Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Ingeniería Eléctrica

Sección de Computación

Implementación de una arquitectura blackboard para sistemas de hipermedia

Tesis que presenta el

Lic . Ricardo Romero Quiroz

Para obtener el grado de

Maestro en Ciencias

En la especialidad de

Ingeniería Eléctrica

Opción

Computación

Director de tesis:

Dr. José Oscar Olmedo Aguirre

Julio de 2003

Ciudad de México, México.

Resumen

Los sistemas de información distribuida, como la World Wide Web, han demostrado la simplicidad y versatilidad con la que los *sistemas de hipermedia* permiten recuperar eficazmente grandes volúmenes de información. En relación a la integración de documentos cuyo formato no es HTML o cuyo contenido no debe alterarse, se han propuesto los *sistemas abiertos de hipermedia* para dar solución a tales problemas de interoperabilidad.

El propósito de éste trabajo, es describir la construcción de un sistema abierto de hipermedia con fácil integración a la Web. El sistema consiste de un número reducido de extensiones a un navegador y de un servidor de enlaces con arquitectura reconfigurable.

Entre las contribuciones de éste trabajo podemos resaltar que el sistema permite:

- Extender el modelo simple de enlace de la Web con otros modelos de enlaces como el específico, genérico, de avance u otros definidos por el usuario.
- Extender el modelo de proceso de la Web con una arquitectura reconfigurable que permite no solo introducir nuevos servidores de enlace sino también organizarlos con diferentes estilos como el conducto-filtro o el de pizarra.
- Hacer el seguimiento de documentos móviles, como aquellos involucrados en aplicaciones *workflows*, mediante el modelo de enlace de avance.
- Demostrar la superioridad expresiva de la arquitectura de pizarra sobre la conducto-filtro.

El sistema se ha usado exitosamente para recuperar información académica de la Sección de Computación del Cinvestav al introducir nuevos enlaces sin alterar el contenido de los documentos Web, un problema bastante común pero en general difícil de resolver cuando no se cuenta con permisos de acceso para modificar los documentos.

Reconocimientos

Quiero dar gracias en primer lugar a mi Dios por el regalo de la vida, ya que por Él las cosas se mueven y son posibles. Gracias al Conacyt por su valioso apoyo. al Cinvestav en general por la oportunidad que me dio. Gracias Dr. Olmedo por su comprensión y paciencia, por el valioso tiempo que dedicó a la elaboración de este trabajo. A los doctores Gerardo de la Fraga y Sergio Chapa por aceptar ser mis sinodales y por sus atinados comentarios. Sofi también para ti muchas gracias.

Mamá gracias por ser mi madre y apoyarme siempre en las adversidades de la vida (sabes a lo que me refiero). Marce te deseo lo mejor del mundo. Héctor con todo y tu carácter duro gracias también a ti.

En forma especial quiero agradecer al enano por haber aceptado la aventura, por comprenderme y por motivarme (El Señor nunca te abandonará).

Por último gracias a todos mis compañeros de generación: Landa, Eloy, Lorena, Joselito, Julio, Juan Manuel, Adriana. A todos mis profesores muchas gracias.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Sistemas Hipermedia

En la actualidad un gran número de sistemas organizan la información textual usando estructuras de directorio. Esta organización permite nombrar y agrupar lógicamente colecciones enormes de textos y de otro tipo de documentos, facilitando considerablemente la búsqueda y localización de la información. Sin embargo, para las aplicaciones que requieren de métodos de recuperación más eficientes, la estructura lineal de un texto ya no es suficiente. Se han inventado numerosos mecanismos que dan soporte a referencias directas de un fragmento de texto a otro. Por otra parte, algunas interfaces de usuario permiten interactuar directamente con estos fragmentos y establecer nuevas relaciones entre ellos. A éstas extensiones de texto es a lo que usualmente llamamos "hipertexto" (también conocido como texto no lineal). *Hipertexto* es una expresión acuñada por Theodor H. Nelson[1] en los años sesenta. Se refiere a un tipo de texto electrónico, una tecnología informática radicalmente nueva y, al mismo tiempo, un modo de edición. Como él mismo lo explica:

"Por hipertexto, me refiero a una escritura no secuencial de un texto que permite al lector elegir dónde continuar su lectura usando una pantalla interactiva. De acuerdo con la noción popular, se trata de una serie de bloques de texto conectados entre sí por enlaces, que forman diferentes itinerarios para el usuario".

Dicho con otras palabras, el hipertexto es una forma de enlazar textos, párrafos o líneas situadas en diferentes documentos para facilitar su lectura; como si estuvieran situadas en el mismo texto. Algo de esto hacíamos desde hace mucho tiempo, cuando a un libro se lo acompañaba con un aparato de notas y citas. Solo que la tecnología del papel hace el proceso confuso y el lector tiene

que elegir entre interrumpir el hilo o esperar a buscar más tarde. Parece mentira como una simple facilidad técnica puede modificar radicalmente la lectura. Como dice el mismo Theodor H. Nelson[1]:

"El hipertexto, que se ha definido como texto diseñado para ser leído de forma no lineal o no secuencial, se presta particularmente al tipo de texto característico de los escritos científicos o humanísticos. Este género de escritos requiere que el lector abandone el texto principal y se aventure a considerar las notas a pie de página, las pruebas estadísticas o de otros autores. Nuestra experiencia en la Universidad Brown indica que el uso del hipertexto enseña a los estudiantes a leer de esta manera avanzada."

Como lector de hipertextos, uno puede escoger entre volver a la exposición del autor o seguir alguna de las conexiones sugeridas por los enlaces, utilizar otras funciones del sistema o buscar nuevas conexiones. La versatilidad del hipertexto, que se manifiesta en múltiples conexiones entre bloques individuales de texto, requiere de un lector activo.

Derrida [2] concuerda con el diseño de los actuales sistemas de hipertexto en los que el lector, se encuentra activamente ocupado en el descubrimiento y exploración del texto, pudiendo hacer intervenir diccionarios con análisis morfológicos que conectan las palabras aisladas con sinónimos, antónimos y derivados. Derrida reconoce acertadamente que una nueva forma de texto más rica, más libre, más fiel a nuestra experiencia y, tal vez a una experiencia más real aún desconocida, depende en última instancia de los fragmentos de texto relacionados.

La noción de hipertexto permite al autor hacer flexible el orden de la lectura. A medida que el lector se mueve por una red de textos, desplaza constantemente su centro de atención y, por lo tanto, cualquier usuario de hipertexto hace de sus intereses propios el eje organizador de su investigación del momento. Hipermedia extiende al hipertexto al introducir diferentes tipos de medios en los documentos como audio, gráficos y video, en donde información textual o no textual puede relacionarse mediante enlaces.

Diferentes autores han elogiado ésta técnica poniendo de manifiesto en cada ocasión diferentes razones:

- **Facilidad de uso de enlaces.** En un sistema que soporta enlaces, todos los enlaces son igualmente fáciles de seguir sin importar en que parte del documento se localicen.
- **Facilidad para crear nuevos enlaces.** Los usuarios pueden crear y mantener sus propias

redes de enlaces o simplemente anotar algún comentario en el documento sin alterar su contenido.

- **Perspectiva global del documento.** Los visualizadores (*browsers*) proveen una variedad de vistas del tipo tabla de contenido, ofreciendo así una forma sencilla de reestructurar documentos grandes y combinando, al mismo tiempo, las ventajas de las vistas globales y parciales.

- **Modularidad de la información.** El mismo fragmento de texto puede referenciarse desde varias partes del documento lo que ayuda a reducir la redundancia.

- **Consistencia de la información.** Cuando un fragmento de texto que incluye enlaces se inserta en otro documento, dependiendo de su naturaleza, los enlaces todavía pueden proveer acceso directo a los documentos referenciados.

- **Colaboración.** Varios autores pueden colaborar usando un documento como medio de comunicación, anotando comentarios e introduciendo referencias, obteniendo de esta forma un entretejido de documentos.

Ahora mencionamos algunos de los problemas asociados al uso de ésta tecnología:

- **Desorientación.** Se manifiesta por la incapacidad del usuario para controlar la información en un vasto espacio de documentos interconectados. Éste problema está íntimamente relacionado al diseño del hiperdocumento.

- **Sobrecarga.** Se ha comprobado que estructurar la información de un hiperdocumento puede suponer un gran esfuerzo para el usuario.

- **Falta de interoperabilidad.** Sistemas de hipermedia como la World Wide Web no se consideran como sistemas abiertos porque la integración de un documento en el sistema requiere su modificación para incorporar el lenguaje de marcado HTML que utiliza la Web.

Estos problemas se ven acrecentados por el hecho de que, al no existir normas o metodologías de diseño de hiperdocumentos, cada autor es libre de escribir hiperdocumentos como crea más conveniente.

1.1.1. Sistemas Abiertos de Hipermedia

Los primeros sistemas de hipermedia eran sistemas monolíticos que usaban formatos propietarios de documentos, lo que impedía su interoperabilidad con otros sistemas. Los sistemas abiertos de hipermedia se propusieron entonces para resolver la naturaleza aislada de los primeros sistemas.

Los sistemas abiertos de hipermedia deben, en principio, ser capaces de interoperar con sistemas externos para intercambiar datos con ellos. Los datos, por lo tanto, no deben modificarse por las aplicaciones que los usen. Para integrar la información de enlaces sin modificar los documentos, los sistemas abiertos usan bases de datos externas, también conocidas como *bases de enlaces* (*linkbases*). Las bases de enlaces indican la localización y la extensión de los fragmentos de documentos que relacionan los enlaces. Así, el contenido del documento se mantiene inalterado siempre que las aplicaciones hipermedia ofrezcan mecanismos suficientes para navegar entre documentos. El mecanismo básico de navegación corresponde al patrón de interacción Modelo-Visualizador-Controlador (*Model-Viewer-Controller, MVC*) introducido en el entorno de programación del lenguaje Small-Talk. La navegación consiste de tres fases:

1. **Selección.** El usuario selecciona un fragmento de texto con el propósito de obtener información adicional. El fragmento junto con información de contexto es enviado al controlador para determinar si posee un enlace asociado.

2. **Procesamiento.** El controlador consulta la base de enlaces para determinar el documento relacionado de acuerdo al modelo de hipermedia.

3. **Recuperación.** El visualizador recupera una copia del documento y despliega su contenido.

En hipermedia, al controlador se le conoce como *procesador de estructura*. El procesador de estructura consiste de un arreglo de componentes conocidos como *servidores de enlaces*, organizados de acuerdo a alguna arquitectura de software específica, como por ejemplo, pizarra, conducto-filtro o cliente-servidor. Cada servidor de enlace implementa un mecanismo de recuperación de documentos de acuerdo a su tipo. Por ejemplo, el enlace especifico, como se habrá de explicar más adelante, funciona como un apuntador fijo a un documento, mientras que el enlace genérico funciona en forma similar a un diccionario. De esta forma, el procesador de estructura resuelve el enlace como producto de la operación coordinada de los servidores de enlace. Como resultado de la descomposición altamente modular del procesador de estructura, ésta organización permite incorporar diferentes tipos de enlaces diseñados y construidos con relativa independencia por los diseñadores de sistemas hipermedia.

A pesar de las ventajas que los sistemas abiertos de hipermedia ofrecen al permitir que cualquier tipo de documento pueda integrarse al sistema, el enfoque posee dificultades que han limitado su amplia aceptación. Entre los problemas más importantes relacionados con el modelo de datos podemos señalar:

- **Edición de hiperdocumentos.** Este problema se observa al modificar el contenido de un documento que contiene fragmentos de texto referenciados por enlaces. Debido a que los documentos y los enlaces se almacenan por separado, la localización de los fragmentos pueden ya no corresponder con las referencias en los enlaces.
- **Mantenimiento de versiones.** El problema surge de la dificultad de elegir la versión apropiada del documento referenciado por un enlace. Debido a que el enlace no contiene información sobre la historia de las versiones, el sistema no puede resolver el conflicto sobre cuál versión del documento recuperar.
- **Pérdida de integridad.** Este problema resulta cuando un enlace no puede resolver un documento válido. Puesto que la localización de un documento puede cambiar después de que el documento ha sido renombrado, movido o borrado, el sistema podría no determinar correctamente su nueva designación, localización o posible destrucción.

En éste trabajo, los problemas relacionados con el modelo de datos no se presentan ya que se asume que los documentos se mantienen relativamente estables por períodos largos. En éstas circunstancias, los documentos se caracterizan porque no son editados, no se mantienen versiones de ellos, ni tampoco cambian de localidad. En sistemas de información como aquellos que describen la información de una institución, por ejemplo, la información académica de una universidad, ésta suposición es razonablemente válida ya que no se realizan modificaciones significativas a los documentos o a sus enlaces por meses.

De mayor importancia para éste trabajo son los problemas relacionados con el comportamiento asociado a los diferentes tipos de enlaces. La selección y construcción de una arquitectura para el procesador de estructura que garantice la correcta interoperación de los servidores de enlaces es el problema central de este trabajo.

1.2. Planteamiento del Problema

En [15] se propuso una arquitectura de coordinación para el procesador de enlaces, con el propósito de analizar la facilidad con la que un diseñador de sistemas hipermedia podía definir nuevos tipos

de enlaces e integrarlos en sistemas abiertos como Microcosm [12].

Al introducir en el procesador de estructura el servidor de un nuevo tipo de enlace, se descubrió una anomalía en la resolución de enlaces. La anomalía surge cuando a pesar de que el servidor de enlace está presente, el procesador de estructura es incapaz de resolver correctamente el enlace. Una descripción más detallada de éste problema se dará en el capítulo 3.

La causa del problema fue identificada en la incapacidad de la arquitectura conducto-filtro para procesar cierto tipo de enlaces. Para resolver este problema se sugirió que este problema no se presentaría en un procesador de estructura basado en una arquitectura de tipo pizarra. Aunque se argumentó que la arquitectura de pizarra posee un poder expresivo superior al de la arquitectura conducto-filtro, no se ha construido un sistema abierto de hipermedia que lo demuestre.

1.3. Objetivo General del Proyecto

El propósito de este trabajo es el de implementar la arquitectura de pizarra en un sistema abierto de hipermedia con fácil integración a la Web.

1.3.1. Objetivos Específicos del Proyecto

El planteamiento de éste objetivo conduce a considerar los siguientes objetivos específicos:

1. Describir el modelo de datos de cada tipo de enlace considerado: específico, genérico y de avance.

2. Describir el modelo de procesos de los servidores para cada tipo de enlace. Definir las reglas de comportamiento de los servidores de acuerdo al modelo de datos.

3. Describir y comparar las arquitecturas del procesador de estructura siguiendo patrones arquitectónicos predefinidos tales como conducto-filtro y pizarra. Integrar los servidores de enlaces especializados en la arquitectura.

4. Demostrar experimentalmente la superioridad expresiva de la arquitectura de pizarra sobre la de conducto-filtro. Usando enlaces de avance en bases de enlaces con integridad débil, demostrar que la arquitectura de pizarra permite recuperar documentos que no es posible recuperar con la arquitectura conducto-filtro.

1.4. Metodología

Para lograr estos objetivos se propone seguir la siguiente metodología:

1. En el modelo de datos de hipermedia, identificar las piezas de información hipermedia (fragmentos de texto, campos, enlaces, etc.) como elementos HTML.

2. En el modelo de procesos de hipermedia, identificar a los servidores de enlaces y a otros componentes de software.

3. Formular las reglas de comportamiento de los componentes de la arquitectura e instrumentarlos como métodos o como procesos ligeros (*Threads*).

4. Diseñar la arquitectura del procesador de enlaces como un Proxy que se sitúa entre el visualizador (*browser*) y el servidor Apache Jakarta-Tomcat para intercambiar mensajes. El Proxy debe revisar todas las peticiones HTTP producidas en la navegación para asociarlas con las selecciones del documento correspondientes, produciendo así los enlaces que serán consultados en las bases de enlaces. Los enlaces válidos se dirigirán al servidor; los enlaces inválidos serán procesados por el servidor de enlaces.

5. Demostrar la implementación de la arquitectura definiendo un escenario de prueba en donde las localidades que visitan los documentos móviles se registran en la base de enlace. La validación incluye ambas arquitecturas. La validación incluye ambas arquitecturas, tanto la conducto-filtro como la de pizarra.

6. Mostrar las situaciones en las cuales la arquitectura de conducto-filtro no puede resolver las referencias a documentos móviles. Mostrar que bajo las mismas circunstancias, la arquitectura de pizarra resuelve correctamente las referencias.

1.5. Organización de la tesis

Esta tesis está organizada de la siguiente manera. En el capítulo 2 hacemos un recorrido sobre diferentes puntos de vista acerca del hipertexto, así como sus características básicas, los elementos del hipertexto como son el enlace y sus propiedades y el nodo; cerramos el capítulo hablando sobre algunos sistemas de hipertexto desarrollados en el siglo pasado. En el capítulo 3 hablamos sobre las bases utilizadas para la construcción del prototipo creado en este trabajo, y en primer término describir los diferentes modelos de enlaces implementados , así como también los dos modelos de procesos implementados. En el capítulo 4 tratamos el caso de estudio, un sistema para

la recuperación de la información académica de la Sección de Computación del Cinvestav, que se desarrollo para probar el sistema. El capítulo 5 es una descripción detallada del diseño del sistema que está distribuido en tres capas que son: la capa de aplicación, la capa de servicio de enlaces y la capa de almacenamiento. Sobre la programación de las arquitecturas implementadas en el prototipo, el capítulo 6 está dedicado a este tema. El manual de usuario que se describe en el capítulo 7, para la correcta instalación y configuración del sistema en cualquier equipo de cómputo que cuente con el sistema operativo Windows de Microsoft, así como también una ayuda de cómo trabajar con el prototipo. En el capítulo 8 finalizamos haciendo un repaso sobre las conclusiones obtenidas de la investigación y la implementación del trabajo desarrollado, así como también el trabajo futuro.

Capítulo 2

Antecedentes

En este capítulo se dan las definiciones básicas de hipertexto así como sus características y sus propiedades. Después, se describen algunos de los sistemas más importantes en la historia de hipertexto que por los conceptos y características novedosas que introdujeron, dejaron un legado de técnicas a los sistemas actuales.

El hipertexto es una tecnología que organiza una base de información en bloques distintos de contenido textual, conectados a través de una serie de enlaces cuya activación o selección provoca la recuperación de información (Díaz [3]). El hipertexto ha sido definido como un enfoque para manejar y organizar información, en el cual los datos se almacenan en una red de nodos conectados por enlaces. Los nodos contienen texto y además gráficos, imágenes, audio, animaciones y video, así como código ejecutable u otra forma de datos se les da el nombre de hipermedio, es decir, una generalización de hipertexto.

Considerando cómo se representa el conocimiento humano, el hombre opera por asociación, saltando de un concepto a otro, en forma casi instantánea. El paradigma hipermedia intenta modelar este proceso con enlaces entre pedazos de información contenidos en nodos.

A diferencia de los libros impresos, en los cuales la lectura se realiza en forma secuencial desde el principio hasta el final, en un ambiente hipermedia la "lectura"puede realizarse en forma no lineal, y los usuarios no están obligados a seguir una secuencia establecida, sino que pueden moverse a través de la información y hojear intuitivamente los contenidos por asociación, siguiendo sus intereses en búsqueda de un término o concepto. En las figuras 2.1, 2.2, 2.3, 2.4 , se representan los estilos secuencial, jerárquico, reticulado y el de hipermedio.

Figura 2.1: *Estilo secuencial*

Figura 2.2: *Estilo jerárquico*

Figura 2.3: *Estilo reticulado*

En términos más sencillos, y a la vez más amplios, un hipermedio es un sistema de bases de datos que provee al usuario una forma libre y única de acceder y explorar la información realizando saltos entre un documento y otro.

En la literatura se utiliza, a veces, en forma indiscriminada los términos hipertexto, sistemas hipertexto e hiperdocumento; sin embargo, es necesario aclarar que se hace referencia a objetos distintos, y que en lo sucesivo se utilizará las siguientes definiciones para evitar confusiones:

Hiperdocumento: es el contenido de información, incluyendo los fragmentos de información y las conexiones entre esos fragmentos, el sistema utilizado para leer o escribir tal documento es independiente del propio documento.

Figura 2.4: *Estilo hipertexto*

Sistema hipertexto: es una herramienta de software que permite leer y escribir hiperdocumentos. Este sistema no contiene hiperdocumentos.

Hipertexto: es un sistema que contiene hiperdocumentos.

Según Landow [4], el hipertexto es una forma distinta de literatura; definen hipertexto *"como el uso del computador que trasciende la linealidad, límites y calidad fija de la tradicional forma de escritura de texto"*.

Otro autor en el campo de la literatura, Bolter [5] compara al hipertexto con formas tradicionales de presentar información textual: *"hipertexto consiste de tópicos y sus conexiones; los tópicos pueden ser párrafos, oraciones o palabras simples. Un hipertexto es como un libro impreso en el cual el autor tiene disponible un par de tijeras para cortar y pegar pedazos de redacción de tamaño conveniente. La diferencia es que el hipertexto electrónico no se disuelve en una desordenada carpeta de anotaciones: el autor define su estructura definiendo conexiones entre esas anotaciones"*.

Conklin [6] da una definición más técnica, centrada en el uso de las computadoras: *"son ventanas, en una pantalla, las cuales están asociadas a objetos en una base de datos, y enlaces provistos entre estos objetos, tanto gráficamente (iconos etiquetados) como en la base de datos (apuntadores)"*.

En publicaciones menos formales como Byte, Fiderio [7] da otra definición igualmente técnica: *"hipertexto, en el nivel más básico, es un manejador de base de datos que permite conectar pantallas de información usando enlaces asociativos. En un nivel mayor, hipertexto es un ambiente de software para realizar trabajo colaborativo, comunicación y adquisición de conocimiento. Los productos de este software emulan la habilidad del cerebro para almacenar y recuperar información haciendo uso de enlaces para un acceso rápido e intuitivo"*.

Para Balzer [8] hipertexto es: *"una base de datos que tiene referencias cruzadas y permite al*

usuario (lector) saltar hacia otra parte de la base de datos, si éste lo desea".
Esta definición clarifica algunos puntos de interés sobre hipertexto: Hipertexto es una base de datos. La información no consta de grupos de bytes, sino que es estructurada y de tamaño considerable, características similares a muchas bases de datos. A pesar de que la estructura de información tiene una forma distinta a las estructuras de bases de datos tradicionales, muchos sistemas de bases de datos son capaces de almacenar información utilizada en los hipertextos. Además la acción típica permitida al usuario es la de saltar entre las partes de la base de datos. Esto es diferente a la típica utilización de bases de datos, en las cuales la obtención de información se realiza a través de consultas.

Finalmente, haciendo una analogía con espacios multidimensionales según Rada [9], el término hipertexto: *"se relaciona con el término 'espacio hiperbólico', debido al matemático Klein, en el siglo XIX. Klein utilizó el término hiperespacio para describir una geometría de muchas dimensiones; por lo anterior, se puede deducir que hipertexto es texto multidimensional, considerándose el texto como una estructura unidimensional".*

2.1. Características básicas del hipertexto

Esta tecnología de información ha sido defendida y elogiada debido a las grandes ventajas que proporciona. Sin embargo, no todos los sistemas de hipertexto que se han implementado y están disponibles en distintas plataformas e instalaciones cumplen cabalmente con las expectativas de los usuarios. Para Lee-Berners [10], un sistema de hipertexto, en términos ideales, debe cumplir con las siguientes características:

- Esta tecnología debe proveer un medio adecuado para organizar y presentar información poco o nada estructurada, no ajustada a esquemas tradicionales y rígidos como es el caso de las bases de datos. Pueden utilizarse esquemas jerárquicos para la utilización de sistemas de documentación de texto tradicionales, muy organizados o simplemente creando estructuras de redes con poco o ningún atributo de precedencia.

- Tener asociada una interfaz de usuario muy intuitiva, pues se pretende imitar el funcionamiento de la mente humana, haciendo uso de modelos cognitivos, por lo que el usuario no debería realizar grandes esfuerzos para obtener la información requerida.

- La información se encuentra distribuida y puede ser accesada en forma concurrente por varios usuarios, por lo tanto es un ambiente de colaboración compartido.

- Es un ambiente colaborativo: un usuario puede crear nuevas referencias entre dos documentos cualesquiera en forma inmediata e independiente de los tipos de contenido, haciendo crecer su hiperdocumento, sin generar cambios en el hiperdocumento referenciado. Estas referencias pueden estar embebidas en el documento, de modo que aunque éste se cambiara de instalación, el enlace seguiría proporcionando acceso a la información referenciada.
- Tener asociados varios mecanismos de recuperación y búsqueda de información a través de las navegaciones, ya sean dirigidas o no dirigidas.

Estas características hacen que este paradigma sea utilizado en una variedad muy amplia de aplicaciones, en las cuales se tienen al menos los siguientes requerimientos: gran cantidad de información organizada en distintos fragmentos y contextos, los cuales pueden estar relacionados entre sí, que el usuario necesita en forma discreta, y que pueda estar implantado en un ambiente electrónico o computacional. Dados estos requerimientos, el dominio de las aplicaciones de hipermedia incluye: ayudas y documentación, diccionarios y enciclopedias electrónicas, herramientas CASE para desarrollo de software, organizadores de ideas, sistemas de información turísticos y geográficos, venta electrónica, soporte para enseñanza y aprendizaje, trabajo colaborativo y comunicaciones. Estas aplicaciones pueden ser implementadas tanto en ambientes cerrados o en ambientes abiertos.

Un ambiente hipermedia cerrado es aquel donde todo el repositorio de información se encuentra concentrado en una única unidad de almacenamiento o servidor, y los enlaces entre hiperdocumentos sólo puede realizarse entre fragmentos de información que residen en el mismo servidor. En cambio, cuando el ambiente es abierto, los contenidos y fragmentos de información pueden encontrarse distribuidos en diversos repositorios de almacenamiento o varios servidores, es decir la información se encuentra físicamente distribuida en servidores distintos y se permiten hacer referencias entre hiperdocumento que residen en servidores distintos: este es el caso del World Wide Web [10].

El hipertexto es una tecnología que organiza una base de información en bloques discretos de contenido llamados nodos, conectados a través de una serie de enlaces cuya selección provoca la inmediata recuperación de la información destino. De otra manera, la organización hipertextual permite enlazar información que esté relacionada, por lo que se puede navegar a través de un entramado de nodos, de acuerdo con las preferencias o las necesidades de adquisición de conocimiento que se tenga en cada momento.

La característica más distinguida del hipertexto es el soporte para las referencias cruzadas. Pero ¿que califica a una referencia cruzada como un enlace? ¿Cuánto esfuerzo es permisible por parte del

usuario quien intenta trazar una referencia? El límite más bajo aceptado para el soporte de referencias puede ser especificado de la siguiente manera: para calificar como hipertexto, un sistema debe requerir no más que un par de teclas (o movimientos de ratón) del usuario para seguir un enlace sencillo. En otras palabras la interfase debe proveer enlaces que actúan como "botones mágicos" para transportar al usuario rápida y fácilmente a un nuevo lugar en el hiperdocumento.

Otra característica esencial es la velocidad con la cual el sistema responde a las referencias solicitadas. Sólo el retraso más breve deben ocurrir (uno o dos segundos a lo mucho). Mucho trabajo diseñado tiene esta característica. Una razón por esta preocupación es que el lector no sabe si seguir un enlace de referencia o solo tener una mirada superficial del nodo referenciado. Si hacer este juicio toma mucho tiempo, el usuario puede llegar a frustrarse y fastidiarse con los enlaces de hipertexto.

Sin embargo no todos los enlaces transversales pueden ser instantáneos. Quizás tan importante es una respuesta rápida como proporcionarle las señales al usuario sobre el posible retraso que toma la consulta. Por ejemplo, alguna característica visual del ícono de enlace podría indicar si el nodo destino está en memoria, en disco, sobre la red, o archivado fuera de línea.

2.2. Propiedades de los enlaces

Para Conklin [6], los enlaces pueden ser utilizados para varias funciones. Entre ellas se tienen:

- Pueden conectar una referencia al mismo documento.
- Pueden conectar un comentario o una anotación al texto del cual esta escrito.
- Pueden proveer información organizacional (Establecer la relación entre dos piezas de texto o entre una tabla de contenidos y su sección).
- Pueden conectar dos piezas sucesivas de texto, o una pieza de texto y todos sus sucesores inmediatos.
- Pueden conectar entradas en una tabla o figura a descripciones largas, o a otra tabla o figura.

Los enlaces pueden tener nombres y tipos. Pueden tener un rico conjunto de propiedades. Algunos sistemas permiten el despliegue de enlaces para ser encendidos o apagados (esto es, removidos del display para que el documento aparezca como texto ordinario).

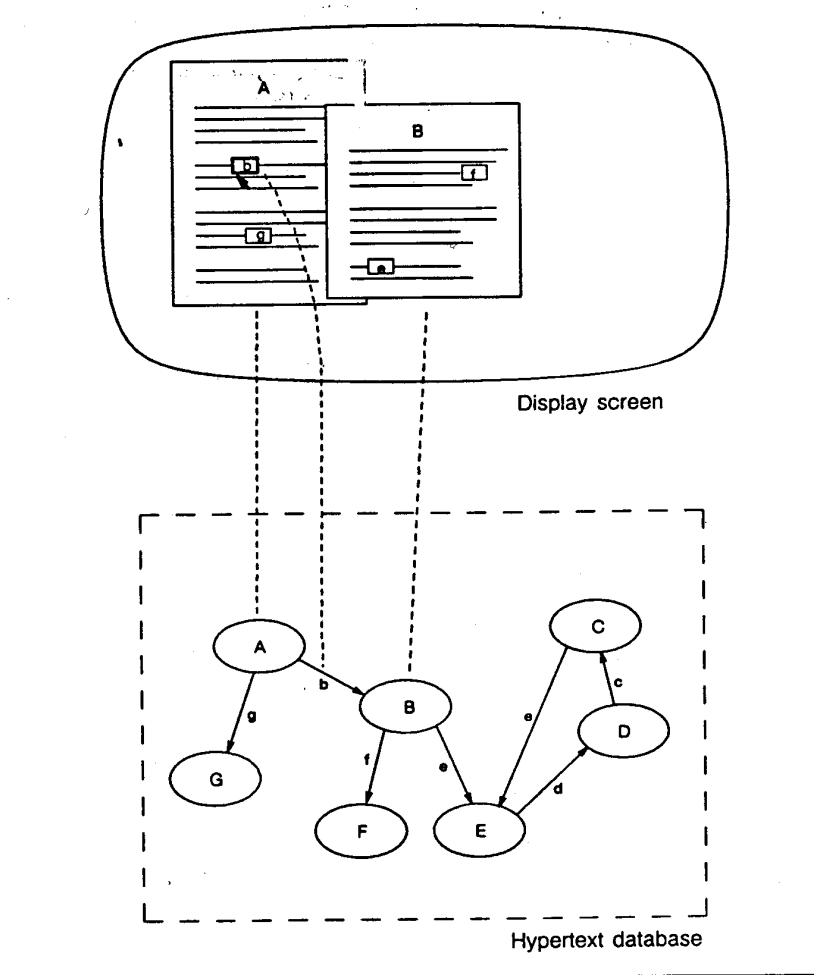

Figura 2.5: *La correspondencia entre ventanas y enlaces en la pantalla, y nodos y enlaces en la base de datos. En este ejemplo cada nodo en la base de datos de enlaces se despliega en una ventana por separado. El enlace "b" en la ventana "A" ha sido activado, abriendo una nueva ventana llamada "B" con el contenido del nodo "B" en la base de datos*

2.3. El nodo

El nodo es un elemento constitutivo del hipertexto que contiene una cantidad discreta de información, pueden contener texto, gráficas, audio, video, así como también el código fuente u otras formas de datos. Los nodos y algunos sistemas de la propia red, son vistos a través de un visualizador interactivo y manipulados a través de un editor.

2.4. La base de enlaces

Los enlaces y otra información de hipermedia son mantenidos en una base de datos por separado llamado *bases de datos de enlaces* (*linkbases*). Una base de datos de enlaces tiene *la propiedad de integridad fuerte* cuando esta no contiene alguna consistencia. Se espera que todas las bases de datos de enlaces mantengan la propiedad de integridad fuerte antes que las aplicaciones puedan modificar su ambiente. Como las aplicaciones pueden eventualmente cambiar el ambiente de las base de datos de enlaces, una base de datos de enlaces puede perder la propiedad de integridad. No obstante, la base de datos de enlaces podría después recobrarla de nuevo manteniendo por lo menos una forma débil de integridad caracterizada por el almacenamiento de información sobre los enlaces rotos y algunas acciones correctivas.

Hasta aquí hemos explicado los diferentes conceptos y definiciones sobre el hipertexto que se escribieron a lo largo de estos años, así como también se expusieron sus características elementales, sus propiedades y sus componentes.

2.5. Sistemas de hipermedia en la historia

La historia del hipertexto es rica y variada porque el hipertexto no es una idea nueva. Mucha gente ha contribuido a esta idea, y cada persona ha tenido siempre en mente algo diferente. A continuación se mencionan algunos de los sistemas de hipertexto más famosos o mayormente aceptados por la comunidad.

Un tipo de hipertexto manual es la tarjeta tradicional de índices de 3 X 5 para tomar notas. Su ventaja es que en un pequeño espacio colocamos trozos de información. El usuario puede fácilmente reorganizar un conjunto de tarjetas cuando se añade nueva información. Obviamente hay un problema para encontrar una tarjeta específica cuando el número de ellas es muy grande.

2.5.1. Memex

A Vannevar Bush, asesor científico del presidente Roosevelt, se le acredita como el primero en describir el hipertexto en su artículo de 1945 “As We May Think” en el cual hace un llamado a hacer un mejor esfuerzo para mecanizar el sistema de literatura científica. En el artículo, introduce una máquina para el trabajo de visualización y crea sus notas en un extensivo sistema de gráficos y texto en línea. Este sistema llamado Memex contenía una librería muy grande como sus notas

personales, fotografías y bosquejos.

Bush describe la característica esencial del Memex como la habilidad para atar dos cosas y motivado por la necesidad de ofrecer formas más naturales de indexamiento y recuperación:

"La mente humana... opera por asociación. El hombre puede esperar para duplicar su proceso mental artificialmente, pero tiene que estar dispuesto para aprender de ello. Uno no puede esperar igualar la velocidad con la cual la mente sigue una huella asociativa, por esto debemos tomar el poder de la mente para considerar la permanencia y claridad de las cosas recuperadas del almacenamiento."

2.5.2. Augment

Justo dos décadas después Douglas Engelbert, en el Stanford Research Institute, fue influenciado por las ideas de Bush. En 1963, escribió *"A Conceptual Framework for the Augmentation of Man's Intellect"* (Un cuadro conceptual para el aumento del intelecto humano). Previó que las computadoras podrían acomodarse en un nuevo estado de la evolución humana, caracterizada por *"la manipulación de símbolos externos automatizados"*:

"En este estado, los símbolos con los cuales el humano representa los conceptos que él está manipulando, pueden ser colocados ante sus ojos, movidos, almacenados, renombrados, operados de acuerdo a reglas extremadamente complejas -todas en una respuesta muy rápida con una cantidad mínima de información proporcionada por el humano, por medio de dispositivos especiales de cooperación tecnológica. En el límite de lo que podemos ahora imaginar, esta podría ser una computadora, con la cual los individuos podrán comunicarse rápidamente y de manera fácil, acoplado con un display a color tridimensional en el cual imágenes extremadamente sofisticadas pueden construirse ...

Su sistema propuesto, H-LAM/T (Humano usando Lenguaje, Artefactos, y Metodología, con la cual se Entrena), incluyó al usuario humano como un elemento esencial: el usuario y la computadora fueron cambiando dinámicamente componentes en una simbiosis la cual ha tenido un efecto "amplificante" en la inteligencia nativa del usuario. Esta es una visión todavía muy común entre desarrolladores de sistemas de hipertexto.

2.5.3. Xanadu

Durante el desarrollo de Augment de Engelbart, otro visionario del hipertexto, Ted Nelson, desarrolló sus propias ideas, pero con énfasis sobre la creación de un ambiente literario unificado

en una escala global. Nelson acuñó el término "hipertexto." Sus ideas y sus escritos son los más extravagantes realizados por cualquier trabajo hecho hasta esa fecha. El nombró a su sistema de hipertexto Xanadu, "el lugar mágico de la memoria literaria" del poema "Kubla Khan" de Samuel Taylor. En Xanadu, el espacio de almacenamiento se guarda con el uso pesado de enlaces. Únicamente el documento original y los cambios hechos a este se guardan. El sistema reconstruía fácilmente versiones previas de documentos. Nelson describe sus objetivos de la manera siguiente:

"Bajo un seguimiento de ideas las cuales no son técnicas pero si literarias, implementamos un sistema para almacenamiento y recuperación de texto ligado. El documento, nuestra unidad fundamental, puede contener ventanas para cualquier otro documento. El cuerpo evoluciona de manera continua sin cambio fundamental. Nuevas ventanas y enlaces pueden adherirse de manera continua, nuevos accesos a material viejo. Algoritmos veloces dan como resultado una fragmentación de datos tolerable para la facilidad de servicio."

El objetivo principal del proyecto Xanadu, ha facilitado el revolucionario proceso de colocar todo un cuerpo literario en línea. En efecto, el diseño de Xanadu hizo una fuerte separación entre la interfase de usuario y, el servidor de base de datos, con mayor énfasis en éste último. Nelson predijo que el advenimiento de bibliotecas en línea crearía un nuevo mercado para la organización e indexación de este inmenso almacenamiento de información.

2.5.4. Intermedia

Uno de los grupos de investigación sobre hipertexto mas viejos y con más tradición que existe está en la Universidad de Brown, en el Institute for Research in Information and Scholarship (IRIS). El proyecto Intermedia fue construido en dos décadas de trabajo y tres generaciones de sistemas de hipertexto.

La primera generación fue el Hypertext Edition System diseñado por Ted Nelson, Andy Van Dam, y varios estudiantes en una IBM 2250 en 1968. Este sistema fue usado por el Centro Espacial de Houston para producir la documentación del Apollo.

La segunda generación fue el File Retrieval and Editing System (FRESS). FRESS fue reforzado con una multiterminal de tiempo compartido diseñado por Van Dam y sus estudiantes. Estuvo disponible en 1969 y fue comercialmente implantado en los 70's. FRESS se usó por cientos en la facultad y estudiantes por más de una década. La tercer generación, el proyecto denominado Elec-

tronic Document System, fue un sistema de hipermedia que enfatizaba los gráficos a color y ayudas en la navegación.

Finalmente, el sistema Intermedia se desarrolló como una colección de herramientas (figura 2.6) que permiten a los autores crear enlaces a documentos de varios medios tales como texto, horarios, diagramas y otras imágenes generadas por computadora, videos documentales, y música. Dos cursos, uno sobre biología celular y otro de literatura inglesa, fueron enseñados usando el sistema. Todavía algunas aplicaciones en la década de los 80's incluyeron InterText, un procesador de texto; InterDraw, un editor gráfico; InterVal, un editor en línea que permite a los usuarios organizar de manera interactiva la información en tiempo y secuencias de fechas; InterSpec, un visualizador de secciones en 3D; e InterPix, un visualizador de imágenes escaneadas. Un editor de video, un editor de animación en 2D, y métodos más complejos para el filtrado.

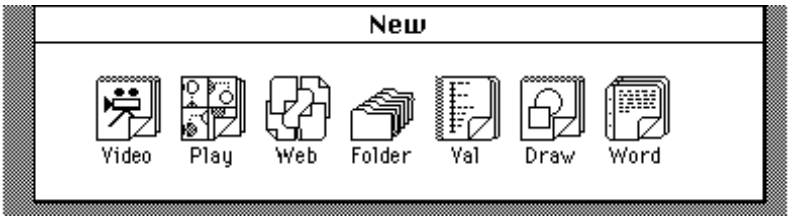

Figura 2.6: *Menú principal del sistema Intermedia*

Intermedia fue desarrollado tanto como una herramienta para los profesores para organizar y presentar sus lecciones vía la computadora como para un medio interactivo para estudiantes para estudiar los materiales y adherir sus propios comentarios y reportes.

2.5.5. Open hipermedia

Hipermedia como campo ha evolucionado desde su inicio por Bush y el trabajo fundacional de Engelbart y Nelson en una tecnología que permite a los usuarios accesar, enviar y administrar información (Conklin, 1987 [6]). El diseño de los sistemas de hipermedia también ha evolucionado, exhibiendo un incremento en los niveles de abstracción en cada generación. Los primeros sistemas de hipermedia (algunos enlistados anteriormente) fueron monolíticos y cerrados caracterizados por utilizar sus propios formatos, embebiendo los enlaces en los documentos, además de una visible falta de extensibilidad, y con esto nos referimos al hecho de que estos sistemas no podían crear enlaces en tiempo de ejecución. Los sistemas abiertos de hipermedia han sido propuestos para direccionar los problemas de interoperabilidad de los primeros sistemas. Davis (1995)[13] define un sistema abierto

de hipermedia como un sistema que es capaz de importar nuevos objetos al sistema. Los sistemas abiertos de hipermedia deben ser capaces de operar con sistemas externos e intercambiar datos con ellos. Contrario a lo que realizan los lenguajes de descripción de documentos tales como HTML o XML, los sistemas hipermedia abiertos no alteran el documento por imposición de marcas en su contenido. Los enlaces y otra información de hipermedia son mantenidos en una base de datos por separado llamado bases de datos de enlaces (*linkbases*). Las desventajas de los sistemas abiertos de hipermedia tienen que ver principalmente con los problemas de consistencia entre una selección persistente y el ancla de la información de enlaces. Entre los mayores problemas a este respecto, Davis (1995)[13] ha identificado el problema de enlaces rotos (*dangling link*). El problema de enlaces rotos ocurre cuando un ancla falla en resolución de un documento válido, debido a que los archivos pueden estar en un lugar diferente al registrado en el sistema de hipermedia.

2.5.6. Microcosm

El sistema abierto que se estudiará con detalle como base de comparación en los resultados, es el sistema Microcosm desarrollado por la Universidad de Southampton. Este permite autores y lectores para integrar grandes colecciones de documentos creados o accesados desde aplicaciones externas sin modificar sus formatos originales.

El modelo básico de Microcosm (ver la figura 2.7) tiene una organización arquitectónica de 4 capas: la de usuario, la de aplicación, de servicio de enlaces y por último la capa de almacenamiento. Concentrándonos en la capa de servicio de enlaces, cuya arquitectura adopta un modelo centrado en procesos para implementar los servicios de enlaces comprendiendo un número de procesos relativamente autónomos llamados filtros que se comunican con los otros a través de un sistema de paso de mensajes. Así, el componente de servicio de enlaces de Microcosm se descompone en una cadena de filtros, vistos como un estilo filtro-tubería (*pipe-filter*).

Los filtros pueden removerse o reordenarse arbitrariamente. Cuando un filtro recibe un mensaje desconocido, este solamente lo pasa al siguiente filtro, en la tubería. Eventualmente, los mensajes que emergen de la tubería de filtros son enviados de nuevo al despachador de enlaces para que presente el resultado al usuario correspondiente Una de las series desventajas de este tipo de arquitectura es que el orden relativo entre filtros puede afectar los resultados. Diferentes ordenamientos de los filtros dentro de la tubería produce salidas diferentes con las mismas entradas.

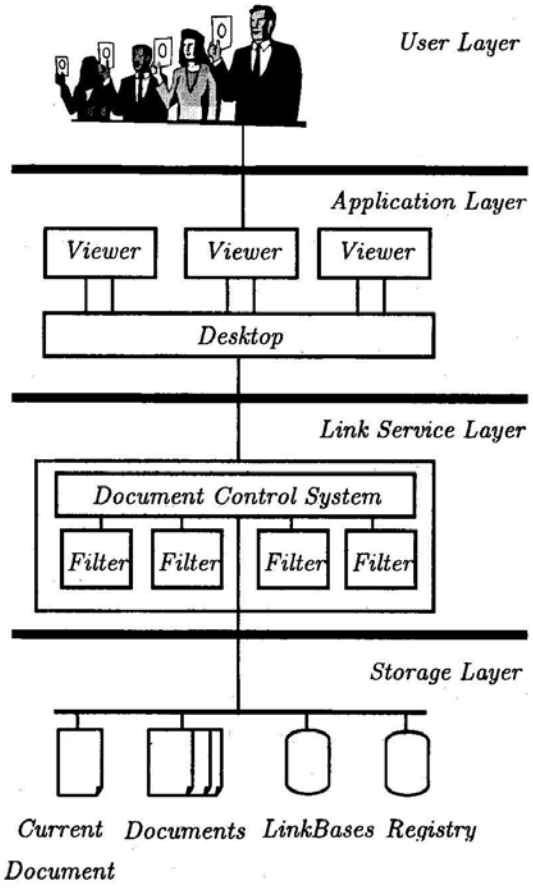

Figura 2.7: *Arquitectura de Microcosm*

2.5.7. La World Wide Web

Para simplificar las dificultades de acceso a información dentro de un entorno tan extenso y heterogéneo como Internet, y avanzar más allá de lo propuesto por Gopher, un grupo de investigadores del CERN (Laboratorio de Física de Partículas de Ginebra) ideó un nuevo servicio de información. Integraba un atractivo formato de presentación de datos, junto con un sistema para estructurar y enlazar tipos muy variados de información. Su nombre, la World Wide Web, "la red que cubre el mundo".

Este nuevo servicio de información tuvo su origen en el año 1989, en un proyecto lidereado por Tim Berners-Lee. El objetivo perseguido era disponer de un sistema de creación y distribución de documentos, que solucionase la problemática de mantener y distribuir información desarrollada en diferentes aplicaciones de edición, en un entorno de usuarios que precisan acceder a documentación común. El sistema Web está formado por:

- Unos clientes y servidores que se encargan de manejar la información. Utilizan un protocolo denominado HTTP (Hypertext Transfer Protocol), desarrollado específicamente para la Web.
- Un nuevo formato de descripción de documentos, denominado HTML (Hypertext Markup Language). Los archivos de hipertexto se desarrollan en el formato HTML, un sencillo lenguaje de descripción de documentos, con sofisticadas capacidades de representación de información.

Los documentos de hipertexto pueden contener texto, con diferentes tipos de letra y tamaño, imágenes, vídeo, sonido, etc., junto con la capacidad de integrar nuevos tipos de información que se planteen en el futuro. Además, y como aspecto fundamental, pueden contener "elementos activos", partes de un documento que al ser pulsados producen la presentación de nueva información en la pantalla de una aplicación cliente; los enlaces construyen "túneles" a través de Internet, conectando información relacionada.

Los documentos de hipertexto se presentan en la pantalla del cliente Web como páginas de un libro, por lo cual es muy común el empleo del término 'página Web'. Sin embargo, los enlaces y elementos activos rompen la estructura lineal de un libro, permitiendo que el acceso a la información se realice por varios caminos diferentes.

Se denomina World Wide Web, a la colección de documentos escritos en HTML que se encuentran en servidores que usan el protocolo petición-respuesta HTTP.

2.6. Resumen

Hasta aquí solo hemos hecho mención de las diferentes vertientes que se tienen sobre el hipertexto, sus definiciones y conceptos, por quienes así también han llevado sus ideas a la práctica, y nos han legado un gran número de sistemas que han revolucionado la idea y el concepto hasta el día de hoy, y podemos decir también que gran parte de la población mundial sin tener por lo menos la mínima idea de este concepto han interactuado con el sin saberlo.

Realizamos un breve repaso a las características ideales sobre el hipertexto, a los componentes y sus propiedades, y para finalizar el capítulo se dio una visión general sobre el funcionamiento y propiedades de los diferentes sistemas construidos, que si bien es cierto en un principio estos sistemas fueron cerrados (la interoperabilidad era nula) su creación sirvió de inspiración a las generaciones futuras de sistemas de hipertexto. Hoy en día aunque los nuevos sistemas han evolucionado hasta convertirse en sistemas abiertos (interoperables), también es cierto que esta evolución ha acarreado

consigo ciertos problemas que los desarrolladores han tenido que enfrentar y darle solución para su correcto funcionamiento.

Capítulo 3

Sistemas Abiertos de Hipermedia

En este capítulo se describe el sistema abierto de hipermedia que se construyó y los principios de diseño que determinaron su implementación. Para el diseño de nuestro sistema tomaremos como punto de partida la descripción de los modelos de enlaces y de procesos que utiliza. Los modelos de enlaces que describen el mecanismo básico de recuperación de documentos. El modelo de procesos describe la organización que rige entre los resolvedores de enlaces. Como referencia del modelo de procesos se usó el sistema Microcosm el cual fue introducido en la sección anterior. Este sistema permite tanto a autores como a lectores integrar grandes colecciones de documentos creados o accesados por aplicaciones externas sin modificar sus formatos originales.

3.1. Modelos de enlaces

A continuación se describirán los modelos de enlaces que fueron considerados en éste trabajo. Los modelos describen informalmente el significado de la acción de seguir un enlace, el cual depende de la selección del lector y que tiene efecto en la recuperación del documento asociado.

3.1.1. Enlaces específicos

Los enlaces específicos representan la forma más básica de enlace en hipermedia. Un enlace específico es un enlace unidireccional que relacionan a un par de documentos. Como se muestra en la figura 3.1, un enlace específico se puede interpretar como una instrucción "goto" que indica al visualizador cuál es el documento a recuperar. El enlace se identifica generalmente mediante un efecto visual que lo hace resaltar sobre el texto normal del documento. A pesar de que éste tipo de enlaces ha sido incluido en diferentes sistemas a lo largo de la historia de hipertexto, estudios realizados por Nordbotten [16] han revelado que el uso de éste enlace la desorientación

Figura 3.1: *El enlace específico*

y la sobrecarga cognitiva pueden impedir a los usuarios recuperar información a pesar de que se encuentre disponible. La desorientación del usuario, un sentimiento de estar "perdido en el espacio", es experimentada con el aumento de la información entrelazada. La sobrecarga cognitiva lleva a la frustración cuando existen muchas opciones que pueden elegirse para recuperar la información.

En comparación a los enlaces que ofrece la Web, los enlaces específicos son similares a ellos en funcionalidad y apariencia pero difieren en su forma de almacenamiento ya que estos se mantienen separados en bases de enlaces. Como se discutió en la introducción, el mantenimiento de la consistencia de la información entre documentos y bases de enlaces da lugar a los problemas de los sistemas abiertos de hipermedia discutidos ya en la sección 1.2. Para evitar éstos problemas se ha recomendado el uso de otras formas de enlaces para sistemas abiertos que no conllevan tantos problemas como los causados por los enlaces específicos.

3.1.2. Enlaces genéricos

Los enlaces genéricos resuelven los problemas introducidos por los enlaces específicos. Como se describió en la sección anterior, los enlaces específicos pueden acarrear muchos problemas para el lector. Por otro lado desde la perspectiva del autor, crear hipertexto a gran escala es una tarea ardua, si el número de posibles nodos crece de manera significativa. La proliferación de enlaces específicos crece entonces geométricamente y puede ocasionar al autor confusión y mucho trabajo. Los enlaces dinámicos implementados en este trabajo los hemos denominado enlaces genéricos. Un enlace genérico es un enlace unidireccional que relaciona un fragmento de texto a un ancla destino

fija; esto es, relaciona un fragmento de texto a uno o varios documentos. Podemos decir que un enlace genérico corresponde a una entrada en un diccionario donde el origen corresponde al término que se define y el ancla destino al documento que tiene la definición. La figura 3.2 muestra en forma clara la forma de actuar de un enlace genérico.

Aunque la mayoría de las interfaces de hipertexto son consideradas apropiadas para las tareas de exploración de información, el creciente aumento en el esfuerzo para la creación de enlaces, sin embargo, impide que ésta sea usada para colecciones grandes de documentos. Tales colecciones requieren de interfaces basadas en consultas. El sistema descrito crea enlaces de manera dinámica. Golovchinsky [17] realizó un experimento para comparar el comportamiento de una búsqueda basada en enlaces creados de manera dinámica y otra en enlaces específicos. El resultado sugiere que los enlaces dinámicos son tan efectivos como las consultas explícitas.

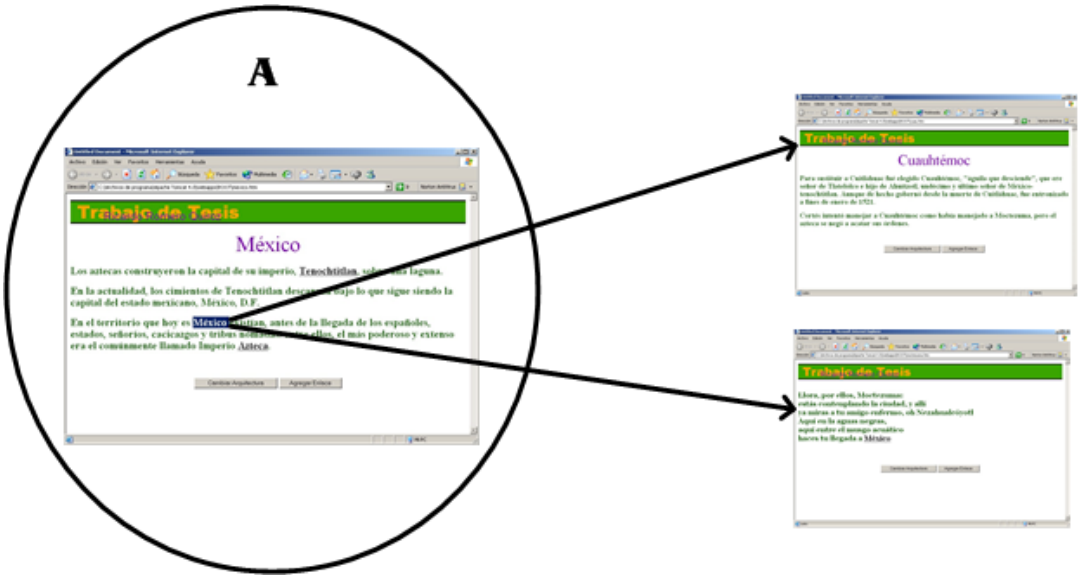

Figura 3.2: *El enlace generico*

3.1.3. Enlaces de avance

La idea de los enlaces de avance es describir el movimiento de un documento desde una localidad previa a su localidad actual. Con esta información, un enlace roto (*dangling link*) puede ser reparado por un enlace de avance a menos que esté sea inválido también. Supongamos que un documento tiene una referencia a otro documento “d” en el nodo “A”, si el documento “d” no se ha movido del nodo, o no ha cambiado no hay problema cada vez que se referencíe, se visualizará en el browser

(figura 3.3).

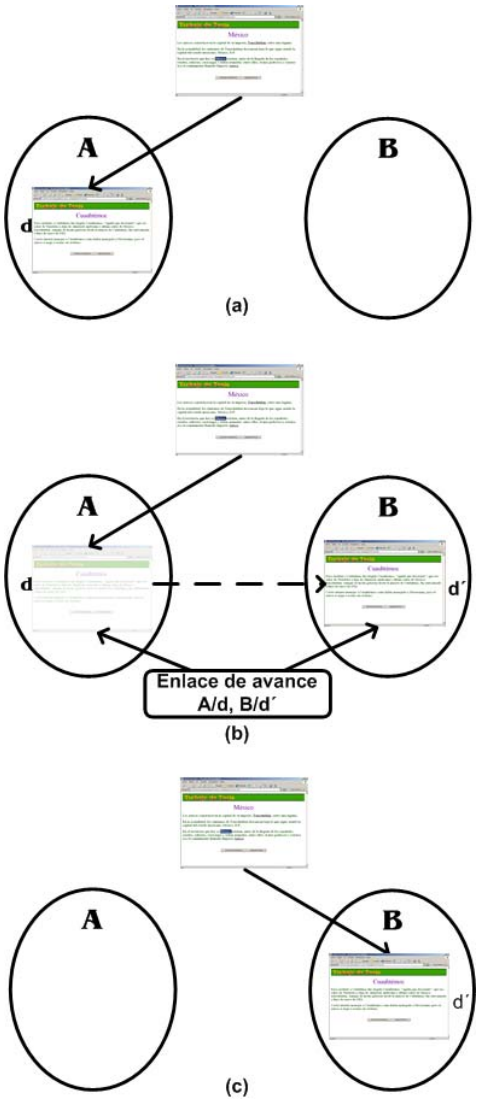

Figura 3.3: *a)Un enlace referenciando un documento válido. b)Enlace inválido. c) Enlace reparado.*

Si el documento "d" se mueve del nodo "A" al nodo "B", ¿Qué pasa?, simple y sencillamente el browser no puede resolver la referencia del documento de origen. El enlace de avance evita que esto ocurra, para ello se registra en el sistema el movimiento del documento "d" de "A" a "B" en la base de datos de enlaces, de modo que cuando se hace referencia al documento "d", el sistema, en primer lugar resuelve el enlace y después verifica que el enlace no contenga un enlace de avance, si es así devuelve al browser el enlace de avance para su visualización.

Con el enlace de avance podemos actualizar la base de datos de enlaces, devolviéndole la pro-

piedad de integridad fuerte, con solo llenar un registro tomado de la base de enlaces de avance. Nuestro propósito es diseñar y construir un enlace de avance para proveer la funcionalidad hipermedia como un servicio que repare los enlaces rotos al tiempo de la exploración. Este experimento proveerá también visiones profundas con respecto los problemas de extensibilidad de la arquitectura de conducto-filtro.

Creemos que el problema de los enlaces rotos no puede ser resuelto de manera satisfactoria a menos que algunas condiciones sean impuestas sobre las propiedades de las bases de enlaces, como se discute a continuación. Una base de enlaces tiene la *propiedad de integridad* cuando esta no almacena alguna inconsistencia. Para todas las bases de enlaces es deseable tener la propiedad de integridad antes que las aplicaciones puedan modificar su ambiente. Como las aplicaciones pueden cambiar eventualmente el ambiente de las bases de enlaces, una base de enlaces puede perder la propiedad de integridad. No obstante, la base de enlaces podría después recobrarla de nuevo manteniendo por lo menos una forma débil de integridad caracterizada por el guardado de información sobre los enlaces rotos y algunas otras acciones correctivas.

Con el enlace de avance podemos actualizar la base de datos de enlaces, devolviéndole la propiedad de integridad fuerte, con solo llenar un registro tomado de la base de enlaces de avance. Nuestro propósito es diseñar y construir un enlace de avance para proveer la funcionalidad hipermedia como un servicio que repare los enlaces rotos al tiempo de la exploración. Este experimento proveerá también visiones profundas con respecto los problemas de extensibilidad de la arquitectura de conducto-filtro.

Asumimos que en una base de enlaces únicamente puede tener dos posibles situaciones ya sea que no hay enlaces rotos o que para cada enlace roto hay exactamente un enlace de avance que lo repara. No obstante, una complicación más fuerte viene cuando consideramos a los enlaces de avance como enlaces de primera clase porque ellos pueden tener enlaces rotos también. Afortunadamente, el proceso de reparación naturalmente se extiende a los enlaces de avance. De acuerdo a la propiedad de integridad débil de las bases de enlaces, un enlace de avance roto puede ser reparado por otro enlace de avance. Porque el movimiento de los documentos ocurre de manera independiente de los enlaces que los referencian, la notificación de una violación de integridad es también independiente de los enlaces creados previamente.

Debido a su naturaleza y propósito, estos enlaces se implementan en una base de datos, que

contiene el origen y el destino de aquellos documentos cuyo movimiento haya sido registrado por el usuario. Por este motivo estos enlaces no aparecen en la capa de aplicación sino que están definidos en las dos capas mas bajas del sistema, y aquí se hace mención de ellos para tener un conocimiento previo de su funcionamiento.

3.2. Modelos de procesos

Como se describió en el capítulo 3, Microcosm es un sistema de hipermedia que permite importar documentos sin imponer ningún tipo de marcado como lo requiere la Web con los documentos HTML. Este carácter abierto es el principio que ha motivado su uso en este trabajo como modelo de referencia para el modelo de procesos. En Microcosm, el lector puede navegar a través de la información multimedia utilizando aplicaciones especializadas llamadas *visualizadores*. El Sistema de Control de Documentos (*Document Control System, DCS*) provee facilidades para la integración de los visualizadores. A través del *DCS*, el usuario interactúa con Microcosm seleccionando fragmentos especiales de texto, conocidos como *anclas*, que se distinguen de otros fragmentos por estar unidos a un enlace (ver figura 2.7). Un visualizador provee la funcionalidad básica para desplegar el ancla de un enlace en un documento usando algún efecto visual. Cuando un lector selecciona el texto de una ancla haciendo clic sobre él, un mensaje que contiene información tanto del fragmento de texto seleccionado como de su contexto es enviado desde el visualizador al *DCS*, el cual pasará el mensaje a través de una cadena de resolvedores de enlaces, también llamados *filtros* en Microcosm.

3.2.1. Arquitectura conducto-filtro

Un filtro es representado por un proceso comunicante que implementa un método para resolver un tipo de enlace. Los filtros pueden ser instalados, removidos o reordenados en un arreglo lineal llamado el *conducto*. Cuando un filtro recibe un mensaje, el filtro envía al siguiente en el conducto o bien una respuesta o bien al mismo mensaje. Cada filtro puede reconocer y responder a un mensaje de acuerdo al modelo de enlace que el filtro implementa. Cuando un filtro reconoce un mensaje del mismo tipo, trata de determinar si existe un enlace asociado en la base de enlaces. Si tal enlace existe, recupera la información y la pasa como respuesta al siguiente filtro. Por otra parte, cuando un filtro recibe un mensaje cuyo tipo es desconocido, éste simplemente pasa el mensaje inalterado al siguiente filtro. Cuando los mensajes emergen eventualmente del conducto, estos son enviados al despachador de enlaces que presenta el resultado obtenido al lector. El despachador de enlace puede solicitar la intervención del lector para seleccionar entre varios enlaces disponibles si hay más de uno. El enlace seleccionado es enviado al *administrador de enlaces* que interpreta su contenido y

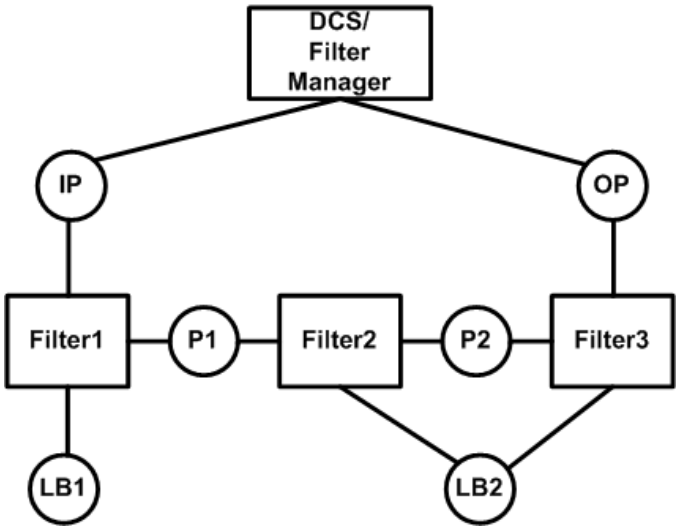

Figura 3.4: *La arquitectura conducto-filtro*

actúa en respuesta. Entre las posibles acciones pueden ser cargar el nuevo documento en el mismo visualizador o iniciar otro y entonces cargar el nuevo documento. Después, el lector podría continuar leyendo el documento recuperado si así lo desea.

En la figura 3.4, los componentes son representados por cajas etiquetadas y comunicadas por espacios conectores de datos representados por círculos etiquetados. Las líneas relacionando pares de componentes y conectores determinan los patrones estructurales entre los elementos de la arquitectura. La característica topología lineal de la arquitectura conducto-filtro puede ser claramente apreciada en la figura 3.4.

En la arquitectura, tanto el visualizador como el *DCS* pueden acceder al documento que recibe el centro de atención. En la implementación de nuestro sistema, el documento es representado conceptualmente por una colección de pares dirección-token, ordenados por dirección, en donde un token es un fragmento de texto de extensión variable. Esta representación permite reconstruir al documento completo a partir de la yuxtaposición de los fragmentos que lo componen, ordenados de acuerdo a su dirección. Al mismo tiempo, ésta representación permite identificar los fragmentos de texto que sirven de ancla para un enlace. Cuando una selección persistente ha sido seleccionada por el lector para seguir un enlace, un mensaje que contiene el par dirección-token es enviado desde el visualizador al *DCS* como se indicó antes.

Al igual que la interpretación dada a los documentos, se puede dar una interpretación a las base

de datos de enlaces ya que pueden ser consideradas como colecciones de pares de anclas origen-destino. Esta representación sugiere un tratamiento uniforme de la representación al considerar tanto a los documentos como a las bases de enlaces como un medio de coordinación basada en espacios de términos [15]. Debido a que nuestra implementación es diferente a la presentada en ese trabajo, la descripción más abstracta del modelo formal de coordinación no será necesario abordarla aquí. Nuestra implementación utiliza mecanismos de sincronización más convencionales tales como peticiones del protocolo HTTP a un servidor Web. Lo único que es relevante para éste trabajo es la organización de los componentes y su comunicación. En la descripción de las arquitecturas, los conectores se interpretan simplemente como un medio de comunicación entre procesos indirecto y asíncrono. Esta forma de comunicación, asegura la interacción desacoplada de los procesos, lo que garantiza el mayor grado de concurrencia que es posible obtener de ellos.

Como se muestra en la figura 3.4, el conducto recibe solicitudes del *DCS* a través del conector *IP*. A su vez, el conducto envía el resultado de regreso al *DCS* por medio del conector *OP*. Un número de conectores (por ejemplo, *P1* y *P2*) son puestos entre pares consecutivos de filtros (*filtro1* y *filtro2*) para permitir el flujo de información a través del conducto. Los filtros usan posiblemente base de enlaces compartidas que podrían distribuirse físicamente en una red de computadoras. En la figura mencionada, se puede apreciar que el *filtro1* utiliza la base de enlaces *LB1* en tanto que el *filtro2* y el *filtro3* comparten la base de enlaces *LB2*. El *DCS* pide al usuario que seleccione un resultado si más de uno se produce en el conducto. Después de descartar los resultados remanentes, el despachador permite que la selección del usuario sea obtenida finalmente por el *DCS*. El nombre del nuevo documento es enviado al visualizador, el cual en respuesta, despliega al usuario el documento recuperado.

3.2.2. Arquitectura de pizarra

Los *sistemas de pizarra* (*blackboard*), derivados de los sistemas de producción, están centrados en la noción de una memoria compartida (Hayes-Roth 1995, Nilson 1998). Una pizarra es una estructura de datos que es leída y modificada por programas llamados *fuentes de conocimiento* o *agentes*, como se muestra en la figura 3.5. Un agente puede codificarse en términos de *reglas de producción*, pares condición-acción, que respectivamente reconocen y modifican patrones relevantes de datos y que actúan de manera oportuna sobre la pizarra. Dicho de otra forma, la pizarra puede verse como un repositorio central de datos que puede utilizarse para habilitar la comunicación entre agentes. La arquitectura de pizarra logra un mayor grado de flexibilidad debido a que las dependencias entre los agentes ya no dependen de la restrictiva secuencialidad impuesta por la

arquitectura conducto-filtro. La comunicación entre los agentes se consigue por un mecanismo de notificación que permite activar al agente apropiado de acuerdo al tipo de dato (mensaje) puesto en la pizarra. El agente revisa entonces el contenido de la pizarra y lo retira si es de su interés. Si hay más de un agente que recibe la notificación, se usa una estrategia para resolver los conflictos que pueden surgir por la aplicación de más de una regla. En este sentido, la pizarra es similar a una memoria transaccional que se rige por el principio de "todo o nada", es decir, que una regla se aplica solamente cuando están todos los datos que satisfacen a su parte condicional. De otra forma, la regla no se aplica y el contenido de la pizarra queda inalterado.

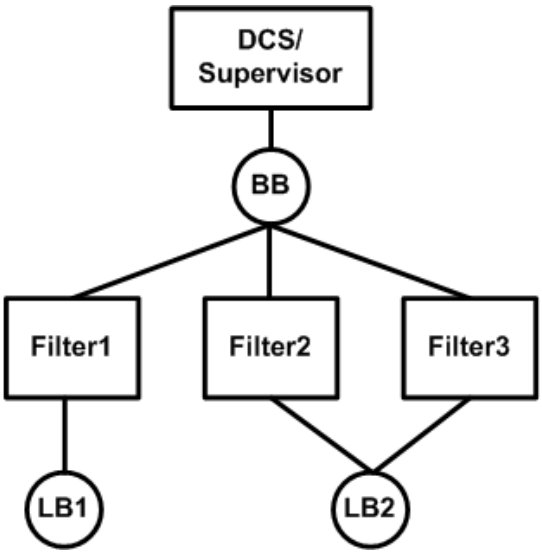

Figura 3.5: *La arquitectura de pizarra*

En la arquitectura de pizarra, el supervisor reemplaza al administrador de filtros de la arquitectura conducto-filtro mientras que los agentes reemplazan a los filtros. El supervisor se comunica con el *DCS* y media todas las actividades de coordinación entre los agentes. El *DCS* envía solicitudes de servicio de enlace al supervisor las cuales pueden ser cualesquiera de las descritas en la siguiente sección. Una solicitud de servicios de enlace es puesta inicialmente en la pizarra por el supervisor quien notifica a los agentes registrados a que revisen el contenido de la pizarra. De acuerdo al tipo de enlace, los agentes pueden producir modificaciones a los datos, por ejemplo, como resultado de la resolución de un enlace. Dicha actividad continúa incesantemente hasta que los agentes ya no pueden obtener más modificaciones. Este criterio de terminación indica intuitivamente que los datos que quedan en la pizarra son aquellos que ya no pueden ser elaborados aún más por los agentes. En este punto, lo único que queda por hacer es que el supervisor intervenga para retirar estos datos y enviarlos al *DCS* como respuesta.

3.3. Comparación de las arquitecturas conducto-filtro y pizarra

En comparación a la arquitectura de conducto-filtro, en la arquitectura de pizarra no es evidente cuál puede ser el flujo de datos entre los agentes ya que la comunicación está centralizada en la pizarra. Cuando la pizarra está vacía, el supervisor coloca una solicitud de resolución de enlace, notificando a todos los agentes registrados a que atiendan la solicitud. Cuando un agente recibe eventualmente una notificación, éste se esfuerza por proveer el servicio tan pronto como la solicitud está disponible en la pizarra. Para eliminar cualquier dependencia, los agentes compiten sobre una base justa para tomar la solicitud que se encuentra en la pizarra. Una vez que todos los agentes han concluido su participación, el supervisor determina si hubo algún cambio en los datos. En caso de que los datos se hayan modificado, el supervisor notifica de nuevo a todos los agentes registrados para que vuelvan a proporcionar sus servicios. Si no hubo cambios en los datos, el supervisor envía el contenido de la pizarra al despachador de enlaces.

El estilo arquitectural conducto-filtro presenta las siguientes ventajas:

- **Simplicidad.** Se deriva del acceso exclusivo que cada filtro tiene sobre la operación que fluye a través del conducto, evitando mecanismos elaborados de coordinación para prevenir conflictos en los accesos a datos.
- **Concurrencia.** Como cada filtro se representa por un proceso con estado local, la arquitectura permite que las solicitudes que están fluyendo a través del conducto pueden procesarse simultáneamente por los filtros del conducto.
- **Flexibilidad.** Resulta de la organización lineal de la arquitectura que permite a nuevos filtros ser instalados a la tubería fácilmente.
- **Análisis formal.** Debido a la independencia de cada filtro, es posible obtener un análisis formal de cada uno.

Sin embargo, esta arquitectura presenta también algunos inconvenientes entre los cuales podemos citar.

- **Anomalías en el ordenamiento de los filtros.** El orden relativo entre los filtros en el conducto puede afectar los resultados. Un ordenamiento diferente de los filtros dentro del conducto produce salidas diferentes con las mismas entradas. Por abstracción los filtros son comparables a funciones que transforman datos y el conducto es similar a la composición

de funciones, el problema naturalmente se da porque la composición de funciones es en general una operación no conmutativa. Como un ejemplo, consideremos los enlaces genéricos, específicos y los de avance. Aunque el orden relativo entre los específicos y los genéricos es irrelevante, la posición de los enlaces de avance es esencial. Para entender porque la colocación de los enlaces de avance es importante considere la siguiente propiedad de ésta arquitectura cuando contiene filtros que producen enlaces inválidos: el filtro divide al conducto en dos partes de modo que cualquier referencia inválida que el filtro produzca deberá aparecer después del filtro. Por lo tanto, si los filtros de los enlaces de avance aparecen antes del filtro de un enlace genérico, no podrán reparar los enlaces rotos porque aún no han sido generados. Por el contrario, si los filtros de los enlaces de avance se encuentren después del enlace genérico, entonces podrán reparar los enlaces rotos que se produzcan.

Para ilustrarlo, supongamos que en la tubería se instalan primero los filtros de enlaces específicos y genéricos y luego el filtro de los enlaces de avance. Por consiguiente, por acomodar el enlace de avance después del genérico y específico, podemos asegurar que los enlaces rotos pueden ser reparados. Por el contrario, colocando el enlace de avance antes del específico y genérico, los enlaces no pueden ser reparados porque el enlace de avance está en la parte de la tubería donde los enlaces no han sido producidos en absoluto.

- **Falta de poder de procesamiento.** Curiosamente, en la arquitectura conducto-filtro, un enlace válido podría no resolverse incluso si un enlace de avance está, instalado en la tubería y si todos los enlaces de avance requeridos para reparación del enlace están disponibles. Incluso asumiendo un orden apropiado de filtros en la tubería, el problema continuaría. La causa del problema está en el estricto acceso secuencial de los filtros a los flujos de datos impuesta por la arquitectura.

  Como se analizó anteriormente, los enlaces de avance son enlaces de "primera clase" que pueden quedar rotos también. Entonces, más de una reparación puede requerirse para obtener un enlace válido. Sin embargo, la arquitectura conducto-filtro podría reparar un enlace a lo más una vez, haciendo imposible poder reparar enlaces de avance rotos. Esto es un fenómeno interesante que demuestra que los enlaces de avance no pueden ser soportados totalmente por la arquitectura conducto-filtro.

- **Servicio a solicitudes no deseadas.** Esto de debe a que todos los filtros están conectados en un arreglo lineal, el flujo de datos siempre fluye a través de todos los filtros. Esto significa que todos ellos deben examinar la solicitud.

3.4. Comparación entre Arquitecturas

Otro objetivo de la tesis es comparar la expresividad entre las arquitecturas conductor-filtro y de pizarra. Podemos afirmar que la de pizarra es la más expresiva como se discute a continuación.

En la arquitectura conducto-filtro, el resultado correcto en la resolución de enlaces depende tanto del ordenamiento de los filtros en el conducto como del número de enlaces de avance involucrados en el seguimiento de un enlace. Cuando el enlace hace referencia a un documento que no se ha movido, no habrán registrados enlaces de avance en la base de datos por lo que no importa entonces el orden de los filtros y, como consecuencia, el documento será recuperado siempre con éxito. Por otra parte, cuando el enlace de avance involucra enlaces de avance porque el documento se ha movido, la colocación del filtro de los enlaces de avance al final del conducto asegura que la resolución del enlace tenga éxito. El peor caso ocurre cuando la resolución del enlace involucra enlaces de avance y cuyos filtros se encuentran antes de los filtros que pueden resolver enlaces como los genéricos. El problema se presenta porque la petición de resolución de enlaces llega primero al filtro de los enlaces de avance, aún cuando no hay nada que reparar, y luego al filtro del enlace genérico, en donde se podría producir entonces un enlace inválido, el cual ya no podría ser reparado porque el filtro de los enlaces de avance ya fue visitado. Una ventaja de esta arquitectura es que el tiempo de procesamiento es igual a la suma de los tiempos de procesamiento de cada filtro en la tubería. Cuando el usuario realiza una consulta, la petición pasa a través de cada filtro una sola vez y no más, sea la resolución correcta o incorrecta.

Dado que elimina la secuencialidad de procesamiento que caracteriza al conducto de filtros, la arquitectura pizarra siempre va a tener éxito porque elimina las dependencias entre los filtros. Las peticiones de enlaces de los usuarios siempre van a obtener el resultado correcto en la resolución del enlace. ¿Porqué? muy sencillo, en esta arquitectura no hay un orden en los agentes de software para la resolución de enlaces, la petición se coloca en un repositorio y cada uno de los agentes en su oportunidad toma la petición para procesarla y al terminar el trabajo coloca sus resultados de nuevo en el repositorio para que otro agente realice el mismo trabajo. Los agentes dejan de trabajar cuando en un momento dado ninguno de ellos logra modificar el repositorio. Pero como todo esta arquitectura tiene una desventaja y esta es el tiempo de procesamiento, cuyo valor no podemos estimar debido a su inherente naturaleza aleatoria. Lo que si podemos decir es que el tiempo de procesamiento va a depender del número de enlaces de avance involucrados en la petición del usuario. Hay que agregar también que al final hay que sumar al tiempo una ronda extra para comprobar que ninguno de los agentes puede realizar más cambios al repositorio.

3.5. Resumen

En este capítulo, se describieron los modelo de enlaces que fueron usados en el sistema así como el funcionamiento de las arquitecturas implementadas en el proyecto. En el capítulo se describen y analizan las características de los enlaces específicos, genéricos y de avance que extienden el modelo simple de enlace de la Web. Después se analizaron los modelos de procesos que se caracterizan por la arquitectura usada para sincronizar y comunicar a los procesos que representan a los servidores de enlaces, también designados como filtros o agentes en este capítulo. Finalmente, se presentó un análisis comparativo entre las arquitecturas filtro-tubería y de pizarra que fueron instrumentadas en el sistema.

Capítulo 4

Aplicación: Enriqueciendo Documentos Web con Referencias Cruzadas

En éste capítulo se plantea y resuelve un problema de recuperación de información en la Web el cual permitirá demostrar el aumento en el poder expresivo conseguido por el sistema de hipermedia desarrollado. La funcionalidad adicional que ha sido incorporada tanto al servidor como al visualizador permite resolver problemas que no se podrían haber resuelto sin las extensiones al modelo de la Web. El problema consiste en enriquecer los documentos públicamente disponibles de la Sección de Computación del CINVESTAV con enlaces a información referente al personal académico, programas de estudio y proceso de admisión. Con las extensiones, si se selecciona, por ejemplo, el nombre de una materia en el texto, el sistema permite obtener mayor información de ésta siguiendo un enlace. Los documentos se encuentran distribuidos en varios servidores, en algunos de los cuales no se cuenta con permiso para modificar los documentos para insertar enlaces. La solución, basada en el enfoque de los sistemas abiertos de hipermedia, consiste en definir un diccionario que permita recuperar la información utilizando enlaces genéricos. Dicha solución permite extender la funcionalidad de la Web sin requerir de permisos de acceso a los documentos para introducir enlaces. Por otra parte, el uso de los enlaces genéricos simplifica enormemente el desarrollo de la aplicación porque no exige introducir un nuevo enlace por cada ocurrencia en el texto de un término definido en el diccionario.

La aplicación permitirá también mostrar como definir nuevos enlaces genéricos y como seleccionar el tipo de arquitectura que mejor se adecúe al tipo de enlaces que usa la aplicación.

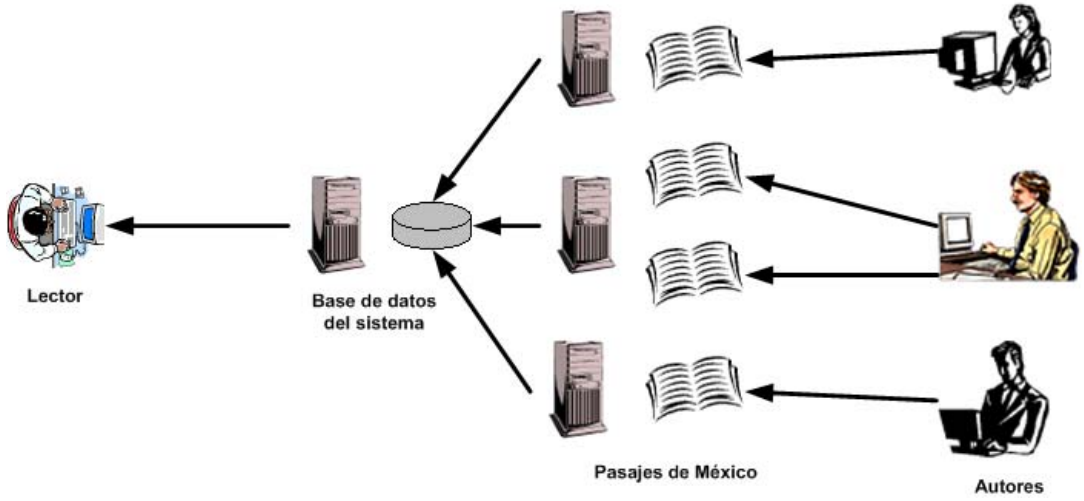

Figura 4.1: *Caso de Estudio.- Documentos de la sección de Computación*

4.1. El servidor Web

El sistema trabaja bajo el ambiente Windows, y en este caso el sistema se probó en la versión de Windows Xp professional Edition, y el servidor http que se instaló fue el Apache Tomcat 4.0 disponible en el sitio de Jakarta Project, la razón para elegir este servido es que Tomcat es la implementación de referencia oficial para las especificaciones Servlet 2.2 y JSP 1.1. Puede ser usado como pequeño servidor para probar páginas JSP y servlets, o puede integrarse en el servidor Web Apache, además de gratuito. La figura 4.2 muestra los logos tanto del Jakarta como de Tomcat.

Ya instalado el servidor Web Apache Tomcat, se crea automáticamente un directorio de nombre Webapps, que es el directorio donde se instalan tanto nuestros servlets como las páginas de prueba del sistema. Tomcat al instalarse configura el puerto 8080, como el puerto predeterminado para trabajar, mismo que utilizamos para hacer las pruebas. Las páginas de prueba se instalan en el directorio ROOT del servidor, y los servlets ya compilados se instalan en el directorio webapps/tesis/WEB-INF/classes. Es necesario dar de alta la aplicación en el descriptor del servidor, este se encuentra en el directorio conf/server.xml de la instalación de Tom Cat; así como también dar de alta los servlets en el descriptor web.xml que se encuentra en webapps/tesis/WEB-INF/. Para facilitar la instalación del sistema estos descriptores se encuentran listos en el disco de la aplicación, el único paso a seguir es sobreescribir estos archivos una vez que se haya completado la instalación del Tom Cat y la aplicación.

Figura 4.2: *Logos de Tomcat y Jakarta*

4.2. El navegador

El navegador utilizado en el sistema es el Microsoft Internet Explorer 6.0, este debido principalmente al soporte que tiene para JavaScript.

4.3. Descripción del problema

Para demostrar la aplicación del prototipo desarrollado, en este trabajo de tesis se considera la colección de documentos relacionados de la Sección de Computación, que tiene como función principal permitir al lector navegar a través de la información académica mediante una asociación de ideas, frases o palabras. Esto es, cuando el lector está leyendo un documento, este puede seguir los enlaces específicos embebidos en el pasaje, o buscar un nuevo documento que esté asociado con una frase o una palabra contenida en dicho documento. Los enlaces embebidos nos llevan a un nuevo documento relacionado según lo haya definido el autor del documento. La imagen 4.1 nos muestra en forma completa todos los pasos que se llevan a cabo en este caso de estudio.

Los pasos son los siguientes:

- Uno o varios autores escriben los documentos con temas relacionados con la sección de computación.
- Estos mismos autores en principio escriben algunos enlaces hipermedia relacionando diferentes documentos.
- Los documentos se almacenan en uno o más servidores Web.
- En el servidor de la aplicación se tiene una base de datos que contiene aquellos enlaces de tipo genérico a los diferentes documentos creados por los diferentes autores.
- El lector solo necesita acceder al servidor de la aplicación usando un visualizador para consultar los documentos de la colección.

Figura 4.3: *Página inicial del sistema*

4.3.1. Actividades del Lector

El lector comienza a recuperar información cuando abre la página principal de la Sección de Computación utilizando un visualizador de Internet. El documento incluye uno o varios enlaces específicos, los cuales son reconocidos por el subrayado dados por el visualizador. La figura 4.3 muestra la página principal así como sus enlaces específicos. Al igual que en la Web, al seleccionar el lector un enlace, el sistema recupera el documento referido presentándolo en el mismo visualizador. Por otra parte, el lector puede crear nuevas rutas de búsqueda, relacionando un texto del documento con algún otro documento. En otras palabras, cuando el lector selecciona una fragmento de texto del documento del que desea obtener mas información, el sistema busca en su base de datos los documentos que están relacionados con el fragmento de texto seleccionado.

Por último el lector si lo desea puede agregar al sistema nuevos enlaces del tipo genérico. Esto con el fin de que el lector tenga la posibilidad de crear nuevas asociaciones dentro de la colección de documentos que integran la información de la Sección de Computación. De esta manera, el lector puede crear sus propios enlaces ya sea para la búsqueda de alguna materia, profesor, proyecto o artículo que esté integrado en la colección de documentos.

Por otro lado, gracias al sistema el lector tiene también la capacidad de seleccionar el método

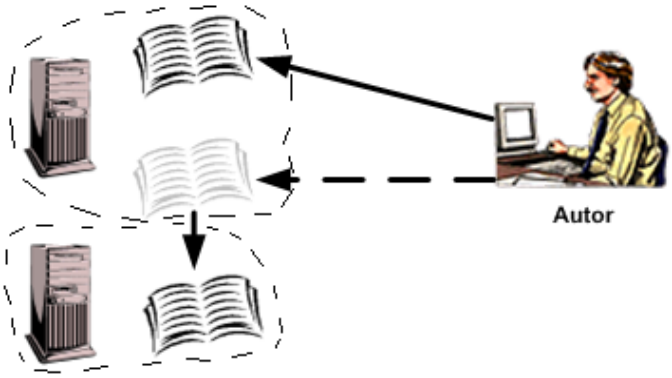

Figura 4.4: *El autor puede modificar o cambiar de ubicación un documento*

de resolución de enlaces que prefiera, ya sea a través de la arquitectura conducto-filtro, como la implementada en el sistema Microcosm, o bien, con la arquitectura de pizarra [15] que ha sido implementada en este sistema (ver figura 4.9).

4.4. Como usar el sistema

El sistema trabaja con los documentos de la sección de computación del departamento de ingeniería eléctrica del Cinvestav.

Es necesario en primer lugar arrancar el servidor Web Apache Tomcat:

`Inicio - Programas - Apache Tomcat 4.0 - Start Tomcat`

Una vez iniciado el servidor, abrir una ventana del Internet Explorer y escribir la dirección siguiente en la barra de direcciones:

`http://localhost:8080`

Con esto se cargará la página de inicio del sistema como se muestra en la figura 4.3 y a continuación seleccionamos el enlace de nombre Cinvestav, para trabajar ya directamente con los documentos de la sección de computación.

4.4.1. Enlaces Específicos

En este momento los enlaces específicos no están implementados en el conjunto de documentos de la sección debido a que no es posible tener acceso a ellos para el trabajo de edición, por este

motivo no se presenta ningún ejemplo de como trabajar con ellos aunque el sistema si los implementa y en los capítulos 5 y 6 se habló de ellos.

4.4.2. Enlaces genéricos

Para generar un enlace genérico se siguen los siguientes pasos (figura 4.5):

- Seleccionar un trozo de texto.
- Elegir la opción copy del menú Edit.

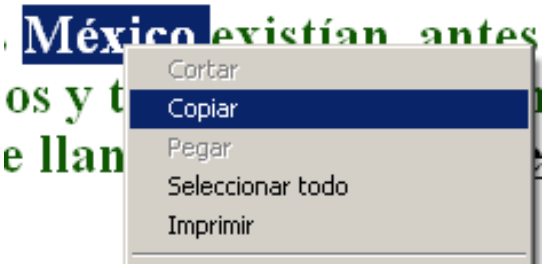

Figura 4.5: *Menu Copy del Navegador*

Una vez que se lleva a cabo el paso 2, automáticamente se abrirá una caja de diálogo para consultar al usuario de si desea o no llevar a cabo la búsqueda de enlaces con el texto seleccionado. (figura 4.6)

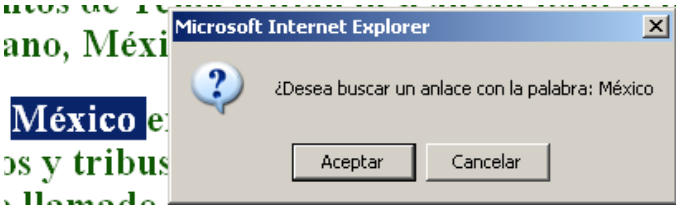

Figura 4.6: *Caja de diálogo del sistema para confirmar la búsqueda del enlace genérico*

La respuesta cualquiera de ambos casos, tanto de los enlaces genéricos como de los específicos, es un documento HTML que muestra los enlaces a los documentos en los cuales está la referencia buscada, esto cuando un enlace contiene más de una referencia en la base de enlaces (véase la figura 4.7), ahora cuando el enlace solo tiene una referencia en la base de enlaces entonces el sistema trae directamente el documento si es que existe un resultado a la búsqueda realizada.

Figura 4.7: *Resultados obtenidos de una búsqueda con más de un resultado.*

4.4.3. Cambio de localidad de documentos

Una de las actividades que se presentan con bastante frecuencia en los sistemas de información es el problema de reubicar documentos. Con esto queremos decir que el autor de algún documento puede mover un documento de un lugar a otro, incluso si el autor así lo desea, puede cambiarle el nombre al documento, véase la figura 4.4. El problema es más que evidente, cuando un lector desea acceder a un documento que fue movido a otro sitio, ya que simplemente no lo encontrará. Esto conlleva a una actividad extra en el sistema, notificar al sistema la nueva posición del documento que en su momento el autor movió de locación. Con esta acción, es posible por lo menos conocer el nuevo destino del pasaje, para mantener actualizado el sistema.

Como se mencionó en el párrafo anterior la solución al problema de modificaciones o traslados de un documento de una locación a otra, es la notificación al sistema. Si el autor realiza una de éstas actividades y no las notifica, entonces el lector no podrá recuperar aquellos documentos que hayan sufrido un cambio de lugar.

Cada vez que un autor realiza la notificación, lo hace vía el formulario de HTML de la figura 4.8. El formulario solicita al autor el URL origen del documento así como el URL destino que contendrá ahora al documento. Cuando se realiza esta acción en el sistema, éste crea entonces en la base de enlaces del servidor principal un enlace de avance para que la base de datos no pierda

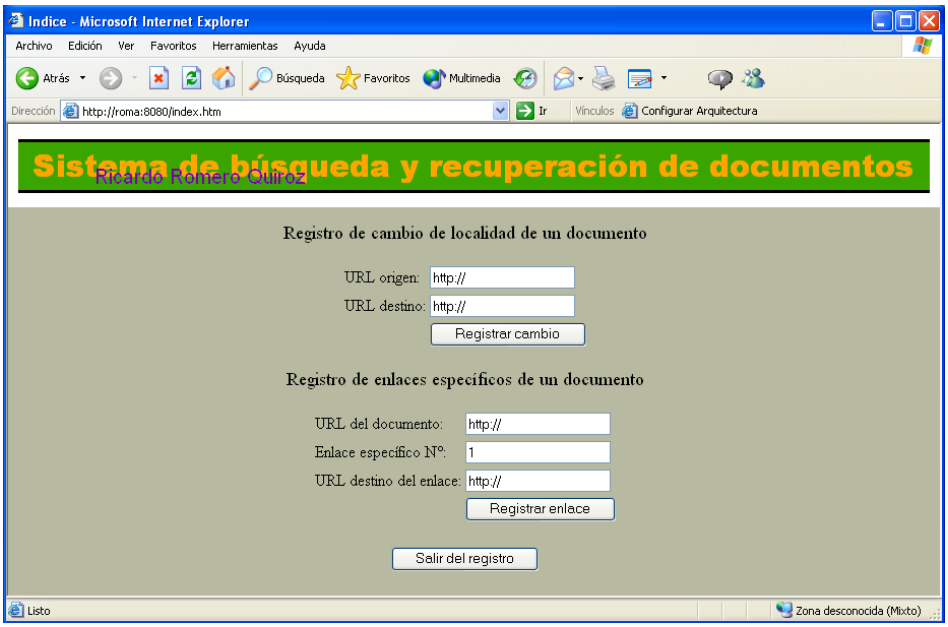

Figura 4.8: *Formulario para el registro de cambios en el sistema*

su propiedad de integridad. De esta manera cualquier lector que utiliza el sistema pueda siempre encontrar el documento que sin importar su localidad final.

Sin embargo, existe otro problema que tiene más bien que ver con el método de resolución de enlaces seleccionado y es, en este problema, en donde se centra otro de los objetivos de este trabajo. La siguiente sección aborda el problema que aquí se menciona.

4.5. Resolución de Enlaces

Cuando el autor decide mover un documento puede notificar al sistema el cambio de ubicación mediante el diálogo de la figura 4.8. Para el lector el sistema debe ocultar estos cambios y por lo tanto debe ser capaz de recuperar los mismos documentos resolviendo los enlaces hipermedia tan eficazmente como sea posible. La resolución eficaz de los enlaces depende de la organización arquitectónica de los filtros, es decir, de la forma como se conectan. El sistema ofrece la posibilidad de organizar los filtros de diferentes formas. La figura 4.9 nos muestra el formulario que contiene el menú de opciones para configurar la arquitectura deseada para la resolución de enlaces. Cuando el lector selecciona una arquitectura, un nuevo formulario HTML para cada arquitectura se presenta al usuario con el fin de configurar apropiadamente cada arquitectura propuesta. Si el lector selecciona

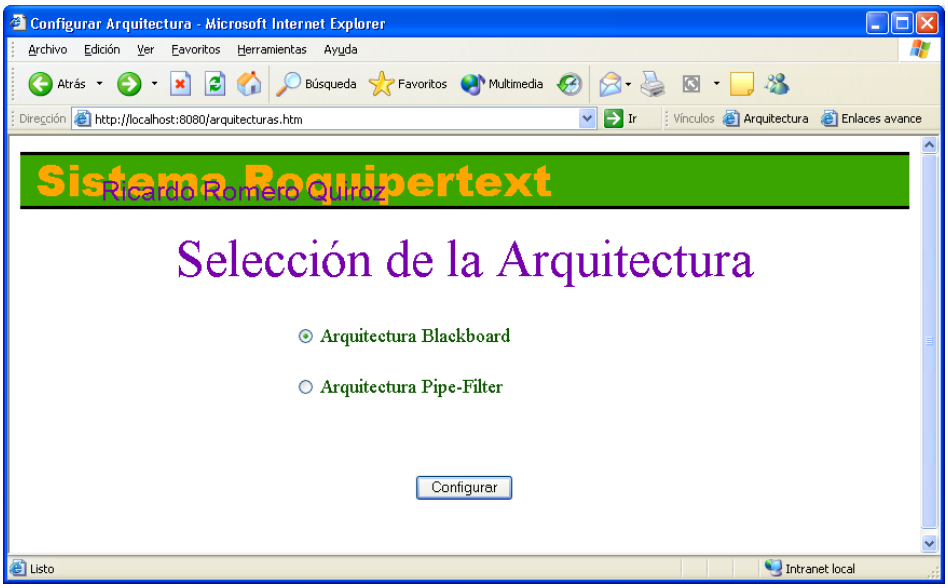

Figura 4.9: *Formulario para el cambio de arquitectura en el sistema*

la arquitectura de pizarra entonces no es necesario seleccionar el orden de los filtros pues como se recordará en la arquitectura de pizarra el orden de los filtros es irrelevante. Por otra parte, si el lector decide trabajar con la arquitectura de conducto-filtro, entonces es necesario configurar el orden de los filtros.

Ahora nos enfocaremos principalmente en el problema del método de búsqueda elegido por el lector. Cuando se selecciona la opción de búsqueda con la arquitectura conducto-filtro, entonces el sistema demanda del usuario el orden en el que deberán quedar los filtros. Este orden es el que determina cuando se puede o no recuperar un documento que haya sido movido de lugar y que en su caso se haya realizado la notificación. El éxito de la búsqueda de la arquitectura conducto-filtro depende estrictamente del orden de los filtros en el sistema. Veamos un ejemplo para ilustrar esto:

- Supongamos que el autor de alguna página personal de la sección ha cambiado de locación su documento y el sistema ha registrado ese movimiento. Todos los movimientos o cambios sufridos por un documento son registrados en una tabla dentro de la base de datos, que almacena el origen y el destino actual del documento, el filtro encargado de verificar esta tabla es el denominado “Avance”.

- El lector ordena los filtros de la siguiente manera: “Avance”, “Genéricos”, y “Específicos”.

Si el lector hace referencia a un documento de la página personal que sufrió un cambio de localidad (figura 4.10), obtendrá como resultado la dirección destino no actualizada del documento,

Curso	Lunes	Martes	Miércoles	Jueves	Viernes	Profesor
Bases de Datos	8:00-10:00		8:00-10:00			Dr. Sergio Chapa
Analisis y Diseño de Algoritmos	10:00-12:00		10:00-12:00			Dr. Guillermo Morales Luna
Arquitectura de Computadoras	12:00-14:00		12:00-14:00			Dr. Adriano De Luca
Redes y Protocolos	16:00-18:00		16:00-18:00			Dr. Jorge Buenabad Chávez
Seminario de Tesis II	18:00-20:00				10:00-12:00	Dr. Luis Gerardo de la Fraga
Redes de Petri			12:00-14:00		12:00-14:00	Dra. Xiaoou Li
Programación Paralela		8:00-10:00			8:00-10:00	Dr. Arturo Díaz Pérez

Figura 4.10: *Documento de origen para ilustrar la búsqueda de enlaces.*

como se ve en la figura 4.11. Aún suponiendo que el filtro de los enlaces de avance se haya colocado en segundo lugar, si el primer filtro no corresponde con el tipo de enlace elegido por el lector, el resultado será el mismo.

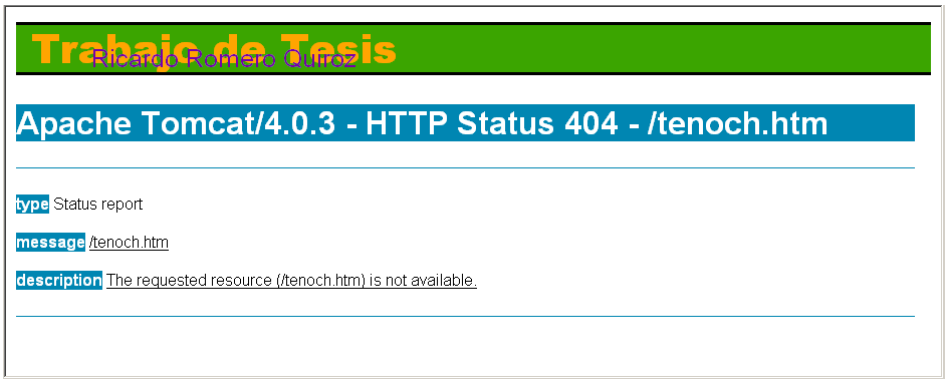

Figura 4.11: *Resultado de la búsqueda utilizando un ordenamiento de filtros no adecuado.*

Siguiendo con este mismo ejemplo, pero ahora suponiendo que el lector ordenó los filtros de la siguiente manera: “Específicos”, “Genéricos” y “Avance”. El lector nuevamente hace referencia a la página personal que sufrió el cambio. En esta ocasión, el lector que hace la búsqueda en el documento 4.10 obtendrá el documento deseado como lo muestra la figura 4.12.

Cuando el lector elige el método de búsqueda con la arquitectura de pizarra, el orden de los filtros es irrelevante, de tal manera que cuando un autor modifica o cambia de locación un pasaje, y este movimiento es notificado al sistema, el documento siempre será localizado. La figura 4.12 muestra el resultado de la misma búsqueda realizada por el lector del ejemplo, sin necesidad de que el lector deba tener conocimiento sobre los filtros o agentes como en el caso de seleccionar la

arquitectura de pizarra.

Figura 4.12: *Resultado de una búsqueda con éxito*

4.5.1. Agregar enlaces genéricos

El sistema nos permite agregar y crear nuevos enlaces genéricos a la base de datos. Los pasos a seguir son:

- Seleccionar el fragmento texto que será el origen del enlace.
- Copiar el texto al clipboard (es decir hacer un Copy del texto seleccionado). Al realizar esta acción, el sistema pregunta si se desea buscar el enlace con el texto seleccionado, se debe elegir la opción cancelar.

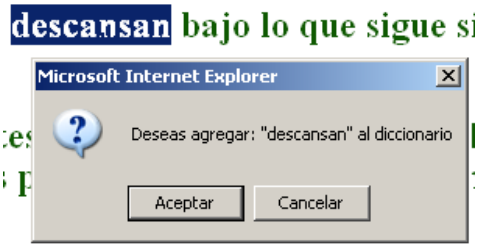

Figura 4.13: *Menu del sistema para agregar un enlace genérico al sistema*

- Al momento en que elegimos cancelar de la opción anterior, inmediatamente se abre otro cuadro de diálogo de la figura 4.13, para confirmar la opción de agregar enlace, en caso de que no se quiera agregar se elige la opción cancelar.
- Una vez que hemos seleccionado la opción "Aceptar" del cuadro de diálogo (figura 4.13), aparecerá un nuevo cuadro de diálogo, que solicita la ruta del documento destino que contiene la referencia del enlace genérico que se va a crear, ver la figura 4.14.

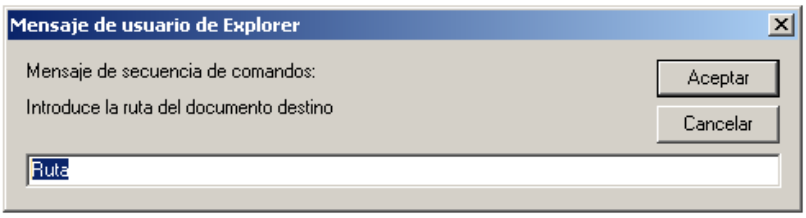

Figura 4.14: *Cuadro de diálogo para introducir la ruta destino del enlace genérico generado*

- Ya escrita la ruta del documento destino, elegimos la opción aceptar y el sistema almacena ese nuevo enlace a la base de datos de enlaces genéricos, si por el contrario en última instancia decidimos no agregar el enlace a la base de datos entonces hay que seleccionar la opción cancelar.

Crear enlaces específicos Aunque en el caso de estudio como lo es la página de la sección de computación no se incluyeron estos enlaces por razones obvias, el usuario puede agregar a sus documentos este tipo de enlaces siguiendo los siguientes pasos:

- Agregar el script especific.js que obtiene el enlace específico del documento actual, y que se encuentra en el directorio scripts los más recomendable es colocar la siguiente línea de código HTML a sus documentos:

```
<script Languaje="JavaScript"
SRC="http://url:8080/scripts/especific.js">
```

- El usuario debe enumerar cada enlace específico de cada documento, es decir cada documento tendrá su propio número de enlaces, y se debe hacer de la siguiente manera:

```
<a href="javaScript:especific(1)">Tenochtitlan</a>
```

Como vemos en este ejemplo el enlace denominado Tenochtitlan es el enlace específico número 1 del documento, y cuando se selecciona este enlace ejecuta la función especific(n) del script.

- Por último hay que dar de alta cada enlace de cada documento a través del formulario que se muestra cuando al iniciar el sistema se tiene el password y la cuenta se es el administrador del sistema (figura 4.15).

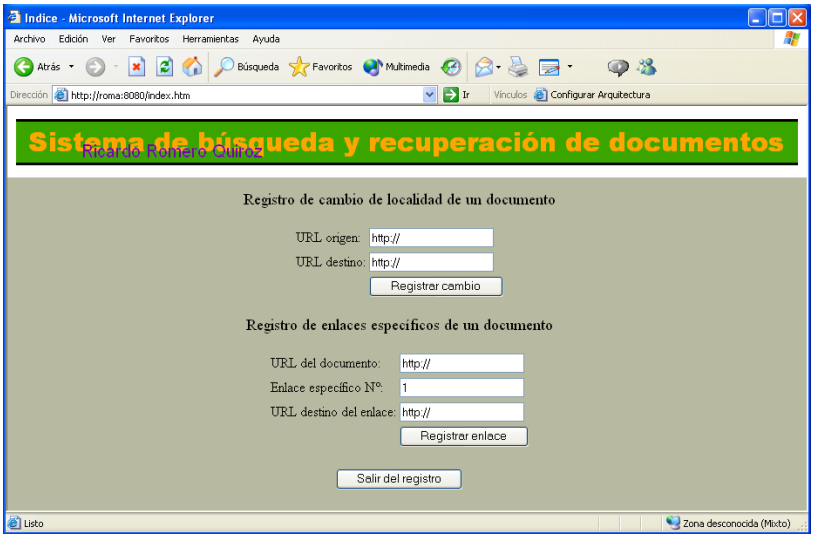

Figura 4.15: *formulario para agregar nuevos enlaces al sistema*

4.6. Análisis de las arquitecturas

La resolución eficaz de enlaces en la arquitectura conducto-filtro depende principalmente de la organización de los filtros en el sistema. Para que el lector obtenga el resultado deseado, es decir el destino actual del documento modificado por el autor, es necesario que el filtro de enlaces de avance esté siempre después del filtro del tipo de enlace que el lector elige. Una vez que el lector ha verificado esta condición, puede estar seguro de obtener siempre el resultado correcto con un tiempo de respuesta más aceptable que utilizando la arquitectura de pizarra. Por la sencilla razón de que en esta arquitectura un filtro solo lleva a cabo su trabajo una sola vez con cada petición hecha al sistema.

Ahora cuando el lector elige la arquitectura de pizarra, por un lado no tiene que preocuparse por la configuración de los agentes de resolución de enlaces en el sistema, y por otro lado, el lector siempre tiene la certeza de que el sistema siempre arrojará el resultado correcto. Esto por supuesto tiene su precio, el número de veces que un agente trabaja en esta arquitectura puede ser mayor a uno. Los agentes en esta arquitectura trabajan mientras que al menos uno de ellos haya obtenido una respuesta a la solicitud de resolución de enlaces. Cuando ninguno de los agentes modifica la petición de resolución es entonces cuando los agentes ya no pueden producir ninguna respuesta

diferente, por lo que la respuesta final es enviada al lector.

4.6.1. Cambio de Arquitectura

Para llevara cabo esa tarea se abre un documento denominado "Configurar Arquitectura" y que es el documento HTML *arquitecturas.htm*, este se encuentra en el directorio ROOT del sistema, y nos presenta un pequeño formulario para seleccionar la arquitectura deseada. Ver la figura 5.4.

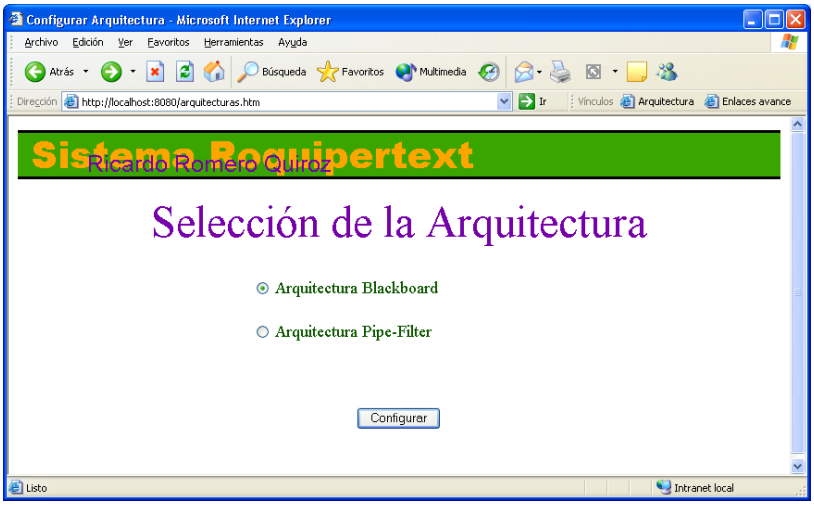

Figura 4.16: *Formulario para la configuración de arquitectura.*

En primer lugar hay que seleccionar la arquitectura con la que deseamos probar al sistema, tenemos en la ventana las dos opciones: la arquitectura de pizarra y la arquitectura conducto-filtro.

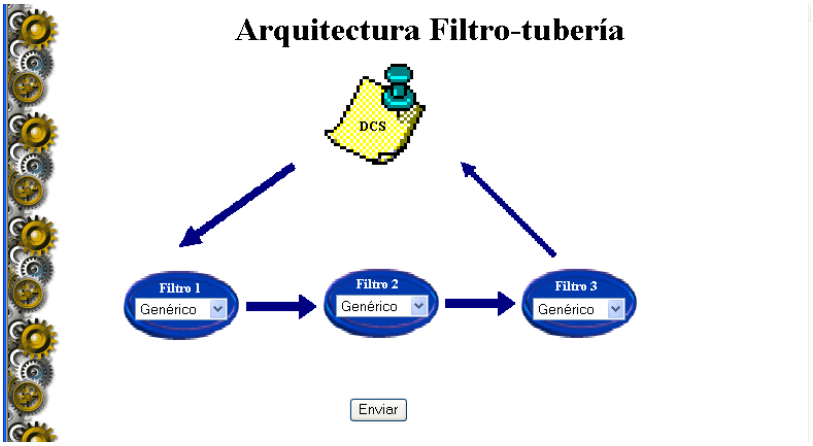

Figura 4.17: *Formulario para la configuración de la arquitectura conducto-filtro.*

Cuando se elige la arquitectura pizarra el orden de los filtros es irrelevante debido a la naturaleza misma de la arquitectura. En el caso de elegir la arquitectura conducto-filtro, debemos también

seleccionar el orden de los filtros para llevar a cabo la búsqueda de los enlaces. Los filtros están numerados del 1 al 3 lo que representa también su posición en la arquitectura.

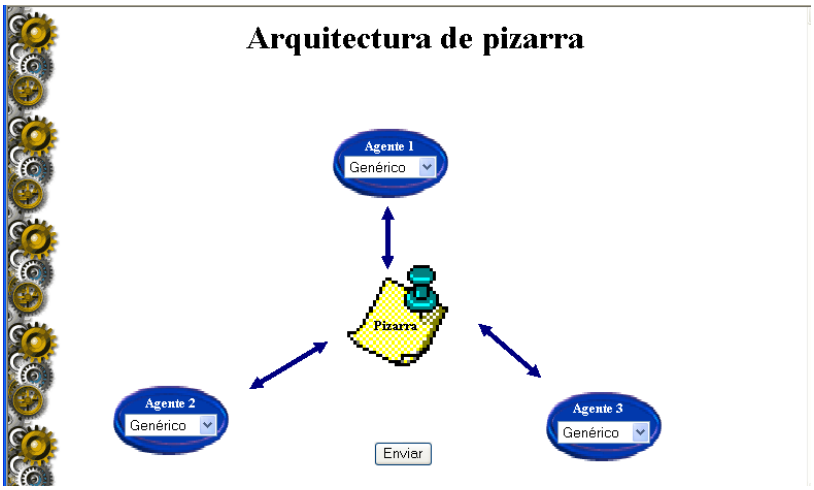

Figura 4.18: *Formulario para la configuración de la arquitectura de pizarra.*

Una vez seleccionada la arquitectura y el orden de los filtros, sólo resta pulsar el botón enviar del formulario y la nueva configuración será almacenada en la base de datos del sistema. El sistema estará trabajando con esta configuración hasta que se realice un nuevo cambio en la configuración de la arquitectura.

4.7. Justificación del caso de estudio

Este sistema ha sido puesto a prueba con la colección de documentos que se integran en los servidores Web del Departamento de Ingeniería Eléctrica y, específicamente, en la Sección de Computación. Esto con el único fin de mostrar la utilidad práctica del sistema. Entre las facilidades que presenta el sistema a este tipo de aplicaciones podemos enumerar las siguientes:

- *Navegación personalizada.* El lector puede navegar entre los documentos de la colección como es usual, es decir, siguiendo los enlaces que aparecen en los documentos, o si lo prefiere, puede generar nuevos enlaces con solo seleccionar un fragmento de texto del documento actual.

- *Creación de Enlaces sin Edición de Documentos.* El sistema permite a cualquier usuario crear sus propios enlaces y almacenarlos en el sistema para utilizarlos después en el momento que lo requiera, con solo seleccionar un fragmento de texto de cualquier documento.

- *Cambio de Localidad de Documentos.* Esta es una de las mayores facilidades o ventajas que presenta el sistema, y para entenderlo vamos a ilustrarla con el siguiente ejemplo: tal como

está el sistema en este momento ¿qué pasa cuando un autor decide cambiar de nombre su documento o, simplemente, lo cambia de lugar? Bueno la tarea de edición de todos los demás documentos que hacen referencia no solo será bastante ardua sino en ocasiones imposible, ya que requiere acceder a los documentos para modificarlos para lo cual se puede necesitar permiso de escritura. Con el sistema propuesto no es necesario siquiera modificar los documentos que hacen referencia a los documentos modificados, ya que al registrar todos los cambios de localidad de estos, el sistema recupera automáticamente la dirección correcta del documento solicitado.

4.8. Resumen

En el caso de estudio presentado en este capítulo, se mostró una de las maneras de como se pueden aplicar el sistema en forma práctica. El sistema permite enriquecer la colección de documentos de la Sección de Computación, haciendo énfasis sobre todo en las diferencias que existen entre las dos arquitecturas en cuanto a su capacidad para resolver enlaces que involucran documentos que han cambiado su localidad.

Capítulo 5

Diseño del Sistema

De acuerdo a los objetivos de diseño del sistema como son la de extender la capacidad del navegador para soportar tipos adicionales de enlaces hipermedia; el navegador fue extendido incorporando mecanismos de resolución de enlaces basados en las arquitecturas de conducto-filtro y de pizarra. En este capítulo revisaremos y describiremos la arquitectura general del sistema, desde la capa de aplicación hasta la de almacenamiento.

5.1. Capa de aplicación

El sistema tiene un diseño por capas que son: La capa de aplicación, la capa de servicio de enlaces y la capa de almacenamiento, como se muestra en la figura 5.1. La capa de aplicación consiste de: el navegador (visualizador) del tipo Mozaic, el cual nos permite el manejo de enlaces y ejecutar un conjunto de scripts, con estas características se implementan los enlaces adicionales. El conjunto de scripts nos permiten la implementación de los enlaces genéricos y específicos, así como también agregar nuevos enlaces genéricos al sistema.

El script que implementa a los enlaces genéricos se mantiene atento a cualquier fragmento de texto que se copie al clipboard del sistema, si el usuario lo decide el script envía este fragmento de texto al servidor de enlaces mediante una petición HTTP, otra opción que nos da el sistema cuando el clipboard tiene un fragmento de texto es la de crear un enlace genérico cuya tarea también la realiza este script. El enlace específico se implementa con un enlace de HTML que en vez de enlazar un URL hace una llamada a un script que se comunica con el servidor de enlaces vía HTTP para solicitar la resolución del enlace elegido por el usuario. Los enlaces de avance no son accesibles al lector, estos enlaces solo los puede generar el autor de algún documento registrado en el sistema.

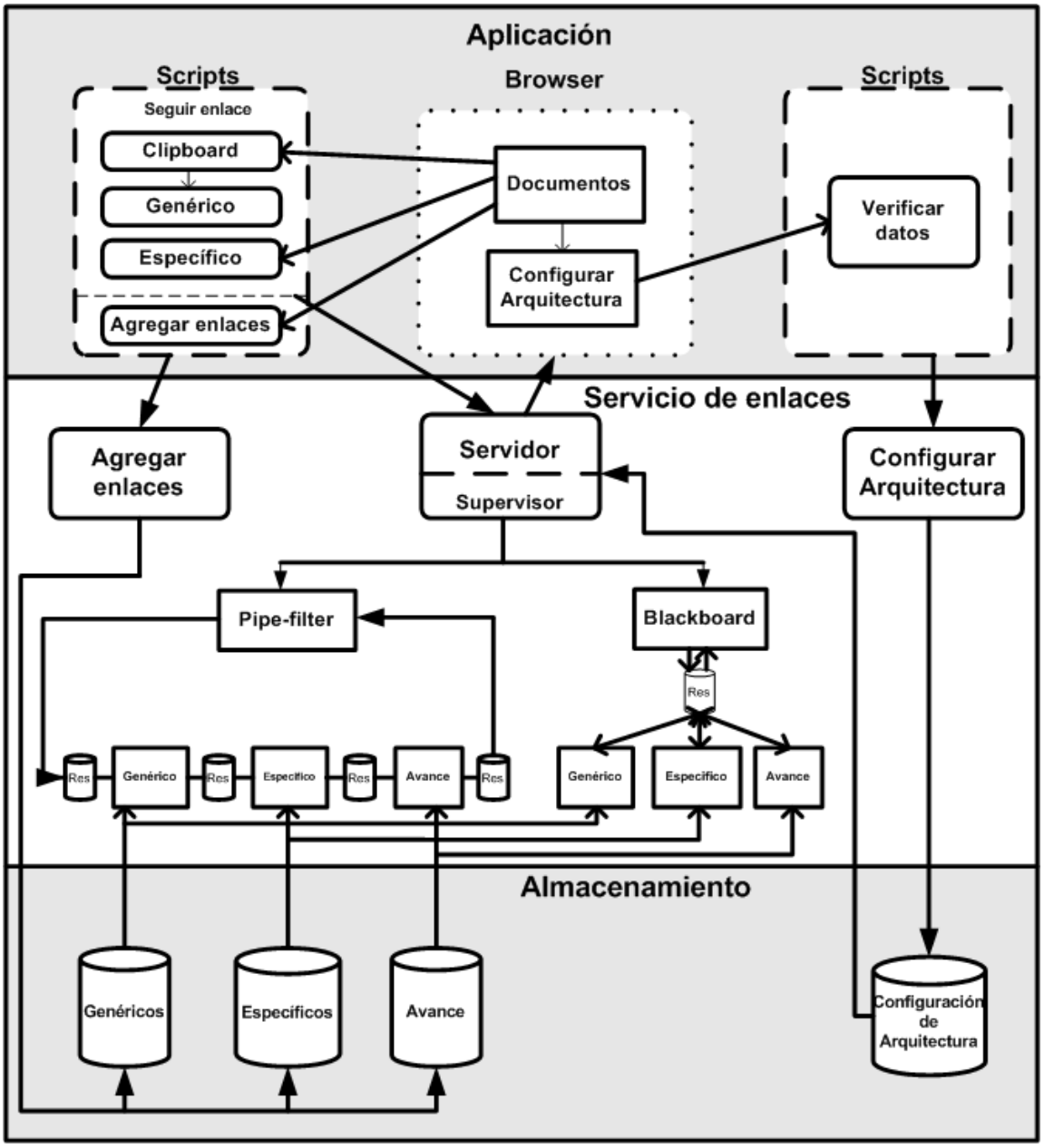

Figura 5.1: *Diseño arquitectónico del sistema*

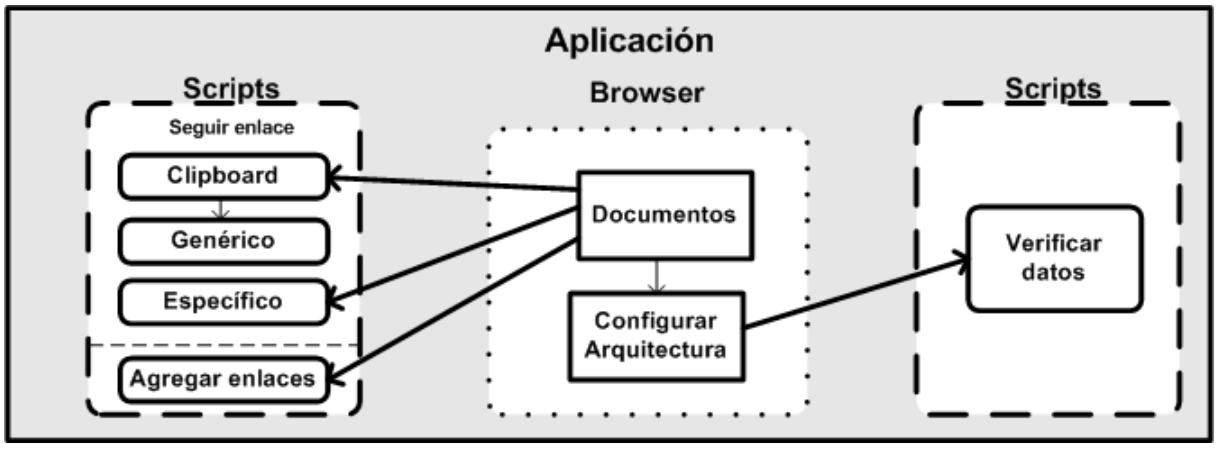

Figura 5.2: *La capa de aplicación del sistema*

El usuario del sistema puede también seleccionar una arquitectura para la resolución de enlaces; esto es posible gracias a un documento que puede ser agregado como mis preferidos, quien así mismo abre un formulario para realizar la tarea de configuración de la arquitectura deseada.

La figura 5.2 indica que los documentos de prueba contienen código script que dan al sistema la funcionalidad para trabajar con los enlaces genérico y específico, por otro lado también se permite registrar un enlace genérico al sistema. El usuario del sistema puede también cambiar de una arquitectura a otra; esto es posible gracias a un enlace embebido en los documentos que abren un formulario para realizar dicha tarea. La página que contiene el formulario también contiene código script cuyo propósito es validar los datos del mismo.

Configurar arquitectura Este componente que se encuentra en la capa de aplicación es un documento HTML que contiene un formulario que nos va a permitir llevar a cabo la configuración de arquitectura desde el browser.

5.2. La capa de servicio de enlaces

Los componentes construidos en esta capa del sistema son los siguientes:

- **Cambio de arquitectura**: Permite al usuario cambiar la configuración de la arquitectura del sistema, con el propósito de que se prueben ambas arquitecturas y se comparen sus resultados.

- **Agregar enlaces**: Útil para agregar nuevos enlaces genéricos al sistema.

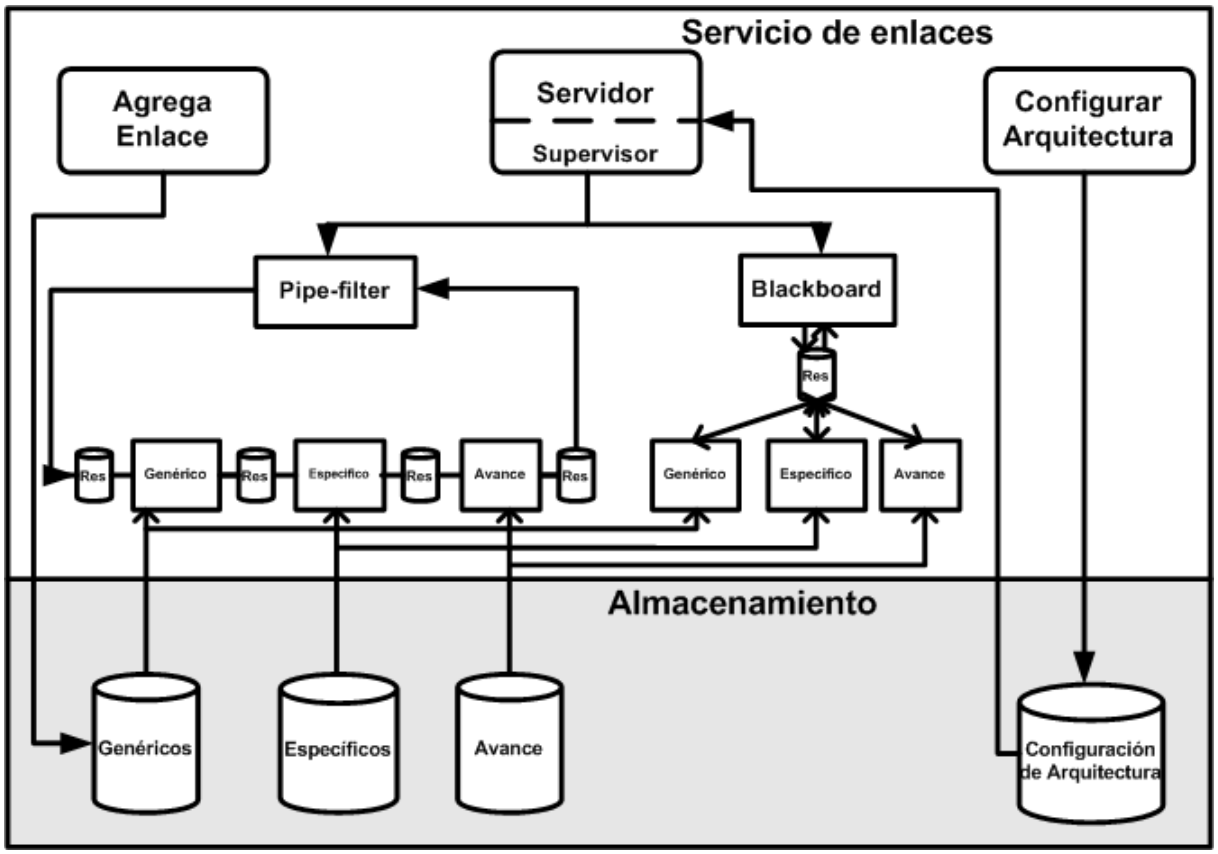

Figura 5.3: *El servicio de enlaces y de almacenamiento*

- **Servidor**: Es el encargado de recibir las peticiones de los usuarios, así como de la comunicación con las arquitecturas y devolver los resultados al browser.
- **Supervisor**: Auxiliar del Servidor para almacenar las peticiones a la base de datos así como de limpiar la misma al final de cada petición.
- **conducto-filtro**: Implementa la forma de trabajo de la arquitectura conducto-filtro del sistema Microcosmos.
- **Pizarra**: Implementa la filosofía de trabajo de las arquitecturas pizarra.
- **Genérico**: Resuelve los enlaces genéricos solicitados.
- **Específico**: Busca en la base de datos de enlaces específicos para dar solución a las solicitudes de los usuarios.
- **Avance**: Al igual que los dos anteriores solo que este realiza su trabajo con la base de datos de enlaces de avance.

La figura 5.3 muestra la arquitectura del servicio de enlaces con todos sus componentes, y su interrelación con la capa de almacenamiento.

5.2.1. Servicio de cambio de arquitectura

Este servicio es ejecutado por un componente quien recibe el conjunto de datos que corresponde a la configuración de la arquitectura con la cual el usuario desea trabajar, para llevar a cabo la búsqueda de los enlaces. La figura 5.4 muestra el proceso del servicio.

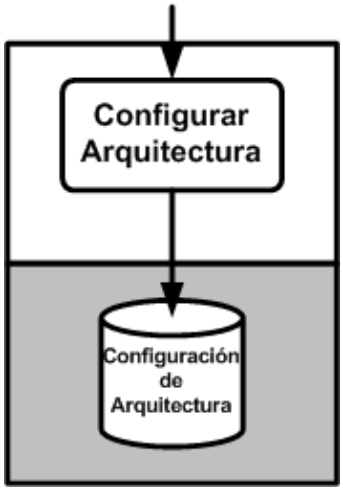

Figura 5.4: *Servicio de cambio de arquitectura*

Es a través de este servicio entonces que podemos cambiar la configuración de la arquitectura cuantas veces sea necesario. Esto nos permitirá también experimentar no solo con las dos arquitecturas, sino también con el orden de los filtros en el caso de la arquitectura conducto-filtro, ya que en la arquitectura de pizarra el orden de los filtros es irrelevante.

5.2.2. Servicio para agregar enlaces

El nombre del servicio es un tanto engañoso, ya que solo es posible agregar enlaces de tipo genérico, nunca de tipo específico y mucho menos de avance. Este servicio lo lleva a cabo un componente cuyo funcionamiento es muy sencillo, pues toma los datos que le son enviados por el usuario y los almacena en la base de datos de los enlaces genéricos, ver la figura 5.5.

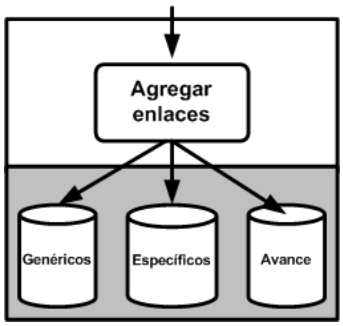

Figura 5.5: *Servicio para agregar enlaces genéricos*

5.2.3. Servidor

Este componente que es el encargado de recibir las peticiones de búsqueda de enlaces de los usuarios, así como también generar dinámicamente el documento de salida al usuario con los resultados de la búsqueda con la arquitectura configurada previamente (ver figura 5.6). Podemos entonces decir que este componente es el DCS (Sistema de control de documento) del sistema Microcosmos, pues es evidente que realizan ambos las mismas funciones. Desde recibir las peticiones de los usuarios, hasta devolver los resultados del trabajo de la arquitectura en turno.

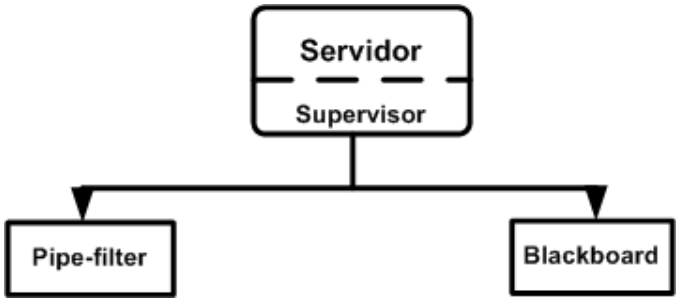

Figura 5.6: *Trabajo del Servidor de manera gráfica*

Podemos darnos cuenta que este componente es quien coordina prácticamente todo el trabajo de resolución de enlaces del sistema, aunque por supuesto el trabajo importante lo llevan a cabo las arquitecturas de resolución de enlaces, cuya implementación describimos más adelante.

Por cuestiones de modularidad y programación se creó una el componente "Supervisor", sus únicos objetivos son: almacenar las peticiones en la base de datos y posteriormente eliminar tanto la petición como los resultados obtenidos por la arquitectura en turno.

5.2.4. Arquitectura conducto-filtro

A diferencia de la arquitectura de pizarra, la arquitectura conducto-filtro adopta un modelo centrado en procesos relativamente autónomos llamados filtros que se comunican con los demás a través del paso de mensajes. De esta manera la arquitectura se puede ver como una cadena de filtros.

En nuestro caso los filtros no se comunican a través del paso de mensajes, sino a través de la base de datos, en la cual cada filtro toma como entrada los datos almacenados en la misma y reescribe sus resultados en la misma tabla, en caso de que existan (Figura 5.7).

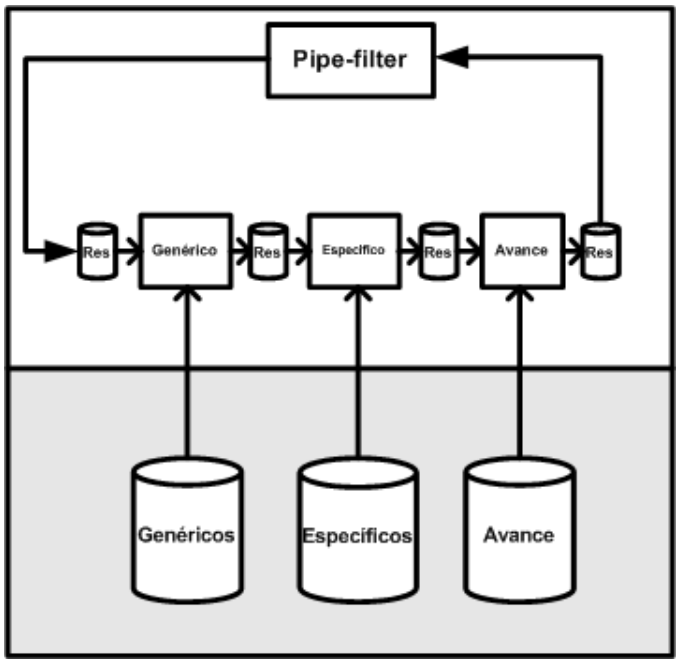

Figura 5.7: *Arquitectura conducto-filtro implementada*

5.2.5. Arquitectura de pizarra

Una pizarra es una estructura de datos que es leída y modificada por programas llamados fuentes de conocimientos o agentes. Un agente puede codificarse en términos de reglas de producción con partes de condición y acción que respectivamente reconocen y modifican patrones relevantes de datos, actuando de manera oportuna sobre la pizarra. La pizarra es un repositorio central de datos que pueden utilizarse para habilitar la comunicación entre agentes. La arquitectura de pizarra logra un alto grado de flexibilidad donde las dependencias de datos entre los agentes de enlaces no

dependen de la restrictiva secuencialidad impuesta por la arquitectura conducto-filtro.

En el sistema el repositorio central de datos o pizarra es una tabla de la base de datos, con la cual trabajan desde el supervisor, los agentes así como el servidor de peticiones para devolver los resultados del trabajo de la arquitectura, como se ve en la figura 5.8.

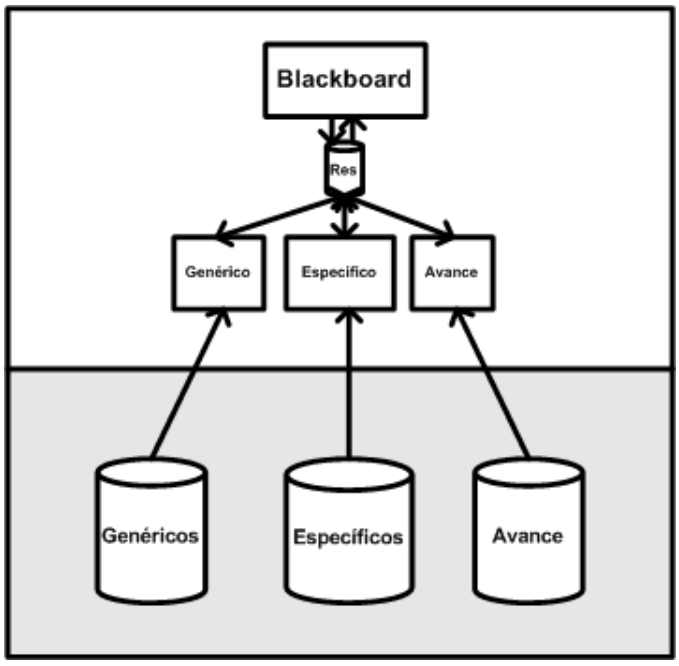

Figura 5.8: *Arquitectura de pizarra implementada*

Como en esta arquitectura la secuencia de los agentes no está predefinida, se crea una un orden de trabajo aleatorio para los mismos, y esto se hace a cada ronda de trabajo de los agentes. Cada uno de los agentes toma uno a uno los datos de la tabla resultados, si resuelve la petición, entonces esta es eliminada de la tabla y se escribe en ella el resultado obtenido por el agente. Una vez que ninguno tiene solución a las peticiones de la tabla en ese momento la arquitectura ha terminado su labor, y el trabajo que sigue le corresponde al servidor de enlaces.

5.2.6. Filtros: Agentes de resolución de enlaces

Dependiendo a que arquitectura hacemos referencia los componentes creados para la resolución de enlaces, los denominamos agentes para la arquitectura de pizarra y filtros para la arquitectura conducto-filtro.

Específico

Este componente tiene como objetivo consultar a la base de datos que contiene los enlaces específicos registrados por el sistema para buscar solución a las peticiones de enlaces colocadas en la base de datos resultados.

Genérico

El enlace genérico tiene la característica de ser un enlace generado "al vuelo", con esto queremos decir que en los documentos no hay ninguna marca que lo indique, por tal motivo estos enlaces pueden generarse con cualquier fragmento de texto que contenga el documento. La resolución de estos enlaces se lleva a cabo con el componente Generico.

Avance

El enlace de avance, nos permite resolver algunos problemas en los sistemas abiertos de hipertexto, nuestros objetivos principales son: mantener la propiedad de integridad en la base de datos y seguir el desplazamiento de aquellos documentos que han sido movidos de una ruta a otra. Este componente toma como datos de entrada los resultados almacenados en la tabla para el mismo fin y busca si estos han sido movidos o modificados (estos cambios están registrados en la base de datos de enlaces de avance), de ser así la solución la sobre escribimos en la misma tabla, y las rutas de los cambios se almacenan en otra tabla para tener registrado el desplazamiento de cada documento.

5.2.7. La capa de almacenamiento

Es en esta última capa donde se almacenan físicamente todos los datos del sistema como son los enlaces y los resultados que se obtienen al hacer trabajar al sistema y las tablas que se utilizan se describen a grandes rasgos a continuación:

- **Arquitectura:** la configuración de la arquitectura.
- **Avance:** Los enlaces de avance registrados en el sistema.
- **Específicos:** Los enlaces específicos.
- **Genéricos:** En esta tabla se almacenan todos los enlaces genéricos que han sido registrados en el sistema y aún los que ingrese el usuario en el futuro.

- **Pizarra:** Almacena de manera momentánea las solicitudes de los usuarios, y los resultados que se obtienen con el trabajo de las arquitecturas.
- **Rutas:** Sirve como almacenamiento temporal para el agente o filtro (según sea la arquitectura) para almacenar los desplazamientos de los documentos que contengan enlaces de avance.

5.3. Análisis del diseño arquitectónico

En esta sección se hace una referencia a las principales ventajas y desventajas del diseño del sistema con respecto a la Web. **Ventajas:**

- *Es un sistema abierto:* Aunque la Web es un sistema abierto también, este diseño tiene la característica de tener ciertas ventajas sobre la misma Web, debido a que nos va a permitir mayor:
  - *Integración:* Si así se desea es posible agregar nuevos agentes o filtros para la resolución de enlaces sin perder la coordinación de todas las actividades.
  - *Interoperabilidad:* El sistema tiene la capacidad de intercambiar datos con otros sistemas (debido principalmente a la naturaleza de los documentos).
  - *Extensibilidad:* Es posible agregar nuevos enlaces al sistema en tiempo de ejecución.
- *Enlaces multidimensionales:* El sistema está diseñado para que los enlaces no sean precisamente unidireccionales como los on los enlaces de la HTML de la Web sino que trabajen de manera multidimensional, un enlace puede hacer referencia a uno o varios documentos.

Sin embargo también se tienen ciertas desventajas como son:

- Perdida de eficiencia: debido principalmente a que el sistema debe estar en constante comunicación con el servidor para la resolución de los enlaces, mientras que en la Web no se lleva a cabo esta acción.

5.4. Resumen

Hemos descrito en forma concisa y detallada la arquitectura del sistema desarrollado, la capa de aplicación, la de servicio de enlaces y la de almacenamiento, así como cada uno de sus componentes. También de discutió sobre las ventajas y desventajas de este diseño con respecto a la Web.

Capítulo 6

Implementación del Sistema

La implementación del sistema se describe en el presenta capítulo. Se hace una descripción de la implementación componente a componente según corresponda a la arquitectura diseñada y explicada en el capítulo anterior.

6.1. JDBC

Antes de iniciar explicamos la forma de utilizar JDBC y de como se utiliza en la implementación del sistema. *JDBC* (*Java DataBase Connectivity*) es una parte del API de *Java* que proporciona clases para conectarse con bases de datos. Dichas clases forman parte del paquete *java.sql*, disponible en el *jdk 1.2.* La conexión con la base de datos se hace por medio de un driver que convierte de *JDBC* a *ODBC*.

Para poder manipular una base de datos en java importamos el paquete *java.sql* que contiene todas las clases necesarias para realizar todas las transacciones que se necesitan, así, con la declaración (abajo) en la cabecera de nuestras clases, estamos importando dichas clases.

```
import java.sql.*;
```

El paso siguiente es cargar el driver que permita a las clases *JDBC* comunicarse con el origen de los datos, que en este caso es *ODBC*:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Con este código estamos creando una nueva instancia del driver. El driver provee los métodos

necesarios para realizar una *Conexión*, pero se requiere de un tipo específico de URL, que utiliza el protocolo jdbc. La forma general del protocolo es: *jdbc:<subprotocolo>:<nombre_origen_datos>*. De tal manera que nuestro url es:

```
static String url = "jdbc:odbc:Linkbases";
```

Con la clase DriverManager, solicitamos una Connection usando el URL y el DriverManager selecciona el driver apropiado, el formato para obtener una conexión es:
DriverManager.getConnection(*URL,Username,Password*);.

```
static Connection conexion;

conexion = DriverManager.getConnection(url,"","");
```

Por lo general el Username y el Password son cadenas vacías (“ ”), a menos que el origen de datos los utilice.
Mientras la clase *Connection* solo realize la conexión apropiada al origen de datos, Java utiliza como Lenguaje Manipulador de Datos (DML) las instrucciones de SQL, que requieren de un objeto denominado *Statement*. Así el siguiente paso es preguntar a la conexión por un objeto *Statement*:

```
static Statement stmt;
stmt = conexion.createStatement();
```

El objeto o interface *Statement* proporciona dos métodos distintos de ejecución de sentencias SQL en función del tipo de sentencia que se vaya a ejecutar:

- executeQuery(String): Sirve para recuperar información contenida en una base de datos. Ejecuta la consulta (query) pasada como parámetro y devuelve un objeto *Resultset* como resultado de la consulta.
- executeUpdate(String): Análogo al anterior, con la diferencia de que este método no sirve para realizar una consulta, sino para *modificar o introducir datos* en la base de datos. Tiene como valor de retorno un *int*, que indica el número de filas actualizadas.

Una vez aclarada la forma de conexión a la base de datos en java, continuamos con la explicación de la implementación del sistema a detalle.

6.2. Capa de aplicación

La capa de aplicación está implementada con: el browser Internet Explorer, HTML para los documentos de prueba y JavaScript para extender la funcionalidad del browser..

6.2.1. Enlaces específicos

De los enlaces específicos hay que recordar que estos se encuentran fijos en alguna parte del documento, y esto da pie a que la implementación sea mucho más sencilla que la de los enlaces genéricos. En esta tesis se trabaja con los enlaces específicos como si fueran enlaces de HTML, aunque en vez de traer el documento directamente, se toma la solicitud de enlace y se envía a través del Servlet a cualquiera de las arquitecturas definidas para su procesamiento.

El enlace específico como cualquier enlace en HTML, se encuentra embebido en el mismo documento:

```
<a href="javaScript:especific(n_enlace)">Cuauhtémoc</a>
```

Cuando el usuario selecciona un enlace del documento, se invoca a una función en JavaScript llamada especific(n_enlace), que recibe como parámetro un entero que le indica a la función que enlace específico es el elegido por el usuario, esto por razones obvias a que un documento puede contener más de un enlace específico, y se necesita determinar a cual de ellos se refiere.

Una vez que se invoca al script, este realiza la llamada al Servlet enviando como datos el documento origen y el número de enlace seleccionado, que son los datos que se necesitan para llevar a cabo la búsqueda de documentos dentro del sistema.

6.2.2. Enlaces genéricos

La implementación de este enlace se llevó a cabo utilizando un script desarrollado en el lenguaje de JavaScript, recordemos que JavaScript es una de las múltiples aplicaciones que han surgido para extender las capacidades del Lenguaje *HTML*. JavaScript es un lenguaje *script* orientado a documento. Nunca podrá hacer un programa, tan sólo podrá mejorar sus páginas *Web*. De esta manera el script de los documentos debe implementar tres operaciones:

1. El usuario del sistema puede seleccionar un fragmento de texto.
2. Del fragmento de texto seleccionado el sistema debe crear un enlace genérico y,
3. Las operaciones anteriores deben estar disponibles al usuario en todo momento.

Para que el usuario pueda seleccionar un fragmento de texto de cualquier documento, se logra utilizando el objeto clipboardData cuya función es proveer acceso a formatos predefinidos de datos en el clipboard en operaciones de edición, a través del menú Edit. Transfiere la información al clipboard del sistema, y retiene los datos hasta que una operación siguiente de edición los suplanta. Este objeto está disponible es JavaScript del Internet Explorer 5.0 en adelante.

Los métodos con los que cuenta este objeto son:

- clearData: remueve uno o más datos del clipboard.
- getData: obtiene el dato en el formato especificado del clipboard.
- setData: transfiere el dato en el formato especificado al clipboard.

Con las herramientas mencionadas arriba nos es posible entonces a través de la opción copy del menú Edit del navegador obtener el fragmento de texto seleccionado por el usuario, para su posterior procesamiento. Recordando indicar al método getData el tipo de datos que se desea recuperar del clipboard.

```
datos = myClipboardData.getData("Text");
```

La segunda operación es solo una línea de código del script:

```
if(buscar)
  document.location="http://xx.xx/Servidor?origen="+datos+"&enlace=0&tipo=0";
```

Una vez que el usuario decide buscar enlaces tomando como referencia el fragmento de texto seleccionado, en ese momento se envía la solicitud al Servlet con el fragmento de texto como datos que recibe el Servlet para su procesamiento.

Por último, la operación que nos permitirá mantener a las anteriores dos funciones ejecutándose cada vez que el usuario seleccione un fragmento de texto de los documentos, se implementó con el

método setTimeout(), en el que podemos especificar que función ejecutar periódicamente según el lapso de tiempo especificado en milisegundos en el propio método, de tal manera que la línea de código:

```
setTimeout("verifica()",100);
```

Ejecuta la función verifica() cada 100 milisegundos, con lo que podemos decir que el script está escuchando, y cuando se copia un fragmento de texto al clipboard, se activa la función verifica() que no hace otra cosa que obtener el texto del clipboard, limpiarlo y si el usuario lo decide hacer una búsqueda de documentos que tengan como referencia el fragmento de texto seleccionado por el usuario.

Una vez que se ha seleccionado el fragmento de texto, el usuario a través del menú Edit del navegador o mediante el menú emergente del mismo se elige la opción copiar que enviará el fragmento de texto seleccionado al clipboard.

Inmediatamente después de elegir la opción copiar del menú Edit, aparecerá una caja de texto pidiendo al usuario la confirmación de la búsqueda de enlaces teniendo como base el fragmento de texto seleccionado. Si decide buscar enlaces en el sistema con el texto seleccionado elegirá la opción aceptar de la caja de texto de los contrario se elige la opción cancelar de la misma.

6.2.3. Cambio de arquitectura

El cambio de arquitectura se lleva a cabo cargando el documento con el formulario que nos permitirá realizar esta acción, este documento solo contiene un script diseñado única y especialmente para validar los datos del formulario, como pueden ser: que el usuario seleccione una de las dos arquitecturas, y si seleccionó la arquitectura conducto-filtro, que ningún filtro se repita en la elección de los mismos. Una vez que los datos del formulario son correctos, lo restante es llamar al componente que almacenará la configuración elegida por el usuario en su respectiva base de datos.

6.3. La capa de servicio de enlaces

Nuestra capa de servicio de enlaces está implementada con las tecnologías de *Servlets y Java*, teniendo como servidor web *Apache Tomcat 4.0.* En esta capa se encuentran tanto los servlets como

las diferentes clases que hacen la conexión a la base de datos, ya sea para escribir o leer datos en la misma.

6.3.1. Servicio de cambio de arquitectura

Este servicio es ejecutado por el servlet "*ConfiguraArqServlet.class*". Cuando el usuario decide cambiar de arquitectura a través del browser se invoca a este servlet, y su trabajo es cambiar la base de datos que contiene la configuración de la arquitectura que está activa. el sistema solo trabaja con una arquitectura a la vez por lo tanto si se requiere utilizar una u otra arquitectura es necesario configurarla en caso de no ser así.

ConfiguraArqServlet.class: Permite cambiar la arquitectura del sistema.

Responsabilidades:	**Colaboradores:**
Recibir los datos del formulario del documento HTML. Decodificar los datos recibidos. Establecer la conexión con la base de datos. Abrir la tabla que contiene la configuración de la arquitectura. Eliminar configuración actual de la tabla. Actualizar configuración en la tabla. Cerrar la conexión a la base de datos.	Ninguno

6.3.2. Servicio para agregar enlaces

El servlet "AgregaGenericoServlet.class" es el encargado de agregar nuevos enlaces genéricos al sistema.

AgregaGenericoServlet.class: Agrega enlaces genéricos a la base de datos.

Responsabilidades:	**Colaboradores:**
Recibir los datos del formulario del documento HTML. Decodificar los datos recibidos. Establecer la conexión con la base de datos. Abrir la tabla que contiene los enlaces genéricos. Agregar el enlace recibido a la tabla. Cerrar la conexión a la base de datos.	Ninguno

6.3.3. Servidor

La clase "ServidorServlet.class" es el servlet principal del sistema, pues es quien recibe las peticiones de resolución de enlaces de los usuarios, así como de poner a la arquitectura configurada para llevar a cabo la tarea. Una vez que la arquitectura termina su labor, el servidor ahora debe enviar los resultados al usuario.

ServidorServlet.class: Realiza prácticamente todo el trabajo, desde invocar a la arquitectura configurada hasta enviar los datos al usuario.

Responsabilidades:	**Colaboradores:**
Recibir los datos del formulario del documento HTML.	Supervisor
Decodificar los datos recibidos.	BSPipe
Establecer la conexión con la base de datos.	BSBlackboard
Abrir la tabla que contiene la configuración de la arquitectura.	
Invocar al supervisor para que sea este quien almacene la solicitud en la base de datos.	
Invocar a la arquitectura configurada.	
Devolver los resultados del trabajo de la arquitectura.	
Llamar al supervisor para limpiar la base de datos de resultados para posteriores peticiones de servicio.	
Cerrar la conexión con la base de datos.	

Este servlet tiene como auxiliar de trabajo a la clase "Supervisor.class" y solo se dedica a escribir y eliminar datos de la base de datos de resultados.

Supervisor.class: Agrega solicitudes y elimina resultados de la base de datos.

Responsabilidades:	**Colaboradores:**
Agregar solicitudes a la base de datos.	Supervisor
Eliminar peticiones de la base de datos.	
Eliminar resultados de la base de datos.	

6.3.4. Arquitectura conducto-filtro

La clase que implementa a la arquitectura conducto-filtro es "BSPipe.class", y tiene a su cargo hacer trabajar a los filtros de acuerdo a la configuración de filtros predeterminada en la base de datos.

BSPipe.class: Implementa a la arquitectura conducto-filtro utilizada en el sistema Microcosm.

Responsabilidades:	**Colaboradores:**
Abrir la conexión a la base de datos.	ServidorServlet
Obtener el orden de los filtros según la configuración almacenada en la base de datos.	BSGenerico
Llamar uno a uno a los filtros de enlaces, de acuerdo al orden configurado.	BSEspecifico
Indicar al Servidor si hubo respuesta a su petición.	BSAvance

6.3.5. Arquitectura blackboard

La clase que implementa a esta arquitectura se denomina "BSBlackboard.class", y al igual que la clase anterior debe hacer trabajar a los agentes de software ya no de acuerdo a una configuración de agentes sino de hacerlos trabajar de manera aleatoria una y otra vez hasta que ninguno de ellos reporte algún cambio en la base de datos de resultados.

BSBlackboard.class: Implementa a la arquitectura blackboard.

Responsabilidades:	**Colaboradores:**
Abrir la conexión a la base de datos	ServidorServlet
Generar un orden aleatorio para el trabajo de los agentes cada vez que se requiera.	BSGenerico
Llamar uno a uno a los agentes de resolucion, de acuerdo al orden generado.	BSEspecifico
Indicar al Servidor si hubo respuesta a su petición.	BSAvance

6.3.6. Filtros o agentes de resolución de enlaces

Específico

Esta clase solo consulta a la base de datos de enlaces específicos con el propósito de resolver una búsqueda de enlace de este tipo.

BSEspecificos.class: Implementa el filtro o agente para la resolución de enlaces específicos.

Responsabilidades:	**Colaboradores:**
Abrir la conexión a la base de datos Verificar si existe una petición en la base de datos. Si hay peticiones consultar a la base de enlaces específicos para intentar dar solución a las peticiones. Almacenar los resultados si los hay. Cerrar la conexión a la base de datos.	Ninguno

Genérico

Esta clase al igual que la clase anterior consulta a la base de datos que contiene almacenados los enlaces genéricos para ver si tiene alguna correspondencia con la petición que se le envía.

BSGenericos.class: Implementa el filtro o agente para la resolución de enlaces genéricos.

Responsabilidades:	**Colaboradores:**
Abrir la conexión a la base de datos Verificar si existe una petición en la base de datos. Si hay peticiones consultar a la base de enlaces genéricos para intentar dar solución a las peticiones. Almacenar los resultados si los hay. Cerrar la conexión a la base de datos.	Ninguno

Avance

Tiene la misma función que las dos clase anteriores solo que esta busca correlaciones de solicitudes que le son enviadas en la base de datos que contiene a los enlaces de avance.

BSAvance.class: Implementa el filtro o agente para la resolución de enlaces de avance.

Responsabilidades:	**Colaboradores:**
Abrir la conexión a la base de datos Verificar si existen resultados en la base de datos "resultados". Buscar si existen enlaces de avance para cada uno de los datos en la base de datos de resultados. Sobre escribir los resultados si los hay. Escribir las rutas seguidas por cada enlace de avance en la respectiva base de datos. Cerrar la conexión a la base de datos.	Ninguno

6.3.7. La capa de almacenamiento

Esta capa está implementada con una base de datos creada en el manejador de base de datos de Microsoft Access, utilizando la Conectividad abierta de base de datos de orígenes de datos (ODBC) para tener acceso a los datos desde la API de java (JDBC). Esta interfaz es un estándar en la industria y es un componente de los servicios de Microsoft windows. La interfase ODBC hace posible que las aplicaciones accedan a los datos con una gran variedad de sistemas manejadores de bases de datos (DBMS). De esta manera, la aplicación es independiente del DBMS a través de una interfaz sencilla. Los usuarios pueden agregar componentes de software a sus aplicaciones llamados drivers, los cuales crean una interfase entre la aplicación y el origen de los datos. Otra de las ventajas de trabajar con una base de datos que soporta la conectividad ODBC es sacar el mayor provecho que esta nos da al crear un cache de datos intermediario entre la aplicación y el almacenamiento físico, lo que da pie a reducir el número de accesos al disco.

El sistema está implementado para trabajar de manera centrada, esto es la base de datos se encuentra en un solo servidor y no de manera distribuida. El contar con una base de datos distribuida daría pie a perder un poco más de velocidad de respuesta, ya que si la resolución de un enlace no se encuentra en el servidor en el que se está consultando actualmente, tiene que preguntar a los demás servidores si alguno de ellos la tiene. Esta estrategia de la información distribuida en varios servidores es factible si la base de enlaces es muy grande como para almacenarla en un solo servidor. La figura 6.1 muestra las tablas creadas en la base de datos.

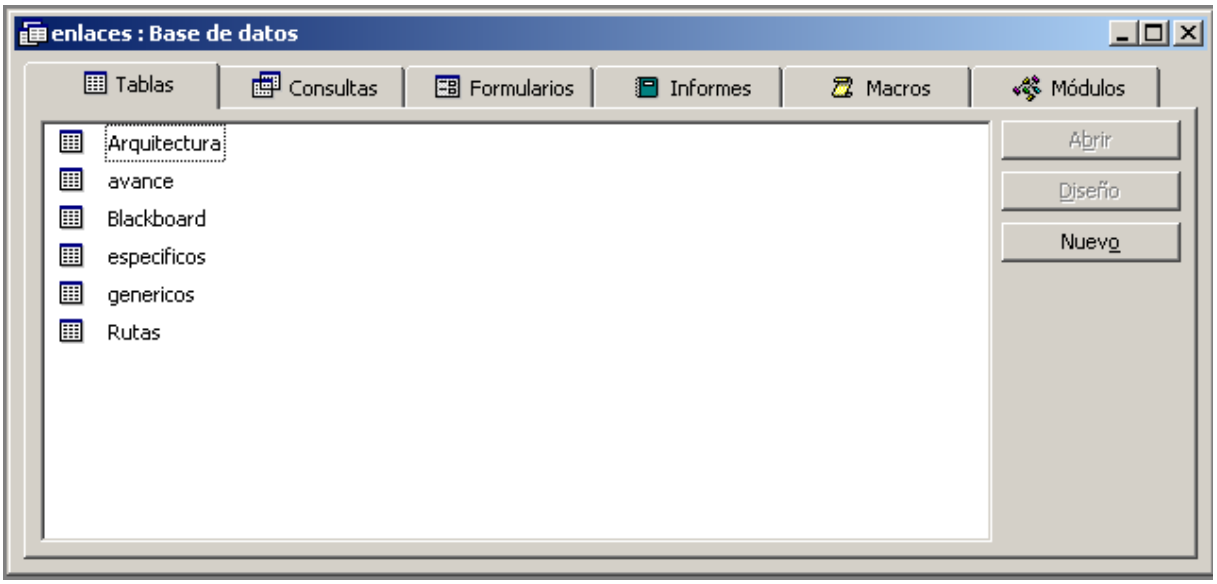

Figura 6.1: *La base de datos de enlaces*

Los datos almacenados en cada tabla se describen a continuación:

- **Arquitectura:** Contiene la configuración de la arquitectura.
  - arquitectura t[1]: Almacena un 1 si la arquitectura es la blackboard o 2 si es la conducto-filtro.
  - filtro_1 t[1]: El filtro número 1 en la conducto de filtros.
  - filtro_2 t[1]: El filtro número 2 en la conducto de filtros.
  - filtro_3 t[1]: El filtro número 3 en la conducto de filtros. Estos 3 campos almacenan el orden de los filtros para la arquitectura conducto-filtro donde los valores del 0 al 2 indican el tipo de enlace (Genérico, Específico y de Avance respectivamente).
- **Avance:** Almacena los enlaces de avance registrados en el sistema.
  - origen t[100]: Contiene el origen del enlace.
  - destino t[100]: Contiene el nuevo destino del enlace.
- **Especificos:** Almacena los enlaces específicos.
  - origen t[100]: Contiene el origen del enlace.
  - enlace t[3]: Contiene el número de enlace según el origen del mismo.
  - destino t[100]: Contiene el destino del enlace.

- **Genericos:** En esta tabla se almacenan todos los enlaces genéricos que han sido registrados en el sistema y aún los que ingrese el usuario en el futuro.
  - origen t[100]: Contiene el origen del enlace.
  - destino t[100]: Contiene el destino del enlace.
- **Blackboard:** Almacena de manera momentánea las solicitudes de los usuarios, y los resultados que se obtienen con el trabajo de las arquitecturas.
  - origen t[100]: Contiene el origen del enlace.
  - destino t[100]: Contiene el destino del enlace.
- **Rutas:** Sirve como almacenamiento temporal para el agente o filtro (según sea la arquitectura) para almacenar los desplazamientos de los documentos que contengan enlaces de avance.
  - origen t[100]: Contiene el origen del enlace.
  - destino t[100]: Contiene el nuevo destino del enlace.

6.4. Tecnologías utilizadas

6.4.1. El protocolo HTTP

El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como la Web. Este es utilizado como el protocolo de comunicación en el sistema, y el puerto utilizado es el 8080.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones

pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

El URL (*Localizador Uniforme de Recursos*) es la forma sintáctica que incluye toda la información necesaria para especificar un elemento remoto. Esta forma sintáctica codifica toda la información en una cadena de caracteres conocida precisamente como el URL. La forma general de un URL es:

`Protocolo://nombre_computadora:puerto/nombre_documento`

En la que *protocolo* es el nombre del protocolo con el que se accede al documento, *nombre_computadora* es el nombre del dominio de la computadora, *:puerto* es un número de puerto de protocolo opcional y *nombre_documento* es el nombre del documento en la computadora especificada.

6.4.2. HTML

HTML es un caso particular de los llamados **SGMLs** (*Standard Generalized Markup Languages*), un complejo sistema de definición de documentos estructurados, normalizado y aceptado internacionalmente. Surgió en 1969, como propuesta para disponer de un sistema único de representación e intercambio de documentos. Su estandarización definitiva se produce en 1986, al ser aceptado por ANSI e ISO.

El lenguaje HTML está definido a partir de unas reglas SGML, recogidas en un documento denominado **HTML DTD** (*Document Type Definition*). En él se especifica la estructura de un documento HTML, el conjunto de etiquetas disponibles, sus opciones, y la forma en que se pueden combinar. Así, un programa de ordenador puede verificar fácilmente la sintaxis de un documento HTML.

HTML (*HyperText Markup Language*) es el lenguaje utilizado para crear documentos de hipertexto. Los archivos HTML describen la distribución y estilo de cada uno de los elementos de una página de Web, a base de combinar el texto de la página con los diferentes comandos del lenguaje. Dentro de un documento HTML se incluyen todos los elementos de texto y formato de una página Web. Las imágenes, sonidos, etc. se almacenan en archivos independientes, y el documento HTML sólo contiene una referencia que los inserta.

HTML está construido a partir de "etiquetas" (*tags*), que marcan el formato de cada uno de

los elementos de la página de hipertexto. Todas las etiquetas son palabras o abreviaturas inglesas rodeadas de los símbolos < >. Las etiquetas pueden contener parámetros u opciones que modifican su comportamiento por defecto.

Todo el proceso de creación de enlaces se controla a través de la etiqueta <A>?<A> (*anchor*). <A> marca una sección de un documento como el origen o destino de un enlace de hipertexto. Para ello, se rodea el elemento deseado, que puede contener tanto texto como imágenes, con la etiqueta pareada <A>. La referencia <A> incluye información que indica un URL; si el usuario selecciona la referencia, el visualizador usa el URL para obtener el documento.

Todos los documentos de prueba en el sistema están creados en HTML, y el tag <A> es usado en el sistema para implementar a los enlaces específicos.

6.4.3. JavaScript

JavaScript es un lenguaje interpretado, inspirado en Java, que se incluye en los documentos HTML para añadir cierta interactividad a sus contenidos, evitando tener que realizar programación en el servidor. Fue desarrollado por Netscape y Sun Microsystems, y se puede usar en los clientes Web de Netscape (a partir de la versión 2.0) y Microsoft (a partir de la versión 3.0); hay ligeras diferencias en los intérpretes JavaScript de cada plataforma, por lo que en caso de duda, se deberá consultar la información de referencia del fabricante. Sus características principales son:

- Es un lenguaje de sintaxis similar a Java, en cuanto a tipos de datos y estructuras de control; sin embargo, al no ser compilado, realiza un control de tipos menos estricto. Por ejemplo, no es necesario declarar las variables, y su tipo puede cambiar durante la ejecución del programa. Todas las referencias entre objetos se resuelven en tiempo de ejecución.

- JavaScript también es interpretado, pero a diferencia de Java, el código JavaScript no se compila, sino que se inserta directamente en los documentos HTML. Por ello, no es necesario disponer de ninguna herramienta de compilación, sólo un browser que lo interprete.

- Utiliza un gestor automático de memoria dinámica, que reserva espacio para crear objetos y lo elimina cuando éstos ya no se utilizan.

- Está basado en un conjunto predefinido de objetos, que pueden ser extendidos. Sin embargo, no es posible crear nuevas clases, o establecer relaciones de herencia.

- Permite utilizar funciones, al estilo de los lenguajes de programación orientados a procedimientos. En Java, las funciones sólo pueden existir como métodos de acceso a un objeto.

Toda la funcionalidad de los documentos de prueba en el sistema está desarrollada en este lenguaje, con el implementamos los enlaces genéricos, así como también los enlaces específicos.

6.4.4. JAVA

Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Como lenguaje de programación para computadores, Java se introdujo a finales de 1995. La clave fue la incorporación de un intérprete Java en la versión 2.0 del programa Netscape Navigator, produciendo una verdadera revolución en Internet. Java 1.1 apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. Java 1.2, más tarde rebautizado como Java 2, nació a finales de 1998.

La compañía Sun describe el lenguaje Java como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico".

Agregamos solo que todo el sistema esta creado en este lenguaje, utilizando las API´s De Servlets y JDBC que describimos a continuación.

6.4.5. API JDBC

JDBC (*Java DataBase Connectivity*) es el estándar de Java para conectarse con bases de datos. Se estima que aproximadamente la mitad del software que se crea en la actualidad incorpora operaciones de lectura y escritura con bases de datos. JDBC está diseñado para ser independiente de la plataforma e incluso de la base de datos sobra la que se desee actuar.

Para conseguir esta independencia, JDBC ofrece un sistema estándar de interconexión con las bases de datos, muy similar al **SQL** (*Structured Query Language*). Los distintos vendedores de bases de datos crean los elementos necesarios que actúan como puente entre JDBC y la propia base de

datos.

La versión JDBC 1.0 forma parte del JDK 1.1. Después de distintas revisiones, actualmente ha aparecido la versión JDBC 2.0, incluida en el JDK 1.2.

6.4.6. Servlets

Como tecnología base utilizada para el desarrollo del sistema se utilizó Servlets de Java, para llevar a cabo la comunicación entre el servidor y el cliente. Los Servlets son módulos que extienden los servidores orientados a petición-respuesta, como los servidores web compatibles con Java. Los Servlets son para los servidores lo que los applets son para los navegadores. Sin embargo, al contrario que los applets, los servlets no tienen interfase gráfico de usuario.

Los servlets pueden ser incluidos en muchos servidores diferentes porque el API Servlet, el que se utiliza para escribir Servlets, no asume nada sobre el entorno o protocolo del servidor. Los servlets se están utilizando ampliamente dentro de servidores HTTP; muchos servidores Web soportan el API Servlet.

El paquete **javax.servlet** proporciona clases e interfaces para escribir servlets. La arquitectura de este paquete se describe en la figura 6.2.

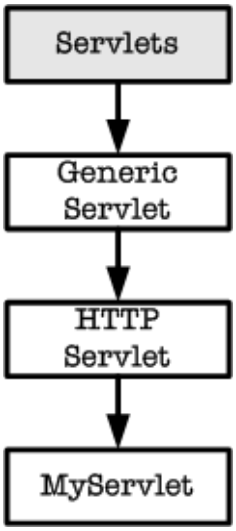

Figura 6.2: *La clase Servlet*

La abstracción central en el API Servlet es la interfase Servlet. Todos los servlets implementan este interfase, bien directamente o, más comúnmente, extendiendo una clase que lo implemente

como HttpServlet. Cuando un servlet acepta una llamada de un cliente, recibe dos objetos.

- Un ServletRequest, que encapsula la comunicación desde el cliente al servidor.
- Un ServletResponse, que encapsula la comunicación de vuelta desde el servlet hacia el cliente.

Tanto ServletRequest como ServletResponse son interfaces definidos en el paquete **javax.servlet**.

6.4.7. Apache Tomcat 4.0

El servidor Tomcat es una aplicación web basada en Java creada para ejecutar servlets y páginas JSP, siendo la implementación oficial de referencia de las especificaciones Servlet 2.3 y JavaServer Pages 1.2. De acuerdo con estas especificaciones, una aplicación web es una colección de servlets, páginas JSP, clases Java, archivos de descripción de la aplicación, documentos estáticos: HTML, XHTML, imágenes, etc. y otros recursos que pueden ser empaquetados y ejecutados en distintos servidores de diferentes proveedores. Es decir, una aplicación web se podría definir como la capa web de cualquier aplicación.

El contenedor que alberga una aplicación web no es más que la estructura de directorios en donde están colocados todos los archivos necesarios para la ejecución de la aplicación web. Por tanto, el primer paso en el desarrollo de cualquier aplicación web consiste en crear la estructura de directorios en donde colocar sus componentes. A continuación se indican los directorios necesarios para la aplicación, que debe colgar del directorio raíz del contenedor de servlets, que puede diferir de unos servidores JSP a otros. En el caso de Tomcat, el directorio a partir del cual se instala cualquier aplicación web debe ser *TOMCAT-HOME/webapps*, en donde TOMCAT_HOME apunta al directorio de instalación de Tomcat.

Una segunda parte muy importante de toda aplicación web es su descriptor de configuración. Este descriptor no es más que un archivo XML de nombre web.xml, localizado en el directorio *WEB-INF*, que contiene la descripción de la configuración correspondiente a la aplicación web.
La información que contiene este descriptor puede incluir los siguientes elementos:

- Parámetros de inicialización del ServletContext
- Configuración de la sesión
- Definiciones de Servlets/JavaServer Pages
- Mapeado de Servlets/JavaServer Pages

- Mapeado de tipos MIME
- Configuración de seguridad
- Páginas de error
- Páginas de bienvenida

Finalmente, la tercera parte importante de una aplicación web es el archivo **WAR** (*Web ARchive*), que es el método estándar empleado para empaquetar una aplicación web y dejarla lista para su distribución y acceso a través de servidores web con soporte para servlets y páginas JSP. Utilizando el archivo WAR se puede distribuir una aplicación web completa, compuesta por cualquier número de recursos, en una única unidad de distribución, en un único archivo.

Con este servidor a través del puerto 8080 se implementó el sistema, al instalar el servidor se crea el directorio Apache Tomcat que contendrá tanto las aplicaciones como los documentos del sistema. Tanto los documentos como los programas fueron colocados en los directorios configurados por default. Los documentos de prueba los instalamos en el directorio *webapps/ROOT*, y los programas servlets en la carpeta *webapps/examples/WEB-INF/classes*.

6.5. Resumen

En este capítulo solo detallamos la implementación del sistema cuyo diseño fue descrito en el capítulo anterior. Hicimos una breve descripción de como manipular en Java una base de datos utilizando la conectividad ODBC, así como también describimos la forma de implementar los enlaces y la funcionalidad de todo el sistema mediante los servlets de Java.

Capítulo 7

Conclusiones

En éste capítulo se discuten las características más importantes del sistema, así como los resultados más relevantes de éste trabajo como la extensibilidad para introducir nuevos tipos de enlace y la versatilidad para elegir el tipo de arquitectura de los servidores de enlaces. El caso de estudio presentado en el capítulo 4 nos permite demostrar la viabilidad de construir un sistema que extienda la funcionalidad tanto del servidor como del cliente Web para incorporar nuevos tipos de enlaces y arquitecturas para facilitar la recuperación de la información con nuevos mecanismos en forma abierta.

7.1. Contribuciones

Las contribuciones de éste trabajo son las siguientes:

- **Extensibilidad del modelo de enlace.** Este trabajo demuestra la forma en que se puede extender el modelo de hipermedia de la Web introduciendo varios modelos de enlace como el específico, el genérico y el de avance. Con ello se establece una forma de incluir nuevos tipos de enlace definidos por autores de hiperdocumentos. Aún cuando el trabajo a nivel de usuario final está lejos de ser sencillo, la arquitectura permite incorporar nuevos componentes escritos en Java y ensamblarlos en una variedad de formas para que respondan mejor a las necesidades de recuperación de información.

- **Arquitectura abierta para la resolución de enlaces.** Los modelos de enlace son implementados por servidores de enlaces los cuales a su vez se organizan de acuerdo a alguna arquitectura de software. En éste trabajo se ha extendido el carácter monolítico y cerrado del resolvedor de enlaces de la Web a una forma modular y abierta que permite la incorporación de diferentes servidores de enlaces.

- **Arquitectura configurable del procesador de enlaces.** El sistema permite elegir entre la velocidad de respuesta del modelo simple de la Web, a modelos más poderosos pero consecuentemente menos eficientes como los modelos conducto-filtro y de pizarra. El caso de estudio ha demostrado la importancia de conocer y seleccionar la arquitectura que más se adecúe al problema dado de recuperación de información.

- **Naturaleza abierta del sistema.** Los documentos se encuentran distribuidos para que tanto lectores como escritores pueden crear enlaces en el sistema sin necesidad de modificar los documentos existentes. Como el caso de estudio ha demostrado, el sistema resuelve el problema de recuperar información aún cuando no se tenga permiso para modificar los documentos.

7.2. Discusión

Ahora responderemos a algunas interrogantes que surgen al intentar comparar el sistema diseñado con respecto a la Web:

- **Manejo de diferentes tipos de enlaces.** En cuanto al manejo de los tipos de enlaces podemos decir que el sistema es mucho más poderoso con respecto a la Web, en principio porque la Web solo maneja un solo tipo de enlaces, mientras que el sistema implementado trabaja con este tipo de enlace y además con los enlaces genéricos y de avance, incluyendo el hecho de que al sistema se le puede ir agregando el manejo de nuevos tipos de enlaces.

- **Facilidad de uso.** Sin duda alguna, el manejo de los enlaces en la Web es mucho más intuitivo, por diferentes razones entre las cuales podemos citar en primer lugar, que todos hemos aprendido a utilizar casi de manera automática los enlaces en cualquier documento Web y en segundo lugar los documentos publicados en la Web escritos en el lenguaje de la Web (HTML) solo manejas un solo tipo de enlace que es reconocido fácilmente por el usuario(enlace aparece resaltado y subrayado). Mientras que en el sistema implementado, el usuario debe aprender a utilizar los nuevos enlaces y esto lleva tiempo. Concluimos que debido a los factores mencionados la Web sigue siendo más fácil de usar por cualquier usuario.

- **Eficiencia.** ¿Cuál de los dos sistemas resuelve un enlace con mayor rapidez? La Web en ningún momento hace una consulta a una base de enlaces para verificar si el enlace existe o está registrado, los enlaces HTML de los documentos Web hacen una referencia directa al documento referenciado por el enlace; no sucede lo mismo con el sistema implementado ya que este antes de devolver la referencia, consulta a su base de enlaces conllevando con esto

todo un proceso de búsqueda que consume tiempo de respuesta al usuario. Así concluimos que en cuanto a eficiencia en el tiempo de respuesta la Web sigue siendo mejor.

- **Flexibilidad.** En este punto el sistema implementado lleva la ventaja, porque (por lo menos hasta la fecha en que se escribió este trabajo) aunque parezca reiterativo, la Web está limitada solo al manejo de un solo tipo de enlaces (manejado por HTML) y no hay más, mientras que como ya se ha mencionado el sistema implementado maneja tres tipos de enlaces y si se desea puede manejar otros tantos con solo agregarlos al sistema.

Hasta aquí hemos solo comparado algunas características entre la Web y el sistema implementado, a continuación hacemos referencia a otras características importantes del sistema implementado y que ayudan a dar una perspectiva más amplia del sistema.

- **Portabilidad.** Una característica deseable de cualquier sistema es la portabilidad, en el caso específico del sistema implementado, toda la programación está hecha en Java, lo que nos permite instalar el código en cualquier sistema operativo sin necesidad de realizar modificaciones que tengan que ver con la funcionalidad de las arquitecturas. La única modificación en este punto se llevaría a cabo en la programación de acceso a la base de datos para realizar la conexión correcta al tipo de base de datos que vaya a ser utilizada, un punto a favor en estos cambios es el hecho de que la programación de acceso a datos está desacoplada con la programación de la funcionalidad del sistema. Lo cual implica un cambio de unas cuantas líneas de código. El documento cliente que se carga en el navegador para escuchar todas las peticiones del usuario está escrito en HTML y JavaScript, que son traducidos por prácticamente todos los navegadores. La base de datos de los enlaces es el único componente que no es portable, ya que el manejador de base de datos utilizado es Access que solo es reconocido por Microsoft.

- **Integración con la Web.** El sistema implementado trabaja de manera integrada con la Web porque cuando el usuario solicita un enlace propio de la Web, este no intercepta la petición sino que la deja pasar para que la misma Web responda a la solicitud hecha por el usuario. Gracias a esta característica del sistema no es necesario un nuevo protocolo de comunicación ya que trabaja de manera integral con la Web, así como tampoco se necesita un nuevo lenguaje de marcado, ya que como se explicó anteriormente el programa cliente instalado en el navegador solo escucha las peticiones de aquellos enlaces involucrados en el sistema aquellos que no lo son se dejan pasar para que la Web se encargue de ellos.

- **Comparación con la directiva redirect de los servidores Web.** Se puede discutir el hecho de que a través de la directiva redirect (direcciona un enlace a un nuevo destino) del

servidor Web de Apache es posible implementar el servicio de enlaces de avance lo cual solo en parte es cierto, por varios motivos:

- La directiva redirect almacena un enlace redireccionado. Esto significa que si el enlace redireccionado falla no hay manera de obtener la nueva ubicación del documento solicitado. En contraste con el sistema implementado que se pueden almacenar tantos enlaces de avance como se necesiten.
- El enlace almacenado en la directiva redirect se configura en tiempo de configuración. Por lo tanto si se desea reescribir el enlace redireccionado hay que tener acceso a la configuración del servidor Web. Mientras que en el sistema implementado el usuario puede agregar enlaces de avance en tiempo de ejecución tantas veces como sea necesario.

7.3. Trabajo a futuro

Concluimos ésta discusión estableciendo algunos de los trabajos a futuro que se pueden realizar tomando como base los resultados obtenidos en ésta investigación. Entre los trabajos a seguir se pueden destacar:

- **Definir nuevos modelos enlaces con mayor sencillez.** Como se discutió antes, los modelos de enlaces están escritos en Java y requieren de un grado de dominio considerable, tanto del lenguaje como del sistema, para poder definir e implementar nuevos modelos de enlaces. Esto se podría resolver con lenguajes visuales declarativos de muy alto nivel que ofrezcan al usuario elementos iconográficos que le sitúen al nivel conceptual del dominio de la aplicación. El sistema compilaría éstas descripciones para producir los componentes Java que implementarían a los servidores de enlaces.
- **Definir nuevas arquitecturas con mayor sencillez.** Al igual que con los modelos de enlaces, se requiere simplificar considerablemente tanto la definición de la arquitectura del procesador de enlace como la integración de los servidores de enlaces en la arquitectura. El problema se puede resolver en forma similar, definiendo lenguajes visuales declarativos que describan los procesos, sus interconexiones y sus interacciones, por ejemplo, usando UML.
- **Integrar éste sistema con sistemas de gestión de documentos.** Una de las aplicaciones más inmediatas de éste sistema es la administración y el control de acceso a documentos móviles Web en un ambiente distribuido. Actualmente, se ha desarrollado un sistema simple de *Workflow* basado en Web-DAV para la gestión de documentos que cambian de localidad

física de acuerdo a las peticiones de usuarios [18]. La integración de ambos sistemas permitiría dar seguimiento a un documento utilizando los enlaces de avance. De esta forma, los enlaces inválidos que generaría el subsistema de *Workflow* serían reparados por el subsistema de hipermedia, lo cual permitiría continuar leyendo el documento sin importar cuál es su localidad física.

Bibliografía

[1] Nelson, Ted H. *Literary Machines,* Swarthmore, PA: Self-published, 1981.

[2] Bloom, Harold, Paul de Man, Jacques Derrida, Geoffrey H. Hartman, and J. Hillis Miller. *Deconstruction and Criticism.* London: Routledge & Kegan Paul, 1979.

[3] Díaz, P., Catenazzi, N. , Aedo, I. (1996): *De la Multimedia a la Hipermedia,* RA-MA Editores, Madrid. 1996.

[4] Landow, G., Delany, P. (1991):*Hypermedia and Literary Studies,* Cambridge: Massachusetts Institute of Technology Press, 1991.

[5] Bolter, J. (1991): *Writing Space: The Computer, Hypertext, and the History of Writing,* Lawrence Erlbaum Associates, 1991. Review: http://bubl.ac.uk/journals/lis/ae/ejournal/v01n0291.htm

[6] Conklin, J.:*Hypertext: An Introduction and Survey,* IEEE Computer, September 1987. pp. 17-41

[7] Fiderio, J. (1988):*A Grand Vision–Hypertext mimics the brain's ability to access information quickly and intuitively by reference,* Byte Magazine, Vol. 13, Nº 10. October 1988. pp.237–244

[8] Balzer, R., Begeman, M., Garg, P., Schwartz, M., Shneiderman, B.: *Hypertext and Software Engineering*, Hypertext 1989. pp.395-396

[9] Rada, R. (1991):*Hypertext: from text to expertext,* McGraw-Hill. 1991.

[10] Berners-Lee, T., Caillaiau, R., Luotonen, A., Nielse, H., Secret, A. (1994): *The World Wide Web,* Communications of the ACM, Vol. 37, Nº 8. Agosto 1994. pp 76-82

[11] Davis, H.C. *Referential Integrity of Links in Open Hypermedia Systems.* In Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, Hypertex'98. K. Gronbaek, E. Mylonas, and F. Shipman III. Pitsburgh, 1998.

[12] Hall, W., Davis, H., and Hutchings, G., *Rethinking Hypermedia The Microcosm Approach.* Computer Networks and ISDN Systems. Vol.28, 7/11, pp. 1255,(1996).

[13] Davis, H.C. *To Embed or Not to Embed...*,Comunications of ACM, 38(8):108-109, (1995).

[14] Moreau, L., and Hall, W., *On the Expressiveness of Links in Hypertext Systems.* The Computer Journal, 41(7)¿459-473,(1988).

[15] Olmedo-Aguirre, J.O., *Design and Construction of a Coordination Languaje with Applications to Hypermedia.* PhD Thesis. University of Southampton,(2000).

[16] Nordbotten, Joan C., and Nordbotten, Svein., *Search Patterns in Hypertext Exhibits.* Proceedings of the 32nd Hawaii International Conference on System Sciences -1999. Dept of Information Science,University of Bergen, N-5020 Bergen, Norway.

[17] Golovchinsky, Gene, *Queries? Links? Is there a difference?.* CHI97 Electronic Publications: papers. FX Palo Alto Laboratory, http://www.acm.org/sigchi/chi97/proceedings/paper/gxg.htm.

[18] Flores Hernández Sais: *Gestión de Documentos en la Web con webDAV*, Tesis de Maestría en Ciencias, Departamento de Ingeniería Eléctrica, Cinvestav. Por publicarse.