FILTRO DE PÁGINAS WEB

LUIS MIGUEL PARDO LAREDO

ETIG / ETIS

Consultor: Maribel March Hermo

RESUMEN

Este proyecto final de carrera tiene la finalidad de elaborar un filtro de contenidos Web de palabra clave. Se ha tratado exclusivamente de crear una aplicación que funcione sobre el protocolo http controlando los datos que circulan a través del puerto 80.

Para la consecución de los objetivos marcados inicialmente, se ha partido de los conocimientos adquiridos por el alumno a lo largo de su carrera universitaria y la búsqueda de información en diferentes documentos, especialmente de los que se pueden obtener de Internet.

Esta memoria intenta plasmar el método seguido en el proceso de realización, partiendo de una planificación inicial que en algunos casos se ha visto modificada a causa de las dificultades encontradas en la concreción de determinas tareas, las cuales se ha considerado importante recoger en el apartado 2.3 de la mima.

Este documento se encuentra dividido en cinco apartados, y va acompañado de una aplicación programada en lenguaje Java. Aunque se podría haber utilizado otros lenguajes de programación, se optó por realizarlo en Java porque es un lenguaje de alto nivel fuertemente relacionado con Internet. Otro aspecto que influyÓ en la decisión fue la portabilidad del código, dado que permite su ejecución en cualquier máquina independientemente del sistema operativo instalado.

Por último comentar que se ha tratado de realizar un proyecto que tuviese una aplicación real y como consecuencia de una necesidad en el desarrollo de la actividad profesional del alumno. El deseo es que suponga un punto de partida para su posterior perfeccionamiento y que pueda implicar a otras personas que deseen continuar la labor iniciada.

Barcelona, 9 de enero del 2005

ÍNDICE

ÍNDICE DE FIGURAS

Página

1. MEMORIA

1.1. INTRODUCCIÓN

Internet ha crecido de forma exponencial en los últimos años. Según un estudio de la Universidad de Macquarie en Australia, hay más de 73 millones de páginas nuevas de Internet por día. Constituye por una parte una importante herramienta educativa, de información y social; pero por otro lado entraña un riesgo importante para los menores.

La introducción de Internet en las aulas educativas y en la mayor parte de los hogares ha puesto de manifiesto una nueva necesidad para la sociedad, poder controlar que información están recibiendo los menores a través de este nuevo canal de información. En Internet pueden circular contenidos calificables como nocivos, perjudiciales u ofensivos para determinados colectivos de la sociedad. La existencia de este tipo de información es totalmente legal aunque puede ser considerada inmoral por algunas personas.

Debido a la imposibilidad de control de los contenidos de un servidor desde el punto de vista legal y la dificultad para los proveedores de Servicios de Internet (ISP) de controlar todo lo que pasa por sus servidores; el mercado informático ha puesto a disposición de los usuarios toda una serie de sistemas para evitar que los niños accedan en Internet a contenidos no adecuados para ellos.

Existen diversas modalidades de filtrado entre las que cabe destacar las siguientes:

- Los filtros de exclusión: verifican una serie de sitios de Internet a través de listas de páginas prohibidas impidiendo a los usuarios el acceso a cualquier lugar presente en dicha relación. Este sistema presenta los siguientes problemas:
    - La base de datos que alimenta la lista de páginas prohibidas debe actualizarse periódicamente. Algunas empresas permiten que se

descarguen las actualizaciones de forma gratuita de la Red, mientras que otras cobran por la actualización.

- Las listas se quedan obsoletas desde el mismo momento en que son creadas ya que cada día se crean cientos de páginas.

- Los filtros de inclusión: funcionan según el mecanismo contrario. Los sitios verificados por el software son los únicos a los que se permite acceder. Es plenamente eficaz y seguro pero muy restrictivo. Presenta el inconveniente que los listados deben revisarse y ampliarse para no quedar obsoletos.

- Filtros basados en una palabra clave: no permiten acceder a textos que contengan las palabras incluidas en una lista. El principal problema es que no se ejerce ningún control sobre otro tipo de documentos que no sean textos, por ejemplo fotografías. Además, es imposible para esta clase de filtros bloquear los contenidos nocivos sin bloquear a la vez un gran número de información perfectamente sana y útil

Existen otro tipo de métodos que no siguen el mismo tipo de política y en algunos casos no son una verdadera solución. Cabe destacar los siguientes métodos:

- Grabar los lugares visitados por los usuarios: dejando constancia de las actividades que han realizado.

- Sistemas de verificación de la edad: Algunas páginas web con contenido para adultos disponen de un sistema para verificar la edad. Este sistema es totalmente ineficaz, ya que se puede acceder al contenido sin cumplir los requisitos.

- Etiquetado: es un medio que permite describir el contenido de un sitio sin tener por qué acceder a él para comprobar qué contiene. El medio estándar de calificación y etiquetado de contenidos más extendido en Internet es el ofrecido por PICS (Plataforma para la Selección de Contenido en Internet). Su funcionamiento se basa en insertar en los documentos Web unas etiquetas electrónicas textuales o icónicas

invisibles para el lector que describen el contenido de esa página concreta. El proyecto PICS es mantenido por ICRA , (Internet Content Rating Association). Una ejemplo de etiqueta podría ser la siguiente:

<meta http-equiv="pics-label" content='(pics-1.1
"http://www.icra.org/ratingsv02.html" l gen true for
"http://www.example.com" r (nz 1 vz 1 lc 1 oz 1 cz 1))' />

En la actualidad existen filtros que combinan ambos sistemas para mejorar la efectividad y aportan otros métodos para controlar las paginas visitadas es el caso de semáforo.net (http://www.semaforo.net/) que es un filtró de contenidos desarrollado en España que, además de chequear la dirección en una base de datos, rastrea el servidor de origen, analiza el nombre del sitio y las denominaciones de las imágenes que contiene, busca palabras clave y analiza los colores que predominan en la página; además crea informes detallados de navegación y controla el uso horario de Internet.

En un futuro y viendo la expansión de Internet en todo el planeta, la solución al problema no debería pasar por la utilización de filtros que por un lado ralentizan la conexión, sino más bien es necesario que cada sitio etiquete su contenido (PICS) incluyendo un encabezado extra y los adultos controlen que contenidos pueden visionar los niños. Ahora bien viendo la tendencia actual y el rechazo de grupos hacia el etiquetado amparados en el derecho de la libertad de expresión, parece difícil que este objetivo se pueda alcanzar.

1.2. JUSTIFICACIÓN DEL TRABAJO FINAL DE CARRERA

Después de tres años desde el inicio de mi carrera universitaria son numerosos los temas tratados en las diferentes materias, así como los conceptos aprendidos a lo largo de estos años.

Desde un principio mi interés se ha centrado en los sistemas operativos, especialmente en el sistema Linux. El hecho de que se trate de un software libre, de código abierto y que se pueda adaptar a las necesidades del usuario son los argumentos principales para mi interés hacia este sistema. Por estos motivos, en un principio pensé en realizar un proyecto final de carrera centrado en adaptar una distribución linux a las necesidades del centro educativo donde realizo mi labor profesional.

En el momento de decidir que proyecto debía realizar, analicé por un lado los conocimientos teóricos que poseía y la problemática que se estaba presentado en el centro escolar en el que realizo la labor de coordinador de informática. En la labor de mantenimiento informático encontraba en numerosas ocasiones páginas con contenido para adultos, no únicamente de contenido sexual sino también con fuerte contenido violento. Por tanto empecé por buscar aplicaciones para poder controlar los contenidos y encontré algunos como: Cyberpatrol, Net Nanny , Proxomitron, FilterGate y Ad Substract Pro … que funcionaban bien, pero en muchos casos no se adaptaban a nuestras necesidades o eran de pago. Los que eran gratuitos; optenet, Naomi, dansguardian, squidguard, etc. resultaban poco adaptables a nuestra problemática.

Un vez centrado el problema opté por desarrollar un proyecto de filtrado de páginas Web cuyo objetivo era controlar el contenido y vocabulario que aparecían en las paginas Web. Dado que el uso que se realiza de Internet en el centro por parte del alumnado es únicamente consultar información, no trate de crear un filtro para todo tipo de aplicaciones ni protocolos. Concreté mi trabajo en un filtrado del protocolo http exclusivamente y en el la adaptación del software a cualquier navegador Web.

En la actualidad se comienza a proponer un cambio del sistema operativo actual, windows, hacia Linux. El departament d'ensenyament de

Catalunya esta impulsando la idea de migrar hacia un sistema de software libre, supongo que por motivos de costes económicos y atraído por la idea de la gran evolución de software libre y sus numerosas aplicaciones. Este motivo me impulso a realizar mi proyecto utilizando el lenguaje Java. El lenguaje Java cumple con el propósito buscado ya que los archivos de código fuente se pueden compilar en cualquier ordenador, solo es preciso que tenga instalado el entorno de ejecución.

Espero poder realizar una aplicación sencilla, que realmente pueda tener un uso generalizado en cualquier centro educativo y que represente una línea inicial de trabajo para involucrar a otras personas preocupadas por la misma problemática, que contribuyan a mejorar el producto inicial para obtener una mayor eficacia.

1.3. OBJETIVOS

El objetivo principal es crear una aplicación situada entre el navegador e Internet, muy similar a un Proxy con cache; con la finalidad de buscar palabras claves en las páginas solicitadas para impedir el acceso a contenidos para adultos (sexo, violencia...). Cuando se deniegue una petición, se servirá una página Web substitutoria que indicará el acceso a una página calificada para adultos. Por tanto el objetivo principal es realizar un filtro basado en palabras clave. La aplicación mantendrá un registro de las páginas Web denegadas para facilitar el proceso si en otra ocasión se quiere acceder a la misma página.

No se contempla la posibilidad de utilizar un segundo criterio de filtrado: comprobar si el dominio o la IP solicitada se encuentra en una lista de lugares Web que traten temática para adultos, es decir un filtro de exclusión tal y como se definió en la introducción. Pero si se cumple la planificación marcada del proyecto y se observa que no supone una carga muy elevada de trabajo se podría llegar a implementar, aunque inicialmente no se incluye entre las tareas a realizar.

Después de realizar un primer análisis de las diferentes posibilidades para resolver el problema, se comprueba la dificultad de realizar un filtrado totalmente eficaz.

Las búsquedas de palabras clave no pueden utilizar información contextual. Aunque las búsquedas pueden identificar la presencia de determinadas palabras en un texto, no pueden evaluar el contexto en que dichas palabras se encuentran. Es posible que algunas páginas de medicina, que contengan palabras como “genitales, pecho…” sean rechazadas, esto se conoce como falsos negativos/positivos.

Las búsquedas de palabras clave no pueden interpretar gráficos. Por lo tanto, una página que contenga imágenes crudamente sexuales sólo será bloqueada si el texto que aparece en la página contiene una o más palabras de la lista de palabras bloqueadas, pero si solo contiene imágenes se permitirá el acceso, son los falsos negativos/positivos.

Si bien es cierto que se producirán falsos negativos/positivos y positivos /negativos, se pretende conseguir el objetivo principal con un porcentaje elevado de efectividad y pensando, como se ha comentado con anterioridad, que este trabajo es una primera aproximación al problema que se nos plantea y que en un futuro es posible que se perfeccionen las técnicas utilizadas para el filtrado Web.

1.4. ENFOQUE Y METODO SEGUIDO

Este proyecto se inicia con la especificación de los elementos necesarios para poder realizar un filtro Web. Después de cursar la materia de: redes, aplicaciones y protocolos de Internet, poseía un conocimiento de los elementos que podían conformar la aplicación a la que me enfrentaba. Además tenía una base teórica inicial en el protocolo Http y experiencia en programación en Java.

PRIMERA FASE

El primer paso fue profundizar en el protocolo http, especialmente debía conocer como realizaba una petición de página Web un explorador y como me respondería el servidor. También necesitaba saber el tipo de datos y mensajes que se intercambiaban en una comunicación. Para localizar la información recurrí a información expuesta en Internet y a un manual especificado en la bibliografía.

En una segunda aproximación a la resolución del problema investigué acerca del funcionamiento de los Proxy. En este apartado mi desconocimiento en la materia era total, pero resultó sencillo encontrar información aunque en muchos casos se trataba de servidores Proxy externos y en el caso que afrontaba, el Proxy debía correr sobre la misma máquina que realizaba la navegación por Internet.

En el último término de la aproximación a los conocimientos teóricos, estudie el funcionamiento de los Sockets en Java, así como clases relacionadas: DataInputStream, DataOutputStream, PrintStream, FileOutputStream… Mi conocimiento en estas clases es bastante amplio dado que durante mi carrera universitaria he tenido que utilizarlas para la realización de aplicaciones. Al mismo tiempo me propuse realizar un repaso de otras clases de Java, String, byte, Array…, que podrían tener una utilidad en el momento de programar la aplicación.

SEGUNDA FASE

Una vez alcanzado este punto me dispuse a buscar en Internet aplicaciones que realizasen la misma función. Me encontré con el inconveniente que muchas eran versiones comerciales que requerían un pago para su descarga. Del software que localicé y pude descargar realice diferentes pruebas en los navegadores Web Explorer y Mozilla para tener una visión del rendimiento y funcionabilidad.

La finalidad de la búsqueda del software de filtrado era tener un visión global se su funcionamiento y poder personalizar mi aplicación a partir del análisis del mismo. En realidad la complejidad del software era tan elevada que se alejaba por completo del planteamiento de mi trabajo.

TERCERA FASE

En esta fase comencé a implementar la aplicación en java, al mismo tiempo que realizaba la memoria del trabajo.

La memoria se fue concretando a medida que avanzaba en la programación de la aplicación. En esta última fue donde encontré más dificultades al principio.

Empecé implementando un Proxy, es decir, un programa que capturase la solicitud del navegador y la enviase al servidor de páginas Web real. Después se encargaba de recoger la información que recibía del servidor y la enviaba al navegador.

Además el Proxy podía seleccionar las páginas Web que serviría al navegador en función de si estas se encontraban en una lista previamente elaborada. Esto era una primera aproximación al filtro Web que tenía que construir.

En esta parte es donde dedique más tiempo que el planificado, debido a que el Proxy no interpretaba correctamente los mensajes del servidor Web. En el apartado 2.3 de la memoria se concreta esta problemática y la solución aportada.

Una vez terminada la implementación del Proxy, me centré en analizar el contenido de la información que enviaba el servidor, concretamente implemente un programa que buscase palabras contenidas en un diccionario.

Según el resultado de la búsqueda se decidía si la información se enviaba al navegador. Asimismo se creó un historial para recoger las páginas que se visitaban conjuntamente con la fecha y la hora.

CUARTA FASE

En las últimas semanas, antes de realizar la entrega, se revisó el código del filtro Web en busca de errores de programación y al mismo tiempo se añadieron los comentarios oportunos para poder hacer más inteligible la funcionabilidad de la aplicación.

Se revisó todos los puntos de la memoria, para comprobar la coherencia del documento, la ortografía y se completaron algunos apartados “2.4 y 2.5” que en un principio no estaban contemplados.

Por último se seleccionó los aspectos más relevantes de la memoria para poder crear las diapositivas que resumiesen todo el proceso de realización del proyecto.

1.5. PLANIFICACIÓN DEL PROYECTO

- *Documentación: 9 de octubre 2005*
    - El protocolo cliente-servidor Http (Hypertext Transfer Protocol).
    - Conceptos básicos de Html y funcionamiento de los principales navegadores Web.
    - Conceptos específicos de programación en Java (Clase Socket)
    - Funcionamiento de un servidor Proxy.
- *Servidor Proxy-cache : 6 de noviembre del 2005*
    - Establecer la funcionabilidad que ha de desarrollar el Proxy-cache.
    - Implementación de la aplicación a partir de les funciones que ha de desarrollar.
    - Elaboración de la documentación de la aplicación.
- *Filtre: 27 de noviembre del 2005*
    - Estudio de páginas Web con contenido para adultos y análisis del tipo de vocabulario utilizado.
    - Elaboración de un diccionario con terminología presente en estas páginas Web.
    - Implementación del Proxy-cache conjuntamente con el filtro de contenidos.
    - Ampliación de la documentación elaborada con las nuevas especificaciones.
- *Memoria: 18 de diciembre del 2005*
    - Finalización y revisión de aspectos pendientes de la memoria. (Durante todo el proceso se irá realizando).
    - Documentación de PowerPoint y elaboración de la presentación del trabajo.
- *Entrega: 9 de enero del 2006*
    - Revisión y últimos retoques de la aplicación y de tota la documentación técnica.
    - Presentación del trabajo final de carrera.

2. ESPECIFICACIÓN DE LA APLICACIÓN

2.1. CONCEPTOS TEÓRICOS

2.1.1. EL PROTOCOLO HTTP

DEFINICIÓN

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. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945.
Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP.

El protocolo HTTP se basa en un paradigma de peticiones y respuestas. Un cliente envía una petición en forma de método, una URI, y una versión de protocolo seguida de los modificadores de la petición de forma parecida a un mensaje MIME, información sobre el cliente y al final un posible contenido. El servidor contesta con una línea de estado que incluye la versión del protocolo y un código que indica éxito o error, seguido de la información del servidor en forma de mensaje MIME y un posible contenido.

Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc., respetando su formato original.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
- Existen tres verbos básicos (hay más, pero por lo general no se utilizan) que un cliente puede utilizar para dialogar con el servidor: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD,

para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).

- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.
- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
- Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.

ETAPAS DE UNA TRANSACCIÓN HTTP

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo *Location* del cliente Web.
- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
- Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del *browser*, datos opcionales para el servidor,…
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.

En la actualidad se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo,

denominado *HTTP Keep Alive*, es empleado por la mayoría de los clientes y servidores modernos. Este es el modo de funcionamiento por defecto en el http/1.1

CABECERAS

Son un conjunto de variables que se incluyen en los mensajes HTTP, para modificar su comportamiento o incluir información de interés. En función de su nombre, pueden aparecer en los requerimientos de un cliente, en las respuestas del servidor o en ambos tipos de mensajes. Los nombres de variables se pueden escribir con cualquier combinación de mayúsculas y minúsculas. Además, se debe incluir un espacio en blanco entre el signo : y su valor. En caso de que el valor de una variable ocupe varias líneas, éstas deberán comenzar, al menos, con un espacio en blanco o un tabulador.

- Content-Type: descripción MIME de la información contenida en este mensaje.
- Content-Length: longitud en bytes de los datos enviados, expresado en base decimal.
- Content-Encoding: formato de codificación de los datos enviados en este mensaje.
- Date: fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación.
-

Cabeceras para peticiones del cliente:

- Accept: campo opcional que contiene una lista de tipos MIME aceptados por el cliente. Se pueden utilizar * para indicar rangos de tipos de datos; tipo/* indica todos los subtipos de un determinado medio, mientras que */* representa a cualquier tipo de dato disponible.
- Authorization: clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización empleado, seguido de la clave de acceso propiamente dicha. La explicación se incluye más adelante.
- From: campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.

- If-Modified-Since: permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.
- Referer: contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.
- User-agent: cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los *browsers* de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I)

Cabeceras para respuestas del servidor:

- Allow: informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta.
- Expires: fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha.
- Last-modified: fecha local de modificación del objeto devuelto. Se puede corresponder con la fecha de modificación de un fichero en disco, o, para información generada dinámicamente desde una base de datos, con la fecha de modificación del registro de datos correspondiente.
- Location: informa sobre la dirección exacta del recurso al que se ha accedido. Cuando el servidor proporciona un código de respuesta de la serie 3xx, este parámetro contiene la URL necesaria para accesos posteriores a este recurso.
- Server: cadena que identifica el tipo y versión del servidor http
- WWW-Autenticate: cuando se accede a un recurso protegido o de acceso restringido, el servidor devuelve un código de estado 401, y utiliza este campo para informar de los modelos de autentificación válidos para acceder a este recurso.

Solicitud del cliente: Este es un ejemplo de petición que realiza el navegador al servidor Web.

GET http://www.google.es/ HTTP/1.0: Solicitud y versión de http.
***Accept: ** : entiende y acepta todos los tipos MIME.
Accept-Language: es: idioma que soporta (preferentemente) el cliente.
Accept-Encoding: gzip, deflate: indica su disposición a utilizar compresión de datos.
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1): nombre del navegador o del programa usado para acceder al recurso solicitado.
Host: www.google.es : nombre del servidor.
Linia en blanco: Indica la finalización de la petición.

Respuesta del servidor: El servidor respondería con la siguiente cabecera si es correcta la petición

HTTP/1.0 200 OK : Estado de la operación, el 200 indica que es correcto.
Date: Sat, 19 Nov 2005 01:12:26 GMT: fecha de la solicitud.
Server: Apache/1.3.29 (Unix): Tipo y versión del servidor.
Expires: Sun, 03 May 1998 16:00:00 GMT: Fecha y hora a partir de la cual el contenido se considera obsoleto. Como es anterior a el campo date no se guardará en la cache.
Connection: close: Indica que se cerrará la conexión en cuanto se finalice de transmitir la respuesta.
Transfer-Encoding: chunked: Indica que el tipo de codificación vendrá indicada en cada una de las cabeceras de los paquetes TCP.
Content-Type: text/html: Indica el tipo de contenido de la página. Se representa según la especificación Mime.
Linia en blanco: Indica la finalización de la cabecera
<html>
..... Cuerpo del mensaje.
</html>

2.1.2. EL SERVIDOR PROXY

Un Proxy es un programa que hace de intermediario en una comunicación y que además acostumbra a ofrecer otros servicios

Típicamente se usan en redes de área local para poder compartir una sola conexión a Internet. Aunque usarlo para aumentar el número de máquinas con acceso a Internet cuando se tienen pocas direcciones IP tiene muchos inconvenientes. El más importante es que mantiene el interior completamente inaccesible desde el exterior. En algunos casos esto puede significar una gran ventaja, por ejemplo desde el punto de vista de la seguridad, pero en numerosas ocasiones supone un inconveniente, especialmente cuando se desea acceder desde el exterior a un ordenador personal.

Los servidores Proxy acostumbran a implementar una cache interna que permite aumentar el rendimiento de las conexiones a Internet. En esta cache se almacena las páginas solicitadas por los usuarios. De esta manera, la próxima vez que se solicite esa página se descargará directamente del proxy-caché, en lugar de hacerlo desde el servidor. Puesto que las peticiones de las páginas almacenadas no tienen que viajar hasta el servidor, la navegación será más rápida.

Estos Proxy no almacenan ni las contraseñas, ni datos privados de usuarios, ya que Las conexiones seguras se realizan mediante otro protocolo, el SSL. Todas las páginas dinámicas (ASP, php) tampoco son almacenadas, así como tampoco las de autentificación.

Se suelen definir los puertos sobre los que trabajará el Proxy que suelen ser: el puerto 80 (el de la navegación Web), el 1755 (streaming de Windows Media), 554 y 7070 (streaming de Real Networks)…

Las peticiones efectuadas por puertos distintos de los anteriores no se cachea, es decir pasan directamente a través del Proxy sin que este realice ninguna función, cuando se trata de un servidor Proxy, o no o pasan a través del Proxy si este se encuentra en la misma máquina.

Los Proxy se pueden clasificar en dos tipos:

- Externos: cuando se encuentran en una maquina que actúa como servidor y es diferente a la que realiza las peticiones, cliente. Son los que facilitan los proveedores de servicios de Internet (ISP), como ya.com, Terra…
- Internos: se encuentran en la misma máquina que el navegador que realiza la petición. El más destacado es Squid ya que posee una de las mejores cachees que existen, y es utilizada por miles de empresas e ISP.

2.1.3. SOCKETS EN JAVA . APLICACIÓN

Para establecer un Proxy son necesarias las clases ServerSocket y Socket.

La clase ServerSocket está pensada para ser utilizada en el extremo del servidor (socket del servidor). Su constructor se encarga de crear un Socket, asignarle un puerto y crear una cola que almacenará las peticiones.

```
ServerSocket escuchandoSocket = new ServerSocket(puertoServicio)
puertoServicio = 8000
```

La clase Socket se hace servir en el extremo del cliente y su constructor crea un Socket y la petición de conexión al servidor a través del puerto especificado.

```
Socket socketCliente = escuchandoSocket.accept();
```

Al crear el socket se utiliza el accept de la clase ServerSocket creando un bucle infinito, quedando el servidor a la espera de aceptar una conexión. Cuando esta conexión se produce esta clase devuelve un objeto de tipo socket que es el que utiliza el servidor para comunicarse con el cliente.

También es necesario crear un DataOutputStream por donde el servidor enviará datos al cliente y un DataInputStream por el cual podrá recibirlos.

```
salida = new DataOutputStream(socketCliente.getOutputStream());
entrada = new DataInputStream(socketCliente.getInputStream());
```

Ambas clases poseen el método Close() que es de vital importancia para el correcto funcionamiento de la comunicación. Considerando que estamos creando un Proxy que atenderá numerosas peticiones, tenemos que cerrar cada socket (socketCliente.close()) en el momento de concluir cada comunicación. Para cada petición se creará un SocketCliente nuevo de esta forma la información no se mezclará entre diferentes peticiones. En cambio el ServerSocket permanecerá a la escucha mientras tengamos la aplicación en funcionamiento y solo se cerrará (ServerSocket.Close()) cuando deseemos finalizarla.

He de comentar que existen diferentes formas de crear Sockets y ServerSockets, utilizando otros constructores con diferentes parámetros, pero para el desarrollo de la aplicación únicamente se precisan los constructores que se han especificado.

2.2. DESARROLLO PRÁCTICO

2.2.1. PRODUCTOS OBTENIDOS Y EJECUCIÓN DE LA APLICACIÓN

Par el funcionamiento de la aplicación es preciso tener instalado el programa Java, concretamente JDK 1.4.1 o una versión superior. Además de disponer de un navegador Web para poder realizar las peticiones de páginas Web y una conexión a Internet.

La aplicación esta formada por los siguientes archivos:

denegar.txt : Contiene direcciones Web que se han denegado con anterioridad o bien que el usuario ha introducido manualmente.

diccionario.txt: Contiene los términos que se pueden encontrar en páginas con contenidos para adultos y que utilizará el filtro como base para rastrear las páginas Web.

historial.txt: Recoge un listado de los sitios Web a los que se accede desde la máquina indicando la fecha y la hora.

prohibido.htm: Pagina Web que se servirá en el caso de intentar acceder a una Web que posea contenidos inadecuados para menores.

paginaHTML.htm: Archivo que recoge en formato texto el código Html que devuelve el servidor a nuestra petición. Se utilizará para analizar el contenido de las palabras que figuran.

Filtro.java, Filtro.class: Esta clase realiza el filtrado, comparando cada palabra del archivo diccionario.txt con paginaHTML.htm. Devolviendo true si encuentra palabras prohibidas o false si no las encuentra.

PeticionHttp.java, PeticionHttp.class:. Esta clase se utiliza para modificar la petición del navegador y extraer la dirección Web del servidor.

Servidor.java, Servidor.class: Es el eje central de la aplicación. Se encarga de recoger la petición Http del navegador analizarla y enviarla al servidor. Posteriormente analiza la respuesta y decide si presentar la página o deniega la petición. Está formada por:

- un núcleo principal (main) que se encarga de recibir las peticiones, enviar la petición al servidor Web real, y devolver la respuesta al navegador.

- Método Denegación: compara la dirección Web solicitada con las introducidas en el archivo denegar.txt.
- Método Log: guarda en historial.txt todas las páginas Web solicitadas con la fecha y hora de solicitud.

Si situamos estos archivos en la carpeta “filtro” y la ruta hasta la misma es: c:\filtro\ .Dado que el filtro Web se presentará tanto con los archivos fuentes (.java) como con sus respectivos archivos compilados (.class), para ejecutar el programa tendríamos que introducir el siguiente comando en una ventana de Dos:

C:\filtro\java Servidor

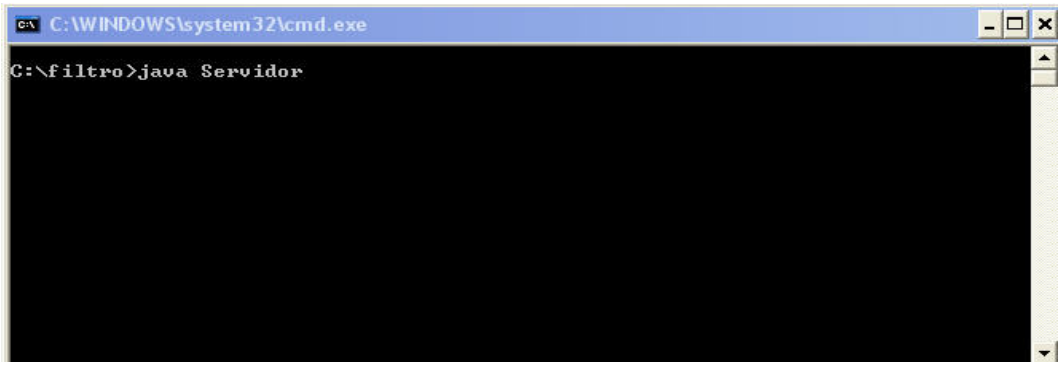

Figura 2.2.1.1 Ejecución de la aplicación

La aplicación no está pensada para ser ejecutada en red sinó que se instalará en cada máquina donde se precise. En el momento de programar se ha tomado la determinación de que todos los archivos fuentes se han de colocar en la misma carpeta. En el caso de introducir alguna modificación será preciso volver a compilar el programa principal con el siguiente comando:

C:\filtro\javac Servidor.java

Figura 2.2.1.2 Compilación de la aplicación

2.2.2. CONFIGURACIÓN DEL NAVEGADOR:

El filtro Web se ha probado en los dos navegadores más utilizados actualmente: Internet Explorer y Mozilla Firefox. El motivo de realizar las pruebas en dos navegadores es evaluar el rendimiento que ofrecen cada uno ante la utilización de la aplicación.

Tal y como se ha comentado con anterioridad el programa actúa como un Proxy, por tanto se tendrá que modificar la configuración que viene definida por defecto en ambos exploradores. En ambos casos definiremos el puerto 8000 para la comunicación entre el navegador y el filtro Web.

En el caso de utilizar Internet Explorer la configuración del Proxy se encuentra en las opciones de Internet, dentro del apartado conexiones.

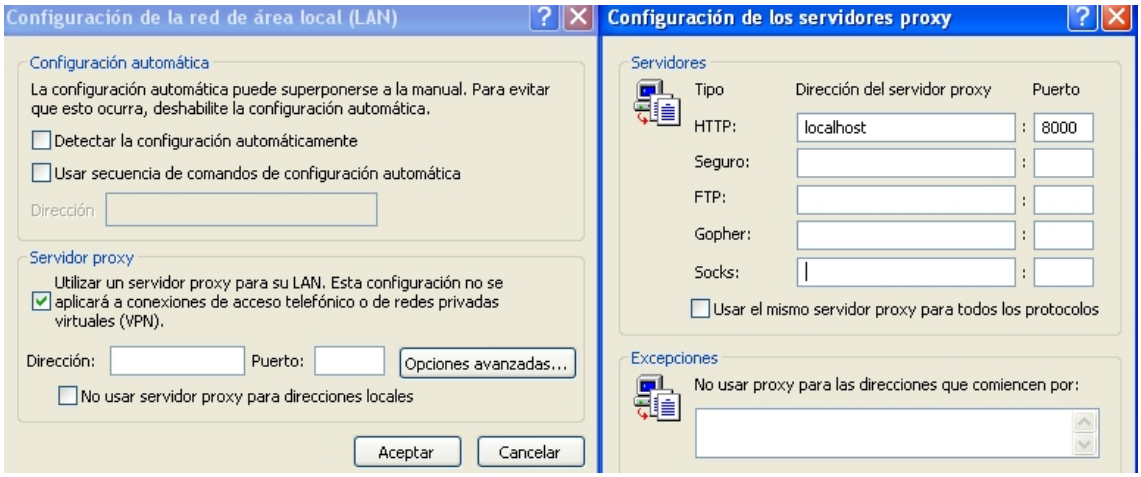

Figura 2.2.2.1 Configuración del navegador Explorer

Si por el contrario, se utilizase Mozilla Firefox, tendremos que abrir el menú Herramientas y seleccionar la pestaña opciones:

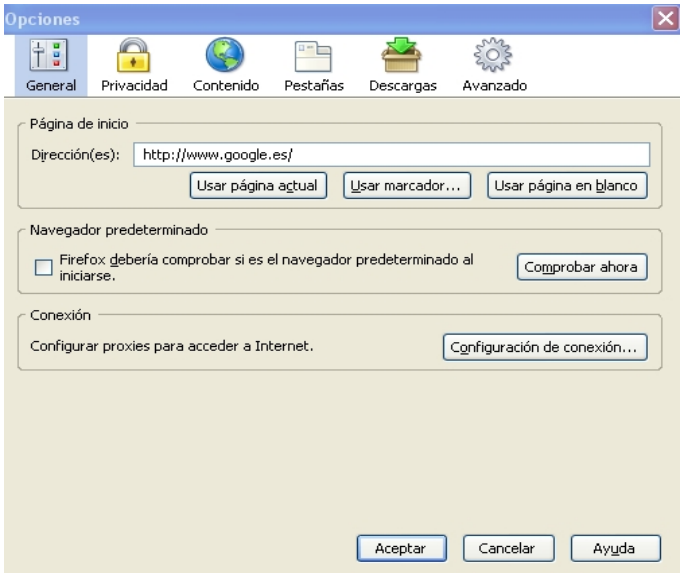

Figura 2.2.2.2 Configuración del navegador Firefox 1

En la ventana opciones que se desplegará, vemos en la imagen anterior que posee un botón para configurar la conexión. Al seleccionarla nos aparecerá una nueva ventana donde podremos definir el puerto que deseamos utilizar:

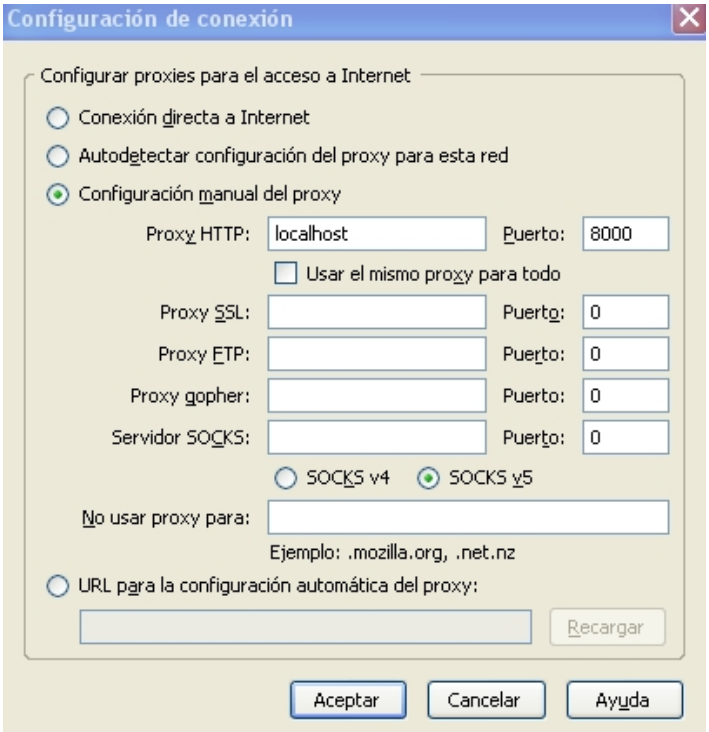

Figura 2.2.2.3 Configuración del navegador Firefox 2

Una vez realizados estas configuraciones, al ejecutar el comando Java explicado en el apartado 2.2.1, el filtro Web estará preparado para poder recibir las peticiones del navegador. En los apartados siguientes se concretará como se realiza todo el proceso de filtrado hasta que la página Web es servida al navegador.

2.2.3. FUNCIONAMIENTO DE LA APLICACIÓN

El filtro Web es un programa implementado en Java mediante la utilización de sockets. Esta aplicación se comporta como un Proxy a través del cual pasan todas las peticiones de páginas Web que realizan los usuarios hacia Internet. Este filtro no permite el acceso a páginas con contenidos inadecuados o a aquellas que están incluidas en una lista de páginas prohibidas.

El usuario realiza una petición de página Web a través del explorador, esta petición la recibe el Proxy que esta escuchando a través del puerto 8000. Se procede al análisis de la dirección Web que el usuario ha solicitado, se comprueba que la página no está en denegar.txt. Nos podemos encontrar ante dos situaciones:

1- www.sex.com: La página esta presente en denegar.txt: Esto significa que cuando se intentó acceder a la página por primera vez en un periodo anterior, el análisis de las palabras que en ella aparecen fue negativo, es decir, contiene palabras que están presentes en el diccionario de denegación.

   Se servirá la página Web PROHIBIDO.HTM que indica el acceso a un recurso con contenidos inadecuados.

2- www.uoc.es: La página no esta presente en denegar.txt. Puede tratarse de una página que contenga contenidos para adultos, en tal caso será nuestro primer acceso; o bien ser una página sin contenidos prohibidos. Esta diferenciación se realizará en una etapa posterior.

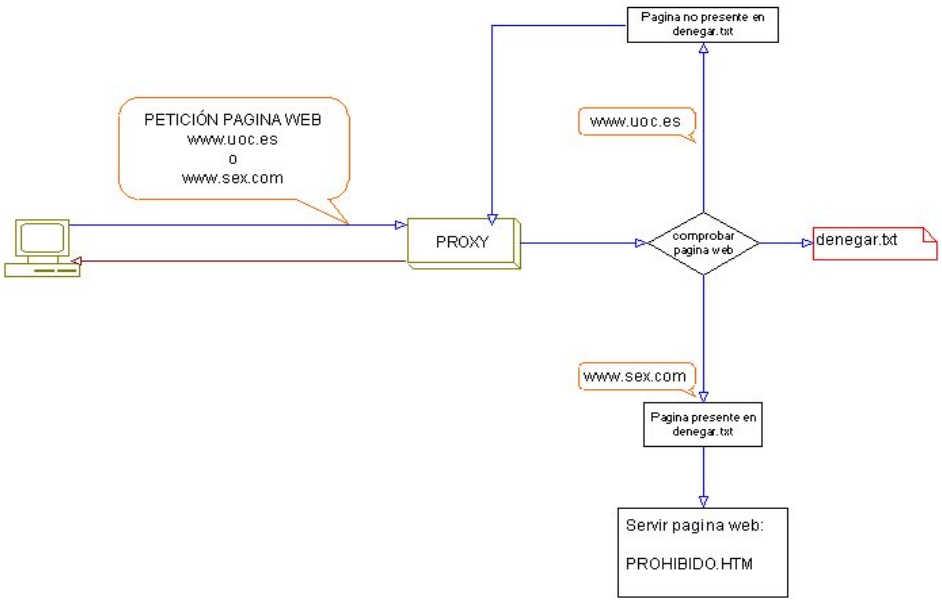

Figura 2.2.3.1 Análisis de la dirección Web

Una vez comprobado que la página no se encuentra en denegar.txt, el Proxy realizará la petición de pagina al servidor real, si la petición es correcta el servidor enviará el código Html de la misma. Este código se recogerá en el archivo paginaHTML.txt para su posterior análisis de las palabras que aparecen en la página mediante un diccionario. Este proceso comprueba que cada palabra que hay en la Web servida no coincide con ninguna de las palabras contenidas en el diccionario que con anterioridad se ha creado.

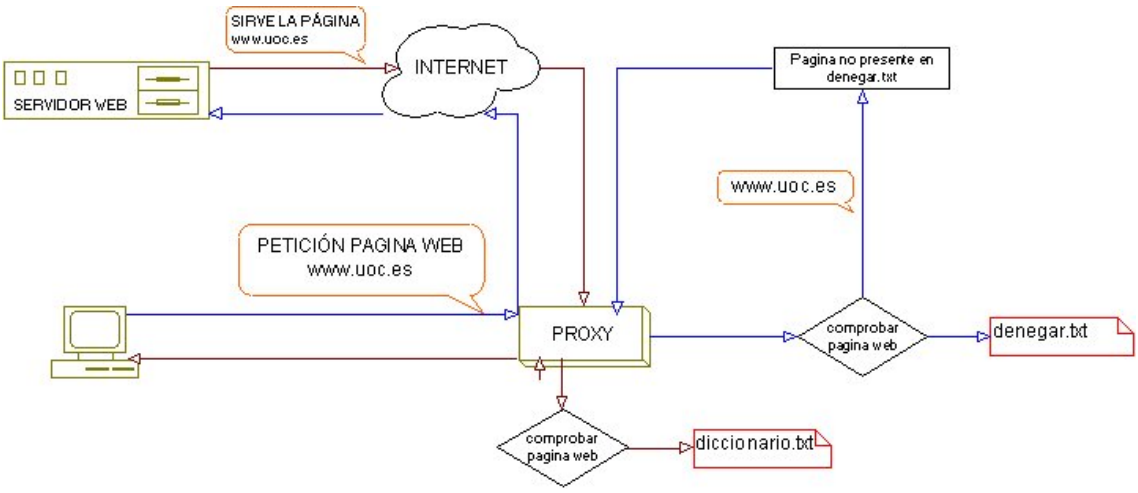

Figura 2.2.3.2 Petición de página Web

En este punto nos podemos encontrar en dos situaciones diferentes:

2.1- www.uoc.es: El análisis demuestra que no hay palabras que puedan representar vulneración del vocabulario incluido en el diccionario. En este caso el filtro Web presenta un resultado positivo, por tanto la página se transmitirá al navegador para que el usuario pueda consultarla.

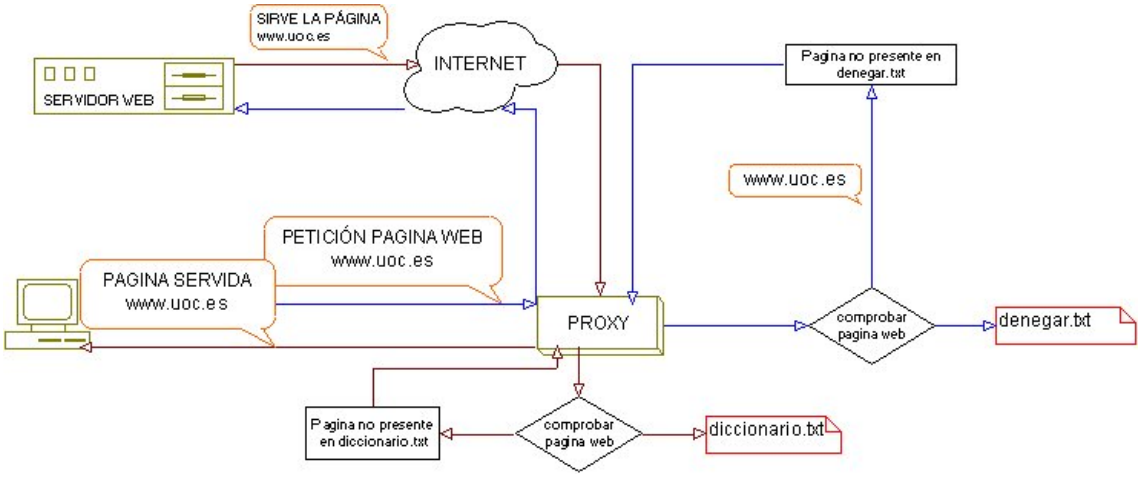

Figura 2.2.3.3 Transmisión de página Web solicitada al navegador

2.2- www.adultos.com: En este caso el resultado de la comprobación es negativa. Se han encontrado coincidencias entre las palabras que presenta la página y las que contiene el diccionario.

Se servirá la página Web PROHIBIDO.HTM que indica el acceso a un recurso con contenidos inadecuados.

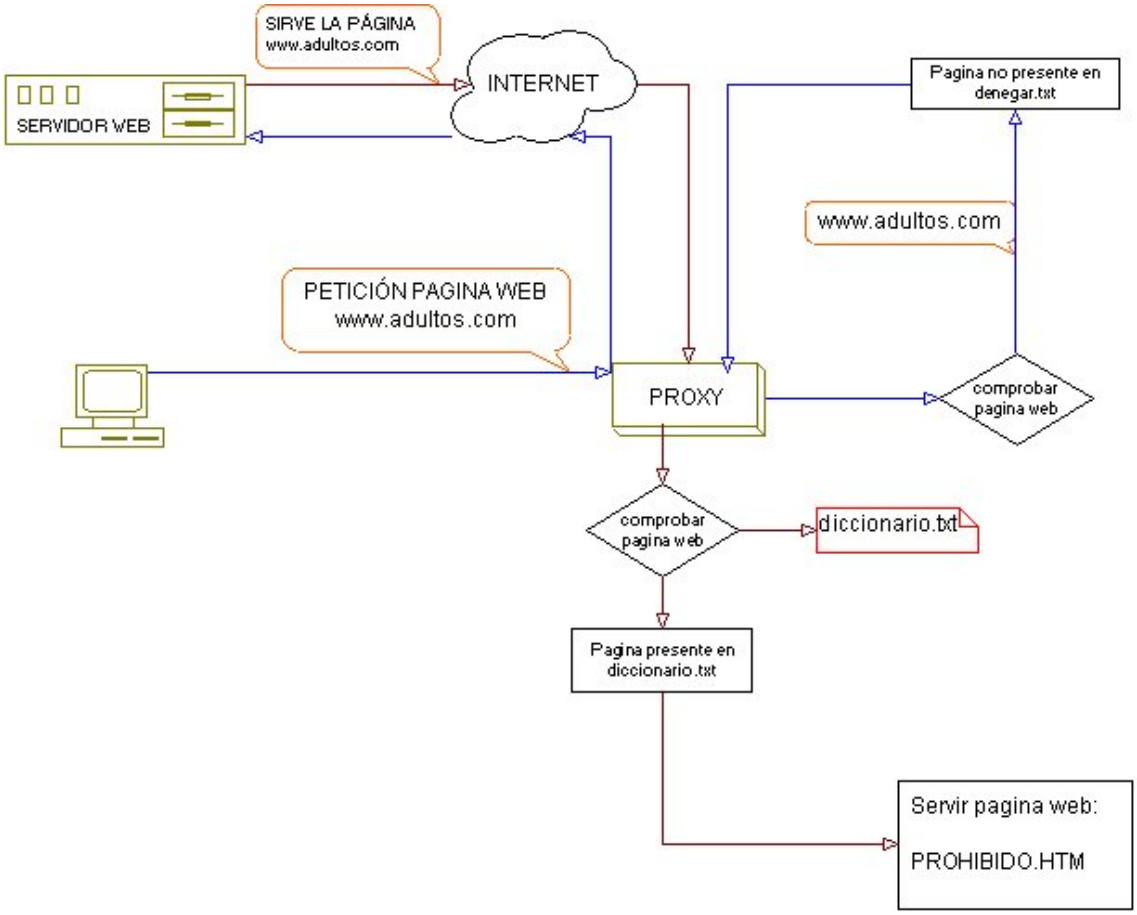

Figura 2.2.3.4 Transmisión de página Web de prohibición al navegador

Cuando se deniega una petición, se debe servir una página Web que sustituya a la solicitada. Esta página Web es genérica y especifica que se ha intentado acceder a un recurso que contienen contenidos inadecuados. El modelo que se utilizará será el siguiente:

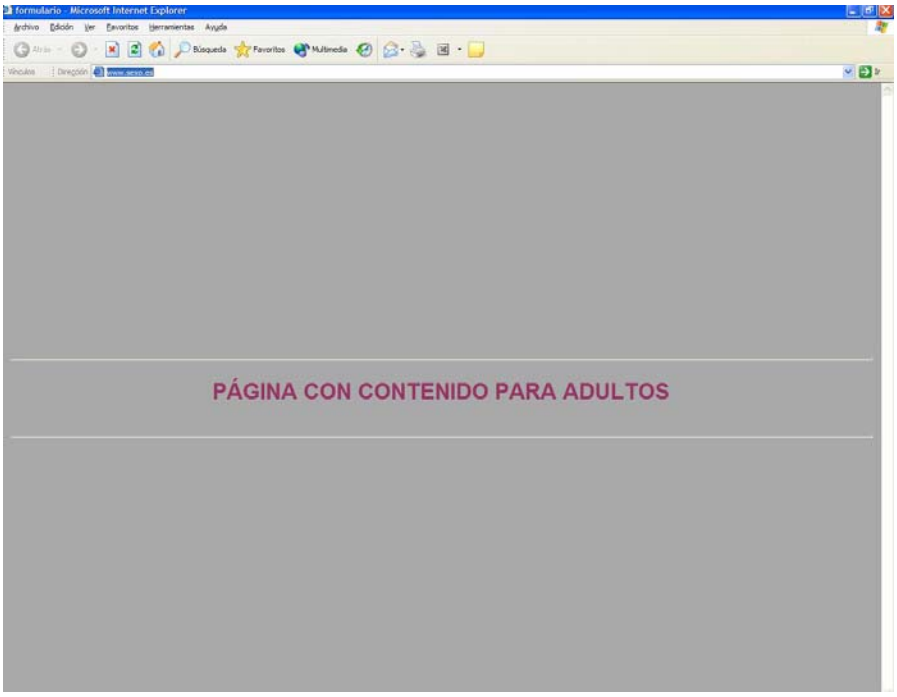

Figura 2.2.3.5 Página Web de prohibición

En el siguiente diagrama se puede observar un esquema de todo el proceso.

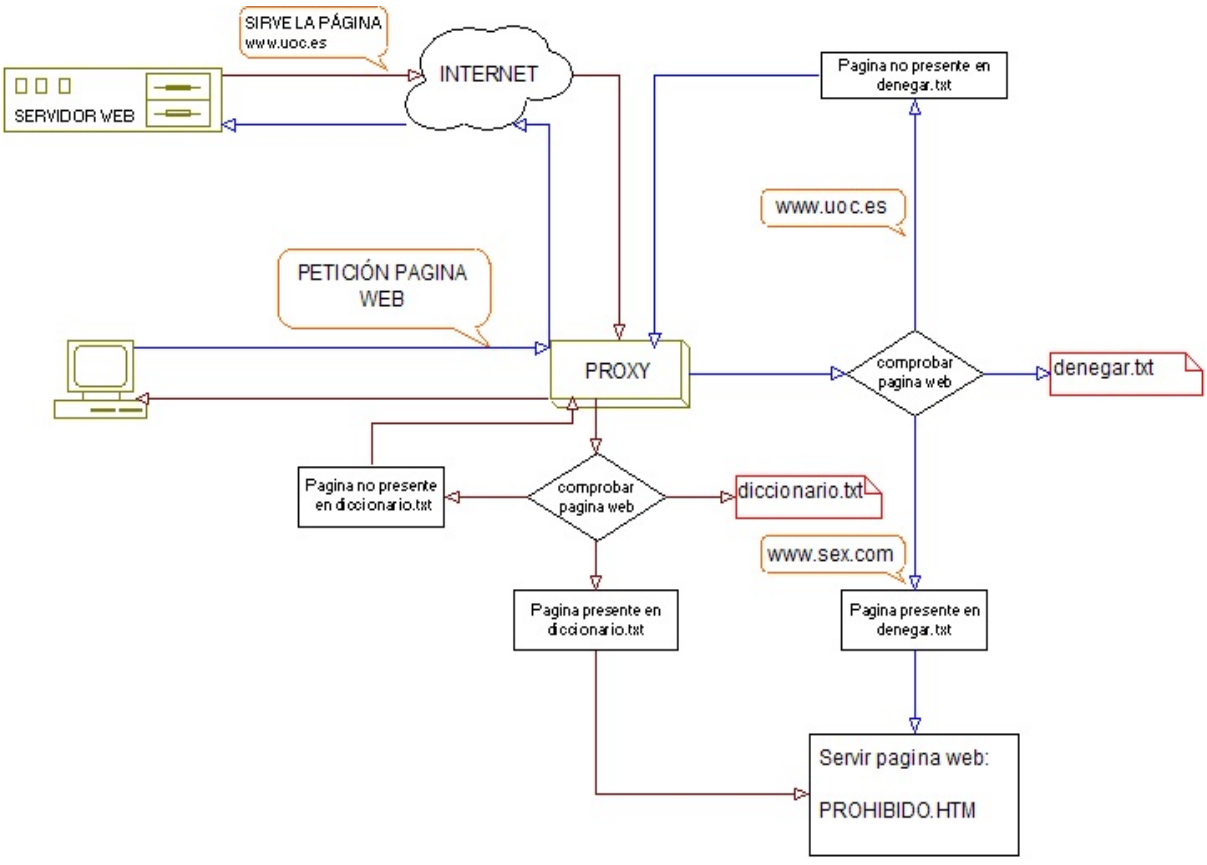

Figura 2.2.3.6 Proceso completo de filtrado Web

2.3 PROBLEMATICAS DE LA IMPLEMENTACIÓN Y SOLUCIONES PRESENTADAS

- Comunicación de Sockets: (Método Flush)

  En el desarrollo de la aplicación he encontrado especial dificultad para sincronizar la comunicación entre el cliente y el servidor Proxy. Como se comenta en el apartado 2.1.3 el Proxy basa su funcionamiento en la utilización de sockets (clase socket i serverSocket), que son clases de java que no presentan gran dificultad. Pero en el momento de programar el Proxy, comprobaba como aunque escuchaba las peticiones del navegador no podía servir la página en determinadas ocasiones. Analizando la documentación y la información disponible en Internet, no lograba encontrar la solución al problema. Me parecía que Lo más lógico era que no se sirvieran nunca las página. En realidad el problema se producía porque después de realizar un flujo de escritura no se limpiaba el flujo, esto provocaba que los datos de diferentes escrituras se mezclasen y el navegador no pudiese interpretar la información.

- Protocolo http 1.0 y 1.1

  La segunda problemática que surgió se debía a la incorrecta configuración del navegador. Es preciso configurar el navegador para que el Proxy funcione con el protocolo http 1.1 ya que la conexión TCP/IP es permanente, en cambio con la versión 1.0 se transmiten los datos por partes. Al tener configurado el navegador para que el Proxy utilice el protocolo 1.0 el tiempo de respuesta es muy elevado y en algunos servidores que reciben muchas peticiones se podría exceder el tiempo de conexión y el Proxy cerraría la conexión.

- Método readline() y trim()

  Cuando realizaba lecturas de archivos y la variable era un string, en el momento de comparar dos strings iguales con el metodo equals() la respuesta era negativa. El error se debía a que en la lectura, los espacios en blanco también los considera como parte del string y en el momento de comparar con otro string no los interpretaba como iguales,

por tanto después de cada lectura desde archivo es preciso utilizar el método trim() para eliminar los espacios en blanco.

- Filtrado de Imágenes

  Como se indica en introducción el filtrado que se implementa es de palabra clave, a pesar de eso, se ha intentado analizar como realizar un filtrado de las imágenes dado que algunas páginas Web están realizadas sin texto, simplemente con la composición de diferentes imágenes. En estas páginas el filtro es totalmente ineficaz. Dado que el tiempo marcado se ha visto superado, ha sido imposible realizar este filtrado. En contraposición se ha introducido un método para impedir el acceso a las páginas que se recogen en el archivo denegar.txt. Únicamente es necesario introducir la URL de estas páginas en el archivo y el filtro impide el acceso. Este filtrado se denomina filtro de exclusión y se comenta su funcionamiento en el apartado 1.1.

He intentado recoger los principales problemas que se han presentado en el momento de la implementación de la aplicación. No quiere esto decir que sean los aspectos más complicados que aparecen en el programa sino aquellos que me han supuesto una mayor dedicación, a causa de la gran cantidad de métodos que proporciona el API de java.

Analizando la dificultad de interpretar el código a causa de su gran extensión se han documentado destalladamente todos los métodos y se ha separado cada proceso en la programación mediante comentarios aclaratorios para la correcta interpretación de la misma.

2.4 POSIBLES MEJORAS DE LA APLICACIÓN

El filtro elaborado, tal y como se define en las especificaciones de la aplicación, únicamente escucha del puerto 80, es decir se trata de un filtro para el protocolo http. Por este motivo cuando se desea realizar una comunicación cifrada, usando el protocolo https(SSL) a través del puerto 443 no permite la conexión.

Inicialmente la necesidad era crear un filtro exclusivamente para el visionado de páginas Web y viendo el tiempo que se necesitaba, desde un principio se descartó la posibilidad de hacer extensible el filtro a otros protocolos. Una de las posibles ampliaciones sería contemplar esta deficiencia y ampliar la funcionabilidad de la aplicación.

Dado que la finalidad era que la aplicación se utilizase en centros escolares, no se cree necesario considerar otros protocolos como FTP,SMTP, IRC…; ya que el uso que se realiza de las tecnologías de la comunicación se basa en la consulta de información a través de Internet.

Otra aspecto a considerar, tal y como se comenta en el apartado anterior, es buscar un mecanismo para poder filtrar imágenes para adultos. Aunque se ha consultado información relativa al tema ha sido imposible implementar ninguna solución.

Por último sería interesante mejorar la velocidad de análisis del filtro, aunque actualmente no es relativamente importante dado la velocidad de transmisión de datos de las redes ADSL, hay métodos de búsqueda más efectivos que el secuencial.

Estos son los aspectos a tener en cuenta en próximas modificaciones atendiendo al objetivo inicial y que el alumno tienen intención de proveer para conseguir un producto final más eficiente.

2.5 SEGUIMIENTO DE LA PLANIFICACIÓN INICIAL

Mirando la planificación inicial en el apartado 1.5 vemos que se ha tratado de elaborar la memoria al mismo tiempo que se avanzaba en la programación de la aplicación.

En un principio se ha seguido esta planificación, hasta que han empezado ha surgir las dificultades que se comentan en el apartado 2.3. A partir del 27 de noviembre en que tenía que estar terminado el Proxy se ha descompensado toda la programación. Además cuando se inicio el trabajo se desconocían algunos aspectos que después de realizar el estudio de la documentación han influido en el desarrollo del proyecto.

Se comento la posibilidad de elaborar un filtro de exclusión basado en las IP que en un principio estaba pensado para el último periodo del proceso, pero después de estudiar el funcionamiento de los Sockets, se vio que representaba una tarea sencilla de implementar al mismo tiempo que se programaba el Proxy. Únicamente que no se buscaba la ip para filtrar, sino que se utilizaba la URL, que a efectos prácticos es lo mismo.

Indicar que este representaba el primer proyecto de larga duración para el alumno y que en el momento de elaborar no se contaba con la experiencia para considerar todos estos aspectos, por este motivo se actuó con cautela y se fijo la finalización del proyecto para el día 18 diciembre, proveyendo que surgirían dificultades que retrasarían la planificación.

A pesar de todo, se finalizó el proyecto el día 4 de enero del 2006 con un retraso aproximado de un mes, habiendo conseguido los objetivos fijados y con un resultado que ha sido plenamente satisfactorio para el alumno, especialmente por el aprendizaje realizado, tanto sobre el tema tratado como en los procedimientos y la metodología adquirida en el desarrollo y planificación del proyecto.

3. CONCLUSIONES

Después del estudio de las diferentes posibilidades de filtrar los contenidos Web, he llegado a la conclusión que actualmente no existen métodos totalmente eficaces. El filtro que se ha implementado además de no se efectivo ante todas las páginas Web, ralentiza la comunicación cliente/servidor. Además los diseñadores de páginas Web cada vez hacen más difícil que estos filtros sean plenamente eficaces.

Como se comenta con anterioridad, cuando se realiza una petición Web el servidor responde con una cabecera que indica una serie de información. Aquí es donde realmente se debería indicar los contenidos, al menos en páginas que pueden resultar dañinas para los menores. Aunque actualmente esto parece totalmente inviable ya que supone una vulneración de la libertad de expresión.

Dada la evolución que se ha producido en las tecnologías de la información y viendo que supone una nueva metodología de aprendizaje para nuestros menores, tendremos que dotar de los mecanismos necesarios para que esto no suponga un perjuicio más que un beneficio.

Este proyecto ha supuesto una forma de plasmar todo lo que se ha aprendido a lo largo de mi carrera universitaria y enfrentarse a un problema real. La valoración a nivel personal es positiva por lo que ha significado su realización y por el esfuerzo que me ha supuesto. Mi deseo es que realmente pueda ser un trabajo que tenga una utilidad en el futuro.

4. GLOSARIO

Cabecera: La parte de la petición y la respuesta HTTP que se envía antes del contenido propiamente dicho, y que contiene meta-información describiendo el contenido.

Cookie: Son pequeños archivos de texto que una página Web coloca en su ordenador, que le identifican como cliente.

Filtro: Un proceso que se aplica a la información que es enviada o recibida por el servidor.

HTML: Hyper Text Markup Language: Sistema para estructurar documentos que pueden ser mostrados por los visores de páginas Web en Internet.

ISP: Es el acrónimo en inglés de Internet Service Provider (Proveedor de Servicios de Internet), empresa dedicada a conectar a Internet la línea telefónica de los usuarios, redes distintas e independientes, ambas.

JAVA: Es un lenguaje de programación especialmente diseñado para el uso distribuido del ambiente Internet. Fue diseñado para tener la sensación y apariencia del lenguaje C++.

MIME-type: Una manera de describir el tipo de documento a ser transmitido. Consiste en dos componentes, uno principal y otro secundario, separados por una barra. Por ejemplo: text/html.

Proxy: Un servidor intermedio que se pone entre el cliente y el *servidor de* origen.

SSL: (Secure Sockets Layer) Es un protocolo diseñado por la empresa Netscape Communications, que permite cifrar la conexión, incluso garantiza la autenticación.

Streaming: Tecnica que permite cargar contenidos multimedia sin necesidad de esperar a que estos se descarguen al disco duro por completo. Consiste en descargar cierta cantidad de información para permitir su iniciación, y mientras nosotros visualizamos ese medio, este sigue descargándose.

TCP: Es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales enviarse datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

URL: Es la cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet.

5. BIBLIOGRAFIA

Filtro de contenidos:

- http://www.informatica-juridica.com/trabajos/problema_contenidos.asp
- http://www.uoc.edu/web/esp/art/uoc/0109038/casacub.html
- http://www.albanet.com.mx/articulos/FILTROS.htm
- http://www.besafeonline.org/spanish/filtros.htm
- http://www.portaley.com/usuario/revista20050331.shtml

Proxy:

- http://www.lasalle.es/benicarlo/monograficos/ired.html

Sockets y java:

- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/V_2.htm
- http://www.htmlweb.net/articulos/sock_java.html
- http://www.infor.uva.es/~cllamas/sd/Curso2001-2002/Extra.htm
- http://usuarios.lycos.es/galadeyahoo/taller%20web/sockets_java.doc
- http://www.geocities.com/chuidiang/java/sockets/socket.html
- http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte20/cap20-10.html
- Janie Jaworski. Java 1.2 al descubierto, ed. Pretince Hall.
- Judy M. Bishop. Java. Fundamentos de programación, ed. Adison Wesley.

Protocolo HTTP:

- http://cdec.unican.es/libro/HTTP.htm
- http://www.lfcia.org/openprojects/camllets/doc/html/node16.html
- http://www.frlp.utn.edu.ar/materias/internetworking/apuntes/http/HTTP-4.pdf
- Jordi Íñigo Griera. Xarxes. Aplicacions o protocols d'internet, UOC, Septiembre 2003

6. ANEXOS

6.1. DESCARGA E INSTALACIÓN DE JAVA

La empresa Sun Microsytems, creadora de java nos ofrece la posibilidad de descargarnos la última versión de java de manera gratuita desde la página Web http://java.sun.com/.

Lo primero que observamos es que se nos ofrecen tres versiones: J2SE, J2EE y J2ME. Para Pc de usuario nos descargaremos la versión estándar, J2SE conjuntamente con el API donde encontraremos toda la documentación técnica. En este momento la versión más actual es la 5.0 y la podemos encontrar en: http://java.sun.com/j2se/1.5.0/download.jsp.

Una vez descargado el SDK de java podemos instalarlo. Se trata de un archivo comprimido auto ejecutable que pulsándolo comienza su instalación. Durante el proceso nos preguntará si deseamos instalar java en los navegadores, aceptaremos para que nos actualice la máquina virtual java de los navegadores. Así mismo nos preguntará el directorio donde deseamos instalar, aceptaremos el que nos ofrece por defecto para facilitar el proceso de configuración.

Por último tendremos que configurar el sistema para que conozca donde se encuentra instalado java. Dependiendo del sistema vamos a ver como se establece la variable PATH en distintas versiones de windows:

- Windows 2000 , XP y NT:

Entramos en Inicio>Configuración>Panel de control y accedemos a sistema. Elegimos la pestaña Avanzado y buscamos la variable PATH en las variables de entorno. Tenemos que añadir la ruta de la carpeta bin de donde hemos instalado java, que si escogimos la ruta que venía por defecto será:

C:\j2SDK1.5.0_01\bin

Dentro de la variable PATH habrá varias direcciones separadas por punto y coma, añadiremos la anterior al final de la misma.

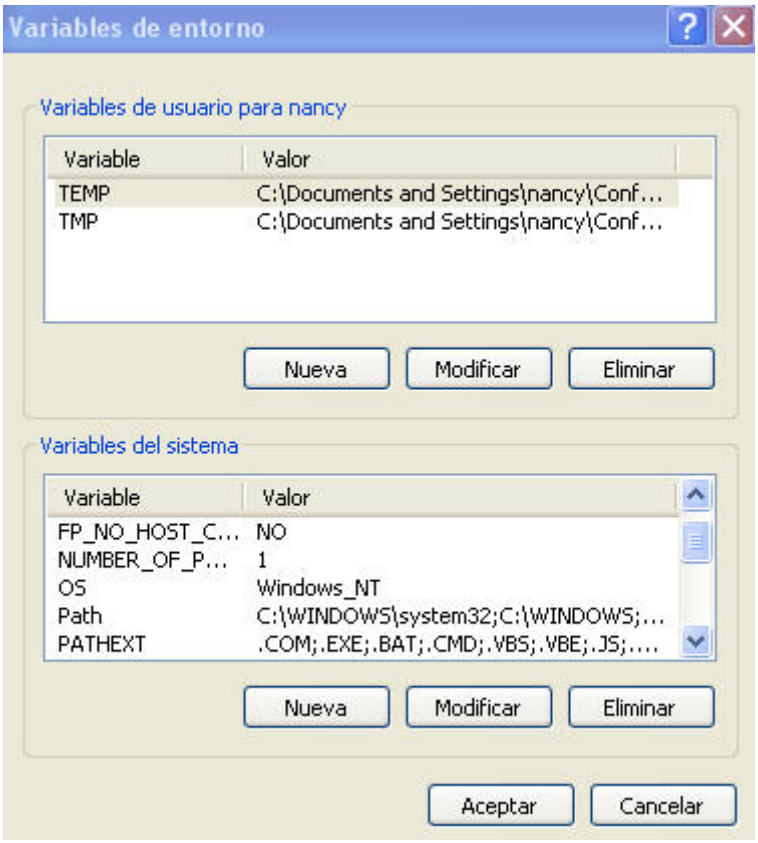

Figura 6.1.1 variables del entorno

Figura 6.1.2 Variable PATH

- Windows ME:

Entramos en Inicio>Programas>Accesorios>Herramientas del sistema> Información del sistema. Elegimos el menú herramientas y seleccionamos las herramientas de configuración del sistema. Pulsamos la pestaña entorno y aquí podemos editar la variable PATH. Como realizamos anteriormente se añade la ruta de la carpeta bin de donde hemos instalado java, que será:

C:\j2SDK1.5.0_01\bin

6.2. DOCUMENTACIÓN DE LA APLICACIÓN

6.2.1 CLASE SERVIDOR

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

Class Servidor

```
java.lang.Object
  |
  +--Servidor
```

---

public class **Servidor**
extends java.lang.Object

Constructor Summary

`Servidor()`

Method Summary

`static java.lang.String`	`denegacion(java.lang.String getHost)` Parametrode entrada: dirrección web
`static void`	`log(java.lang.String direccionWEB)` Parametrode entrada: dirrección web
`static void`	`main(java.lang.String[] args)`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Servidor

```
public Servidor()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

denegacion

`public static java.lang.String` **`denegacion`**`(java.lang.String getHost)`

Parametrode entrada: dirrección web. * Devuelve noPermitir si la pagina está en denegar.txt * y permitir si la pagina no está. *

---

log

`public static void` **`log`**`(java.lang.String direccionWEB)`

Parametrode entrada: dirrección web. * Crea un historial de las paginas visitadas * anotando su dirección, hora y fecha de consulta * y la ip desde la que se consutó *

6.2.2 CLASE PETICIONHTTP

Package **Class** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

Class PeticionHttp

```
java.lang.Object
  |
  +--PeticionHttp
```

---

public class **PeticionHttp**
extends java.lang.Object

Constructor Summary

`PeticionHttp``()` Constructor con parámetros
`PeticionHttp``(java.lang.String peticion)` Constructor a partir de un String que es la linia de petición
`PeticionHttp``(java.lang.String pMetodo, java.net.URI pURI, java.lang.String pVersion)` Constructor con parámetros:

Method Summary

`java.lang.String`	**`getHost`**`()`
`java.lang.String`	**`getMetodo`**`()`
`java.lang.String`	**`getPath`**`()`

int	**getPort**()
java.net.URI	**getUri**()
java.lang.String	**getVersion**()
java.lang.String	**peticionProxy**()
void	**setMetodo**(java.lang.String pMetodo)
void	**setUri**(java.net.URI uri)
void	**setVersion**(java.lang.String pVersion)
java.lang.String	**toString**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

PeticionHttp

```
public PeticionHttp()
```

Constructor con parámetros

PeticionHttp

```
public PeticionHttp(java.lang.String pMetodo,
                    java.net.URI pURI,
                    java.lang.String pVersion)
```

Constructor con parámetros:

Parameters:

`pURI` - - java.net.URI nombre del recurso URI

`pVersion` - - String versión del protocolo

PeticionHttp

```
public PeticionHttp(java.lang.String peticion)
```

Constructor a partir de un String que es la linia de petición

Method Detail

getMetodo

```
public java.lang.String getMetodo()
```

Returns:
- String

---

getUri

```
public java.net.URI getUri()
```

Returns:
- java.net.URI

---

getVersion

```
public java.lang.String getVersion()
```

Returns:
- String

---

setMetodo

```
public void setMetodo(java.lang.String pMetodo)
```

---

setUri

```
public void setUri(java.net.URI uri)
```

Parameters:
`uri` -

---

getHost

```
public java.lang.String getHost()
```

---

getPath

```
public java.lang.String getPath()
```

---

getPort

```
public int getPort()
```

---

peticionProxy

```
public java.lang.String peticionProxy()
```

---

setVersion

```
public void setVersion(java.lang.String pVersion)
```

---

toString

```
public java.lang.String toString()
```

Overrides:

`toString` in class `java.lang.Object`

Returns:

- String

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

6.2.3 CLASE FILTRO

Package **Class** Tree Deprecated Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

Class filtro

```
java.lang.Object
  |
  +--filtro
```

---

public class **filtro**
extends java.lang.Object

Constructor Summary

`filtro()`

Method Summary

`boolean`	`getFiltro()`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

filtro

```
public filtro()
```

Method Detail

getFiltro

```
public boolean getFiltro()
```
