
Del mismo modo y siguiendo la metodología en esta fase de pruebas se realiza la implementación y ejecución de los caso de prueba diseñados para éste módulo. La figura 5.15 resume las fallas y éxitos obtenidos en la ejecución de la pruebas al módulo “Edición árbol de Taxonomía”.

EJECUCIÓN DE PRUEBAS								
	Iteración 1		**Iteración 2**		**Iteración 3**		**Iteración 4**	
CASO DE PRUEBA	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**
Taxonomía	2		1			Éxito		
Adicionar referencia URL		Éxito		Éxito		Éxito		
Adicionar tema al árbol taxonómico		Exito		Éxito		Éxito		
Eliminar tema ó documento del árbol taxonómico	1			Éxito		Éxito		
Expandir y contraer árbol taxonómico		Éxito		Éxito		Éxito		

Tabla 5.15 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo Edición árbol de Taxonomía, indicando el número de defectos encontrados por caso de prueba e iteración

En la tabla 5.16 se muestra las pantallas de salida del Meta Generador para cada caso de prueba del módulo de edición árbol de taxonomía una vez alcanzado la operación satisfactoria del módulo. Del mismo modo que en los procesos de prueba anteriores para la clasificación, se selecciona un archivo de taxonomía en formato rdm de ejemplo para ejecutar los casos de prueba diseñados para este módulo. Para el caso de prueba “Taxonomía” se verificó la funcionalidad del módulo al abrir el archivo tipo “rdm” de una taxonomía, decodificar el formato del mismo e instanciar un objeto árbol que muestra la taxonomía abierta, así mismo se verifico la operación a los eventos de expandir y contraer el árbol taxonómico. Para el caso de prueba “Adicionar Referencia URL” se verifico la funcionalidad para recuperar la referencia URL

de la página Web activa en el navegador del Meta Generador, la selección del nodo tema del árbol taxonómico y la adición de esta referencia URL al nodo seleccionado. Por último para los demás casos de prueba se verificó la funcionalidad de eliminar temas del árbol taxonómico considerando la validación del usuario para eliminar junto con el nodo la herencia correspondiente si este es el caso, la adición de nuevos temas al árbol y la edición del nombre de los temas.

RESULTADOS DE LA PRUEBA DEL MÓDULO EDICIÓN ÁRBOL DE TAXONOMÍA	
Caso de Prueba	**CORRIDA**
Taxonomía	
Adicionar referencia URL	

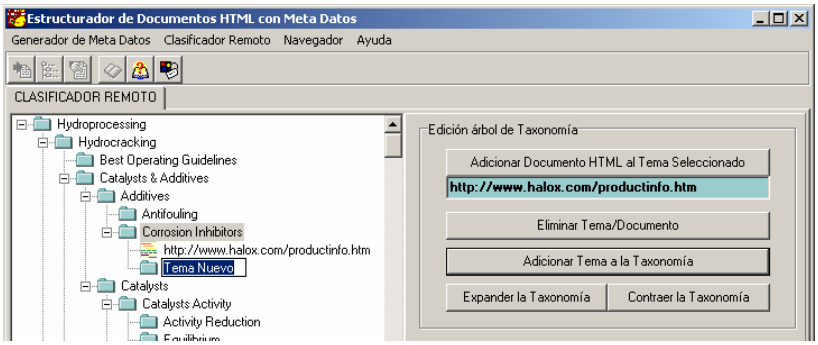
RESULTADOS DE LA PRUEBA DEL MÓDULO EDICIÓN ÁRBOL DE TAXONOMÍA	
Caso de Prueba	**CORRIDA**
Adicionar tema al árbol taxonómico Eliminar tema ó documento del árbol taxonómico	

Tabla 5.16 Salida del Meta Generador a la prueba del módulo de edición árbol de taxonomía

En la tabla 5.17 se indican la estrategia, recursos y programa de actividades para definir el plan de prueba del módulo “Navegador”.

Prueba del módulo Navegador

PLAN DE LA PRUEBA	
Introducción:	Abrir una taxonomía de clasificación y ejecutar las opciones de navegación
Estrategia	Seleccionar una taxonomía de prueba y realizar las funciones de navegación considerando: Abrir una referencia URL del árbol taxonómico Exportar la taxonomía al navegador Verificando el resultado obtenido en cada caso.
Recursos	Archivo de Taxonomía y referencias URL
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Clasificador Remoto” Abrir un documento Web en el navegador Cargar una taxonomía Expandir el árbol taxonómico y seleccionar nodo con referencia URL Abrir referencia en el navegador Exportar taxonomía al navegador Verificar los resultados obtenidos

Tabla 5.17 Plan de la prueba del módulo “Navegador”

Siguiendo el mismo procedimiento las tablas 5.18 y 5.19 muestran el diseño y ejecución de la prueba para el módulo “Edición árbol de Taxonomía”.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Abrir referencia URL en navegador	Seleccionar taxonomía de prueba Cargar y abrir taxonomía de prueba Seleccionar referencia URL del árbol taxonómico Abrir referencia en el navegador Verificar resultado
Exportar taxonomía a navegador	Seleccionar taxonomía de prueba Cargar y abrir taxonomía de prueba Adicionar referencias URL a los temas del árbol Exportar taxonomía al navegador Verificar resultado

Tabla 5.18 Diseño de la prueba para el módulo “Navegador”

EJECUCIÓN DE PRUEBAS								
	Iteración 1		**Iteración 2**		**Iteración 3**		**Iteración 4**	
CASO DE PRUEBA	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**	**No. Fallas**	**Éxito**
Abrir referencia URL en navegador		Éxito		Éxito		Éxito		Éxito
Exportar taxonomía a navegador	4		2		1			Éxito

Tabla 5.19 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo Navegador, indicando el número de defectos encontrados por caso de prueba e iteración

En la tabla 5.20 se muestra las pantallas de salida del Meta Generador para cada caso de prueba del módulo navegador una vez alcanzado la operación satisfactoria del módulo. Del mismo modo que en los procesos de prueba anteriores, se selecciona un referencia de URL de ejemplo para ejecutar los casos de prueba diseñados para este módulo. Para el caso de prueba “Abrir referencia URL en navegador” se verificó la funcionalidad para seleccionar una referencia URL del árbol taxonómico y solicitar la página de Web correspondiente a través del navegador del Meta Generador. Para el caso de prueba “Exportar taxonomía al navegador” se verificó la

funcionalidad de transportar el objeto árbol con su contenido a un documento de Web codificado en HTML.

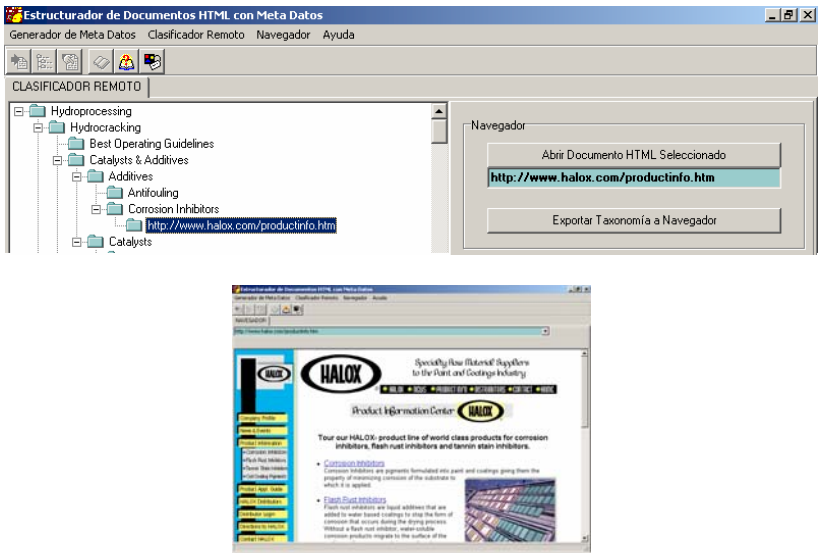
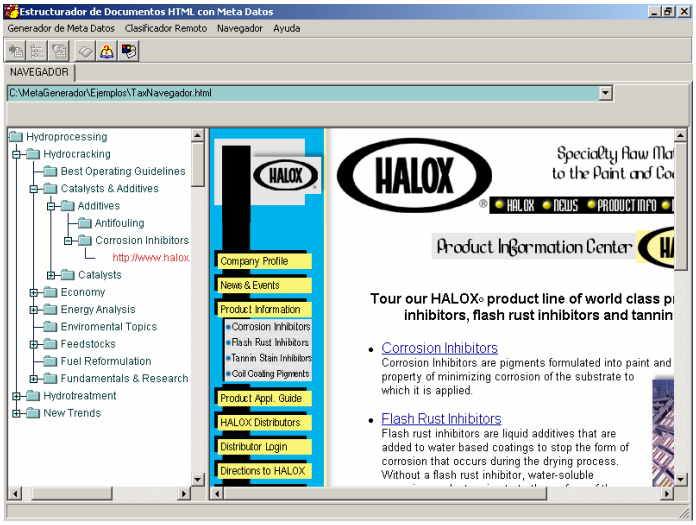
RESULTADOS DE LA PRUEBA DEL MÓDULO NAVEGADOR	
Caso de Prueba	**CORRIDA**
Abrir referencia URL en navegador	
Exportar taxonomía a navegador	

Tabla 5.20 Salida del Meta Generador a la prueba del módulo de navegador

Aplicando la metodología del mismo modo que en los módulos anteriores para el módulo "Respaldo de Taxonomía "se definió un plan, diseño, implementación y ejecución de pruebas. Estos se muestran en las tablas 5.21, 5.22 y 5.23 respectivamente.

Prueba del módulo Respaldo de Taxonomía.

PLAN DE LA PRUEBA	
Introducción:	Abrir una taxonomía de clasificación y ejecutar las opciones de respaldo
Estrategia	Seleccionar una taxonomía de prueba y realizar las funciones de respaldo de taxonomía considerando: Guardar taxonomía con datos de referencias Abrir taxonomía con datos de referencias Generar taxonomía en formato RDM Verificando el resultado obtenido en cada caso.
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de "Clasificador Remoto" Abrir un documento WEB en el navegador Cargar una taxonomía Expandir el árbol taxonómico y adicionar referencias URL Generar taxonomía en formato RDM Verificar los resultados obtenidos

Tabla 5.21 Plan de la prueba del módulo "Respaldo de Taxonomía"

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Guardar Taxonomía	Seleccionar taxonomía de prueba Cargar y abrir taxonomía de prueba Adicionar referencias URL a los temas del árbol Guardar taxonomía con referencias Verificar resultado
Abrir Taxonomía	Seleccionar taxonomía con referencias Cargar y abrir taxonomía seleccionada Expandir árbol taxonómico con referencias Verificar resultado
Generar Taxonomía en formato RDM	Seleccionar taxonomía de prueba Cargar y abrir taxonomía de prueba Adicionar referencias URL a los temas del árbol Genera taxonomía en formato RDM Verificar resultado

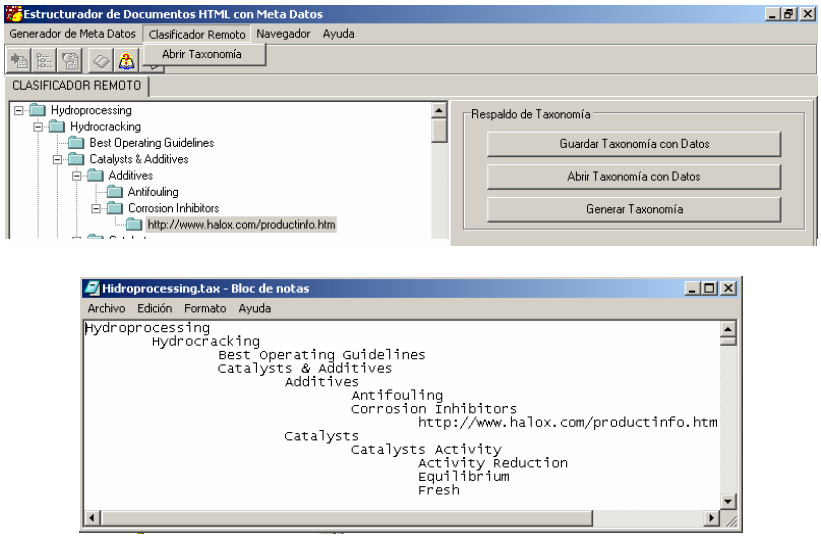
Tabla 5.22 Diseño de la prueba para el módulo "Respaldo de Taxonomía"

EJECUCIÓN DE PRUEBAS								
	Iteración 1		Iteración 2		Iteración 3		Iteración 4	
CASO DE PRUEBA	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito
Guardar Taxonomía	2			Éxito		Éxito		Éxito
Abrir Taxonomía	1			Éxito		Éxito		Éxito
Generar Taxonomía en formato RDM	3		2		1			Éxito

Tabla 5.23 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo de Respaldo de Taxonomía, indicando el número de defectos encontrados por caso de prueba e iteración

En la tabla 5.24 se muestra las pantallas de salida del Meta Generador para cada caso de prueba del módulo respaldo de taxonomía una vez alcanzado la operación satisfactoria del módulo. Para el caso de prueba “Guardar taxonomía “ y “Abrir taxonomía” se verificó la funcionalidad al crear un archivo texto plano con la estructura del árbol taxonómico incluyendo las referencia URL adicionadas, así como recuperar la taxonomía con referencias URL desde el archivo texto creado e instanciar un objeto árbol. Por último para el caso de prueba “Generar taxonomía en formato RDM” se verificó la funcionalidad al crear un archivo tipo “rdm” a partir del árbol taxonómico creado dentro del Meta Generador.

Con esto se terminó la verificación de la funcionalidad esperada del Clasificador de Documentos Remotos del Meta Generador de acuerdo a los requerimientos definidos en el modelo de casos de uso, para de esta manera proseguir a la validación de los resultados a través del análisis de los defectos encontrados en el proceso de pruebas.

<table>
<tr><th colspan="2">RESULTADOS DE LA PRUEBA DEL MÓDULO RESPALDO DE TAXONOMÍA</th></tr>
<tr><td>Caso de Prueba
Guardar Taxonomía
Abrir Taxonomía</td><td>CORRIDA
 

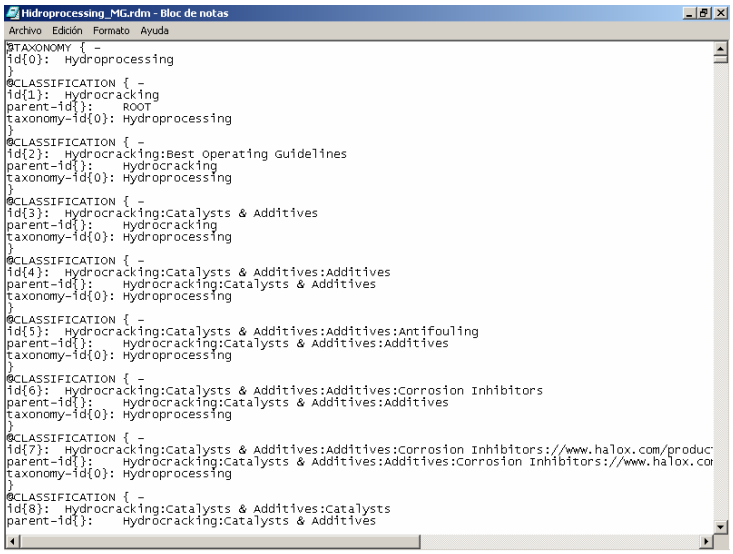
</td></tr>
<tr><td>Generar Taxonomía en formato RDM</td><td> 
</td></tr>
</table>

Tabla 5.24 Salida del Meta Generador a la prueba del módulo respaldo de taxonomía

De esta manera se ha cubierto el ciclo de vida de pruebas diseñadas para verificar la funcionalidad del caso de uso “Clasificador de Documentos Remotos” y alcanzar los objetivos planteados por la metodología en esta fase de pruebas.

5.1.3 Caso de Uso “Navegador”.

Por último del mismo modo que en el caso de uso anterior, se aplicó la metodología para definir el plan, diseño, implementación y ejecución de pruebas para verificar la funcionalidad del caso de uso “Navegador” y con éste terminar la fase de pruebas del Meta Generador.

En la tabla 5.25 se indican la estrategia, recursos y programa de actividades del plan de prueba diseñado para el caso de uso “Navegador”.

Prueba del caso de uso “Navegador”

PLAN DE LA PRUEBA	
Introducción:	Abrir el navegador y realizar las funciones propias de un navegador
Estrategia	Abrir el navegador considerando: Direccionamiento de URL Procesamiento de sitios de WEB Verificando el resultado obtenido en cada caso.
Recursos	Funciones de control del explorador WEB de windows
Programa de actividades	Ejecutar el Meta Generador Seleccionar la carpeta de “Navegador” Realizar las funcionas propias de un navegador Verificar los resultados obtenidos

Tabla 5.25 Plan de la prueba del caso de uso “Navegador”

Siguiendo el mismo procedimiento las tablas 5.26 y 5.27 muestran el diseño y ejecución de la prueba para caso de uso “Navegador”.

DISEÑO DE LA PRUEBA	
Caso de Prueba	**Procedimiento de prueba**
Navegador	Ejecutar el Meta Generador Seleccionar la carpeta de “Navegador” Realizar las funciones propias de un navegador Verificar los resultados obtenidos

Tabla 5.26 Diseño de la prueba para el caso de uso “Navegador”

EJECUCIÓN DE PRUEBAS								
	Iteración 1		Iteración 2		Iteración 3		Iteración 4	
CASO DE PRUEBA	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito	No. Fallas	Éxito
Taxonomía		Éxito						

Tabla 5.27 Tabla de resultados de la ejecución de las pruebas diseñadas para el módulo de Navegador, indicando el número de defectos encontrados por el caso de prueba e iteración

En la tabla 5.28 se muestra la pantalla de salida del Meta Generador para el caso de prueba del módulo navegador una vez alcanzado la operación satisfactoria del módulo. Con el caso de prueba para el navegador del Meta Generador se verifico la funcionalidad dada por el componente "Active X" para la operación esperada de un navegador de Internet para este módulo.

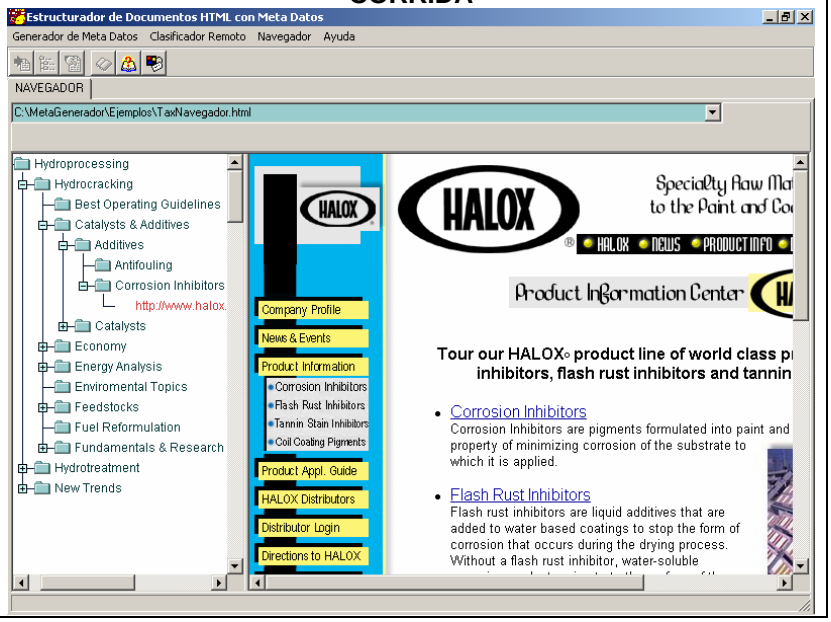
RESULTADOS DE LA PRUEBA DEL MÓDULO RESPALDO DE TAXONOMÍA	
Caso de Prueba	**CORRIDA**
Taxonomía	

Tabla 5.28 Salida del Meta Generador a la prueba del Navegador

De esta manera se ha cubierto el ciclo de vida de pruebas diseñadas para verificar la funcionalidad del caso de uso “Navegador” y alcanzar los objetivos planteados por la metodología en esta fase de pruebas.

Con esto se terminó la verificación de la funcionalidad esperada de todos los módulos que componen al Meta Generador de acuerdo a los requerimientos definidos en el modelo de casos de uso, para de esta manera proseguir a la validación de los resultados a través del análisis de los defectos encontrados en el proceso de pruebas.

5.2 Validación de Resultados

La metodología de “objectory process” propone dos parámetros de medida para los procedimientos de prueba. La primera manera es medir la integridad de la prueba y está basada en el alcance de ésta ó el alcance del código ejecutado. La segunda medición es de confiabilidad y está basada en el análisis de los defectos descubiertos durante la prueba.

Validación por Integridad

La validación por integridad se realiza a través del alcance de la prueba que mide cuantos procedimientos de prueba ó casos de prueba han sido verificados de acuerdo al total de casos de prueba.

Alcance de Ejecución de Prueba = AEP = CPE/TCP

Donde CPE son los Casos de Prueba Ejecutados y TCP son el Total de Casos de Prueba.

Así también el alcance de la prueba con éxito, indica para un conjunto específico de casos de prueba cuantos de éstos se han realizado con éxito.

Alcance de la Prueba con Éxito = APE = CPEE/TCP

Donde CPEE son los casos de prueba ejecutados con éxito.

Estas dos relaciones nos indican un parámetro cuantificable de la integridad de la prueba al expresarlos en términos de porcentaje, es decir, que porcentaje de casos de prueba han sido cubiertos con un cierto porcentaje de éxito.

Las tablas 5.29, 5.30 y 5.31 muestran los valores de estas mediciones para los casos de prueba diseñados.

Caso de Uso "Generador Local de Meta Dato"						
Prueba del módulo Meta Dato Genérico	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	11	4	39	28.20	10.25	0.36
Iteración 2	11	7	39	28.20	17.95	0.63
Iteración 3	11	10	39	28.20	25.64	0.90
Iteración 4	11	11	39	28.20	28.20	1
Caso de Uso "Generador Local de Meta Dato"						
Prueba del módulo Meta Dato Clasificación	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	9	4	39	23.08	10.26	0.44
Iteración 2	9	6	39	23.08	15.39	0.67
Iteración 3	9	7	39	23.08	17.95	0.77
Iteración 4	9	9	39	23.08	23.08	1
Prueba del módulo Meta Dato P. Clave	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	8	5	39	20.51	12.82	0.63
Iteración 2	8	6	39	20.51	15.38	0.75
Iteración 3	8	8	39	20.51	20.51	1

Tabla 5.29 Tabla de resultados de integridad para el Caso de Uso "Generador Local de Meta Dato"

Caso de Uso "Clasificador de Documentos Remotos"						
Prueba del módulo Edición árbol de Taxonomía	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	5	3	39	12.82	7.69	0.60
Iteración 2	5	4	39	12.82	10.25	0.80
Iteración 3	5	5	39	12.82	12.82	1
Prueba del módulo Navegador	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	2	1	39	5.13	2.56	0.50
Iteración 2	2	1	39	5.13	2.56	0.50
Iteración 3	2	1	39	5.13	2.56	0.50
Iteración 4	2	2	39	5.13	5.13	1
Prueba del módulo Respaldo Taxonomía	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	3	0	39	7.69	0	0
Iteración 2	3	2	39	7.69	5.13	0.67
Iteración 3	3	2	39	7.69	5.13	0.67
Iteración 3	3	3	39	7.69	7.69	1

Tabla 5.30 Tabla de resultados de integridad para el Caso de Uso "Clasificador de Documentos Remotos"

Caso de Uso "Navegador"						
Prueba del módulo Edición árbol de Taxonomía	**CPE**	**CPEE**	**TCP**	**AEP %**	**APE %**	**Integridad APE/AEP**
Iteración 1	1	1	39	2.56	2.56	1

Tabla 5.31 Tabla de resultados de integridad para el Caso de Uso "Navegador"

De estas tablas se puede observar que los casos de prueba con éxito alcanzan en promedio el 70% de integridad con respecto al total en la segunda iteración, lo que nos indica que el desarrollo tiene una tendencia de calidad buena alta ya que muestra poco índice de defectos en un periodo medio de desarrollo. Es importante recordar que es una metodología para garantizar que el desarrollo sea probado exhaustivamente y que se tenga un parámetro de medición respecto a la calidad del software desarrollado al comparar con los nuevos desarrollos.

5.2.2 Validación por Defectos

Mientras que la medición por el alcance de las pruebas con éxito proporciona una medición de integridad, una evaluación de los defectos encontrados durante la ejecución de las pruebas proporciona la mejor indicación de la calidad del software. El método para el análisis de defectos se hace revisando la distribución de defectos sobre el valor de uno a más de los parámetros asociados al defecto, hay cuatro parámetros de defectos comúnmente usados para este análisis **[ObjMan14]**:

El estado del defecto (detectado, estable, resuelto, etc.).- Este parámetro indica el estado en el que se encuentra el defecto, un primer estado para el defecto es cuando éste es detectado, un segundo estado es cuando este defecto es estable y se mantiene abierto hasta identificar las posibles causas que lo provocan, un tercer estado es cuando el defecto es resuelto. También se pueden tener estados del defecto como; crítico, no muy crítico, ó alguno que indique el grado de severidad del defecto.

La prioridad que el defecto tiene para ser resuelto ó eliminado.- Este parámetro indica el nivel de prioridad con el que se debe atacar el defecto, teniéndose cuatro niveles: Inmediato, Alta, Normal y Baja.

Grado de impacto del defecto.- Este parámetro indica el grado de impacto que provoca el defecto sobre el usuario final, la organización ó terceras partes y permite definir la prioridad para su solución.

Componentes que deben ser reparados para eliminar el defecto.- Este parámetro indica las causas que provocan el defecto y que permiten resolver el mismo.

La metodología de "Objectory Process" propone tres clases de reportes para el análisis de defectos **[ObjMan14]**:

Reporte de distribución de defectos.- Este tipo de reporte permite que la cuantificación del defecto sean demostradas en función de uno ó dos parámetros del defecto. Los criterios de la prueba generalmente incluyen una declaración sobre el número permisible de defectos abiertos en categorías particulares, tales como el grado de severidad. Este criterio se comprueba fácilmente con una evaluación de la distribución del defecto. El análisis de la distribución del defecto proporcionan una buena referencia de la eficacia de la prueba y de las actividades en la solución del defecto. Por ejemplo, si la mayoría de los defectos sin resolver está en un estado de pendiente-validación, es probable que no se tengan bastantes recursos que sean aplicados al esfuerzo por resolverlos.

Reporte del periodo del defecto.- Son un tipo especial de reporte de la distribución del defecto. Los reportes del periodo del defecto demuestran cuánto tiempo un defecto ha estado en un estado en particular, tal como el periodo de tiempo desde que fue detectado y se comportó de manera estable un determinado defecto.

Reporte de tendencia de defectos.- Este tipo de reporte permite mostrar el número de defectos por su estado (Detectado, estable, abierto ó cerrado) en función del tiempo. La tendencia identifica índices del defecto y

proporcionan una vista particularmente buena del estado de la prueba. Las tendencias del defecto siguen un patrón bastante previsible en un ciclo de la prueba (iteración). Tempranamente en el ciclo de la prueba, el índice del defecto se levantan rápidamente. Después alcanzan un pico y caen en un cierto plazo a un índice más lento. La tendencia en la solución del defecto debe seguir una misma tendencia pero en sentido inverso, este perfil puntual indica que el programa del ciclo de la prueba está cumpliéndose satisfactoriamente y representan un esfuerzo acertado. Si sus tendencias se desvían dramáticamente de estos perfiles, pueden indicar un problema e identificar cuando se requieren recursos adicionales que deben ser aplicados a las áreas específicas del desarrollo ó prueba.

A continuación se presentan estos reportes para los casos de uso más representativos del generador de Meta Datos.

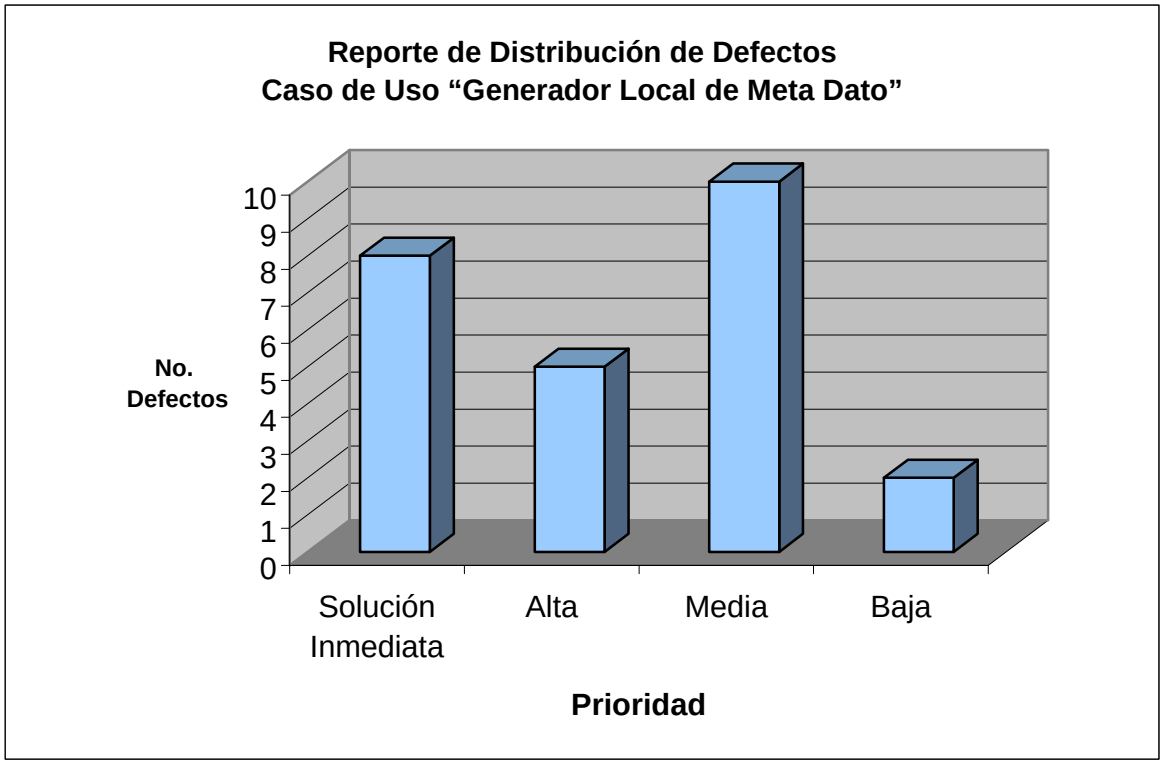


Figura 5.2. Distribución de Defectos "Generador Local de Meta Dato"

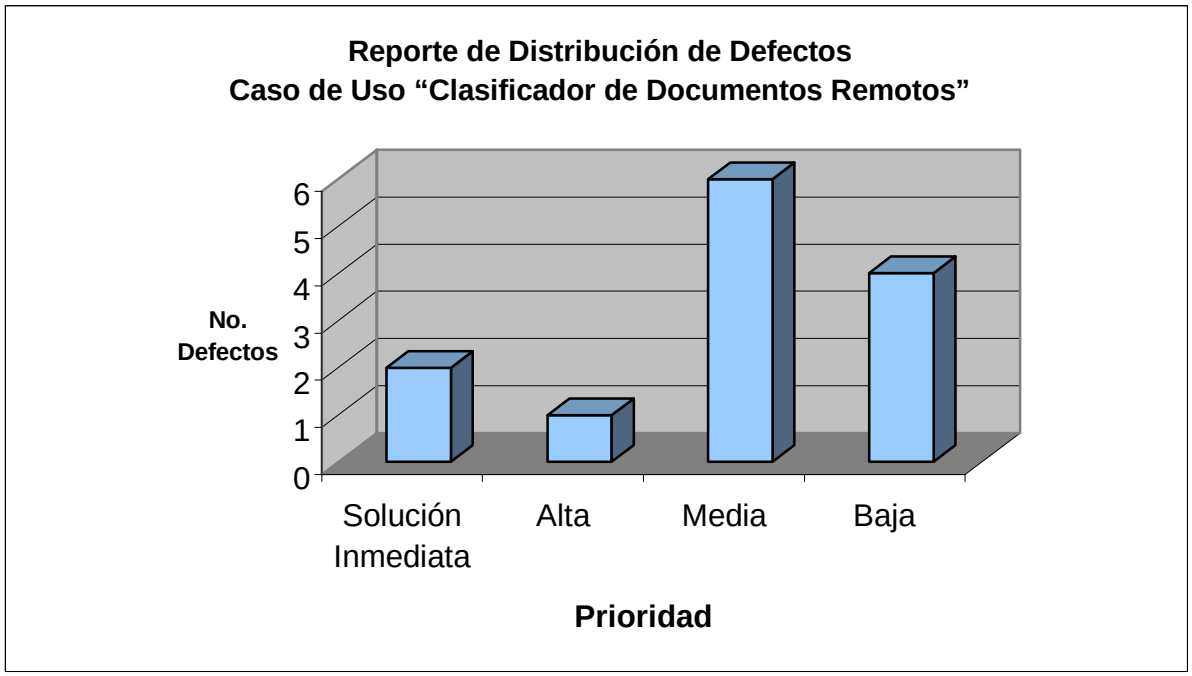


Figura 5.3. Distribución de Defectos "Clasificador de Documentos Remotos"

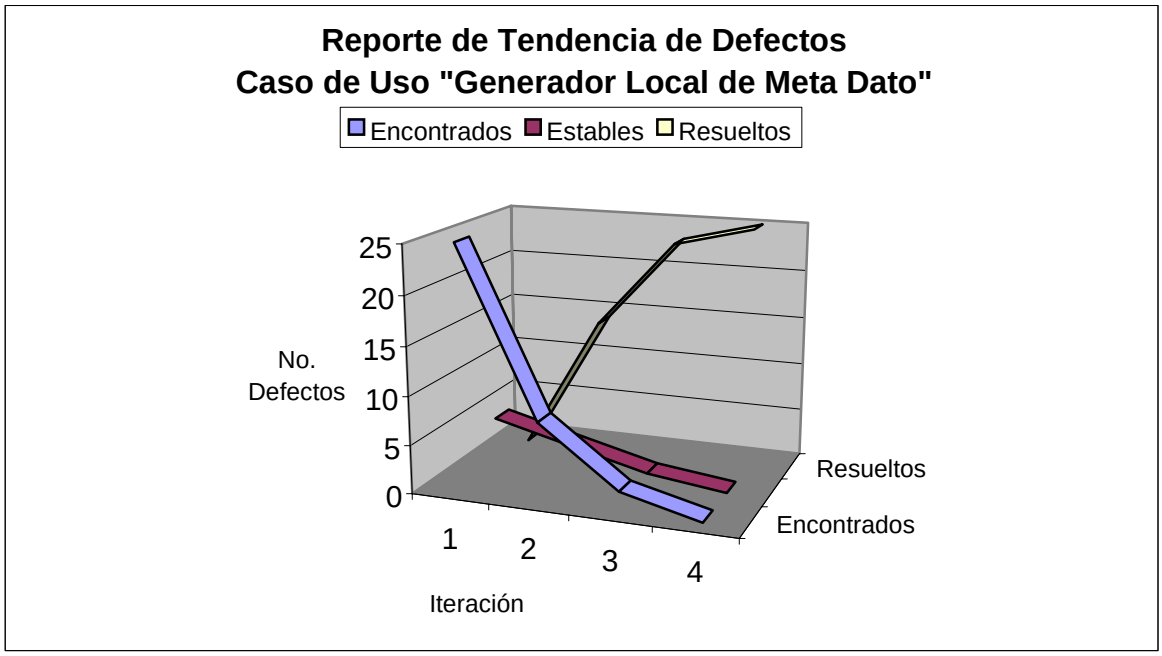


Figura 5.4 Tendencia de Defectos "Generador Local de Meta Dato"

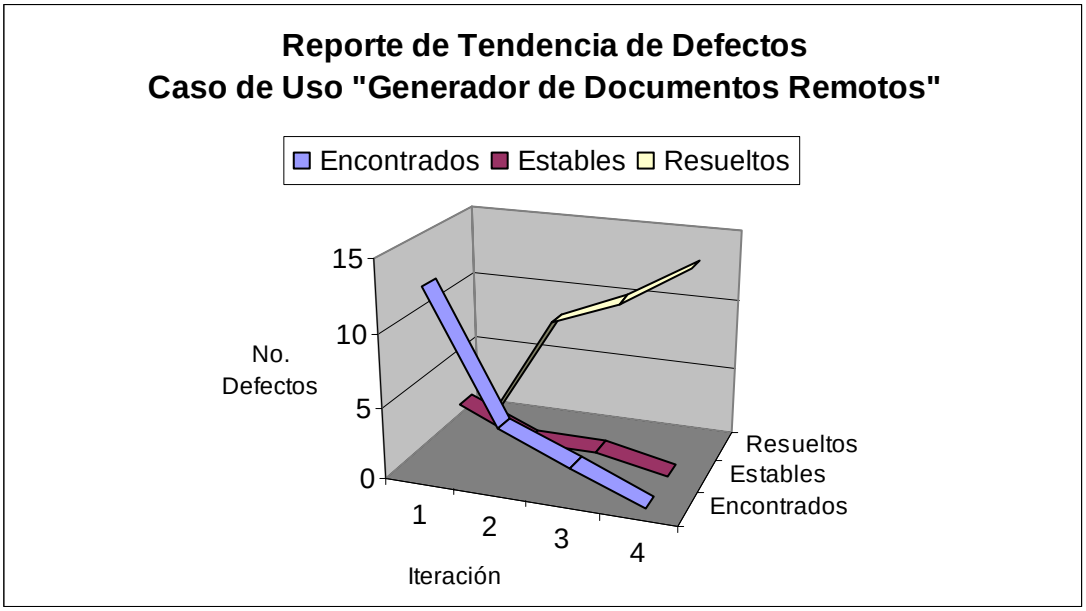


Figura 5.5 Tendencia de Defectos “Clasificador de Documentos Remotos”

Resumen

En este capítulo se definieron casos de prueba para cubrir la funcionalidad total del modelo de casos de uso y se ejecutaron cuantificando el número de fallas obtenidas por cada prueba, para de esta manera realizar una validación de resultados por integridad y por defectos del software desarrollado, así mismo se hizo un análisis gráfico para hacer tangible la distribución y tendencia de defectos de cada caso de uso durante el ciclo de vida del desarrollo.

Se aplicó la metodología propuesta por “Rational – Objectory Process” para el desarrollo de aplicaciones de software orientadas a objetos, generando una herramienta de software amigable en ambiente de ventanas para el API de Windows que satisface los requerimientos definidos.

Requerimientos que fueron analizados, diseñados, implementados y probados de acuerdo a esta metodología. Solo resta delinear los beneficios y limitantes de esta herramienta desarrollada y los trabajos futuros, tema que se tratará en el siguiente capítulo.

CAPÍTULO 6
CONCLUSIONES

En este último capítulo se describen los beneficios y limitaciones de la herramienta de software desarrollada y se discuten los trabajos futuros.

6.1 Beneficios y limitaciones

Debido al conocimiento vertido en los documentos generados por los especialistas e investigadores y que son publicados en el Web, se hace valioso contar con una herramienta de software que facilita el proceso de generación de Meta Datos y permitir estructurar estos documentos de acuerdo a una adecuada clasificación definida a través de una taxonomía en alguna área de conocimiento. La herramienta desarrollada en este trabajo de tesis cumple satisfactoriamente con esta necesidad, además de también ofrecer la posibilidad de generar nuevas taxonomías ó editar las ya existentes sin la necesidad de contar con un software que involucra la tecnología de un Robot de Búsqueda, lo que implica mayor esfuerzo y costos adicionales.

Otro beneficio que proporciona la herramienta de software desarrollada es tener la ventaja de poder abrir una taxonomía específica y a través de su propio navegador colocar referencias a documentos encontrados en el Web y que el especialista o investigador clasifiquen en algún tema del árbol taxonómico, para posteriormente exportar ésta taxonomía a un documento en formato HTML y realizar las consultas a estas referencias. Con esto es posible instalar en un servidor de Web y ofrecer un sistema de navegación jerárquico, de donde se pueden acceder desde la propia herramienta o de cualquier navegador las referencias clasificadas en el árbol taxonómico.

Así se contribuye enormemente a la aplicación en sistemas de información orientados a tecnologías de vanguardia en el campo de la administración del conocimiento y sistemas de inteligencia tecnológica.

De esta manera el Meta Generador desarrollado cumple con los objetivos planteados en él capítulo 1 para la clasificación de documentos no estructurados y resuelve su problemática, proporcionando herramientas adicionales que facilitan y complementan el proceso de estructuración a través de un ambiente amigable de manejo de ventanas y de las facilidades y características que ofrece el ambiente gráfico de Windows.

Así mismo este desarrollo esta limitado a una aplicación local que solo permite una comunicación indirecta con un robot de búsqueda, a través de un archivo compatible (archivo RDM), que por el momento satisface las necesidades de los usuarios a los que esta herramienta esta orientada. Sin embargo a pesar de ser una limitación y como se menciona en el párrafo de “Trabajos Futuros”, la herramienta cuenta con la plataforma para continuar con el desarrollo y establecer comunicación con éstos.

Es conveniente mencionar que la metodología utilizada para el desarrollo de esta herramienta de software permitió tener una alta calidad en la funcionalidad y operación de sus módulos y cumplir satisfactoriamente con los requerimientos definidos a través de casos de uso, utilizando tecnología orientada a objetos con programación orientada a componentes y la validación de resultados con una metodología que permite tener un alto grado de calidad como se indica en el capítulo 5.

Trabajos a Futuro

Esta herramienta de software puede ampliarse para establecer una conexión a la base de datos de los Robots de Búsqueda y almacenar directamente las referencias URL clasificadas por el “MetaGenerador” a través de mensajes RDM, con lo cual se logrará una mayor integración de esta aplicación con los sistemas de indexación de algunos Robots de Búsqueda de Internet. Así mismo también con la funcionalidad de exportar la taxonomía procesada en formato HTML, esta herramienta de software puede también ser aplicada a la creación de contenidos para la construcción del material de aprendizaje de cursos en línea en proyectos de Universidad virtual, ó en la navegación de documentos en portales a través de categorías ó temas jerárquicos.

ANEXO A. CÓDIGO FUENTE

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("MetaGenerador.res");
USEFORM("Entrada.cpp", FrmEntrada);
USEFORM("GLocalMetaDato.cpp", FrmGenMD);
USEFORM("Acerca_De.cpp", AcercaDe);
USEFORM("BarraAvance.cpp", BusquedaPC);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
   try
   {
      Application->Initialize();
      Application->CreateForm(__classid(TFrmEntrada), &FrmEntrada);
      Application->CreateForm(__classid(TFrmGenMD), &FrmGenMD);
      Application->CreateForm(__classid(TAcercaDe), &AcercaDe);
      Application->CreateForm(__classid(TBusquedaPC), &BusquedaPC);
      Application->Run();
   }
   catch (Exception &exception)
   {
      Application->ShowException(&exception);
   }
   return 0;
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Acerca_De.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TAcercaDe *AcercaDe;
//---------------------------------------------------------------------------
__fastcall TAcercaDe::TAcercaDe(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Entrada.h"
#include "GLocalMetaDato.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFrmEntrada *FrmEntrada;
//---------------------------------------------------------------------------
__fastcall TFrmEntrada::TFrmEntrada(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFrmEntrada::ImagenEntradaClick(TObject *Sender)
{
    FrmEntrada->Hide();
    Release();
    FrmGenMD->Show();
}
//---------------------------------------------------------------------------
#include <string.h>
#include <stdio.h>

#include <vcl.h>
#pragma hdrstop

#include "GLocalMetaDato.h"
#include "Acerca_De.h"
#include "BarraAvance.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "SHDocVw_TLB"
#pragma link "cdiroutl"
#pragma resource "*.dfm"

int Flg1=0; //Ya se abrio archivo HTML (Flg1=1), Ya se creó un Meta Clas.(Flg1=2)
int Flg2=0; //Limitador de Tópicos seleccionados
int Flg3=0; //Encontro Meta de Clasificación el archivo HTML seleccionado
int Flg4=0; //Para adicionar Metas de clasificación
int Flg51=0; //Ya fue guardado el archivo HTML modificado (MGenrico)
int Flg52=0; //Ya fue guardado el archivo HTML modificado (MClasificación)
int Flg53=0; //Ya fue guardado el archivo HTML modificado (MPClave)
int Flg6=0; //Se creó una nueva taxonomía
int Flg7=1; //Busqueda de Palabras Clave (KEYWORD)
int Flg8=0; //Encontra palabra igual
int Flg9=0; //Encontro Meta de KeyWord en el archivo HTML seleccionado
int Flg10=0;//Para adicionar Metas de KeyWord
int Flg14=0; //Ya se creó un Meta de KeyWord
int Flg15=0; //Ya fue guardado el archivo HTML modificado (Meta KeyWords)
```

```
int FrRep=0;

FILE *in;
FILE *out;

char Line[1024];
char Linea[1024];
char LineaAux[1024];
char LineaAux1[1024];
char LineaAux2[1024];

char IdHijo[1024];
char IdPadre[1024];
char IdTaxo[1024];

char Meta[1024];
char NomHTML[520];
char NomHTMLa[520];

wchar_t buff[100];
char palabra[64];

TFrmGenMD *FrmGenMD;
//---------------------------------------------------------------------------
__fastcall TFrmGenMD::TFrmGenMD(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::SMenuSalirClick(TObject *Sender)
{
   if(Application->MessageBox("¿Esta seguro de que desea Terminar?","TERMINAR",
                MB_OKCANCEL)==IDOK)
   {
      if(Flg51==1)BotonGuardarHTMLpgClick(this);
      if(Flg52==1)SMGuardarDocumentoHTMLClick(this);
      if(Flg53==1)BotonGuardarClick(this);
      Application->Terminate();
   }
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::FormCloseQuery(TObject *Sender,
     bool &CanClose)
{
   if(Application->MessageBox("¿Esta seguro de que desea Terminar?","TERMINAR",
                MB_OKCANCEL)==IDOK)
   {
      if(Flg51==1)BotonGuardarHTMLpgClick(this);
      if(Flg52==1)SMGuardarDocumentoHTMLClick(this);
      if(Flg53==1)BotonGuardarClick(this);
      Application->Terminate();
   }
   else CanClose = false;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::SMAbrirTaxonomiaClick(TObject *Sender)
{
   if(Flg6==1)
   {
      BotonCrearMDC->Visible=true;BotonCrearMDC->Enabled=false;
      BotonAbrirHTML->Visible=true;BotonAbrirHTML->Enabled=false;
      BotonEditarHTML->Visible=true;BotonEditarHTML->Enabled=false;
      ArbolTaxonomia->Visible=true; ArbolTaxonomia->Enabled=false;
      LTaxCargada->Visible=true; LTaxCargada->Enabled=true;
      LTopicosSel->Visible=true; LTopicosSel->Enabled=true;
      LAbrirTax->Visible=true; LAbrirTax->Enabled=true;

      //SMAbrirTaxonomia->Enabled=false;BotonAbrirTax->Enabled=false;
      Flg6=0;
   }
   else {SMAbrirTaxonomia->Enabled=true;BotonAbrirTax->Enabled=true;}
   if (AbrirTaxonomia->Execute())
   {
      FILE *in;
      TTreeNode *Nodo;
      //SMAbrirTaxonomia->Enabled=false;BotonAbrirTax->Enabled=false;
      ArbolTaxonomia->Items->Clear();
      ListaTopicos->Clear();

      //char Line[512];
      //char LineAux[512];    //TreeView1->Selected
      //int Nivel=0;
      //int contL=0;

      in = fopen(AbrirTaxonomia->Files->Strings[0].c_str(), "rt");
      while(fgets(Line, sizeof(Line), in))
      {
         if(StrPos(Line,"@TAXONOMY"))
         {
            fgets(Line, sizeof(Line), in);*(StrPos(Line,"\n"))='\0';
            ArbolTaxonomia->Items->Add(NULL,StrPos(Line,":")+2);
         }
         if(StrPos(Line,"@CLASSIFICATION"))
         {
          fgets(IdHijo, sizeof(Line), in);*(StrPos(IdHijo,"\n"))='\0';     //Identificador de Hijo
          fgets(IdPadre, sizeof(Line), in);*(StrPos(IdPadre,"\n"))='\0';    //Identificador de Padre
          fgets(IdTaxo, sizeof(Line), in);*(StrPos(IdTaxo,"\n"))='\0';     //Identificador Taxonomia

          if(StrPos(IdPadre,"taxonomy"))
          {
            StrCopy(LineaAux1,IdPadre);
            StrCopy(IdPadre,IdTaxo);
            StrCopy(IdTaxo,LineaAux1);
          }
          if(StrPos(IdPadre,"ROOT"))
          {
            Nodo = ArbolTaxonomia->Items->Item[0];
            ArbolTaxonomia->Items->AddChild(Nodo,(StrPos(IdHijo,":")+2));
```

```
          }
          else
          {
            StrCopy(Linea,(StrPos(IdHijo,":")+2));
            for(;StrPos(Linea,":");)
            {
               StrCopy(LineaAux,(StrPos(Linea,":")+1));
               StrCopy(IdHijo,LineaAux);
               StrCopy(Linea,LineaAux);
            }
            StrCopy(LineaAux,(StrPos(IdPadre,":")+2));
            StrCopy(IdPadre,LineaAux);
            for(;StrPos(LineaAux,":");)
            {
               StrCopy(LineaAux1,(StrPos(LineaAux,":")+1));
               StrCopy(IdPadre,LineaAux1);
               StrCopy(LineaAux,LineaAux1);
            }

            for(int cnt=0;cnt<ArbolTaxonomia->Items->Count;cnt++)
                 if(ArbolTaxonomia->Items->Item[cnt]->Text==IdPadre)
                   Nodo = ArbolTaxonomia->Items->Item[cnt];
            ArbolTaxonomia->Items->AddChild(Nodo,IdHijo);
          }
         }
      }
      BotonAbrirHTML->Enabled=true;
      SMAbrirDocumentoHTML->Enabled=true;
      LAbrirTax->Visible=false;
      LContinuaHTML->Visible=true;
      fclose(in);
   }
   else {SMAbrirTaxonomia->Enabled=true;BotonAbrirTax->Enabled=true;}
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::ArbolTaxonomiaClick(TObject *Sender)
{
   BotonCrearMDC->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::ArbolTaxonomiaMouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if(Flg2<10)
   {
      int rep=0;

      THitTests HT;
      TTreeView *pTV = ArbolTaxonomia; //(TTreeView *)Sender;
      HT = pTV->GetHitTestInfoAt(X,Y);


      if (HT.Contains(htOnItem))
```

```
      {
         if(pTV->GetNodeAt(X,Y)==ArbolTaxonomia->Items->Item[0])return;
         StrCopy(LineaAux1," ");
         TTreeNode *PadreNodo = pTV->GetNodeAt(X,Y);
         for(;PadreNodo->Parent!=NULL;)
         {
           StrCat(LineaAux1,":");
           StrCat(LineaAux1,PadreNodo->Text.c_str());
           PadreNodo=PadreNodo->Parent;
         }
         StrCopy(LineaAux2,"");
         while(StrPos(LineaAux1,":"))
         {
           StrCat(LineaAux2,(StrRScan(LineaAux1,':')+1));
           *StrRScan(LineaAux1,':')='\0';
           if(StrPos(LineaAux1,":"))StrCat(LineaAux2,":");
         }
         for(int i=0;i<ListaTopicos->Items->Count;i++)
         {
            if(!StrComp(LineaAux2,ListaTopicos->Items->Strings[i].c_str()))
            {
             rep=1;
             Application->MessageBox("Este Tópico ya fué
seleccionado","TOPICOS",MB_OK);
            }
         }
         if(rep==0)
         {
            ListaTopicos->Items->Add(LineaAux2);
            Flg2++;
         }
      }
   }
   SMLimpiarVentanadeTopicos->Enabled=true;
   SMEliminarTopico->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMLimpiarVentanadeTopicosClick(
     TObject *Sender)
{
   Flg2=0;
   ListaTopicos->Clear();
   SMLimpiarVentanadeTopicos->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMEliminarTopicoClick(TObject *Sender)
{
   for(int i=0;i<ListaTopicos->Items->Count;i++)
   {
      if(ListaTopicos->Selected[i])
      {
         ListaTopicos->Items->Delete(i);
```

```
            Flg2--;
            break;
        }
    }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMFuenteHTMLClick(TObject *Sender)
{
    //HTML->ViewSource=!HTML->ViewSource;
    //if(HTML->ViewSource)
    //    SMFuenteHTML->Caption="Documento";
    //else
    //    SMFuenteHTML->Caption="Fuente";
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonCrearMDCClick(TObject *Sender)
{
    int i=0;
    while(i<=MemoHTMLS->Lines->Count)
    {
        StrCopy(Line,MemoHTMLS->Lines->Strings[i].c_str());
        StrLower(Line);
        if(StrPos(Line,"<meta name=\"classification\""))
        {
            Flg3=1;
            MemoMetas->Clear();
            MemoMetas->Lines->Append(Line);
            if(!StrPos(Line,"\">"))
            {
                StrCopy(Line,MemoHTMLS->Lines->Strings[++i].c_str());
                MemoMetas->Lines->Append(Line);
            }
        }
        i++;
    }

    int In=0;
    if(ListaTopicos->Items->Count==0)
    {
        if(Application->MessageBox("NO HAY TÓPICO SELECCIONADO",
                                   "TÓPICOS",MB_OK)==IDOK)
        {
            return;
        }
    }
    if(Application->MessageBox("¿Esta seguro de que desea Crear Meta?",
                               "META DE CLASIFICACIÓN",MB_OKCANCEL)==IDOK)
    {
        //int i;
        //in = fopen(AbrirHTMLmg->Files->Strings[0].c_str(), "r+");
        MemoHTMLS->Visible=false;
```

```
      out = fopen("Temp.cls", "w+");

      if(Flg3==1)
      {
         if(Application->MessageBox("    ¿Desea Eliminar el META ENCONTRADO?",
                  "META DE CLASIFICACIÓN ENCONTRADO",MB_OKCANCEL)==IDOK)
         {
         }
         else
         {
            if( Application->MessageBox
                  ("¿Desea dejar el META ENCONTRADO y adicionar Tópicos
seleccionados?",
                   "META DE CLASIFICACIÓN ENCONTRADO",MB_OKCANCEL)==IDOK)
            {
               Flg3=0;Flg4=1;
            }
            else return;
         }
      }

      if(Flg4==0)
      {
         while(In<=MemoHTMLS->Lines->Count)
         {
            StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
            StrCopy(Linea,Line);
            StrLower(Linea);
            fputs(Line,out);
            if(StrPos(Linea,"<head>"))
            {
               fputs("<meta name=\"Classification\"\n",out);
               fputs("content=\"",out);
               break;
            }
            else In++;
         }

         int m=0;//char *c;
         do
         {
            //StrCopy(Meta,ArbolTaxonomia->Items->Item[0]->Text.c_str());
            //c=StrEnd(Meta);c=c-1;
            //for(i=0;Meta!=c;c--)
            //fputc(Meta[i++],out);
            fputs(ArbolTaxonomia->Items->Item[0]->Text.c_str(),out);
            fputs(":",out);

            //StrCopy(Meta,ListaTopicos->Items->Strings[m].c_str());
            //c=StrEnd(Meta);c=c-1;
            //for(i=0;Meta!=c;c--)
            //fputc(Meta[i++],out);
            fputs(ListaTopicos->Items->Strings[m].c_str(),out);
```

```
        if(m==ListaTopicos->Items->Count-1)fputs("\">\n",out);
        else  fputs(";",out);
        m++;
    }while(m<ListaTopicos->Items->Count);
    //fputs(Meta,out);
    In++;
}
else
{
    while(In<=MemoHTMLS->Lines->Count)
    {
        StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
        StrCopy(Linea,Line);
        StrLower(Linea);
        if(StrPos(Linea,"<meta name=\"classification\""))
        {
            char *c;
            int i=0;
            while(!StrPos(Line,"\">"))
            {
                fputs(Line,out);
                In++;
                StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
            }
            c=StrEnd(Line);c=c-3;
            for(;Line!=c;c--)
            fputc(Line[i++],out);
            fputs(";",out);
            int m=0;
            do
            {
                //StrCopy(Meta,ArbolTaxonomia->Items->Item[0]->Text.c_str());
                //c=StrEnd(Meta);c=c-1;
                //for(i=0;Meta!=c;c--)
                //    fputc(Meta[i++],out);
                fputs(ArbolTaxonomia->Items->Item[0]->Text.c_str(),out);
                fputs(":",out);

                //StrCopy(Meta,ListaTopicos->Items->Strings[m].c_str());
                //c=StrEnd(Meta);c=c-1;
                //for(i=0;Meta!=c;c--)
                //    fputc(Meta[i++],out);
                fputs(ListaTopicos->Items->Strings[m].c_str(),out);

                if(m==ListaTopicos->Items->Count-1)fputs("\">\n",out);
                else  fputs(";",out);
                m++;
            }while(m<ListaTopicos->Items->Count);
            //fputs(Meta,out);
        }
        else fputs(Line,out);
        In++;
    }
    Flg4=0;
```

```
}

if(Flg3==1)
{
   while(In<=MemoHTMLS->Lines->Count)
   {
      StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
      StrCopy(Linea,Line);
      StrLower(Linea);
      if(StrPos(Linea,"<meta name=\"classification\""))
      {
         while(!StrPos(Line,"\">"))
         {
            In++;
            StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
         }
         if(StrPos(Line,"\">"))
         {
            In++;
            StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
         }
      }
      fputs(Line,out);
      In++;
   }
   Flg3=0;//MemoMetas->Clear();
}
else
{
   while(In<=MemoHTMLS->Lines->Count)
   {
      StrCopy(Line,MemoHTMLS->Lines->Strings[In].c_str());
      fputs(Line,out);
      In++;
   }
}

//fclose(in);
fclose(out);

MemoHTMLS->Clear();
in = fopen("Temp.cls", "rt");
//out = fopen(AbrirHTML->Files->Strings[0].c_str(), "w+");
out = fopen("TempAux.cls", "w+");
while(fgets(Line, sizeof(Line), in))fputs(Line,out);
fclose(in);
fclose(out);

MemoGenericos->Clear();
MemoHTMLS->Clear();
MemoMetasPC->Clear();
in = fopen("TempAux.cls", "rt");
while(fgets(Line, sizeof(Line), in))
{
```

```
            MemoGenericos->Lines->Append(Line);
            MemoHTMLS->Lines->Append(Line);
            MemoMetasPC->Lines->Append(Line);
        }
        fclose(in);
        Flg52=0;

        MemoHTMLS->Visible=true;LHTMLGenerado->Visible=true;
        BotonCrearMDC->Enabled=false;
        //BotonOtraTax->Enabled=true;
        //BotonMetaOtroArch->Enabled=true;
        //ListaTopicos->Enabled=false;
        //ArbolTaxonomia->Enabled=false;
        LExpTaxo->Visible=false;LExpTopicos->Visible=false;
        Flg1=2;BotonEditarHTML->Enabled=true;
        SMGuardarDocumentoHTML->Enabled=true;
        BotonGuardarHTMLclas->Enabled=true;
        Flg52=1;
    }
    else {ListaTopicos->Enabled=true;Flg1=1;}
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMGuardarDocumentoHTMLClick(TObject *Sender)
{
    if( Application->MessageBox
        ("¿Desea Guardar el Archivo HTML modificado, con Meta Dato de Clasificación?",
        "GUARDAR HTML MODIFICADO",MB_OKCANCEL)==IDOK)
    {
      GuardarHTML->FileName=AbrirHTMLmg->Files->Strings[0].c_str();
      if (GuardarHTML->Execute())
      {
        FILE *out;
        out = fopen(GuardarHTML->Files->Strings[0].c_str(), "w+");

        int i=0;
        while(i<=MemoHTMLS->Lines->Count)
        {
            fputs(MemoHTMLS->Lines->Strings[i].c_str(),out);
            i++;
        }
        fclose(out);
        Flg52=0;

        SMGuardarDocumentoHTML->Enabled=false;
        //Button6->Enabled=false;
        MemoHTMLS->ReadOnly=true;
        //Abrir2->Enabled=true;Button5->Enabled=true;
      }
    }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonEditarHTMLClick(TObject *Sender)
```

```
{
   MemoHTMLS->ReadOnly=false;
   Flg52=0;
   Flg1=2;
   //SMGuardarDocumentoHTML->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonMetaOtroArchClick(TObject *Sender)
{
   if(Flg52==1&&Flg1==2)FrmGenMD->SMGuardarDocumentoHTMLClick(this);
   Flg1=1;
   SMAbrirDocumentoHTMLmgClick(this);
   MemoHTMLS->Visible=false;
   ListaTopicos->Enabled=true;
   ArbolTaxonomia->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonOtraTaxClick(TObject *Sender)
{
   if(Flg52==1&&Flg1==2)FrmGenMD->SMGuardarDocumentoHTMLClick(this);
   //if(Flg15!=1)Form1->Button13Click(this);
   Flg1=0;
   BotonAbrirTax->Enabled=true;
   SMAbrirTaxonomia->Enabled=true;
   MemoHTMLS->Visible=false;LHTMLGenerado->Visible=false;
   HTML->Visible=false;LHTMLSel->Visible=false;
   SMFuenteHTML->Enabled=false;
   MemoHTMLS->Clear();
   ArbolTaxonomia->Enabled=false;
   ListaTopicos->Enabled=true;
   BotonCrearMDC->Enabled=false;
   BotonEditarHTML->Enabled=false;
   LAbrirTax->Visible=true;
   //Button4->Enabled=false;

   ArbolTaxonomia->Items->Clear();
   ListaTopicos->Clear();Flg2=0;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMFuenteHTMLpcClick(TObject *Sender)
{
   //HTMLpc->ViewSource=!HTMLpc->ViewSource;
   //if(HTMLpc->ViewSource)
   //    SMFuenteHTMLpc->Caption="Documento";
   //else
   //    SMFuenteHTMLpc->Caption="Fuente";
}
```

```
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonBuscarPCClick(TObject *Sender)
{
   //fpos_t filepos;

   //char palabrac[64];
   //char Linea[512];
   FrRep=MaskERepeticion->Text-0;
   //int Rep=0;
   //ListBox2->Items->Add(String(FrRep));
   int i=0;
   int tm=0,tm1=0,min=0,seg=0;
   node *root = NULL;

   float tprc = 0.0018f;
   tm= MemoMetasPC->Lines->Count*tprc/60;
   min=((MemoMetasPC->Lines->Count*tprc/60-tm)*100)/60+tm;
   tm1= ((MemoMetasPC->Lines->Count*tprc/60-tm)*100)/60;
   seg=(((MemoMetasPC->Lines->Count*tprc/60-tm)*100)/60-tm1)*60;
   BusquedaPC->LTmpEstm->Caption=min;
   BusquedaPC->LTmpEsts->Caption=seg;

   BusquedaPC->BarraAvPClave->Max=MemoMetasPC->Lines->Count;
   BusquedaPC->BarraAvPClave->Position=0;
   BusquedaPC->Show();BusquedaPC->Update();
   FrmGenMD->Enabled=false;

   //out = fopen("tkeyw.cls", "wt+");
   while(i<=MemoMetasPC->Lines->Count)
   {
      StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
      StrLower(Line);
      if(StrPos(MemoMetasPC->Lines->Strings[i].c_str(),"body"))
         break;
      i++;
      BusquedaPC->BarraAvPClave->StepBy(1);
      BusquedaPC->LLineaProc->Caption=BusquedaPC->BarraAvPClave->Position;
      BusquedaPC->LTotLin->Caption=MemoMetasPC->Lines->Count;
      BusquedaPC->Update();BusquedaPC->Repaint();
   }
   //fclose(out);
   MemoMetasPC->Visible=true;

   //in = fopen("tkeyw.cls","rt");

   i++;
   while(i<=MemoMetasPC->Lines->Count)
   {
      StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
      //StrLower(Line);

     while(Line[0]!=NULL&&Line[0]!='\n')
```

```
    {
      int x=0,y=0;
      for(;Line[x]==' '||Line[x]=='!'||Line[x]=='"'||Line[x]=='#'
         ||Line[x]=='%'||Line[x]=='&'||Line[x]=='/'||Line[x]=='('
         ||Line[x]==')'||Line[x]=='='||Line[x]=='?'||Line[x]=='¿'
         ||Line[x]=='¡'||Line[x]=='*'||Line[x]=='~'||Line[x]=='\\'
         ||Line[x]=='+'||Line[x]=='-'||Line[x]=='_'||Line[x]==','
         ||Line[x]=='.'||Line[x]==';'||Line[x]==':'||Line[x]=='\''
         ||Line[x]=='@'||Line[x]=='$'||Line[x]=='®'||Line[x]>='0'&&Line[x]<='9';x++);
      StrCopy(Line,Line+x);
      for(x=0;Line[x]=='<';)
      {
         if(StrPos(Line,">"))
            StrCopy(Line,StrPos(Line,">")+1);
         else
         {
            Line[x]=NULL;i++;
            break;
         }
      }

      for(x=0;Line[x]!=' '&&Line[x]!='<'&&Line[x]>='A'&&Line[x]<='Z'
         ||Line[x]>='a'&&Line[x]<='z'||Line[x]=='ü'||Line[x]=='é'
         ||Line[x]=='â'||Line[x]=='ä'||Line[x]=='à'||Line[x]=='å'
         ||Line[x]=='ê'||Line[x]=='ë'||Line[x]=='è'||Line[x]=='ï'
         ||Line[x]=='î'||Line[x]=='ì'||Line[x]=='Ä'||Line[x]=='Å'
         ||Line[x]=='É'||Line[x]=='ô'||Line[x]=='ö'||Line[x]=='ò'
         ||Line[x]=='û'||Line[x]=='ù'||Line[x]=='ÿ'||Line[x]=='Ö'
         ||Line[x]=='Ü'||Line[x]=='á'||Line[x]=='í'||Line[x]=='ó'
         ||Line[x]=='ú'||Line[x]=='ñ'||Line[x]=='Ñ'||Line[x]=='Á'
         ||Line[x]=='Â'||Line[x]=='À';x++,y++)palabra[y]=Line[x];
      if(y>0)
      {
         palabra[y]=NULL;StrCopy(Line,Line+y);
         root = addnode(root);
      }
    }
    i++;
    FrmGenMD->Update();
    BusquedaPC->BarraAvPClave->StepBy(1);
    BusquedaPC->LLineaProc->Caption=BusquedaPC->BarraAvPClave->Position;
    BusquedaPC->Update(); BusquedaPC->Repaint();
  }
  crealista(FrRep,ListaPCEncontradas,root);
  ListaPCEncontradas->Enabled=true;
  BotonBuscarPC->Enabled=false;
  BotonGenerarMPC->Enabled=true;
  BusquedaPC->Hide();
  FrmGenMD->Enabled=true;
}

node *TFrmGenMD::addnode(node *p)
{
  int code;
```

```
  if(p == NULL)
  {  p = (node *)malloc(sizeof(node));
     p->pword = (char *)malloc(strlen(palabra) + 1);
     strcpy(p->pword,palabra);
     p->count = 1; p->left = p->right = NULL;
  } else
  {
     code = strcmp(palabra, p->pword);
     if(code < 0)p->left = addnode(p->left); else
     if(code > 0)p->right = addnode(p->right); else
     p->count++;
  }

  return p;
}

void TFrmGenMD::crealista(int FrRep,TListBox *ListaPCEncontradas,node *p)
{
   if(p != NULL)
   {
     crealista(FrRep,ListaPCEncontradas,p->left);
     if(p->count>=FrRep)ListaPCEncontradas->Items->Add(p->pword);
     crealista(FrRep,ListaPCEncontradas,p->right);
   }
}

//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::MaskERepeticionChange(TObject *Sender)
{

   Flg7=1,Flg8=0;
   BotonBuscarPC->Enabled=true;
   BotonEditar->Enabled=false;
   BotonGuardar->Enabled=false;
   ListaPCEncontradas->Clear();
   //ListBox3->Clear();
   //MemoMetasPC->Visible=false;
   //MemoMPCenc->Visible=false;
   //LHTMLMpc->Visible=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonGenerarMPCClick(TObject *Sender)
{
   int i=0;
   while(i<=MemoMetasPC->Lines->Count)
   {
      StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
      StrLower(Line);
      if(StrPos(Line,"<meta name=\"keywords\""))
      {
         Flg9=1;
         MemoMPCenc->Clear();
```

```
        MemoMPCenc->Lines->Append(Line);
        if(!StrPos(Line,"\">"))
        {
           StrCopy(Line,MemoMetasPC->Lines->Strings[++i].c_str());
           MemoMPCenc->Lines->Append(Line);
        }
     }
     i++;
  }


  if(ListaPCEncontradas->SelCount==0)
  {
     if(Application->MessageBox("NO HAY PALABRAS SELECCIONADAS",
                              "PALABRAS CLAVE",MB_OK)==IDOK)
     {
        return;
     }
  }
  LHTMLMpc->Enabled=true;
  MemoMPCenc->Enabled=true;


     if(Application->MessageBox("¿Esta seguro de que desea Generar el Meta?",
                        "PALABRAS CLAVE",MB_OKCANCEL)==IDOK)
     {
        //int i;

        //in = fopen(OpenDialog1->Files->Strings[0].c_str(), "r+");
        out = fopen("Temp.cls", "w+");

        if(Flg9==1)
        {
           if(Application->MessageBox("    ¿Desea Eliminar el META ENCONTRADO?",
                 "META DE KEYWORD ENCONTRADO",MB_OKCANCEL)==IDOK)
           {
           }
           else
           {
              if( Application->MessageBox
                 ("¿Desea dejar el META ENCONTRADO y adicionar Tópicos
seleccionados?",
                  "META DE KEYWORD ENCONTRADO",MB_OKCANCEL)==IDOK)
              {
                 Flg9=0;Flg10=1;
              }
              else return;
           }
        }

        if(Flg10==0)
        {
           i=0;
           while(MemoMetasPC->Lines)
```

```
    {
       StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
       StrCopy(Linea,Line);
       StrLower(Linea);
       fputs(Line,out);
       if(StrPos(Linea,"<head>"))
       {
          fputs("<meta name=\"Keywords\"\n",out);
          fputs("content=\"",out);
          break;
       }
       i++;
    }
    i++;
    int m=0,FlgI=0;char *c;
    do
    {
       if(ListaPCEncontradas->Selected[m])
       {
          if(FlgI!=0)fputs(",",out);
          StrCopy(Meta,ListaPCEncontradas->Items->Strings[m].c_str());
          c=StrEnd(Meta);//c=c-1;
          for(int ii=0;Meta!=c;c--)
             fputc(Meta[ii++],out);
          FlgI=1;
       }
       m++;

    }while(m<ListaPCEncontradas->Items->Count);
    fputs("\">\n",out);
    //fputs(Meta,out);

}
else
{
    i=0;
    while(i<=MemoMetasPC->Lines->Count)
    {
       StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
       StrCopy(Linea,Line);
       StrLower(Linea);
       if(StrPos(Linea,"<meta name=\"keywords\""))
       {
          char *c;
          int ii=0;

          //fputs(Linea,out);
          if(!StrPos(Line,"\">"))
          {
             fputs(Linea,out);
             StrCopy(Line,MemoMetasPC->Lines->Strings[++i].c_str());
          }
          c=StrEnd(Line);c=c-3;
          for(;Line!=c;c--)
```

```
                    fputc(Line[ii++],out);
                fputs(",",out);
                int m=0,FlgI=0;
                do
                {
                    if(ListaPCEncontradas->Selected[m])
                    {
                        if(FlgI!=0)fputs(",",out);
                        StrCopy(Meta,ListaPCEncontradas->Items->Strings[m].c_str());
                        c=StrEnd(Meta);//c=c-1;
                        for(ii=0;Meta!=c;c--)
                            fputc(Meta[ii++],out);
                        FlgI=1;
                    }
                    m++;
                }while(m<ListaPCEncontradas->Items->Count);
                fputs("\">\n",out);
                //fputs(Meta,out);
            }
            else fputs(Line,out);
            i++;
        }
        Flg10=0;
    }


    if(Flg9==1)
    {
        //i=0;
        while(i<=MemoMetasPC->Lines->Count)
        {
            StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
            StrCopy(Linea,Line);
            StrLower(Linea);
            if(StrPos(Linea,"<meta name=\"keywords\""))
            {
                //fputs(Line,out);
                if(!StrPos(Line,"\">"))
                {
                    StrCopy(Line,MemoMetasPC->Lines->Strings[++i].c_str());
                }
                if(StrPos(Line,"\">"))StrCopy(Line,MemoMetasPC->Lines-
>Strings[++i].c_str());
            }
            fputs(Line,out);
            i++;
        }
        Flg9=0;MemoMPCenc->Clear();
    }
    else
    {
        i=0;
        while(i<=MemoMetasPC->Lines->Count)
            {
```

```
                StrCopy(Line,MemoMetasPC->Lines->Strings[i].c_str());
                fputs(Line,out);
                i++;
            }
        }

        //fclose(in);
        fclose(out);

        MemoGenericos->Clear();
        MemoHTMLS->Clear();
        MemoMetasPC->Clear();
        in = fopen("Temp.cls", "rt");
        while(fgets(Line, sizeof(Line), in))
        {
           MemoGenericos->Lines->Append(Line);
           MemoHTMLS->Lines->Append(Line);
           MemoMetasPC->Lines->Append(Line);
        }
        MemoMetasPC->Visible=true;

        fclose(in);
        Flg15=0; Flg14=1;

        BotonGenerarMPC->Enabled=false;
        BotonEditar->Enabled=true;
        BotonGuardar->Enabled=true;
        SMGuardarDocumentoHTMLpc->Enabled=true;
        Flg53=1;
      }
      //else {MemoMetasPC->Enabled=false;LHTMLMpc->Enabled=false;}
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonEditarClick(TObject *Sender)
{
   MemoMetasPC->ReadOnly=false;
   BotonEditar->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonGuardarClick(TObject *Sender)
{
   if( Application->MessageBox
      ("¿Desea Guardar el Archivo HTML modificado, con Meta Datos de P. Clave?",
      "GUARDAR HTML MODIFICADO",MB_OKCANCEL)==IDOK)
   {
     GuardarHTML->FileName=AbrirHTMLmg->Files->Strings[0].c_str();
     if (GuardarHTML->Execute())
     {
       FILE *out;
       out = fopen(GuardarHTML->Files->Strings[0].c_str(), "w+");

       int i=0;
```

```
        while(i<=MemoMetasPC->Lines->Count)
        {
           fputs(MemoMetasPC->Lines->Strings[i].c_str(),out);
           i++;
        }
        fclose(out);
        Flg53=0;

        MemoMetasPC->ReadOnly=true;
        BotonEditar->Enabled=true;
        SMGuardarDocumentoHTMLpc->Enabled=false;
        BotonGuardar->Enabled=false;
      }
    }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonNuevoHTMLClick(TObject *Sender)
{
    if(Flg15!=1&&Flg14==1)FrmGenMD->BotonGuardarClick(this);
    Flg7=1;Flg8=0;Flg9=0;Flg10=0;Flg14=0;Flg15=0;
    BotonBuscarPC->Enabled=false;
    BotonBuscarPC->Enabled=true;
    BotonGenerarMPC->Enabled=false;
    BotonEditar->Enabled=false;
    BotonGuardar->Enabled=false;
    BotonAbrirHTMLpc->Enabled=true;
    SMFuenteHTMLpc->Enabled=false;
    SBAbrirDocumentoHTMLpc->Enabled=true;
    SMGuardarDocumentoHTMLpc->Enabled=false;
    ListaPCEncontradas->Clear();ListaPCEncontradas->Enabled=false;
    //CrearPalabrasClaveClick(this);
    MemoMetasPC->Clear();MemoMetasPC->Visible=false;
    MemoMPCenc->Clear();MemoMPCenc->Enabled=false;
    MaskERepeticion->Enabled=false;
    LMDPCEncontrados->Enabled=false;
    LHTMLMpc->Enabled=false;
    LRepticiones->Enabled=false;
    LHTMLpc->Enabled=false;
}
//---------------------------------------------------------------------------


void __fastcall TFrmGenMD::FolderMClasificacionShow(TObject *Sender)
{
    SMenuGenerico->Enabled=false;
    SMenuClasificacion->Enabled=true;
    SMenuPalabrasClave->Enabled=false;
    BotonAbrirTax->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::FolderMDPClaveShow(TObject *Sender)
{
```

```
    SMenuGenerico->Enabled=false;
    SMenuClasificacion->Enabled=false;
    SMenuPalabrasClave->Enabled=true;
    BotonAbrirTax->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::FolderMGenericoShow(TObject *Sender)
{
    SMenuGenerico->Enabled=true;
    SMenuClasificacion->Enabled=false;
    SMenuPalabrasClave->Enabled=false;
    BotonAbrirTax->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::MenuMDatoClick(TObject *Sender)
{
    FoldersGMetaDatos->Visible=true;
    FolderClasifRemoto->Visible=false;
    FolderNavegador->Visible=false;
    BotonMDGenerico->Enabled=true;
    BotonMDClasificacion->Enabled=true;
    BotonMDPClave->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::MenuCRemotoClick(TObject *Sender)
{
    FoldersGMetaDatos->Visible=false;
    FolderClasifRemoto->Visible=true;
    FolderNavegador->Visible=false;
    BotonMDGenerico->Enabled=false;
    BotonMDClasificacion->Enabled=false;
    BotonMDPClave->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::MenuNavegadorClick(TObject *Sender)
{
    FoldersGMetaDatos->Visible=false;
    FolderClasifRemoto->Visible=false;
    FolderNavegador->Visible=true;
    BotonMDGenerico->Enabled=false;
    BotonMDClasificacion->Enabled=false;
    BotonMDPClave->Enabled=false;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMenuAcercadeClick(TObject *Sender)
{
    AcercaDe->ShowModal();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::SMAbrirDocumentoHTMLmgClick(TObject *Sender)
{
   //Acciones del MetaClasificación
   ArbolTaxonomia->Enabled=true;
   ListaTopicos->Visible=true;
   MemoMetas->Clear();
   LContinuaHTML->Visible=false;
   BotonEditarHTML->Enabled=false;

   if (AbrirHTMLmg->Execute())
   {
      in = fopen(AbrirHTMLmg->Files->Strings[0].c_str(), "r+");
      StrCopy(NomHTML,AbrirHTMLmg->Files->Strings[0].c_str());
      StrCopy(NomHTMLa,"file:///");
      StrCat(NomHTMLa,NomHTML);
      //HTMLMGenericos->RequestDoc(NomHTMLa);
      //HTML->RequestDoc(NomHTMLa);
      //HTMLpc->RequestDoc(NomHTMLa);

      URLComboBox->Text= NomHTMLa+8;
      MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,
         NomHTMLa,-1,buff,sizeof(buff));
      HTMLMGenericos->Navigate(buff,0,0,0,0);
      HTML->Navigate(buff,0,0,0,0);
      HTMLpc->Navigate(buff,0,0,0,0);
      Navegador->Navigate(buff,0,0,0,0);
      URLComboBox->Items->Insert(0,NomHTMLa);
      URLComboBox->Text= NomHTMLa;
      EditURLdeNvg->Text= NomHTMLa;

      MemoMetasGEnc->Clear();
      MemoMetasPC->Clear();MemoMetasPC->Visible=false;
      ListaPCEncontradas->Clear();
      MemoHTMLS->Clear();
      ETitulo->Clear();EAutor->Clear();EGenerador->Clear();
      EDescripcion->Clear();
      MemoHTML->Clear();
      MemoGenericos->Clear();
      MemoGenericos->Visible=false;
      MemoGenericos->ReadOnly=true;
      SMGuardarDocumentoHTMLmg->Enabled=false;

      //Recupera HTML para Meta Dato Genérico
      while(fgets(Line, sizeof(Line), in))
      {
         StrCopy(Linea,Line);
         StrLower(Linea);
         if(StrPos(Linea,"<title>"))
         {
           char Texto[64],TextoAux[64];int i=0;
           StrCopy(TextoAux,(StrPos(Line,">")+1));
           for(;TextoAux[i]!='<';i++)Texto[i]=TextoAux[i];
```

```
  Texto[i]='\0';
  ETitulo->Text=Texto;
  MemoMetasGEnc->Lines->Append(Line);
}
if(StrPos(Linea,"<meta name=\"author\""))
{
   LMGencontrados->Enabled=true;
   while(!StrPos(Line,"\">"))
   {
      MemoMetasGEnc->Lines->Append(Line);
      MemoHTML->Lines->Append(Line);
      fgets(Line, sizeof(Line), in);
   }
   if(StrPos(Line,"\">"))
   {
      char Texto[64],TextoAux[64];int i=0;
      StrCopy(TextoAux,(StrRScan(Line,'=')+2));
      for(;TextoAux[i]!='\"';i++)Texto[i]=TextoAux[i];
      Texto[i]='\0';
      EAutor->Text=Texto;
      MemoMetasGEnc->Lines->Append(Line);
   }
}
if(StrPos(Linea,"<meta name=\"lenguaje\""))
{
   LMGencontrados->Enabled=true;
   while(!StrPos(Line,"\">"))
   {
      MemoMetasGEnc->Lines->Append(Line);
      MemoGenericos->Lines->Append(Line);
      fgets(Line, sizeof(Line), in);
   }
   if(StrPos(Line,"\">"))
   {
      char Texto[64],TextoAux[64];int i=0;
      StrCopy(TextoAux,(StrRScan(Line,'=')+2));
      for(;TextoAux[i]!='\"';i++)Texto[i]=TextoAux[i];
      Texto[i]='\0';
      //EAutor->Text=Texto;
      MemoMetasGEnc->Lines->Append(Line);
   }
}
if(StrPos(Linea,"<meta name=\"generator\""))
{
   LMGencontrados->Enabled=true;
   while(!StrPos(Line,"\">"))
   {
      MemoMetasGEnc->Lines->Append(Line);
      MemoGenericos->Lines->Append(Line);
      fgets(Line, sizeof(Line), in);
   }
   if(StrPos(Line,"\">"))
   {
      char Texto[64],TextoAux[64];int i=0;
```

```
            StrCopy(TextoAux,(StrRScan(Line,'=')+2));
            for(;TextoAux[i]!='\"';i++)Texto[i]=TextoAux[i];
            Texto[i]='\0';
            EGenerador->Text=Texto;
            MemoMetasGEnc->Lines->Append(Line);
         }
      }
      if(StrPos(Linea,"<meta name=\"description\""))
      {
         LMGencontrados->Enabled=true;
         while(!StrPos(Line,"\">"))
         {
            MemoMetasGEnc->Lines->Append(Line);
            MemoGenericos->Lines->Append(Line);
            fgets(Line, sizeof(Line), in);
         }
         if(StrPos(Line,"\">"))
         {
            char Texto[64],TextoAux[64];int i=0;
            StrCopy(TextoAux,(StrRScan(Line,'=')+2));
            for(;TextoAux[i]!='\"';i++)Texto[i]=TextoAux[i];
            Texto[i]='\0';
            EDescripcion->Text=Texto;
            MemoMetasGEnc->Lines->Append(Line);
         }
      }
      if(StrPos(Linea,"<meta name=\"date\""))
      {
         LMGencontrados->Enabled=true;
         while(!StrPos(Line,"\">"))
         {
            MemoMetasGEnc->Lines->Append(Line);
            MemoGenericos->Lines->Append(Line);
            fgets(Line, sizeof(Line), in);
         }
         if(StrPos(Line,"\">"))
         {
            char Texto[64],TextoAux[64];int i=0;
            StrCopy(TextoAux,(StrRScan(Line,'=')+2));
            for(;TextoAux[i]!='\"';i++)Texto[i]=TextoAux[i];
            Texto[i]='\0';
            //EFecha->Date=StrToDate(Texto);
            MemoMetasGEnc->Lines->Append(Line);
         }
      }
      MemoGenericos->Lines->Append(Line);
   }

   //Recupera HTML para Meta Dato de Clasificación
   rewind(in);
   while(fgets(Line, sizeof(Line), in))
   {
      StrCopy(Linea,Line);
      StrLower(Linea);
```

```
    if(StrPos(Linea,"<meta name=\"classification\""))
    {
       Flg3=1;
       MemoMetas->Clear();

       while(!StrPos(Line,"\">"))
       {
          MemoMetas->Lines->Append(Line);
          MemoHTMLS->Lines->Append(Line);
          fgets(Line, sizeof(Line), in);
       }
       if(StrPos(Line,"\">"))
       {
          MemoMetas->Lines->Append(Line);
          MemoHTMLS->Lines->Append(Line);
       }

       //fgets(Line, sizeof(Line), in);
    }
    else MemoHTMLS->Lines->Append(Line);
}

//Recupera HTML para Meta P.Clave
rewind(in);
while(fgets(Line, sizeof(Line), in))
{
    StrCopy(Linea,Line);
    StrLower(Linea);
    if(StrPos(Linea,"<meta name=\"keywords\""))
    {
       //Flg3=1;
       MemoMPCenc->Clear();

       while(!StrPos(Line,"\">"))
       {
          MemoMPCenc->Lines->Append(Line);
          fgets(Line, sizeof(Line), in);
       }
       if(StrPos(Line,"\">"))
       {
          MemoMPCenc->Lines->Append(Line);
       }

       fgets(Line, sizeof(Line), in);
    }
}
rewind(in);
while(fgets(Line, sizeof(Line), in))MemoMetasPC->Lines->Append(Line);
rewind(in);

fclose(in);
//SMAbrirDocumentoHTMLmg->Enabled=false;
SMFuenteHTMLmg->Enabled=true;
BotonCrearMGen->Enabled=true;
```

```
        BotonGuardarHTMLpg->Enabled=false;
        SMGuardarDocumentoHTMLmg->Enabled=false;
        GBoxGenericos->Enabled=true;
        LTitulo->Enabled=true;LAutor->Enabled=true;
        LGenerador->Enabled=true;LFecha->Enabled=true;
        LDescripcion->Enabled=true;LLenguaje->Enabled=true;

        //Acciones Meta Clasificación
        MemoMetas->Visible=true;LMDEncontrado->Visible=true;
        HTML->Visible=true;LHTMLSel->Visible=true;
        SMFuenteHTML->Enabled=true;
        BotonGuardarHTMLclas->Enabled=false;
        SMGuardarDocumentoHTML->Enabled=false;
        Flg1=1;
        //BotonAbrirHTML->Enabled=false;
        //SMAbrirDocumentoHTML->Enabled=false;
        LExpTaxo->Visible=true;LExpTopicos->Visible=true;
        LContinuaHTML->Visible=false;

        //Acciones Meta Palabras Clave
        MemoMPCenc->Visible=true;LMDPCEncontrados->Visible=true;
        LHTMLpc->Enabled=true;
        SMGuardarDocumentoHTMLpc->Enabled=false;
        BotonGuardar->Enabled=false;
        SMFuenteHTMLpc->Enabled=true;
        //MemoHTMpc->Visible=true;
        BotonBuscarPC->Enabled=true;
        //BotonAbrirHTMLpc->Enabled=false;
        //SBAbrirDocumentoHTMLpc->Enabled=false;
        SMFuenteHTMLpc->Enabled=true;
        LRepticiones->Enabled=true;
        LMDPCEncontrados->Enabled=true;
        MemoMPCenc->Enabled=true;
        MaskERepeticion->Enabled=true;
    }
    else
    {
        //ArbolTaxonomia->Enabled=false;
        LContinuaHTML->Visible=true;
        LExpTaxo->Visible=false;LExpTopicos->Visible=false;
    }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::SMFuenteHTMLmgClick(TObject *Sender)
{
    //HTMLMGenericos->ViewSource=!HTMLMGenericos->ViewSource;
    //if(HTMLMGenericos->ViewSource)
    //    SMFuenteHTMLmg->Caption="Documento";
   // else
    //    SMFuenteHTMLmg->Caption="Fuente";
}
//---------------------------------------------------------------------------
```

```
void __fastcall TFrmGenMD::BotonMDGenericoClick(TObject *Sender)
{
   FoldersGMetaDatos->ActivePage=FolderMGenerico;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonMDClasificacionClick(TObject *Sender)
{
   FoldersGMetaDatos->ActivePage=FolderMClasificacion;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonMDPClaveClick(TObject *Sender)
{
   FoldersGMetaDatos->ActivePage=FolderMDPClave;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonCrearMGenClick(TObject *Sender)
{
   int NoL=0;
   out = fopen("Temp.cls", "w+");

   //MemoGenericos->Clear();
   while(NoL < MemoGenericos->Lines->Count)
   {
    StrCopy(Line,MemoGenericos->Lines->Strings[NoL].c_str());
    StrCopy(Linea,Line);
    StrLower(Linea);
    if(StrPos(Linea,"<meta name=\"author\""))
    {
     while(!StrPos(Line,"\">"))
     {
      NoL++;
     }
     NoL++;
    }
    else
    {
     if(StrPos(Linea,"<meta name=\"generator\""))
     {
      while(!StrPos(Line,"\">"))
      {
       NoL++;
      }
      NoL++;
     }
     else
     {
      if(StrPos(Linea,"<meta name=\"lenguaje\""))
      {
       while(!StrPos(Line,"\">"))
       {
```

```
          NoL++;
         }
         NoL++;
        }
        else
        {
         if(StrPos(Linea,"<meta name=\"date\""))
         {
          while(!StrPos(Line,"\">"))
          {
           NoL++;
          }
          NoL++;
         }
         else
         {
          if(StrPos(Linea,"<meta name=\"description\""))
          {
           while(!StrPos(Line,"\">"))
           {
            NoL++;
           }
           NoL++;
          }
          else
          {
           if(StrPos(Linea,"<title>"))
           {
            fputs("<title>",out);fputs(ETitulo->Text.c_str(),out);fputs("</title>\n",out);
            NoL++;
           }
           else
           {
            fputs(Line,out);
            if(StrPos(Linea,"<head>"))
            {
             fputs("<meta name=\"author\" content=\"",out);fputs(EAutor-
>Text.c_str(),out);fputs("\">\n",out);
             fputs("<meta name=\"generator\" content=\"",out);fputs(EGenerador-
>Text.c_str(),out);fputs("\">\n",out);
             fputs("<meta name=\"lenguaje\" content=\"",out);fputs(StrRScan(ComboLenguaje-
>Text.c_str(),' '),out);fputs("\">\n",out);
             fputs("<meta name=\"date\" content=\"",out);fputs(DateToStr(EFecha-
>Date).c_str(),out);fputs("\">\n",out);
             fputs("<meta name=\"description\" content=\"",out);fputs(EDescripcion-
>Text.c_str(),out);fputs("\">\n",out);
            }
            NoL++;
           }
          }
         }
        }
       }
      }
```

```
    }

    fclose(out);

    MemoGenericos->Clear();
    MemoHTMLS->Clear();
    MemoMetasPC->Clear();
    in = fopen("Temp.cls", "rt");
    while(fgets(Line, sizeof(Line), in))
    {
        MemoGenericos->Lines->Append(Line);
        MemoHTMLS->Lines->Append(Line);
        MemoMetasPC->Lines->Append(Line);
    }
    MemoGenericos->Visible=true;
    fclose(in);

    BotonEditarHTMLFinal->Enabled=true;
    BotonGuardarHTMLpg->Enabled=true;
    Flg51=1;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonEditarHTMLFinalClick(TObject *Sender)
{
    MemoGenericos->ReadOnly=false;
    BotonEditarHTMLFinal->Enabled=false;
}
//---------------------------------------------------------------------------



void __fastcall TFrmGenMD::MemoGenericosChange(TObject *Sender)
{
    if(SMGuardarDocumentoHTMLmg->Enabled==false)
    {
        SMGuardarDocumentoHTMLmg->Enabled=true;
        BotonGuardarHTMLpg->Enabled=true;
    }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonGuardarHTMLpgClick(TObject *Sender)
{
    if( Application->MessageBox
        ("¿Desea Guardar el Archivo HTML modificado para Meta Datos Genéricos?",
        "GUARDAR HTML MODIFICADO",MB_OKCANCEL)==IDOK)
    {
      GuardarHTML->FileName=AbrirHTMLmg->Files->Strings[0].c_str();
      if (GuardarHTML->Execute())
      {
        FILE *out;
        out = fopen(GuardarHTML->Files->Strings[0].c_str(), "w+");
```

```
        int i=0;
        while(i<=MemoGenericos->Lines->Count)
        {
           fputs(MemoGenericos->Lines->Strings[i].c_str(),out);
           i++;
        }
        fclose(out);
        Flg51=0;

        MemoGenericos->ReadOnly=true;
        BotonEditarHTMLFinal->Enabled=true;
        SMGuardarDocumentoHTMLmg->Enabled=false;
        BotonGuardarHTMLpg->Enabled=false;
     }
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::URLComboBoxKeyPress(TObject *Sender,
     char &Key)
{
   if(Key == VK_RETURN)
   {
      Key=0;
      MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,
         URLComboBox->Text.c_str(),-1,buff,sizeof(buff));
      Navegador->Navigate(buff,0,0,0,0);
      URLComboBox->Items->Insert(0,URLComboBox->Text);
      EditURLdeNvg->Text=URLComboBox->Text;
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::MemoHTMLSChange(TObject *Sender)
{
   if(SMGuardarDocumentoHTML->Enabled==false)
   {
      SMGuardarDocumentoHTML->Enabled=true;
      BotonGuardarHTMLclas->Enabled=true;
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::ListaPCEncontradasClick(TObject *Sender)
{
   BotonGenerarMPC->Enabled=true;
   //MemoMetasPC->Visible=false;
}
//---------------------------------------------------------------------------


void __fastcall TFrmGenMD::MemoMetasPCChange(TObject *Sender)
{
   if(SMGuardarDocumentoHTMLpc->Enabled==false)
```

```
   {
      SMGuardarDocumentoHTMLpc->Enabled=true;
      BotonGuardar->Enabled=true;
   }
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::AbrirTaxonoma1Click(TObject *Sender)
{
   if (AbrirTaxonomia->Execute())
   {
      FILE *in;
      TTreeNode *Nodo;
      ArbolClasRemoto->Items->Clear();

      char Line[512];

      in = fopen(AbrirTaxonomia->Files->Strings[0].c_str(), "rt");

      StatusBar2->SimpleText="¡ CARGANDO TAXONOMIA espere un momento...!";
      while(fgets(Line, sizeof(Line), in))
      {
         if(StrPos(Line,"@TAXONOMY"))
         {
            fgets(Line, sizeof(Line), in);*(StrPos(Line,"\n"))='\0';
            ArbolClasRemoto->Items->Add(NULL,StrPos(Line,":")+2);
         }
         if(StrPos(Line,"@CLASSIFICATION"))
         {
          fgets(IdHijo, sizeof(Line), in);*(StrPos(IdHijo,"\n"))='\0';     //Identificador de Hijo
          fgets(IdPadre, sizeof(Line), in);*(StrPos(IdPadre,"\n"))='\0';    //Identificador de Padre
          fgets(IdTaxo, sizeof(Line), in);*(StrPos(IdTaxo,"\n"))='\0';     //Identificador Taxonomia

          if(StrPos(IdPadre,"taxonomy"))
          {
            StrCopy(LineaAux1,IdPadre);
            StrCopy(IdPadre,IdTaxo);
            StrCopy(IdTaxo,LineaAux1);
          }
          if(StrPos(IdPadre,"ROOT"))
          {
            Nodo = ArbolClasRemoto->Items->Item[0];
            ArbolClasRemoto->Items->AddChild(Nodo,(StrPos(IdHijo,":")+2));
          }
          else
          {
            StrCopy(Linea,(StrPos(IdHijo,":")+2));
            for(;StrPos(Linea,":");)
            {
               StrCopy(LineaAux,(StrPos(Linea,":")+1));
               StrCopy(IdHijo,LineaAux);
               StrCopy(Linea,LineaAux);
            }
            StrCopy(LineaAux,(StrPos(IdPadre,":")+2));
            StrCopy(IdPadre,LineaAux);
```

```
            for(;StrPos(LineaAux,":");)
            {
               StrCopy(LineaAux1,(StrPos(LineaAux,":")+1));
               StrCopy(IdPadre,LineaAux1);
               StrCopy(LineaAux,LineaAux1);
            }

            for(int cnt=0;cnt<ArbolClasRemoto->Items->Count;cnt++)
                 if(ArbolClasRemoto->Items->Item[cnt]->Text==IdPadre)
                   Nodo = ArbolClasRemoto->Items->Item[cnt];
            ArbolClasRemoto->Items->AddChild(Nodo,IdHijo);
          }
         }
      }
      fclose(in);
      StatusBar2->SimpleText=" TAXONOMIA CARGADA TOTALMENTE";
      BotonExpTax->Enabled=true;
   }
}
//---------------------------------------------------------------------------


void __fastcall TFrmGenMD::ArbolClasRemotoExpanded(TObject *Sender,
     TTreeNode *Node)
{
   //Node->ImageIndex=1;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::ArbolClasRemotoCollapsed(TObject *Sender,
     TTreeNode *Node)
{
   //Node->ImageIndex=0;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonAddNodoClick(TObject *Sender)
{
   if(ArbolClasRemoto->Selected==NULL)
      ShowMessage("¡ No hay tema seleccionado !");
   else
      if(ArbolClasRemoto->Selected->ImageIndex!=2&&Navegador->LocationURL!=NULL)
      {
        ArbolClasRemoto->Items->AddChildObject(ArbolClasRemoto->Selected,Navegador-
>LocationURL,Navegador)->ImageIndex=2;
        BotonGenerarTax->Enabled=true;
      }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonElimNodoClick(TObject *Sender)
{
  if(ArbolClasRemoto->Selected==NULL)
    ShowMessage("¡ No hay tema seleccionado !");
```

```
  else
    if(ArbolClasRemoto->Selected->getFirstChild()==NULL)
    {
        if( Application->MessageBox
        ("¿Esta seguro de que desea eliminar?",
        "GUARDAR HTML MODIFICADO",MB_OKCANCEL)==IDOK)
        {
            ArbolClasRemoto->Items->Delete(ArbolClasRemoto->Selected);
            BotonGenerarTax->Enabled=true;
        }
    }
    else
    {
        if( Application->MessageBox
        ("El Nodo seleccionado tiene descendencia\n¿Desea eliminarlo con todo su
contenido?",
        "GUARDAR HTML MODIFICADO",MB_OKCANCEL)==IDOK)
        {
          ArbolClasRemoto->Items->Delete(ArbolClasRemoto->Selected);
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::ArbolClasRemotoChange(TObject *Sender,
      TTreeNode *Node)
{
    //Node->ImageIndex=2;
    //StatusBar2->SimpleText=Node->Text+Node->ImageIndex;
    EditURLaNvg->Text=Node->Text;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::ButtonAbrirHTMLNavdorClick(TObject *Sender)
{
  if(ArbolClasRemoto->Selected==NULL)
    ShowMessage("No hay TAXONOMIA creada");
  else
    if(ArbolClasRemoto->Selected->ImageIndex==2)
    {
        MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,
            ArbolClasRemoto->Selected->Text.c_str(),-1,buff,sizeof(buff));
        Navegador->Navigate(buff,0,0,0,0);
        URLComboBox->Items->Insert(0,ArbolClasRemoto->Selected->Text);
        URLComboBox->Text= ArbolClasRemoto->Selected->Text;
        EditURLdeNvg->Text= ArbolClasRemoto->Selected->Text;
        MenuNavegadorClick(this);
    }
    else Application->MessageBox("El elemento seleccionado no es un Docuemnto
HTML","ERROR",MB_OK);
}
//---------------------------------------------------------------------------


void __fastcall TFrmGenMD::URLComboBoxClick(TObject *Sender)
```

```
{
   if(URLComboBox->Text != "")
   {
      MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,
         URLComboBox->Text.c_str(),-1,buff,sizeof(buff));
      Navegador->Navigate(buff,0,0,0,0);
      EditURLdeNvg->Text=URLComboBox->Text;
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::NavegadorDownloadBegin(
      TObject *Sender)
{
   //StatusBar1->SimpleText= " Conectando a " +  URLComboBox->Text + "...";
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::NavegadorDownloadComplete(
      TObject *Sender)
{
   //StatusBar1->SimpleText= " Conectado";
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::NavegadorBeforeNavigate(
      TObject *Sender, BSTR URL, long Flags, BSTR TargetFrameName,
      Variant *PostData, BSTR Headers, VARIANT_BOOL *Cancel)
{
      StatusBar1->SimpleText= " Conectando a " +  URLComboBox->Text + "...";
      //StatusBar1->SimpleText= URL;
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::NavegadorNavigateComplete(
      TObject *Sender, BSTR URL)
{
   StatusBar1->SimpleText= " Conectado";
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonAddTemaClick(TObject *Sender)
{
   if(ArbolClasRemoto->Selected==NULL)
      ShowMessage("¡ No hay Nodo seleccionado !");
   else
   {
      if(ArbolClasRemoto->Selected->ImageIndex!=2)
      {
         ArbolClasRemoto->Items->AddChild(ArbolClasRemoto->Selected,"Tema Nuevo")-
>EditText();
         BotonGenerarTax->Enabled=true;
      }
   }
```

```
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonGuardarTaxDClick(TObject *Sender)
{
   if (GuardarTaxD->Execute())
   {
      ArbolClasRemoto->SaveToFile(GuardarTaxD->Files->Strings[0].c_str());
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonExpanderTaxClick(TObject *Sender)
{
   ArbolClasRemoto->FullExpand();
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonContraerTaxClick(TObject *Sender)
{
   ArbolClasRemoto->FullCollapse();
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonAbrirTaxDClick(TObject *Sender)
{
   if (AbrirTaxD->Execute())
   {
      ArbolClasRemoto->Items->Clear();
      ArbolClasRemoto->LoadFromFile(AbrirTaxD->Files->Strings[0].c_str());
      for(int id=0;id<ArbolClasRemoto->Items->Count;id++)
      {
        if(StrPos(ArbolClasRemoto->Items->Item[id]->Text.c_str(),"http")
           || StrPos(ArbolClasRemoto->Items->Item[id]->Text.c_str(),"file:///"))
          ArbolClasRemoto->Items->Item[id]->ImageIndex=2;
      }
      BotonExpTax->Enabled=true;
   }
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonExpTaxClick(TObject *Sender)
{
  int Nodo=0;

  //char buffer[256];
  //GetWindowsDirectory(buffer, sizeof(buffer));
  //AnsiString asFileName = FileSearch("TaxNavegador.html", GetCurrentDir() +
AnsiString(";") + AnsiString(buffer));
  //if (asFileName.IsEmpty())
  //  ShowMessage(AnsiString("No se encuentra archivo") + "TaxNavegador.html" + ".");
  //else
  //  ShowMessage(AnsiString("Se encontro ") + ExtractFileDir(asFileName) + ".");
```

```
  out = fopen("ArbolTax.dat", "w+");
  while(Nodo<ArbolClasRemoto->Items->Count)
  {
    int NoPdrs=1;
    TTreeNode *PadreNodo = ArbolClasRemoto->Items->Item[Nodo];
    for(;PadreNodo->Parent!=NULL;NoPdrs++)
        PadreNodo=PadreNodo->Parent;
    StrCopy(Linea,IntToStr(NoPdrs).c_str());
    StrCat(Linea," \"");
    StrCat(Linea,ArbolClasRemoto->Items->Item[Nodo]->Text.c_str());
    if(ArbolClasRemoto->Items->Item[Nodo]->ImageIndex==2)
    {
        StrCat(Linea,"\" \"");
        StrCat(Linea,ArbolClasRemoto->Items->Item[Nodo]->Text.c_str());
        StrCat(Linea,"\" 0 \"OVU1.gif\"\n\0");
    }
    else StrCat(Linea,"\" \"\" 0\n\0");
    fputs(Linea,out);
    Nodo++;
  }
  fclose(out);

  char buffer1[MAXPATH];
  getcwd(buffer1, MAXPATH);
  StrCat(buffer1,"\\TaxNavegador.html");
  //ShowMessage(buffer1);

  MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,
    buffer1,-1,buff,sizeof(buff));
  Navegador->Navigate(buff,0,0,0,0);
  URLComboBox->Items->Insert(0,buffer1);
  URLComboBox->Text= buffer1;
  EditURLdeNvg->Text= buffer1;
  MenuNavegadorClick(this);
}
//---------------------------------------------------------------------------
void __fastcall TFrmGenMD::ArbolClasRemotoDblClick(TObject *Sender)
{
   ButtonAbrirHTMLNavdorClick(this);
}
//---------------------------------------------------------------------------

void __fastcall TFrmGenMD::BotonGenerarTaxClick(TObject *Sender)
{
   if (GuardarTax->Execute())
   {
      out = fopen(GuardarTax->Files->Strings[0].c_str(), "w+");
      fputs("@TAXONOMY { -\n",out);
      StrCopy(Linea,"id{0}:\t");
      StrCat(Linea,ArbolClasRemoto->Items->Item[0]->Text.c_str());
      StrCat(Linea,"\n}\n");
      fputs(Linea,out);
      for(int i=1;i<ArbolClasRemoto->Items->Count;i++)
```

```
      {
        fputs("@CLASSIFICATION { -\n",out);
        StrCopy(LineaAux,"id{");StrCat(LineaAux,IntToStr(i).c_str());
        StrCat(LineaAux,"}:");
        StrCopy(LineaAux1," ");
        TTreeNode *PadreNodo = ArbolClasRemoto->Items->Item[i];
        for(;PadreNodo->Parent!=NULL;)
        {
          StrCat(LineaAux1,":");
          StrCat(LineaAux1,PadreNodo->Text.c_str());
          PadreNodo=PadreNodo->Parent;
        }
        StrCopy(LineaAux2,"\t");
        while(StrPos(LineaAux1,":"))
        {
           StrCat(LineaAux2,(StrRScan(LineaAux1,':')+1));
           *StrRScan(LineaAux1,':')='\0';
           if(StrPos(LineaAux1,":"))StrCat(LineaAux2,":");
        }
        StrCat(LineaAux,LineaAux2);
        StrCat(LineaAux,"\n");
        fputs(LineaAux,out);
        if(StrPos(LineaAux2,":"))
        {
          *StrRScan(LineaAux2,':')='\0';
          StrCopy(LineaAux,"parent-id{");
          //StrCopy(LineaAux,"id{");StrCat(LineaAux,IntToStr(i).c_str());  PENDIENTE
          StrCat(LineaAux,"}:");
          StrCat(LineaAux,LineaAux2);
        }
        else   //El padre es la Raiz de la Taxonomía
        {
          StrCopy(LineaAux,"parent-id{");
          //StrCopy(LineaAux,"id{");StrCat(LineaAux,IntToStr(i).c_str());  PENDIENTE
          StrCat(LineaAux,"}:\tROOT");
        }
        StrCat(LineaAux,"\n");
        fputs(LineaAux,out);
        StrCopy(LineaAux1,StrPos(Linea,"\t"));
        StrCopy(Linea,"taxonomy-id{0}:");
        StrCat(Linea,LineaAux1);
        fputs(Linea,out);
      }
      fclose(out);
    }
}
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "BarraAvance.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TBusquedaPC *BusquedaPC;
//---------------------------------------------------------------------------
__fastcall TBusquedaPC::TBusquedaPC(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TBusquedaPC::FormClose(TObject *Sender,
      TCloseAction &Action)
{
   return;
}
//---------------------------------------------------------------------------
```

ANEXO B. MANUAL DE USUARIO

INSTALACIÓN

1. Colocar CD de instalación en unidad de CD-ROM.
2. Espere a que el programa de arranque se execute automáticamente.
3. Siga las instrucciones que le indica el instalador.

ARRANQUE

4. 1. Seleccione el ícono del METAGenerador en el escritorio de su PC.
5. 2. Dar doble clic y el programa cargará la siguiente imagen:

1. 3. Dar un clic sobre esta imagen y el METAGenerador se ejecutará, desplegándose la ventana principal como se observa en la Figura B.1.
2.

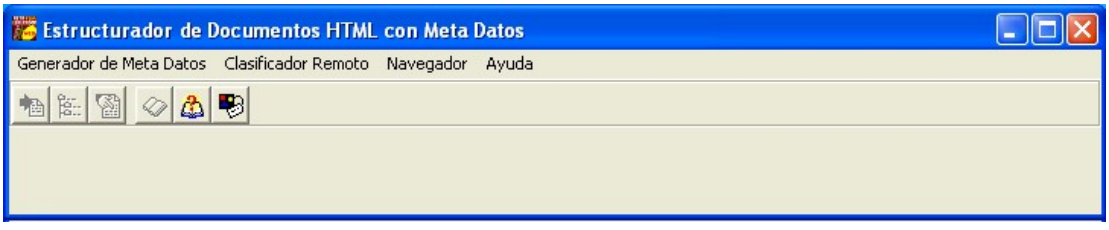


Figura B.1. Ventana principal del MetaGenerador

GENERACIÓN DE META DATOS

1. Seleccione del menú principal “Generador de Meta Datos”.

   De la ventana principal seleccione la carpeta identificada como “Meta Datos Genéricos”.

   Dar un clic sobre esta carpeta, espere a que se despliegue la ventana de trabajo que muestra tres fólderes, desplegándose por omisión la que corresponde a la captura de Meta Datos genéricos. Ver Figura B.2.

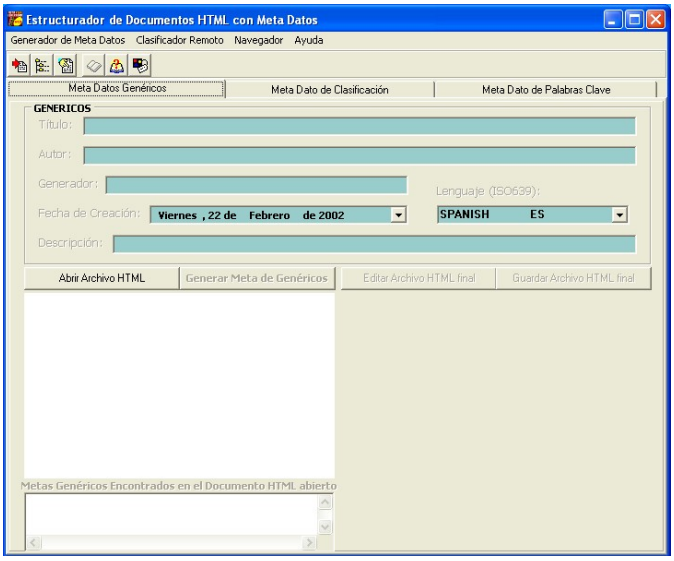


Figura B.2. Folder para la generación del Meta Dato Genérico

Abrir un documento HTML donde desea insertar los Meta Datos generados seleccionado el botón identificado como “Abrir Archivo HTML”. Ver Figura B.3.

Figura B.3. Ventana para buscar documento HTML

El programa detecta en forma automática si el documento HTML abierto contiene Meta Datos genéricos, pero puede capturar los campos de acuerdo a la información que desea. Ver Figura B.4.

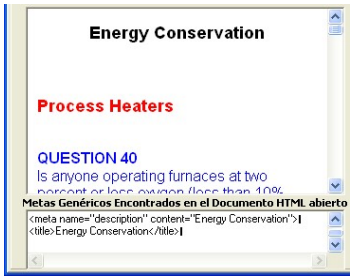


Figura B.4. Componente del navegador y área de texto del Meta Dato encontrado

De un clic en el botón identificado como “Generar Meta de Genéricos”.

Seleccione las opciones que el programa le pida para generar el Meta Dato.

Una vez generado el meta de genéricos, podrá editar el código HTML para modificaciones seleccionando el botón identificado como “Editar Archivo HTML Final”. Ver Figura B.5.

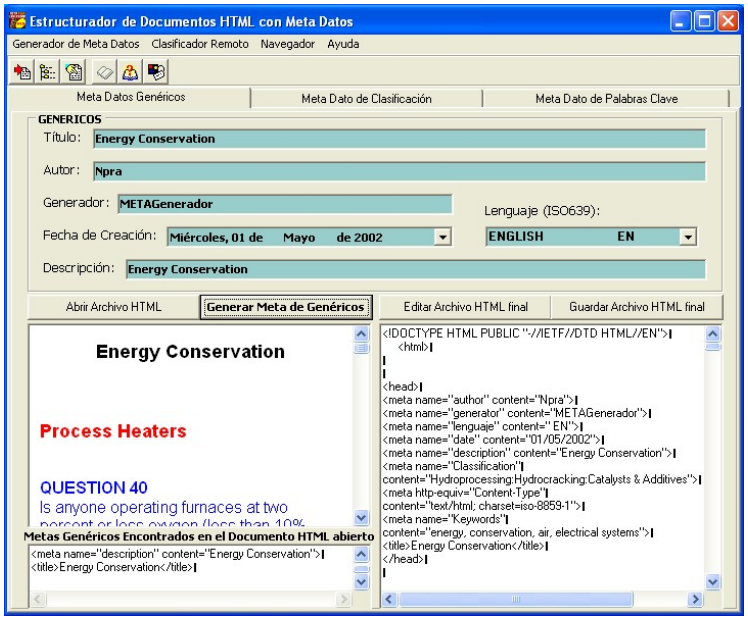


Figura B.5. Meta Datos Genéricos generados

Seleccionando el botón identificado como “Guardar Archivo HTML Final” para salvar los Meta Datos genéricos creados.

Meta Dato de Clasificación

1. De la ventana principal seleccione la carpeta identificada como “Meta Dato de Clasificación”.
2. Dar un clic sobre esta carpeta, espere a que se despliegue la ventana de trabajo que muestra la ventana que corresponde a la generación del Meta Dato de clasificación. Ver Figura B.6.

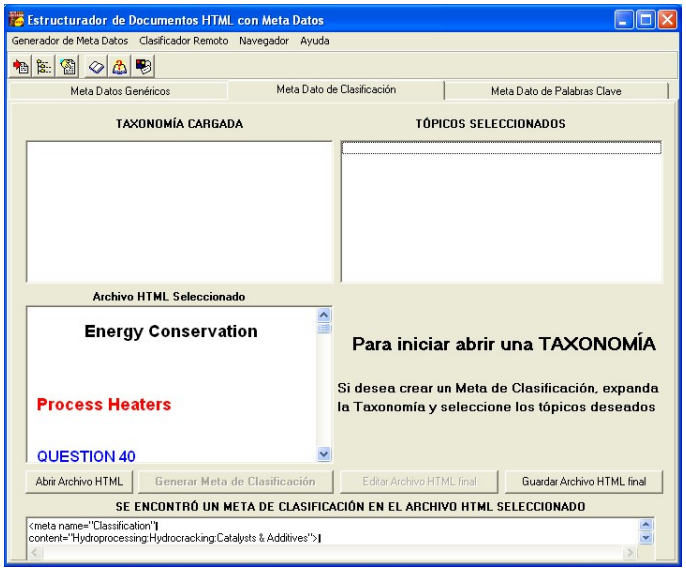


Figura B.6. Fólder para la generación del Meta Dato de Clasificación

3. Del menú “Clasificación” seleccionar el submenú “Abrir Taxonomía” y abrir un archivo de taxonomía de donde se seleccionarán los tópicos que correspondan a la clasificación del documento HTML deseado. Como se muestra en la Figura B.7.

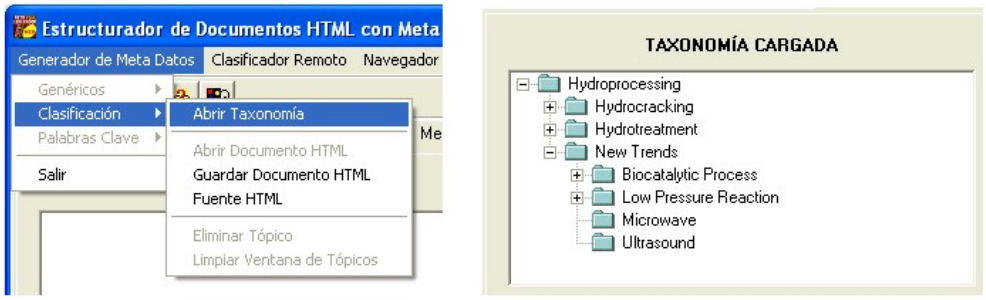


Figura B.7. Cargado de archivo de taxonomía

4. Abrir un documento HTML donde desea insertar el Meta Dato generado seleccionando el botón identificado como “Abrir Archivo HTML” o puede trabajar sobre el documento abierto actualmente.
5. Expanda el árbol taxonómico y seleccione los tópicos de interés, como se observa en la Figura B.8.

Figura B.8. Selección de tópicos del árbol taxonómico

6. Genere el Meta Dato de clasificación seleccionando el botón correspondiente.
7. El programa detecta en forma automática si el documento HTML abierto contiene Meta Dato de clasificación y solicita seleccionar de las opciones para eliminar el Meta Dato encontrado o adicionar a éste la clasificación que corresponde a los tópicos seleccionados, como se indica en la Figura B.9.

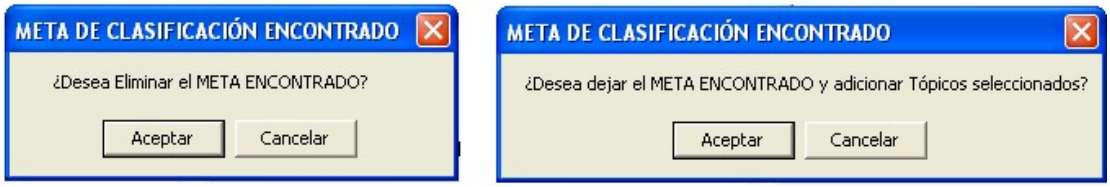


Figura B.9. Ventanas de diálogo para determinar la generación el Meta Dato

8. Una vez generado el meta de clasificación, podrá editar el código HTML para modificaciones seleccionando el botón identificado como "Editar Archivo HTML Final". Ver Figura B.10.

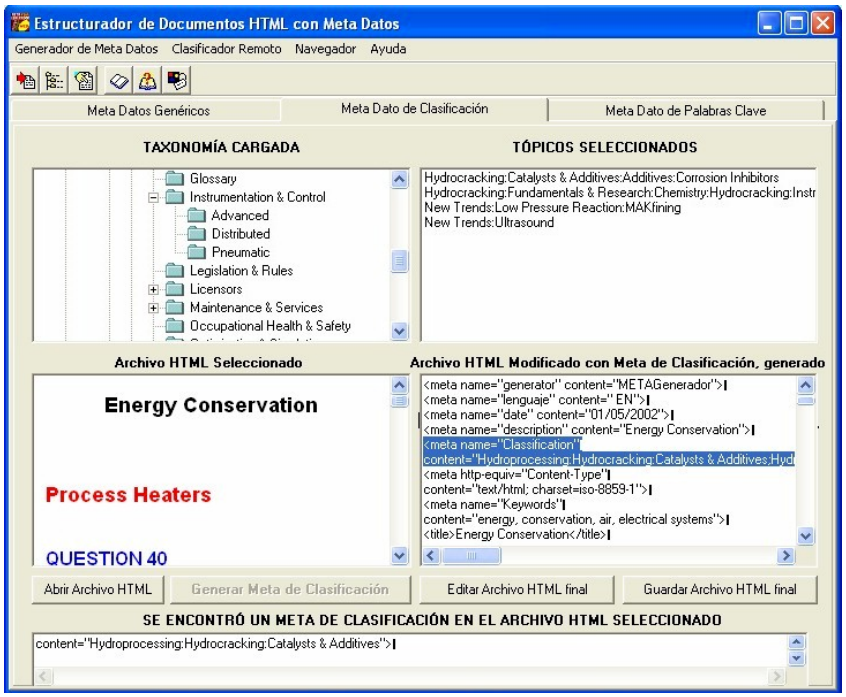


Figura B.10. Meta Dato de Clasificación generado

9. Seleccionando el botón identificado como "Guardar Archivo HTML Final" para salvar los Meta Datos genéricos creados.

Meta Dato de Palabras Clave

De la ventana principal seleccionar la carpeta identificada como “Meta Dato de Palabras Clave”.

Dar un clic sobre esta carpeta, espere a que se despliegue la ventana de trabajo que muestra la ventana que corresponde a la generación del Meta Dato de palabras clave. Como se muestra en la Figura B.11.

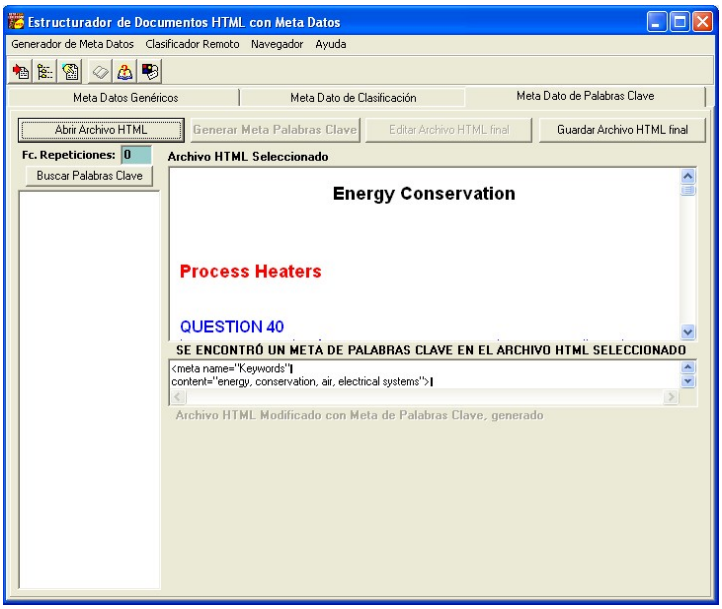


Figura B.11. Fólder para la gneración del Meta Dato de Palabras Clave

Abrir un documento HTML donde desea insertar el Meta Dato generado seleccionado el botón identificado como “Abrir Archivo HTML” o puede trabajar sobre el documento abierto actualmente.

Seleccione la frecuencia de repetición para la búsqueda de palabras clave en el documento HTML abierto y capturar en el campo identificado como “Fc. Repeticiones”. Ver Figura B.12.

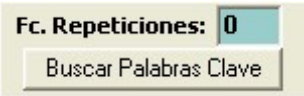


Figura B.12. Selección de la frecuencia de repetición

Realizar la búsqueda de palabras clave seleccionando el botón identificado como “Buscar Palabras Clave”. Ver Figura B.13.

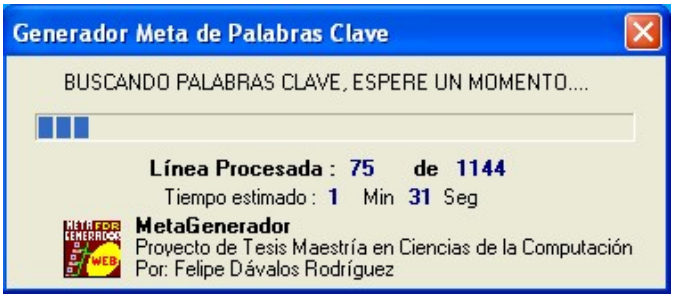


Figura B.13. Ventana que indica el avance de la búsqueda

Seleccionar de la lista las palabras clave que deseé insertar en el Meta Dato a generar, como se muestra en la Figura B.14.

Figura B.14. Lista de Palabras Clave encontradas

Generar el Meta Dato de palabras clave seleccionando el botón correspondiente.

El programa detecta en forma automática si el documento HTML abierto contiene Meta Dato de palabras clave y solicita seleccionar de las opciones para eliminar el Meta Dato encontrado o adicionar a éste la clasificación que corresponde a los tópicos seleccionados, como se indica en la Figura B.15.

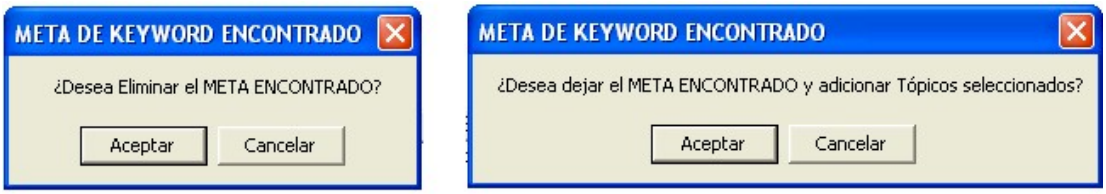


Figura B.15. Ventanas de diálogo para determinar la generación del Meta Dato.

Una vez generado el meta de clasificación, podrá editar el código HTML para modificaciones seleccionando el botón identificado como “Editar Archivo HTML Final”. Ver Figura B.16.

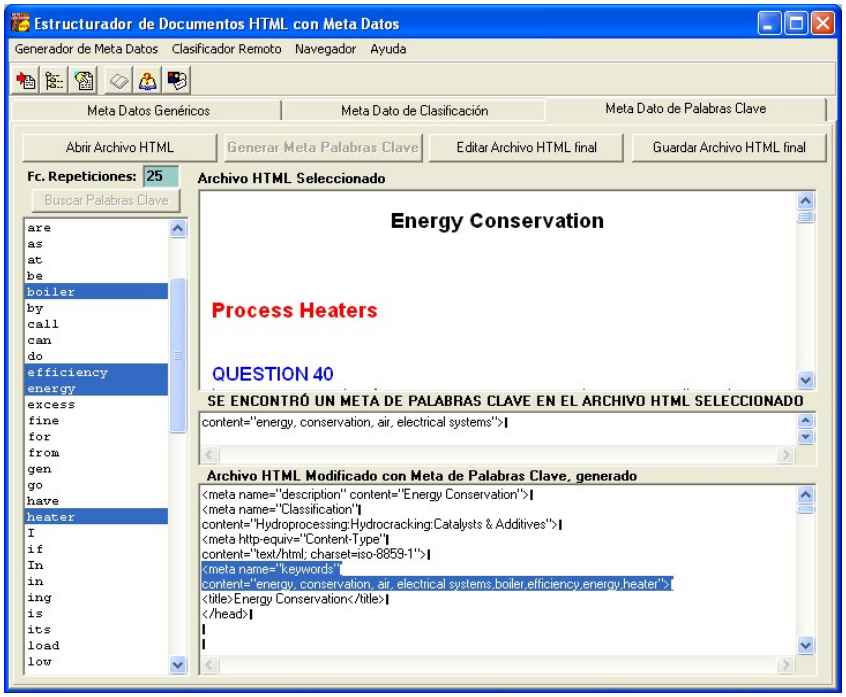


Figura B.16. Meta Dato generado

Seleccionando el botón identificado como “Guardar Archivo HTML Final” para salvar los Meta Datos genéricos creados.

CLASIFICADOR REMOTO

1. Seleccione del menú principal “Clasificador Remoto”, se mostrará la ventana de trabajo.
2. Del menú “Clasificador Remoto” seleccionar el submenú “Abrir Taxonomía” y cargar el archivo correspondiente, como se muestra en la Figura B.17.

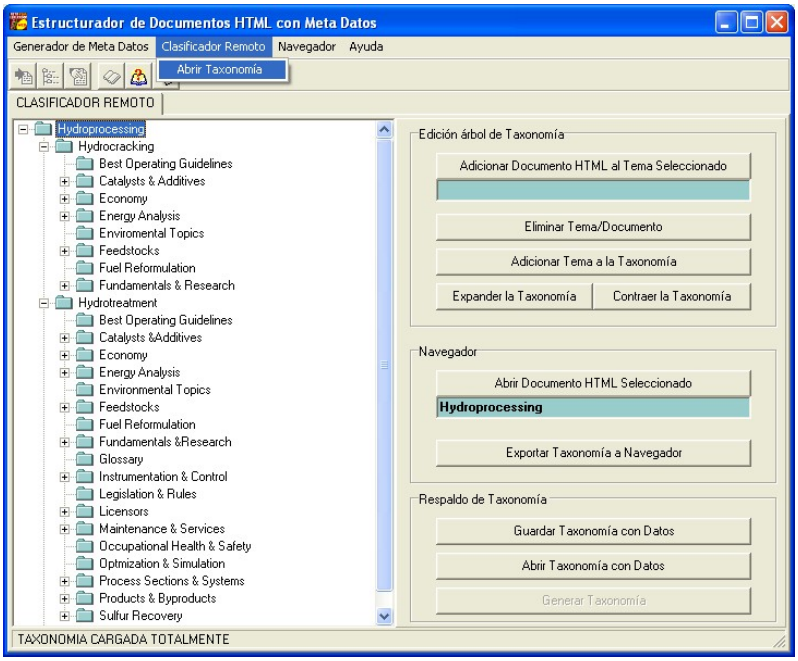


Figura B.17. Cargado de árbol taxonómico

3. La sección de “Edición árbol de Taxonomía” permite adicionar la referencia URL de un documento activo en el navegador, eliminar o adicionar nuevos temas a la taxonomía, así como expandir o contraer el árbol taxonómico.
4. La sección “Navegador” permite abrir en el navegador alguna referencia que se seleccione del árbol taxonómico, como se muestra en la Figura B.18.

Figura B.18. Ventana del navegador

5. Así también es posible exportar la taxonomía a formato HTML y enviarlo al navegador, como se muestra en la Figura B.19.

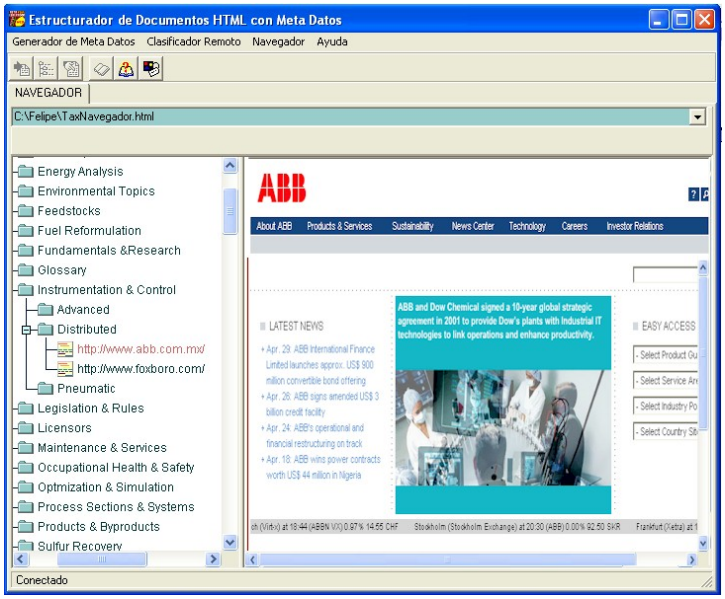


Figura B.19. Taxonomía exportada a documento HTML

6. La sección “Respaldo de Taxonomía” permite salvar y recuperar la taxonomía editada con referencias URL en un formato propio de la aplicación.
7. En esta misma sección es posible crear nuevas taxonomías ya que permite generarlas en el formato convencional “rdm” para que sea leída por los robots de búsqueda.

NAVEGADOR

1. Seleccionar del menú principal “Navegador”, se mostrará la ventana de trabajo.
2. Este navegador es compatible 100% con el navegador Internet Explorer de Microsoft. Ver Figura B.20.

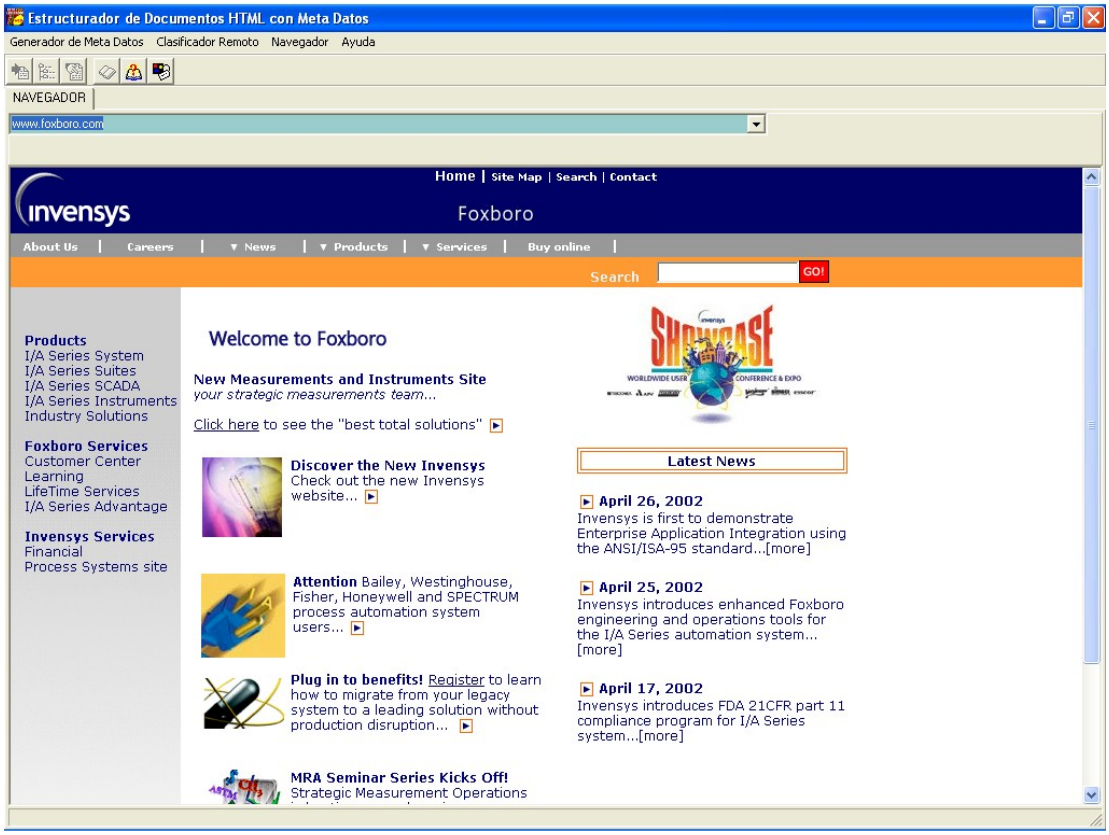


Figura B.20. Ventana del Navegador

GLOSARIO DE TÉRMINOS

Actor.- Alguien o algo, fuera del sistema que interactúa con el sistema.

Administración del conocimiento.- La "administración del conocimiento" se refiere a la habilidad de almacenar, extraer, y entregar el conocimiento tal que puede recuperarse y usarse para tomar decisiones o para apoyar los procesos de negocio.

Arquitectura.- El concepto del nivel más alto de un sistema en su ambiente. La arquitectura de un sistema de software (en un punto dado en el tiempo) es su organización o estructura de los componentes significativos que interactúan a través de interfaces, que son compuestos de componentes sucesivamente más pequeños y sus interfaces.

Caso de Prueba.- Esta formado por un conjunto de datos de entrada de la prueba, condiciones de la ejecución y los resultados previstos desarrollados para un objetivo particular, por ejemplo para provocar una respuesta particular del programa o para verificar funcionalidad con un requisito específico.

Caso de Uso.- Una secuencia de acciones que un sistema realiza y que genera un resultado observable del valor a un actor en particular.

Clase Control.- Una clase usada para modelar el comportamiento específico de uno, o varios casos de uso.

Clase Entidad.- Una clase usada para modelar la información que ha sido almacenada por el sistema, y el comportamiento asociado. Una clase genérica, reutilizada por muchos casos de uso, a menudo con características

persistentes. Una clase de entidad define un conjunto de objetos entidad, que participan en varios casos de uso y sobreviven típicamente en esos casos de uso.

Clase interfaz.- Una clase usada para modelar la comunicación entre el ambientes del sistema y sus funcionamientos interno.

Clase.- Una clase es la descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, cualidades, y semántica.

Descriptor de Recursos.- Consiste en una lista de parejas, atributo-valor (Ej. autor = Darren robusto, título = RDM) que se asocian a un recurso vía una referencia URL. Los robots de búsqueda puede generar los RD's automáticamente (Ej. una robustes de WWW), o puede escribir los RD's manualmente (Ej. un bibliotecario o un autor). Una vez que un descriptor de recurso esté creado, el servidor puede exportarlo vía RDM de manera programática para que los robots de WWW descubran y recuperen los RD's.

Diagrama de Clase.- Un diagrama de la clase muestra una colección de elementos modelo (estáticos) declarativos, tales como clases, paquetes indicando su contenido y sus relaciones.

Diagrama de Colaboración.- Un diagrama de colaboración describe un patrón de la interacción entre objetos; demuestra los objetos que participan en la interacción por sus acoplamientos el uno al otro y los mensajes que se envían.

Diagrama de Secuencias.- Un diagrama que describe un patrón de la interacción entre objetos, dispuesto en orden cronológica; muestra los

objetos que participan en la interacción y los mensajes que envían el uno al otro.

DNS.- (Domain Name System) es un servidor que se utiliza para proveer a las computadoras de los usuarios (clientes) un nombre equivalente a las direcciones IP. El uso de este servidor es transparente para los usuarios cuando este está bien configurado.

Hipertexto.- Sistema que permite realizar enlaces entre zonas de texto dentro de un documento, para facilitar el desplazamiento entre ellas.

Hipervínculo.- Recurso utilizado en multimedia y páginas de la World Wide Web para enviar o enlazar o vincular (saltar) hacia los datos vinculados que se encuentran ubicados en otro lugar. Por ejemplo, sin estar en red se pueden ver estos enlaces en las ayudas de Windows: una lista de enlaces permiten llegar directamente al sector buscado. El término también se utiliza para referirse tanto a las comunicaciones (una línea, canal o circuito sobre el cual se transmiten los datos) como a la administración de datos (puntero incluido en un registro que se refiere a datos o a la posición de los datos en otro registro) En el texto, un hipervinculo por defecto suele estar resaltado por una palabra, frase o imagen del documento, mediante alguna forma de animación o de color diferenciado.

HTML.- (HyperText Markup Language) El lenguaje de código que se emplea para crear documentos de hipertexto para uso en WWW. El código HTML aparece delimitado un bloque de texto que indica como debe aparecer el documento, adicionalmente en HTML se puede especificar que un bloque de texto, o una letra esté unida a otro archivo en Internet. Los archivos HTML son para ser vistos empleando un software Cliente del WWW, como el Internet Explorer de Microsoft, Netscape, Mosaic u otra.

HTTP.- (HyperText Transport Protocol) El protocolo para transferir archivos tipo hipertexto a lo largo de todo Internet. Requiere un programa cliente HTTP en un lado de la conexión y del otro un programa servidor HTTP. Este protocolo es el más importante usado en World Wide Web (WWW).

Inteligencia Tecnológica.- El Termino de Inteligencia Tecnológica (IT) denota una serie de procesos y prácticas que una empresa lleva a cabo sistemáticamente para mantenerse alerta de los cambios tecnológicos y científicos en su medio ambiente. Asimismo, la Inteligencia Tecnológica Competitiva (ITC) no solo abarca el hecho de permanecer al tanto de los cambios y avances científicos y tecnológicos, sino de los competidores y de las oportunidades comerciales, se dice que (ITC) es una herramienta de gestión que permite a los directivos de una institución tener la sensibilidad sobre oportunidades, amenazas y desarrollos científicos y tecnológicos externos que puedan afectar su situación competitiva en función de los recursos con los que cuenta, con el fin de elaborar planes, programas y proyectos relevantes (solleiro y castañón 1998).

Internet .- Red mundial que intercomunica computadoras a través de todo el planeta. Utiliza el protocolo TCP/IP, que controla y lleva a cabo la transferencia de datos. Cada componente de esta red posee una dirección particular, denominada IP Address, (ver DIRECCIÓN IP) que está administrada por un sistema estandarizado (Domain Name System) (ver DNS) que se encarga de asociar un nombre específico (por ejemplo, www.disney-com a una dirección numérica) Internet fue originalmente creada para fines militares y científicos. Su auge se debió al crecimiento del hardware, que permitió dos desarrollos casi simultáneos: por un lado, la aparición de los módems de alta velocidad (a más de 14.400 bps) y el

perfeccionamiento de las interfaces gráficas; por el otro, el crecimiento de uno de sus servicios más populares: la WORLD WIDE WEB.

IP.- Protocolo de Internet que define los códigos numéricos que identifican a cada computador conectado a Internet. Es un número único que consiste en cuatro partes separadas por puntos.

Meta Dato.- Conjunto de elementos propuesto por “Dublin Core” que poseen nombres descriptivos que pretenden transmitir un significado semántico de los mismos, para promover una interoperabilidad global. Una descripción del valor de algunos elementos podrá ser asociada a vocabularios controlados (ver taxonomía). Se asume que otros vocabularios controlados serán desarrollados para asegurar esta interoperabilidad en dominios específicos. Cada elemento es opcional y puede repetirse. Además, los elementos pueden aparecer en cualquier orden. Aunque algunos entornos, como HTML, no diferencian entre mayúsculas y minúsculas, es recomendable escribir correctamente cada meta dato según su definición para evitar conflictos con otros entornos, como XML (Extensible Markup Language). Podemos clasificar estos elementos en tres grupos que indican la clase o el ámbito de la información que se guarda en ellos: Elementos relacionados principalmente con el contenido del recurso, elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual y elementos relacionados principalmente con la instanciación del recurso.

Modelo de Casos de Uso.- Un modelo que refleja lo que se espera deba hacer un sistema y el ambiente alrededor de este.

Modelo de Diseño.- Un modelo de objeto que describe la realización de los casos del uso, que sirve como abstracción del modelo de implementación y su código fuente.

Modelo de Implementación.- Este modelo es una colección de componentes y los subsistemas que los contienen.

Relación de Agregación.- Una forma especial de asociación que modela una relación de todo-parte entre un agregado (el todo) y sus partes.

Relación de Asociación.- Una relación que modela una conexión semántica bidireccional entre casos de uso.

Relación de Generalización.- Una relación taxonómica entre un elemento más general y un elemento más específico. El elemento más específico es consistente con el elemento más general y contiene información adicional. Un instancia del elemento más específico puede ser utilizado donde el elemento más general lo permite.

Requerimiento-. Describe una condición o la capacidad de un sistema, derivado directamente de necesidades del usuario e indicado en el contrato, estándar, especificación, o documento formalmente avalado.

Robot de Búsqueda.- Software que contiene un mecanismo que permite indexar información de acuerdo a un conjunto de Meta Datos o su contenido. La información que indexa un robot se encuentra en documentos accesibles en la red mediante un URL.

Taxonomía.- El estudio de los principios y prácticas de la clasificación, es decir es el proceso del establecimiento y definición de los grupos

sistemáticos o sistemas de clasificación que expresen de la mejor manera posible los diversos grados de similitud entre diferentes temas o tópicos, también se le conoce como vocabulario controlado.

Tipo de Actor.- Conjunto de actores, en el cual cada instancia de actor juega el mismo rol en relación al mismo sistema.

URL.- (Universal Resource Locator) Dirección de un sitio de Internet. La Web utiliza los URL´s para especificar las direcciones de diversos servidores en Internet y los documentos ubicados en cada uno de ellos. Por ejemplo, el URL de IBM es www.ibm.com (WWW significa *World Wide Web* y com significa comercial).

WWW .- *La World Wide Web* (más conocida simplemente como la *Web)* es el segmento de mayor crecimiento en Internet. Fue creada en 1989 por un grupo de científicos del Laboratorio Europeo de Partículas (CERN), quienes deseaban desarrollar un método mejorado para compartir la información de sus investigaciones, ya que los existentes hasta el momento eran obsoletos. En 1992, finalmente, el CERN comenzó a anunciar el proyecto *World Wide Web* (cuya traducción sería Gran Red Mundial), y pronto se instalaron los servidores específicos para publicar la información en la Red. Posteriormente, el increíble avance en el desarrollo de los navegadores o “browser” abrió el acceso a la Red desde una gran variedad de sistemas operativos. Desde aquel origen científico, la Web ha pasado a ser un medio de comunicación masivo, abierto a todas las actividades humanas (educación, comercio, entretenimiento, información, comunicación etc.) La WWW está formada por un conjunto de documentos interconectados, que se encuentran almacenados en computadoras dispersas en todo el mundo. Para acceder a la red es necesario ejecutar un navegador, también conocido como explorador (Internet Explorer y Netscape son los más conocidos). Este

programa sabe cómo interpretar los documentos de la Web y se los presenta al usuario. Al accesar los documentos en hipertexto, es decir un texto con enlaces a otros documentos relacionados. Cuando se recupera un documento, el texto se visualiza gracias a que fue escrito en un lenguaje convencional (ver HTML). Además, para que sea posible acceder a todos los contenidos de la Web, se necesita un protocolo de transferencia de archivos (ver HTTP).