Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

DESARROLLO DE UN PRECLASIFICADOR DE CONTENIDOS BASADO EN HASHES DIFUSOS PARA UNA PLATAFORMA DE INTERCEPCIÓN LEGAL DE COMUNICACIONES

Autor: D. Roberto Díaz Ramos

Tutor: D. Manuel Urueña Pascual

Leganés, Abril de 2012

Título: DESARROLLO DE UN PRECLASIFICADOR DE CONTENIDOS BASADO EN HASHES DIFUSOS PARA UNA PLATAFORMA DE INTERCEPCIÓN LEGAL DE COMUNICACIONES

Autor: Roberto Díaz Ramos
Director: Manuel Urueña Pascual

EL TRIBUNAL

Presidente: Iría Estévez Ayres

Vocal: Alfonso Muñoz Muñoz

Secretario: Marcelino Lázaro Teja

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 27 de Abril de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero dar mis agradecimientos a:

mis tutores, Manuel Urueña Pascual y Carlos Garcimartín, por orientarme en las ideas que se me ocurrían para la realización del proyecto fin de carrera (PFC),

mis profesores y compañeros de la Universidad Carlos III de Madrid (UC3M), por la formación recibida, tanto en el ámbito ingenieril, como en el personal,

mis padres, Daniel y Julia, por la educación recibida, de hecho, de no ser por ellos, muy probablemente, no podría ni haber comenzado ni terminado esta carrera universitaria,

mi novia, Laura por el apoyo recibido y la compresión tenida.

Nunca andes por el camino trazado, pues te conducirá únicamente hacia donde los otros fueron.

Alexander Graham Bell

Resumen

El presente proyecto intenta dar solución a un problema que ha surgido debido al auge de internet como canal de distribución de información. Todos los usuarios que utilizan internet se están intercambiando información continuamente, pero esta información no es siempre legal. Existen usuarios que se intercambian tanto contenidos lícitos como delictivos (e.g. pornografía infantil), pero para poder detectar este contenido y poder perseguir legalmente a estos usuarios, la normativa actual sobre la intercepción legal de las comunicaciones establece que es un juez quien debe ordenar dicha intercepción e indicar el delito por el que es investigado un sospecho. Una vez que un juez autoriza la intercepción legal de las comunicaciones del sospechoso, los analistas de las fuerzas de seguridad se pueden encontrar con el problema de que el sospechoso se haya intercambiado una cantidad inmanejable de datos de toda índole, por lo que el análisis y clasificación por parte de los analistas puede llevar un tiempo y unos medios de los cuales no dispongan los cuerpos de seguridad.

Por este motivo vemos la necesidad de implementar un sistema de pre-clasificación de contenidos ilícitos, para facilitar al analista el trabajo de la clasificación. Cabe destacar que debido a consideraciones legales y prácticas es el analista el último responsable sobre la calificación de los contenidos interceptados.

La forma habitual para realizar la pre-clasificación de contenidos es la utilización de listas de firmas de contenidos ilegales incautados con anterioridad y calculadas mediante hashes criptográficos. Sin embargo para eludirlos basta con cambiar mínimamente el contenido. Para evitar esos posibles intentos de engañar al sistema de detección por parte de los sospechosos, el pre-clasificador está implementado mediante un sistema que permite detectar contenidos similares, Fuzzy hashing. De esta forma se intenta evitar que una pequeña modificación en el contenido haga fallar al pre-clasificador. Si el pre-clasificador estuviera implementado mediante hash criptográficos no podríamos detectar contenidos similares entre sí, ya que el objetivo de los hash criptográficos es encontrar ficheros idénticos.

Adicionalmente también se ha considerado necesario dotar al fuzzy hashing implementado, de una clave de seguridad para el cálculo de los hash de los contenidos. De esta forma la firma generada tiene mayor seguridad ante posibles ataques por parte de los sospechosos, que se adapten a las técnicas de Fuzzy hashing.

Este trabajo se enmarca dentro del proyecto de investigación Europeo ‘INDECT’ de forma que el sistema desarrollado pueda ser integrado con la plataforma de interceptación legal de datos sobre redes IP que está desarrollando la UC3M, y que está basada en la herramienta forense Xplico.

Palabras Clave: Fuzzy Hashing, intercepción legal de las comunicaciones, Xplico, INDECT.

Abstract

This Project tries to solve a problem arisen from increase of internet as information distribution channel. All internet users are continuously exchanging information, but this information is not always legal. There are internet users that exchange whether legal content or illicit (i.e child pornography), however in order to detect this content and legally pursue these users , current regulations about legal interception of communications state that should be a Judge who must order such interception and show the crime because a suspicious is being investigated. Once the Judge authorizes legal interception of suspicious people communications, security forces analyst can find the problem that suspicious people has exchanged an enormous quantity of data of whichever nature, therefore analysis and classification tasks to be perform by analysts can spend time and means nor available by security forces.

That is the reason why we see the necessity to implement a system for pre-classification of illicit content, in order to facilitate to analyst the classification tasks. Emphasize that due to legal and useful consideration, the analyst is the final responsible about content interceptions qualification.

Habitual way to perform content pre-classification is using signature list about illicit content previously confiscated and calculated by means of hashes cryptographic. However for eluding that it is enough with minimum changes in content. To avoid those possible attempt to fool the detection system by the suspicious people, pre-classification is implemented by means of a system that allow to detect similar contents, Fuzzy hashing. On this way, it tries to avoid that a small contents modification could make pre-classification fail. Whether pre-classification system were be implemented by means of hash cryptographic we can't detect similar contents, because of hash cryptographic aim is finding identical files.

In addition it has been considered necessary to equipped the Fuzzy hashing with a password to calculate the hash of the contents. On this way, password generated is securer against suspicious possible attack who adapted to the Fuzzy hashing techniques.

This piece of work is included on the European research project "INDECT" with the aim that developed system can be integrated in the data legal interception platform about IP networks that is developing UC3M and that it is based on forense Xplico tool.

Keywords: Fuzzy hashing. Legal interception of communication, Xplico, INDECT.

Índice general

Índice de figuras

Índice de tablas

Glosario

CSV	Comma Separated Values
DoS	Denial of Service
FH	Fuzzy Hashing
FNV	Fowler–Noll–Vo
FTP	File Transfer Protocol
GLP	GNU Public License
GNU	GNU No es Unix
HTTP	Hypertext Transfer Protocol
INDECT	Intelligent information system supporting observation, searching and detection for security of citizens in urban environment
IP	Internet Protocol
LS6B	least significant 6 bits
LZW	Lempel-Ziv-Welch
P2P	Peer-to-peer
PCAP	packet capture
PFC	Proyecto Final de Carrera
PHP	Hypertext Preprocessor
RGB	Rotating Graphics Base (Red, Gren, Blue)
RLE	Run-length encoding
UC3M	Universidad Carlos III Madrid
ZIP	Zigzag In-Line Package

Capítulo 1

Introducción y objetivos

1.1 Introducción y motivaciones

Internet es el canal más utilizado en la actualidad para el intercambio de información. Esto es debido a que nos proporciona información en tiempo real permitiendo consultar millones de páginas con información de lo más variada, compartir contenidos o comunicarse con otros usuarios situados a miles de kilómetros. Pero no siempre estas ventajas son utilizadas para actividades licitas, ya que hay usuarios que utilizan el potencial de Internet y el presunto anonimato que nos brinda, para realizar actividades delictivas mediante el intercambio de contenidos ilegales (imágenes pedófilas, manuales de terrorismo, etc.…).

Debido a la proliferación de este intercambio de contenidos ilícitos entre usuarios de todas las partes del mundo, surge la necesidad la interceptación legal de las comunicaciones, y en particular de poder diferenciar contenidos lícitos de los que no los son de forma efectiva.

Una orden judicial autorizará la intercepción de las comunicaciones de un sospechoso y bajo el delito que se le está siendo investigado, para posteriormente poder diferenciar los contenidos lícitos de los que no lo son (pre-clasificarlos) entre todos los interceptados.

Un sospechoso puede intercambiar muchísima información, y en esa información pueden existir tanto contenidos lícitos como ilícitos. Por lo que se considera necesario

poder pre-clasificar el contenido interceptado de forma automática y así el analista de las fuerzas de seguridad se pueda centrar en los contenidos más importantes, en este caso los contenidos relacionados con el delito en el que se le investiga. El responsable último de la valoración de la información interceptada es el analista del sistema, apoyándose en la pre-clasificación realizada de forma automática.

De esta forma el pre-clasificador solo alertará al analista de los contenidos semejantes a otros previamente identificados bajo el mismo delito. Ya que si se detectaran contenidos fuera del alcance de la orden judicial se podría llegar a cuestionar la validez de las pruebas obtenidas mediante este procedimiento.

Este Proyecto Final de Carrera (en adelante PFC) analiza los contenidos interceptados, para poder pre-clasificarlos. Para ello se tiene que desarrollar un sistema de pre-clasificación rápido, robusto, y que nos dé el menor número posible de falsos positivos y ningún falso negativo. Y posteriormente evaluar su efectividad.

Para pre-clasificar contenidos contaremos con una serie de listas negras de contenidos ilegales incautados previamente, con las que poder comparar la información interceptada. Estas listas estarán especializadas en las categorías de delitos en las que se van a analizar los contenidos interceptados.

En las listas negras aparecerá un identificador único del contenido, la importancia o gravedad del mismo para el analista que lo intercepto y lo calificó, y un comentario descriptivo del mismo. Existirán listas negras por cada categoría en la cual es ilícito dicho contenido.

Es muy importante que en las listas negras aparezca toda la información anterior, ya que, en ciertos casos, la fuerzas de seguridad no puede poseer contenidos ilegales para compararlos con otros interceptados. Siendo el identificativo único del contenido el método para poder compararlo con el resto de contenido interceptado.

Gracias a la existencia de estas listas negras el analista tiene una lista reducida de contenido ilegal sin tener la necesidad de visualizarlo. Y a su vez es útil para el pre-clasificador ya que comparando los contenidos interceptados con los existentes en estas listas podrá pre-clasificarlos dentro de una categoría u otra.

La manera más eficiente para identificar de forma univoca archivos es mediante la utilización de una firma o hash. Por lo que el identificador único que se buscaba será el resultado del hash calculado. En un primer momento se puede plantear para la pre-clasificación de contenidos la utilización de los denominados hash criptográficos, que ofrecen una forma eficiente de comparar dos contenidos.

Pero la comparación de contenidos por los resultados obtenidos mediante hash criptográficos plantea la problemática, de que el sospechoso puede intentar cambiar mínimamente el contenido para evitar su detección.

Los hash criptográficos son idóneos para decidir si dos contenidos son idénticos o no. Pero en el momento que se realiza una pequeña modificación en el contenido (contenidos homólogos), los hash criptográficos no son validos (una pequeña modificación produce una firma totalmente diferente, dado que su objetivo es

justamente ese, detectar si el contenido ha sido modificado). Por lo que el sistema a desarrollar debe tener las bondades de los hash criptográficos, pero además debe permitir componer su firma a partir firmas parciales del contenido, como es el caso de fuzzy hashing [15] y así tener cierta tolerancia a cambios. Mediante fuzzy hashing se pueden dar como resultado firmas similares para contenidos homólogos.

Actualmente existen herramientas que implementan fuzzy hashing o alguna variante de este, consiguiendo resultados tan eficientes como los hash criptográficos y además de ser tolerantes a modificaciones en el contenido. Gracias a la tolerancia a las modificaciones se puede indicar el porcentaje de similitud existente entre dos contenidos. Consideramos que la utilización de esta modalidad de hash es la indicada para implementar un pre-clasificador efectivo.

El presente PFC de pre-clasificación de contenidos, es uno de los módulos que va a desarrollar la Universidad Carlos III de Madrid (UC3M), para implementar una plataforma de intercepción legal de comunicaciones y dentro del proyecto de investigación Europeo INDECT (*Intelligent information system supporting observation, searching and detection for security of citizens in urban environment*).

INDECT, es un proyecto investigación para la mejora de la seguridad ciudadana financiado por la Unión Europea e iniciado en 2009 por la Plataforma para la Seguridad Nacional de Polonia, al que se le han ido agregando distintas cuerpos de seguridad de la Unión Europea y las más importantes universidades Europeas.

INDECT desarrolla herramientas para mejorar la seguridad de los ciudadanos y la protección de la confidencialidad de la información. Sus principales objetivos son:

- desarrollar un sistema de información inteligente para la detección automática de amenazas y el reconocimiento de la conducta criminal.

- desarrollar nuevas herramientas para facilitar las actividades diarias de los agentes de policía, incluyendo herramientas para la detección de amenazas en Internet. En este objetivo es el que se centra el presente PFC, analizando y desarrollando un sistema pre-clasificador de contenidos.

- desarrollar técnicas de protección de datos y privacidad en el almacenamiento y transmisión de datos.

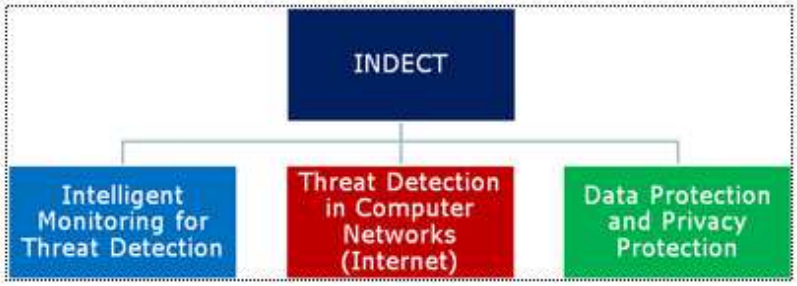


Figura 1. Objetivos del proyecto de investigación INDECT [6]

El proyecto INDECT resultará útil en investigaciones de las fuerzas de seguridad Europeas, tanto para entornos virtuales como reales:

- Pornografía infantil.
- Terrorismo
- Promoción de símbolos totalitarios.
- Tráfico de órganos humanos.
- Propagación de redes de bots, virus, malware.

Además también sirve para la detección de amenazas tales como:

- Abandono de equipajes.
- Obstáculos en lugares inadecuados (e.g. en una carretera).
- Incendios.
- Etc.

1.2 Objetivos

La plataforma de intercepción legal de comunicaciones que está desarrollando la UC3M y de la cual forma parte el presente PFC, se basa en la herramienta Xplico (Ver apartado 2.1 de la presente memoria), por lo que el sistema a desarrollar en este PFC debe ser fácil de integrar con Xplcio.

El sistema a desarrollar en este PFC debe ser capaz de pre-clasificar contenidos interceptados legalmente. Para ello se va a implementar un sistema de pre-clasificación basado en distintas tecnologías hash (fuzzy hashing y hash criptográficos) y evaluar el resultado obtenido.

Una vez evaluados los resultados obtenidos se establecerá un umbral a partir del cual se podrá determinar si dos contenidos son similares o no, este umbral está definido en el capítulo de Evaluación de la presente memoria.

Las características buscadas en el pre-clasificador son:

- Rapidez.
- Robustez.
- Menor número posible de falsos positivos.
- Ningún falso negativo.

Los resultados del módulo de firma serán cadenas de texto de longitud variable en función del tamaño del contenido, reduciendo drásticamente la probabilidad de falsos positivos, al depender la longitud de la firma del tamaño del contenido. El modulo a implementar debe evitar los falsos negativos, con el fin de evitar la posibilidad de que contenido similar al existente en la lista negra no lo pre-clasifique como tal y pueda pasar desapercibido en la investigación. Es mejor que el sistema lance un mayor número de falsos positivos que lance un falso negativo.

Actualmente existen dos herramientas que utilizan fuzzy hashing para el cálculo de su firma son ssdeep [2] y deeptoad [4]. El desarrollo del módulo se va a basar en la herramienta más efectiva de las mencionadas anteriormente, ya que esa es una de las premisas marcadas en la presente memoria.

Por último, una vez implementado el módulo de pre-clasificación se realizara la evaluación del mismo, comprobando su eficiencia ante cambios ocasionados en los contenidos y estudiando su viabilidad en casos prácticos

1.3 Esquema de la memoria

A continuación se muestra un breve resumen de cada capítulo presente en la memoria

2. Estado de la cuestión

En este capítulo se encuentran las tecnologías utilizadas y en las que se ha basado el desarrollo del sistema.

3. Diseño

En este capítulo se puede encontrar los casos de uso de la aplicación así como el diagrama de arquitectura de la misma.

4. Implementación

En este capítulo contiene una descripción detallada de la aplicación implementada junto con sus diagramas de flujos.

5. Evaluación

En este capítulo se puede encontrar el estudio de evaluación realizado con distintos contenidos en el módulo desarrollado y la efectividad del pre-clasificador implementado.

6. Presupuesto

En este capítulo listan los costes directos e indirectos con los que ha contado el presente PFC

7. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se puede encontrar las conclusiones obtenidas tras el desarrollo del presente PFC y posibles trabajos futuros a implementar a partir de este PFC.

Anexos

Este Anexo detalla las instrucciones de instalación de la herramienta en cualquier servidor web y la configuración del mismo.

Capítulo 2

Estado de la cuestión

Para el desarrollo del sistema de pre-clasificación y su posterior evaluación se ha utilizado tecnologías muy dispares, desde el lenguaje de programación java y entorno de desarrollo en eclipse hasta el estudio y uso de distintos tipos de hash, pasando por el uso de Apache o SQLite. En este apartado se describen dentro de las tecnologías utilizadas, las más importantes para el desarrollo del módulo.

La plataforma de intercepción legal de comunicaciones que está desarrollando la Universidad Carlos III de Madrid y de la cual forma parte este PFC está basada en Xplico, como habíamos mencionado anteriormente. Este es el motivo por el cual, de todas las tecnologías utilizadas para su desarrollo, Xplico sea la considerada más importante ya que condicionará la elección de buena parte de las tecnologías empleadas.

Dentro de la plataforma de intercepción legal de comunicaciones que está desarrollando la UC3M existen otros PFCs que se están desarrollando en paralelo. Uno de ellos consiste en el desarrollo de un Gestor de módulos de pre-clasificación (Plugin Manager) que se integrará en Xplico y se comunicará con este y con otros módulos. Para facilitar la compatibilidad entre el Plugin Manager y los módulos a desarrollar dentro de la plataforma de intercepción legal de comunicaciones, todos ellos se desarrollaran en lenguaje de programación JAVA.

2.1 Xplico

Xplico es un proyecto GLP, que tiene por objeto decodificar el tráfico de red capturado mostrando datos útiles para el usuario a través de una interfaz gráfica web. Trata de abstraer al usuario de toda la información técnica proporcionada por otros analizadores de tráfico (*sniffers*) convencionales como: *wireshark*, *tcpdump*, etc. *Xplico* toma el tráfico capturado y reconstruye los protocolos de las aplicaciones, almacenado en un registro los contenidos con sus correspondientes características. Toda esta información se almacena en bases de dato del tipo SQLite, es por ello que nuestro módulo trabaje también con este tipo de base de datos.

La información capturada es presentada en un interfaz de usuario amigable desarrollada en CakePHP, para que el usuario pueda interpretar los datos de una forma más sencilla y visual. Ver Figura 3 de la presente memoria.

Xplico permite capturar información de la red en tiempo real mediante un *sniffer* que tiene integrado, o utilizar como entrada de datos ficheros de captura existentes, en formato PCAP. Este software extraerá toda la información relevante de ella, como puede ser:

- Imágenes transferidas por HTTP
- Ficheros descargados mediante FTP
- Contraseñas que viajen en plano
- Conversaciones de Chat
- Ficheros enviados a impresoras
- Sitios web o webmails utilizados.

El decodificador de tráfico que incorpora *Xplico* se compone de módulos llamados *Disectors* utilizados para extraer de los diversos protocolos de red, los datos del nivel de aplicación que son en los que realmente está interesado el usuario. Los módulos que actualmente se encuentran desarrollados o en desarrollo para *Xplico* se muestran en la siguiente figura, la cual ha sido extraída de la página oficial:

Dissector	Status	Note	Dissector	Status	Note
ARP	90%	—	IPP	90%	—
Radiotap	90%	—	PJL	90%	—
Ethernet	100%	—	NNTP	95%	—
PPP	90%	—	MSN	10%	—
VLAN	95%	—	IRC	85%	—
L2TP	70%	—	YAHOO	0%	—
IPv4	98%	—	GTALK	0%	—
IPv6	98%	—	EMULE	0%	—
TCP	95%	—	SSL/TLS	0%	with keys
UDP	100%	—	IPsec	0%	with keys
DNS	80%	—	802.11	60%	no encryp.
HTTP	100%	—	LLC	60%	—
SMTP	95%	—	MMSE	95%	over HTTP
POP	95%	—	Linux cooked	95%	SLL
IMAP	95%	—	TFTP	90%	—
SIP	80%	—	SNOOP	100%	Format
RTP	70%	—	PPPoE	90%	—
RTCP	60%	—	Telnet	90%	—
SDP	70%	—	WebMail	90%	—
FB chat	90%	—	Paltalk Exp.	60%	—
FTP	90%	—	Paltalk	90%	—

Figura 2. Protocolos de red soportados por Xplico [7]

Xplico está licenciado bajo la GNU General Public License lo cual quiere decir que podemos trabajar con ella abiertamente, ampliándola de manera sencilla, gracias a su diseño modular. El software lo podemos obtener de la página oficial: http://www.xplico.org/download

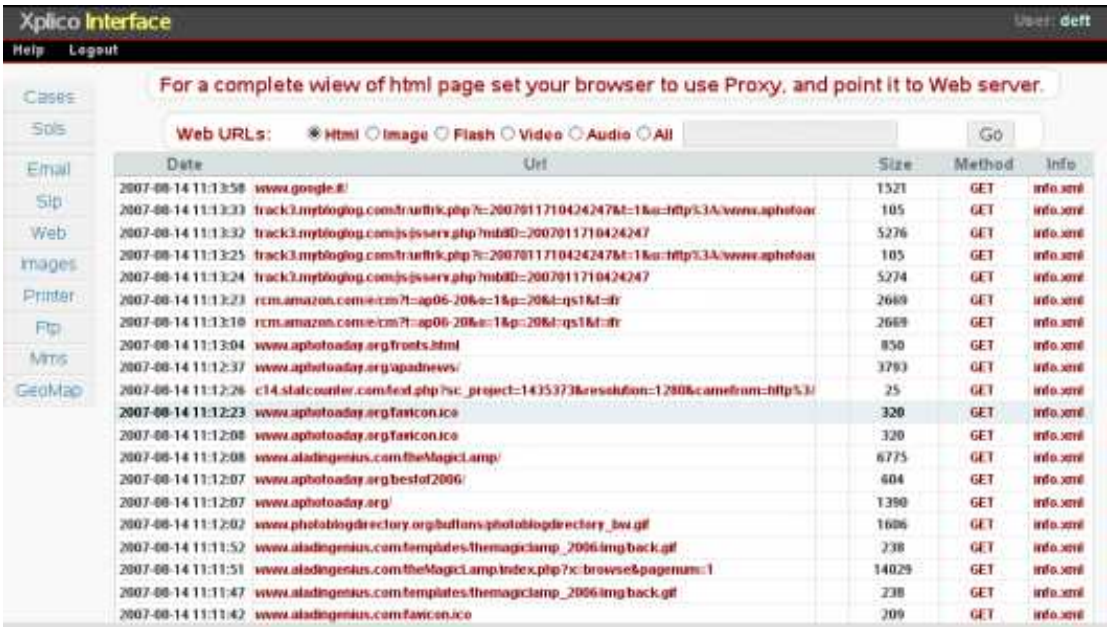


Figura 3. Interfaz web de Xplico [7]

2.2 CakePHP

En este proyecto se utiliza como interfaz de usuario, un interfaz Web programado en CakePHP Versión 1.3.10 (última versión estable), con la única finalidad de facilitar la integración de nuestro módulo de pre-clasificación en Xplico. Nos podemos descargar la última versión estable de CakePHP de la página del proyecto CakePHP [5].

CakePHP es un framework libre para el desarrollo en PHP, basado en el paradigma Modelo-Vista-Controlador. Permite al desarrollador centrarse en la lógica de negocio, ya que simplifica tanto la parte de representación como la de acceso a la base de datos.

- Permite una programación en PHP fácil y rápida.
- Solo se necesita configurar la Base de datos a utilizar y está listo para funcionar.
- Incorpora las librerías y configuraciones necesarias para ser utilizado sin una costosa configuración.

2.3 SQLite

SQLite es una base de datos SQL embebida en la aplicación que no requiere procesos externos ni una configuración compleja. Esta base de datos será utilizada para almacenar la información necesaria para el módulo (listas negras, cache…).

Además de por sus ventajas para los proyectos que no tiene demasiados requisitos de almacenamiento, como el nuestro, SQLite es la base de datos utilizada en Xplico

2.4 Fuzzy Hashing

El objetivo que perseguimos es ser capaces de detectar mediante firmas basadas en técnicas de hashing si dos contenidos son similares.

Como ya hemos visto, los hashes criptográficos tradicionales no sirven puesto que basta con cambiar un bit del contenido para que su firma sea completamente distinta al del original, y por lo tanto los sospechosos podrían evadirlos fácilmente.

Para entender cómo funcionan las técnicas de Fuzzy hashing vamos primero a intentar crear un mecanismo de comparación de contenidos similares basado en funciones de hash criptográficas convencionales. Por ejemplo si suponemos que el sospechoso sólo modifica un pequeña parte del contenido, se podría proponer para llegar a dicho objetivo dividir el contenido en partes iguales y calcular un hash criptográfico sobre cada una de ellas. Esta combinación hace que los hash calculados en ficheros

homólogos, sean similares, pero con muchas limitaciones, como se puede ver más adelante.

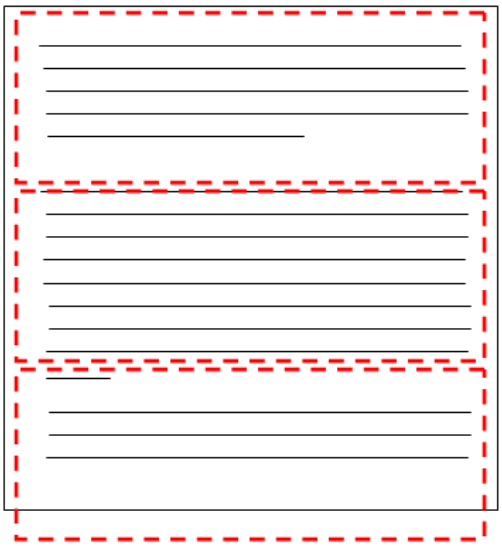


Figura 4. División de un contenido en bloques de tamaño fijo.

Una vez que se tiene el contenido dividido en partes de idénticas dimensiones, se podría realizar el cálculo de los hash de las distintas divisiones mediante el uso de hash tradicionales como puede ser **md5.** La firma resultante del contenido sería la combinación de las firmas parciales calculadas en las distintas partes en las que se ha divido el contenido de entrada.

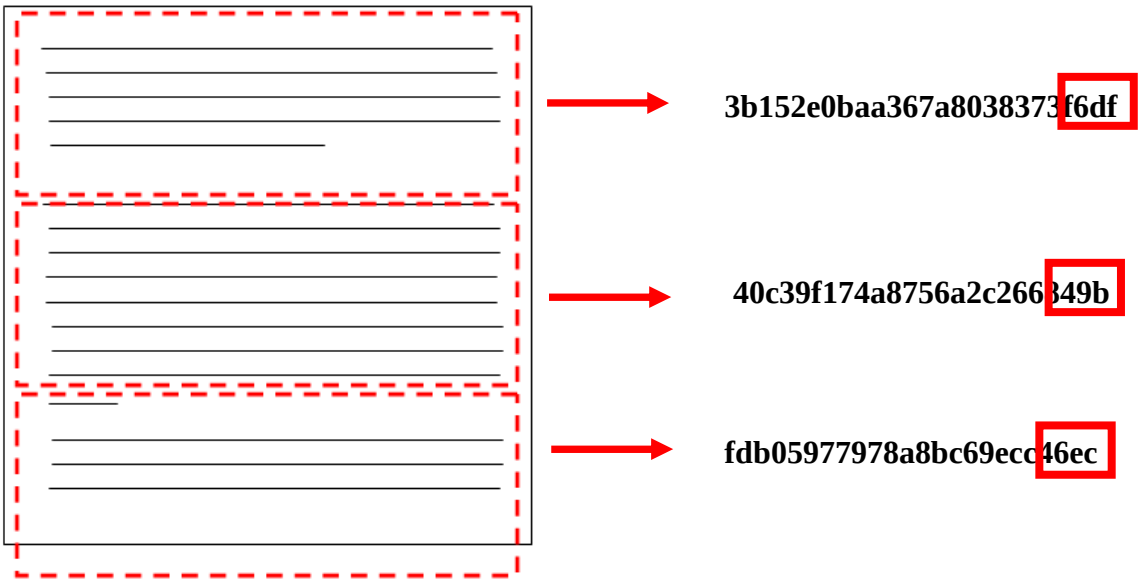


Figura 5. Calculo de las firmas de cada bloque.

Produciendo salidas de tamaño constante basadas sólo en el fichero de entrada. No es necesario coger toda la firma, basta con un pequeño fragmento que resuma cada parte. Un ejemplo de firma sería:

Hash: f6df 849b 46ec

Con este procedimiento se podrían identificar ficheros similares a ficheros conocidos. Pero con las siguientes carencias:

- El cálculo del hash md5 para cada uno de las partes en los que se divide el contenido de entrada supondría un coste computacional muy elevado.

- Las partes en las que se divide el contenido a analizar al ser de tamaño constante, hacen que ciertas modificaciones (e.g. introducir un carácter al principio del fichero) cambien la información de cada bloque, consiguiendo que todas las firmas parciales sean diferentes. Y de esta forma no se obtendría una firma similar a la del contenido original. Para el caso de supresión de un carácter el comportamiento sería análogo. Por lo que no es robusto ni ante la supresión ni la inserción de contenido.

- Sólo se pueden comparar archivos divididos en el mismo número de bloques. Si se cambia el tamaño del fichero, por ejemplo concatenándole basura, y el número de bloques es constante, la firma también cambiaría completamente.

Para evitar el problema de la supresión o inserción de contenido, fuzzy hashing [12] prescinde de los tamaños fijos de bloque, y procede a calcular el hash en busca de puntos de disparo pseudo-aleatorios que deciden cuándo acaba un bloque y empieza el siguiente. Este método se denomina Rolling Hash.

Rolling hash que es un mecanismo de ventana virtual deslizante. Típicamente el tamaño de esta ventana es de 7 bytes, donde el valor del hash depende sólo de las inmediaciones de los bytes dentro de la ventana y no del resto de bytes del contenido. Calculando el hash del contenido únicamente de los bytes que se encuentran dentro de la ventana virtual.

A continuación se muestra un ejemplo gráfico de cómo funciona el mecanismo de rolling hash.

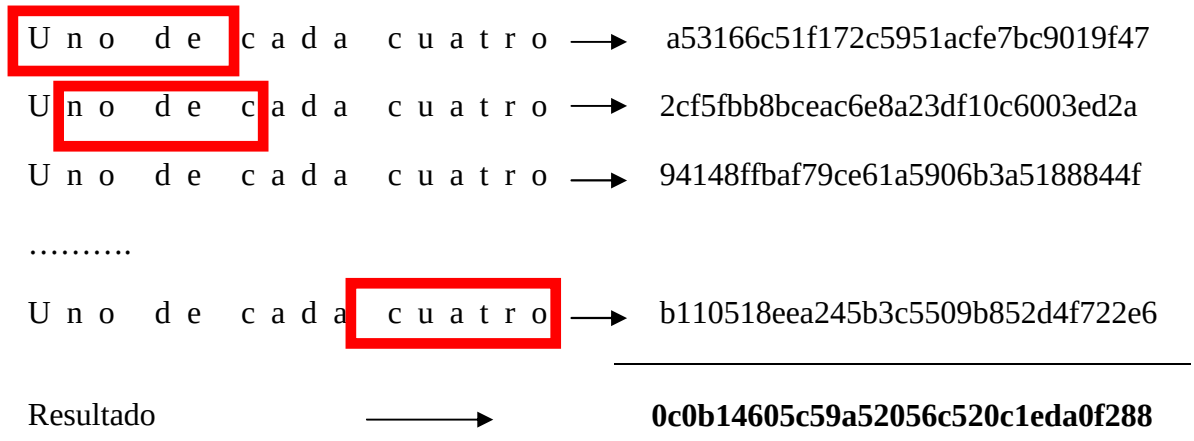


Figura 6. Ejemplo de funcionamiento del Rolling Hash

Como se puede ver en el ejemplo anterior, se va calculado el hash únicamente de los byte de la ventana, una vez calculado el hash de la ventana, esta se desplaza un byte para volver a repetir la operación. Una vez calculados el hash de todo el contenido se opera con los hash parciales obtenidos para conseguir el resultado final del Rolling hash.

Para calcular el punto de disparo basta con comprobar si un valor parcial módulo el tamaño medio de bloque coincide con un valor arbitrario dado, por ejemplo el propio tamaño menos uno (i.e. RH mod B == B - 1). De esta forma, aunque la firma del bloque depende de todo el contenido, el punto de disparo sólo depende de la última ventana, y suponiendo que el valor del hash es básicamente un valor aleatorio uniformemente distribuido, surgirá un punto de disparo en media cada B bytes.

Es importante que el punto de disparo no se vea afectado por cambios menores en el bloque, porque el cambio de un punto de disparo afectaría a la firma de los dos bloques consecutivos que éste separa. Más adelante veremos una mejora introducida para evitar que el sospechoso pueda aprovecharse de esto.

Mediante esta técnica se recorre el contenido calculando un hash criptográfico hasta encontrar un valor de disparo, momento en el cual introducimos dicho valor en la firma del contenido.

Para mejorar el rendimiento fuzzy hashing el hash calculado en el proceso de rolling hash debe tener un coste computacional muy bajo por ello se utiliza el hash Fowler / Noll / Vo (FNV). [16]

El valor de disparo está directamente relacionado con el tamaño de bloque en los que se divide un contenido. Mediante la siguiente formula se establece el tamaño del mismo [2]:

$$b_{\text{init}} = b_{\min} 2^{\lfloor \log_2(\frac{n}{S b_{\min}}) \rfloor}$$

Figura 7. Fórmula para calcular el punto de disparo

Siendo:

B_{init} = Tamaño medio de los bloques
B_{min} = 3
N = longitud del contenido en bytes
S = 64

La firma de fuzzy hashing está compuesta por caracteres en base 64. La obtención de estos caracteres se realiza mediante los 6 bits menos significativos (LS6B) del valor del rolling hash. Para no limitar la comparación de las firmas a contenidos con el mismo tamaño de bloque, fuzzy hashing genera su firma con dos partes, cada una de las partes está, calculada con un tamaño de bloque diferente, con la única intención de ser más flexible y poder corregir desviaciones en el tamaño del contenido. La longitud de la primera parte de la firma está calculado con el tamaño original de bloque y tiene una longitud máxima de 64 bits y de la segunda está calculada con la mitad del tamaño original del bloque, teniendo una longitud de 32 bits.

En la siguiente tabla se puede ver el rango de tamaño de los contenidos, en byte, que comprende cada tamaño de bloque.

Tamaño de bloque	Tamaño mínimo del contenido	Tamaño máximo del contenido
3	0	192
6	193	384
12	385	768
24	769	1536
48	1537	3072
96	3073	6144
192	6145	12288
384	12289	24576
768	24577	49152

Tabla 1. Tamaño de fuzzy hashing vs. tamaño del contenido

Esto quiere decir que, por ejemplo, se podría comparar contenidos que tuviesen un tamaño medio de bloque igual 384 con cualquier otro contenido que tuviera un tamaño de bloque igual a 192, 384 y 768. Lo que supone que se podrían comparar firmas generadas con fuzzy hashing de contenidos con tamaños hasta 4 veces el tamaño el uno del otro.

Al utilizar caracteres en base 64, para componer la firma de fuzzy hashing facilitamos la comparación de distintos contenidos mediante las firmas resultantes. Pudiendo realizar promedios ponderados de la distancia de edición entre firmas y establecer de esta forma el porcentaje de similitud.

A continuación se muestra el diagrama de flujo de fuzzy hashing, para explicar mejor su funcionamiento:

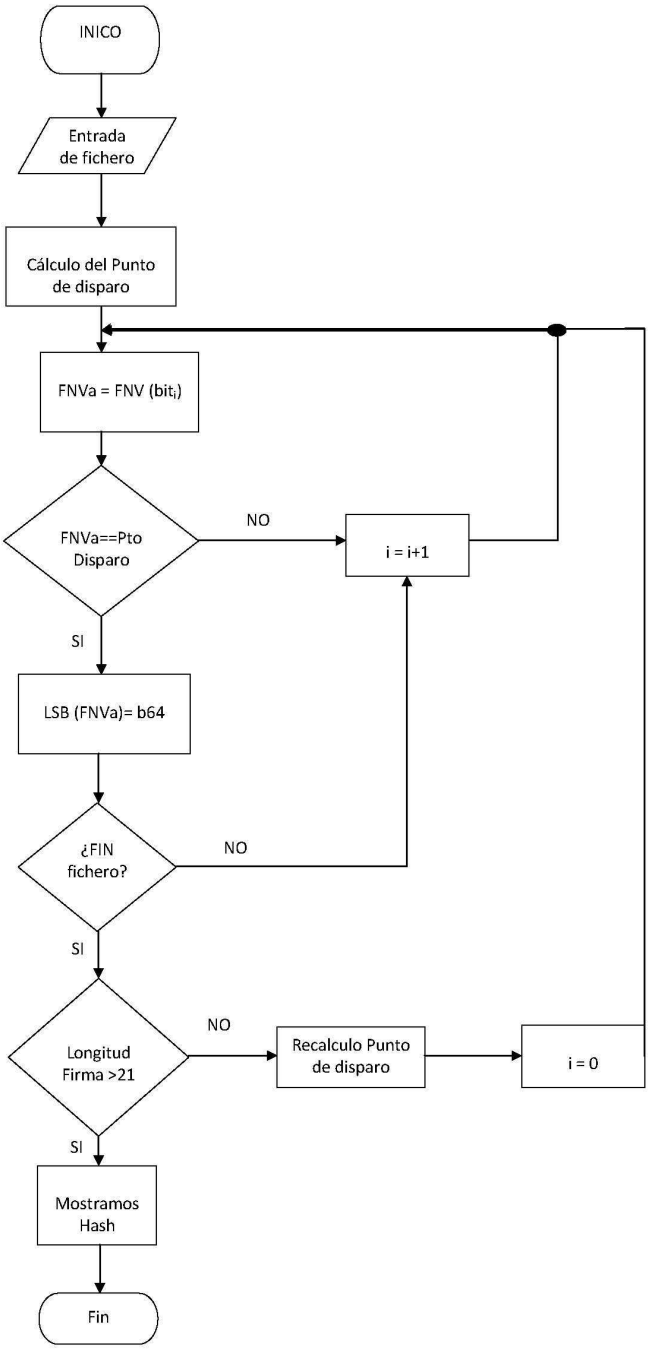


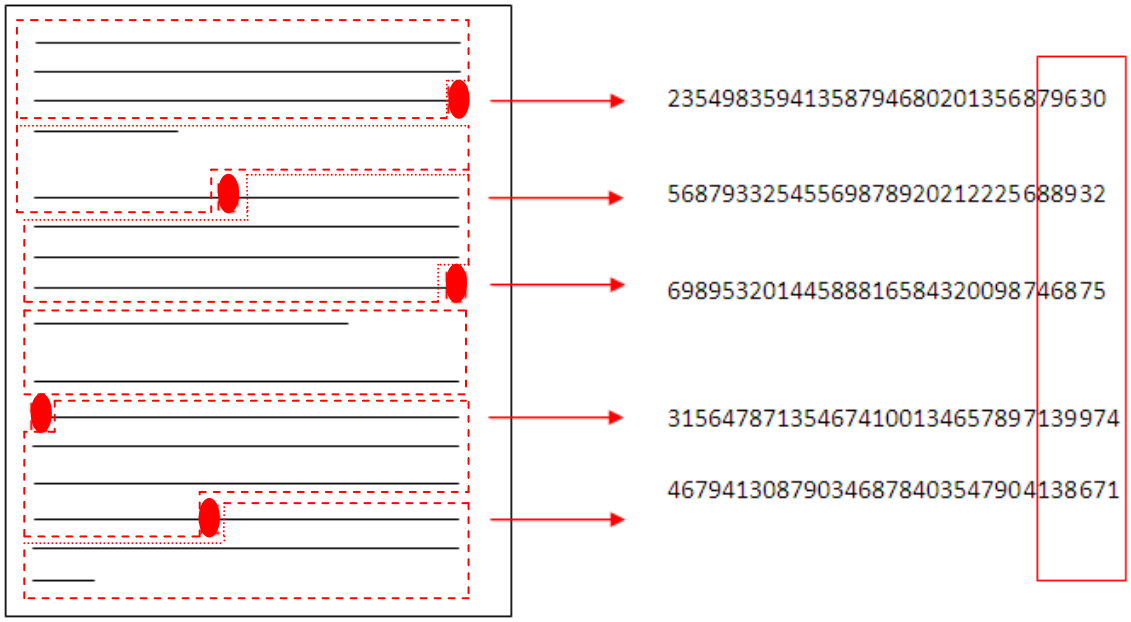
Figura 8. Diagrama de flujo de fuzzy hashing.

Fuzzy hashing realiza el cálculo del valor de disparo mediante la fórmula indicada en la Figura 7 de la presente memoria. Una vez que se tiene calculado el valor de disparo, se procede al cálculo del rolling hash y del FNV de los distintos bytes del contenido, comparando cada valor del rolling hash con los condicionantes de disparo. En el caso que el valor del rolling hash cumpliese unas condiciones determinadas sería indicativo que se ha alcanzado un valor de disparo.

Una vez terminado de recorrer el contenido y si la firma no tiene la longitud mínima de 21 bytes, se procede a repetir el mismo proceso, pero modificando el valor de disparo calculado inicialmente, hasta obtener una hash del tamaño adecuado. El objetivo de fijar una longitud mínima para la firma es evitar posibles colisiones.

De forma más gráfica en la siguiente figura se puede ver con puntos rojos, los valores de disparo y recuadrados los distintos segmentos en los que se ha dividió el contenido. El resultado del hash acumulado en estos valores de disparo es a los que se les calcula el LS6B e introducimos dicho valor en la firma correspondiente.

Los resultados de fuzzy hashing serán cadenas de texto de longitud variable comprendías entre 21 a 64 bytes en función del tamaño del archivo, reduciendo drásticamente la probabilidad de falsos positivos, al depender la longitud de la firma del tamaño del contenido.



Firma Final: GCVZGhr83h3bc0ok3892m12wzgnH5w2pw+sxNEI58:FIVkH4x73h39LH+2w+sxaD

Figura 9. Puntos de disparo y firma de fuzzy hashing

Cada valor de la firma del fichero analizado depende sólo de una parte de la entrada, y cambios limitados en el contenido en la entrada darán como resultado cambios localizados en la firma.

Cabe destacar que tanto la implementación de la técnica rolling hash y el cálculo de FNV se basan únicamente en operaciones de bajo nivel y por lo tanto, se obtiene un alto rendimiento en el cálculo de la firma del contenido.

2.5 Herramientas de Fuzzy hashing

Como habíamos mencionado anteriormente, existen dos herramientas que utilizan fuzzy hashing para el cálculo de su firma: ssdeep [2] y deeptoad [4]. Ambas herramientas se basan en dividir el contenido en partes variables y obtener una firma del contenido analizado. El desarrollo del módulo se va a basar en la herramienta más efectiva de las mencionadas anteriormente, ya que esa va a ser una de las premisas marcadas en los objetivos de la presente memoria.

Para ello se ha realizado un test de rendimiento a ambas. El test de rendimiento consiste en obtener la firma de distintos archivos utilizando ambas herramientas y comparar el tiempo de ejecución de cada una de ellas. En la comparativa también se introducen los hash criptográficos más comunes, para poder analizar la penalización en el uso de fuzzy hashing frente a los hashes criptográficos.

En la siguiente tabla se muestra el tiempo de ejecución (en segundos) de md5, sha1, ssdeep y de deeptoad para distinto contenido, en función de su tamaño. Esta prueba se realiza con un equipo HP 6730b el cual cuenta con una valoración Global igual a 110 puntos en el programa para el cálculo de rendimiento (Benchmark) BAPCo Sysmark 2007 Preview.

KB	Md5	Sha1	Ssdeep	deeptoad
24	0,021 s	0,003 s	0,002 s	0,050 s
116	0,022 s	0,003 s	0,011s	0,111 s
2.389	0,069 s	0,029 s	0,073 s	1.551 s
34.201	0,703 s	0,388 s	0,863 s	20.991 s

Tabla 2. Tiempo de ejecución de hash

A la vista de los resultados obtenidos se puede determinar que ssdeep es mucho más eficiente que deeptoad. Los motivos por los que deeptoad es menos eficiente que ssdeep son porque deeptoad carga el archivo completo en memoria [4] y es demasiado lento en el cálculo de la firma debido a su programación [4]. Errores a evitar en el módulo. También se puede observar que los tiempos de ejecución ssdeep frente md5 son muy similares.

Se puede comprobar que ssdeep [3] es un proyecto mucho más activo que deeptoad [4], en sus correspondientes changelog publicados en las páginas de sus desarrolladores. Este dato da ciertas garantías que ssdeep es una herramienta consolidada y ampliamente testeada.

Capítulo 3

Diseño

En este capítulo se explicará el diseño del sistema de pre-clasificación desarrollado. Para ello se mostrarán los casos de uso de la aplicación, así como la arquitectura general de la misma.

3.1 Arquitectura de la aplicación

A continuación se define la arquitectura física y lógica de la aplicación implementada dentro del alcance del presente PFC, para ello vamos a basarnos en la siguiente figura.

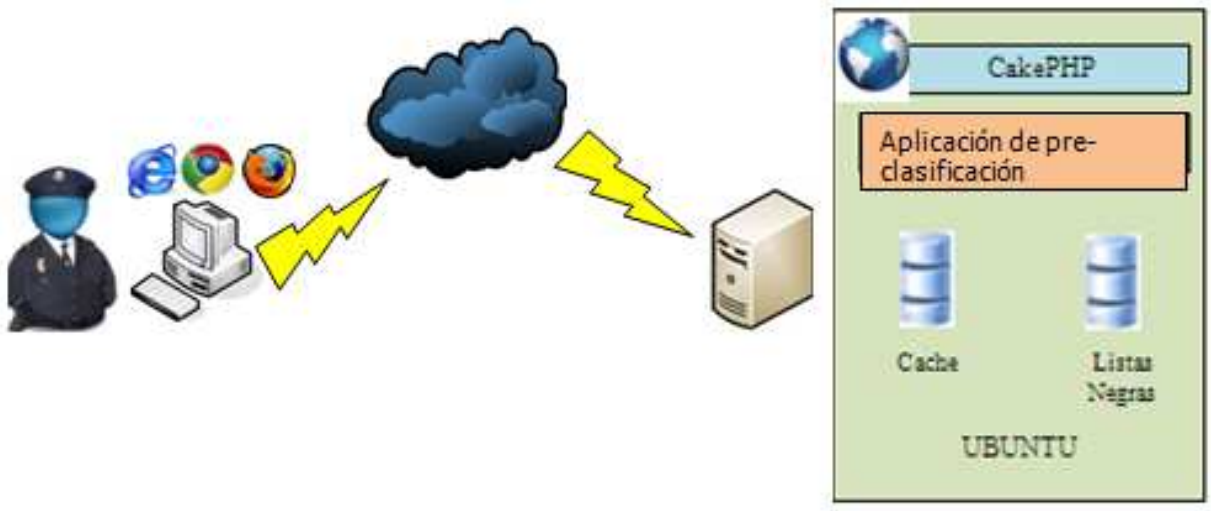


Figura 10. Arquitectura física y lógica de la aplicación

Como podemos ver en la figura anterior, la aplicación se ha implementado como un escenario cliente web y servidor. En la parte de cliente web tendremos al analista de la policía que accederá a la aplicación a través de su navegador favorito a la aplicación que reside en un servidor web Ubuntu.

El analista lanzara las consultas al servidor y esté le presentara la información solicitadas al analista de la policía.

En cuanto a la arquitectura lógica de la aplicación, se puede dividir en dos partes, una parte de presentación web desarrollada en CakePHP con la intención de facilitar su futura integración con Xplico. Y una capa de *backend,* en la cual se encuentra la lógica del sistema a desarrollar y las bases de datos que necesita la aplicación para su funcionamiento (caché y listas negras) en SQLite.

La aplicación cuenta con distintos procesos:

1. Calculo de fuzzy hashing
2. Modificación de los contenidos a formatos neutros
3. Comparación de los contenidos con los existentes en las listas negras
4. Comparación de los contenidos con la cache

Una vez que se tiene definida tanto la arquitectura física como lógica de la aplicación, se tendría que proceder a la implementación de la misma, como podemos ver en el siguiente capítulo.

3.2 Casos de uso de la aplicación

A continuación se muestran los casos de uso que ofrece nuestra aplicación a los usuarios. Por un lado se muestran los casos de uso que ofrece la aplicación y por otro los que ofrece la cache del sistema.

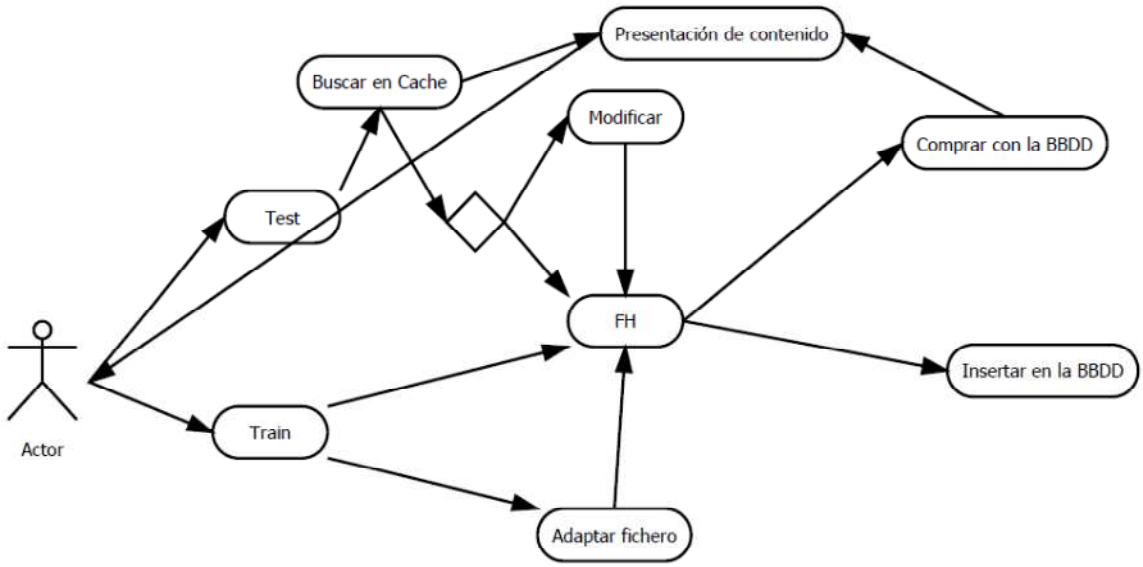


Figura 11. Casos de Uso del entrenamiento y evaluación del sistema

El usuario puede entrenar al sistema con nuevos contenidos (Train), esta acción a su vez lanza una serie de de acciones internas al sistema, como son:

1. Entrenamiento (Train)
   1.1 Calculo de Fuzzy hashing
   1.2 Adaptación del contenido a un formato neutro
   1.3 Calculo de Fuzzy hashing
   1.4 Inserción en la Base de datos.

Otras de las acciones que permite la aplicación al usuario es realizar una evaluación (Test) de un contenido, esta acción al igual que pasa con la acción de entrenamiento, hace que el sistema ejecute una serie de acciones:

2. Evaluación (Test)
   2.1 Calcular el md5 del fichero
      2.1.1 Búsqueda en cache de esta misma evaluación
   2.2 Calcular el Fuzzy Hashing del fichero
   2.3 Modificar el contenido a formatos homogéneos
   2.4 Calcular el Fuzzy Hashing del fichero modificado
   2.5 Inserción de la información del contenido en la cache.
   2.6 Mostrar los resultados tras la evolución al usuario.

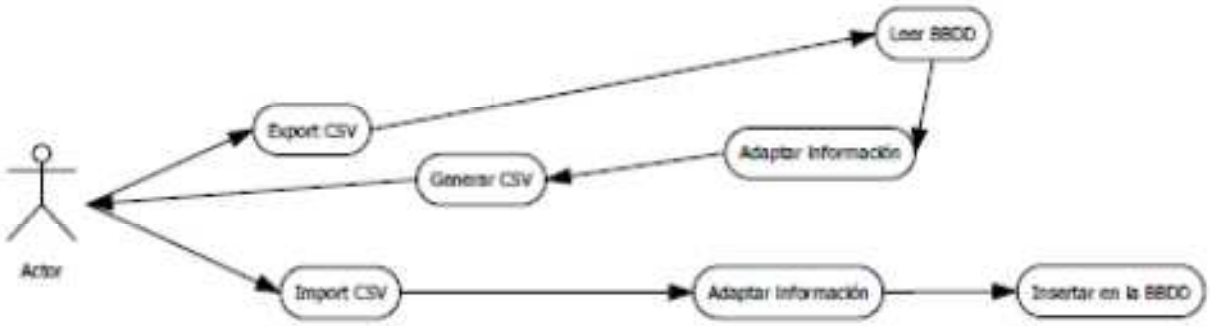


Figura 12. Casos de Uso del módulo mediante CSV

El usuario puede exportar el contenido a archivos CSV e importar de archivos CSV contenido. Estas acciones pueden resultar útiles para la carga máxima de contenido en el sistema o la exportación de la información recopilada por el sistema a terceros. Estas acciones al igual que las anteriores hacen que el sistema lance acciones internas:

3. Export CSV
   3.1 Leer de las listas negras
   3.2 Adaptar la información de las listas negras a formato CSV
   3.3 Generar fichero CSV
   3.4 Devolver al usuario el fichero CSV generado
4. Import CSV
   4.1 Adaptar la información del fichero a la estructura de la lista negra
   4.2 Añadir a la lista negra la información proveniente del fichero CSV

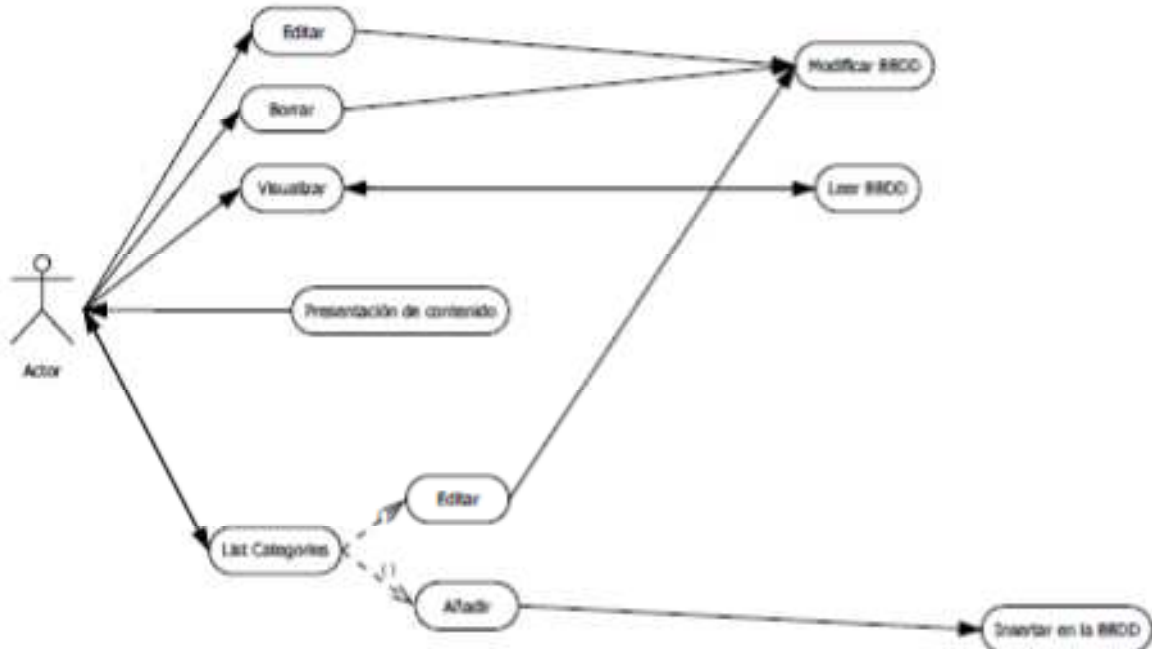


Figura 13. Casos de Uso del módulo Edición y Visionado

Como muestra la figura anterior, la aplicación también ofrece una gran variedad de acciones básicas de modificación de los contenidos que se tienen en las listas negras de contenidos:

5. Editar la información existente

    5.1 Modificar la lista negra

6. Borrar un contenido

    6.1 Modificar la lista negra, eliminando dicho contenido

7. Visualizar toda la información de un contenido existente en la lista negra.

    7.1 Leer de la lista negra

8. Presentación de todos los contenidos bajo una categoría

    8.1 Leer de la lista negra

El usuario también puede listar las categorías en las cuales puede clasificar los distintos contenidos, llegando a incluir nuevas categorías o editar las existentes. Esto a su vez lanza las siguientes acciones de sistema:

9. Mostrar Todas las categorías que maneja nuestro sistema

    9.1 Editar una categoría

    9.2 Añadir una nueva categoría.

Ahora se procede a analizar en más detalle el diagrama de uso de la cache del sistema

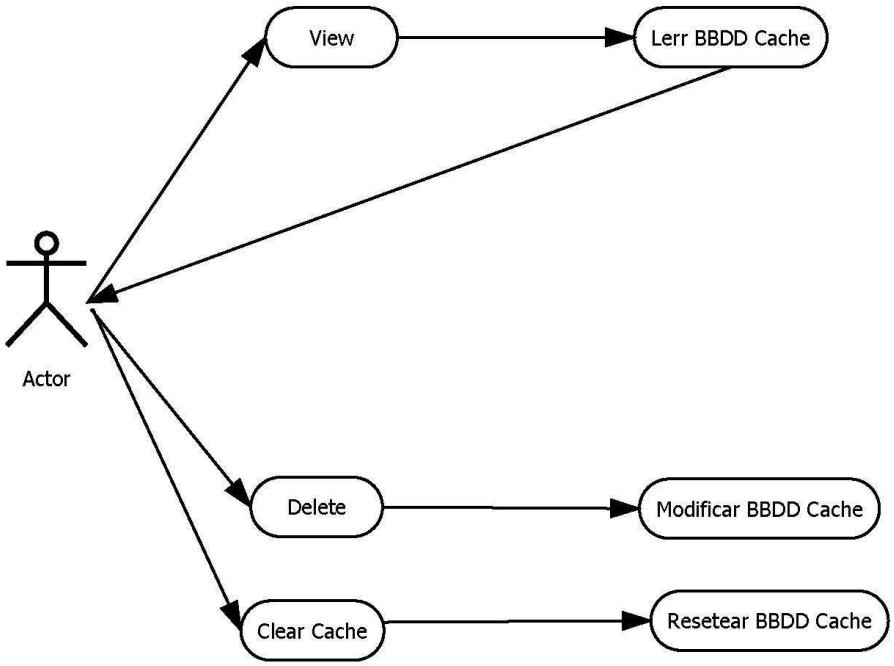


Figura 14. Casos de Uso de la caché del sistema

Como se puede comprobar las acciones que permiten al usuario esta funcionalidad de la aplicación es mucho más limitada que las expuestas anteriormente. En este caso solo se pueden realizar las siguientes acciones:

10. Ver cache

    10.1 leer el detalle de cache de uno de los contenidos almacenados en ella

11. Borrar entrada a cache

    11.1 Modificar la cache

12. Limpiar cache

    12.1 Resetear la cache

La acción “View” (Ver Cache) permite al usuario ver el detalle de la entrada de la cache seleccionada. Esto a su vez lanza una serie de acciones del sistema, que son leer de la Base de datos de la cache y mostrar dicha información al usuario.

La acción “Delete” (Modificar cache) permite al usuario borrar la entrada seleccionada de la cache, esta acción origina la acción del sistema de modificación de la base de datos.

Por último la acción “Clear cache” (limpiar cache) permite al usuario borrar la cache al completo, lanzando una acción del sistema de reseteo de la base de datos del cache.

3.3 Algoritmo implementado

A continuación se explicará cómo se debe implementar el algoritmo del sistema pre-clasificador de contenidos, para ello en la siguiente figura se muestra el diagrama de flujo del módulo y a partir de este se irán detallando uno a uno los distintos casos de uso.

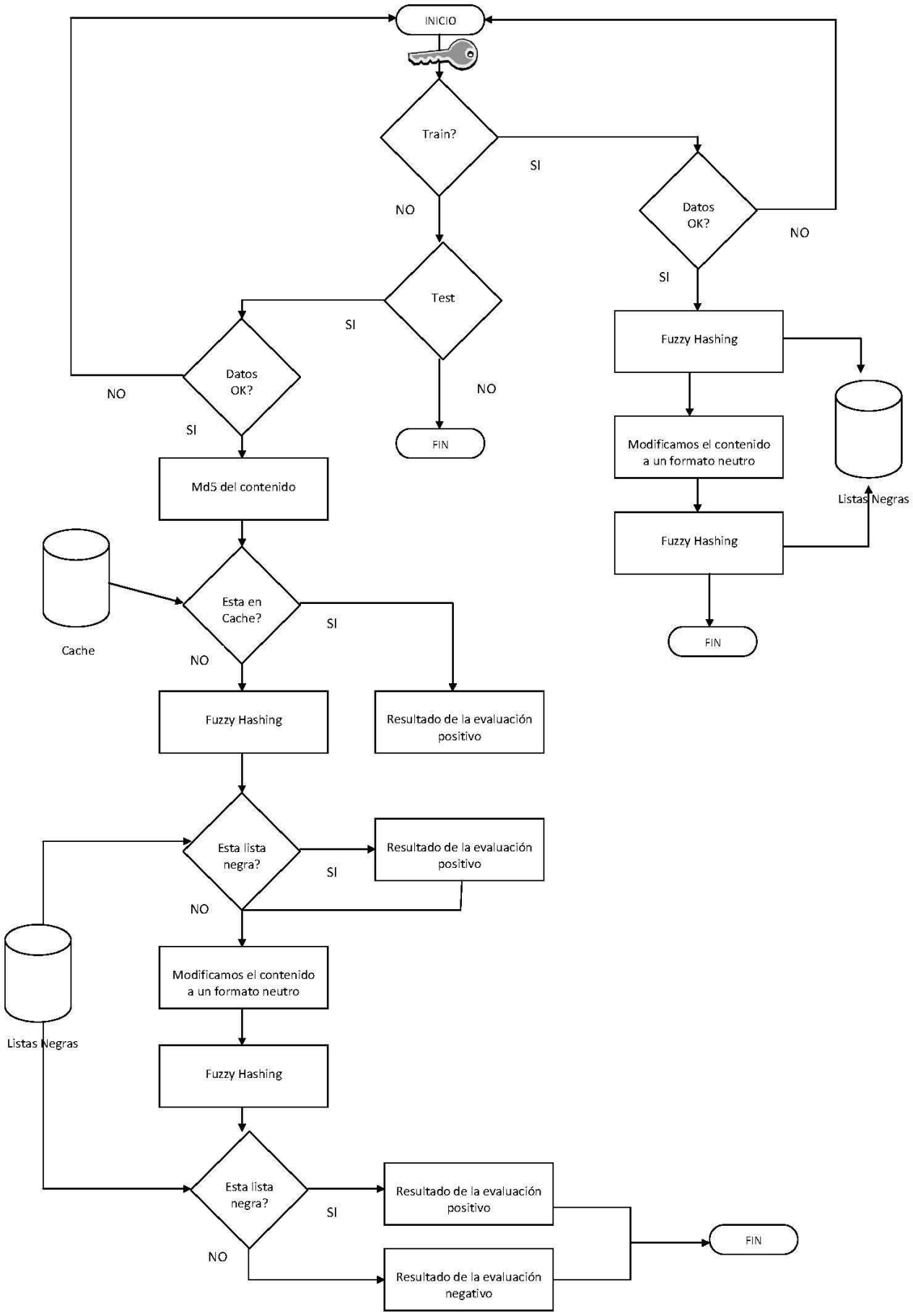


Figura 15. Diagrama de Flujo del modulo pre-clasificador

Como se ve en el diagrama de flujo anterior, el sistema necesita ser iniciado por parte del usuario para ejecutarse. Una vez que el usuario facilita el fichero con el que quiere trabajar, la clave de seguridad y los demás datos complementarios necesarios para la ejecución del sistema en sus dos modos (entrenamiento y evaluación). El sistema empieza a lanzar sus procesos internos.

Lo primero que hace el sistema es evaluar en qué modo se va a ejecutar, es decir, en modo entrenamiento o en modo evaluación. Una vez definido el modo en el cual se va a ejecutar el sistema, se comprueban que los datos facilitados son correctos para poder funcionar. Y una vez realizadas esta serie de comprobaciones, se calcula el Fuzzy hashing del contenido facilitado por el usuario.

En este punto nuestro módulo tendrá una serie de modificaciones respecto a la implementación de fuzzy hashing.

- **Seguridad:** El usuario facilita una clave al módulo y este mediante esta clave calcula los puntos de disparo del contenido. De esta forma se evita que personas ajenas sepan el método del cálculo del fuzzy hashing e intenten romperlo mediante diversos ataques o conozca a priori los puntos de disparo y por lo tanto el valor final de la firma. Los ataques que puede sufrir son que el sospecho intente forzar que un contenido que se encuentra en una lista negra, mediante modificaciones puntuales (inserciones, añadir basura, cambiar unos caracteres por otros, etc.) obtenga un hash completamente distinto, evitando de esta forma ser detectado.

- **Eficiencia:** Adicionalmente el módulo cuenta con un procesado del contenido, con la intención de adaptarlo a un formato que maximice los resultados que fuzzy hashing ofrece en la comparación de contenidos. Las modificaciones que se realizan en los contenidos se detalla en el capítulo de evaluación. El sistema se implementará de forma que calculará el fuzzy hashing del fichero sin modificación alguna y posteriormente lo volveremos a calcular con las modificaciones que se indican en el capítulo de evaluación.

El módulo a implementar permitirá realizar comparaciones de dos firmas con la única intención de determinar si los contenidos de los que se obtuvieron son similares. Los pasos realizados para calcular la similitud entre dos firmas generadas por el sistema son:

- El primer paso que se realiza es comprobar el tamaño del bloque, si el tamaño del bloque de ambos es compatible (tamaño de bloque del contenido son iguales o uno es el doble del otro) pueden ser comparados, en caso contrario las firmas de los contenidos no pueden ser comparados

- El siguiente paso es buscar secuencias recurrentes en la firma con la intención de eliminarlas. El motivo es porque las secuencias repetidas no transmite mucha información sobre el contenido del archivo, por lo que es conveniente eliminarlo.

- Por último calculamos un promedio ponderado de la distancia de edición entre las dos firmas, mediante la distancia Levenshtein [17]. La distancia Levenshtein es el

> número mínimo de operaciones (inserción, eliminación o la sustitución de un carácter) requeridas para transformar una cadena de caracteres en otra.

Dado el promedio ponderado de la distancia de edición entre las dos firmas, calculado en el modulo, el sistema de preclasificación toma dicho valor y el valor de importancia del fichero existente en las listas negras al que es similar, y da una importancia al fichero comparado. La importancia de los contenidos tiene un valor numérico comprendido entre 1 y 5, siendo 5 muy importante, y 1 poco relevante.

Para mejorar el rendimiento, el sistema cuenta con una memoria cache en la cual se almacenan todos los contenidos calculados, para evitar calcular varias veces el mismo fuzzy hashing. De esta forma liberamos a la máquina de la carga computacional que supone este cálculo.

Cuando el usuario quiere evaluar una contenido el sistema lo primero que hará es consultar su menoría cache si este ha sido evaluado con anterioridad. La memoria cache cuenta como identificador de contenidos el resultado del hash criptográfico md5.

Capítulo 4

Implementación

4.1 Bases de datos

El sistema implementado cuenta con una base de datos SQLite, en una se almacenan las tablas relacionadas propias de la aplicación y de la caché de la misma

La base de datos de la aplicación cuenta con dos tablas

- La primera tabla que se denomina **categories** almacena los distintos tipos de categorías que nuestro sistema puede analizar.

    - **Id,** identificador del campo. Valor numérico autoincremental
    - **Type**, tipo de categoría. Campo de texto que califica la categoría

    A modo ejemplo la tabla categories estaría compuesta por:

Id	Type
1	Terrorism
2	Pedophilia

Tabla 3. Tabla de categorías de la BBDD de la aplicación

- La segunda tabla la denominamos **signatures**, almacena los fuzzy hashing de los distintos archivos. Esta tabla está compuesta por los campos:
    - **Id,** identificador del campo. Valor numérico autoincremental
    - **Category,** categoría en la cual esta analizado el contenido.
    - **Value,** valor asignado por el analista. Valor numérico comprendido entre 1 y 5, siendo 1 poca importancia y 5 muy importante
    - **URL,** dirección URL donde el contenido está disponible, si es el caso.
    - **FH_hash,** firma calculado para comprar el contenido.
    - **Coment,** comentarios del analista

  A modo ejemplo la tabla signatures estaría compuesta por:

Id	Category	Value	URL	FH_hash	Comment
1	Categories.1 (Terrorism)	1	http://.....	14asdfad157…	Archivo de …
2	Categories.2 (Pedophilia)	3	http://.....	4asdfasd473…	Imagen de …
3	Categories.2 (Pedophilia)	5	http://.....	4fsdf78sfs3…	Imagen de …
4	Categories.1 (Terrorism)	5	http://.....	7788refasd…	Manual de …
5	Categories.1 (Terrorism)	2	http://.....	aaaAAfasd5…	Video de …
6	Categories.1 (Terrorism)	4	http://.....	aFFaffaf2ff…	Manual de …
7	Categories.2 (Pedophilia)	1	http://.....	44811eeeE1…	Audio de …

Tabla 4. BBDD de los contenidos analizados y calificados.

La tabla **categories** y **signatures** están relacionadas, para que en el campo **category** nos aparezca el id, que representa al **type** de **categories.** De esta forma conseguimos que el usuario seleccione de una lista de posibles categorías, la categoría adecuada para el contenido que este analizando. La tabla de contenidos es utilizada como lista negra en el presente trabajo.

La tabla de cache se almacenará toda la información calculada en interacciones anteriores de la evaluación de contenido, el objetivo de esta caché es agilizar los resultados de la aplicación al no tener que analizar varias veces el mismo contenido. Los

campos existentes en esta base de datos son:

- **md5,** identificador único del contenido
- **value,** valor resultante de la pre-clasificación
- **Coment,** comentario del analista

Utilizamos el hash md5, para comprobar si un archivo ha sido o no analizado previamente.

En la siguiente imagen se puede ver la relación existente entre las tablas de nuestra base de datos de aplicación.

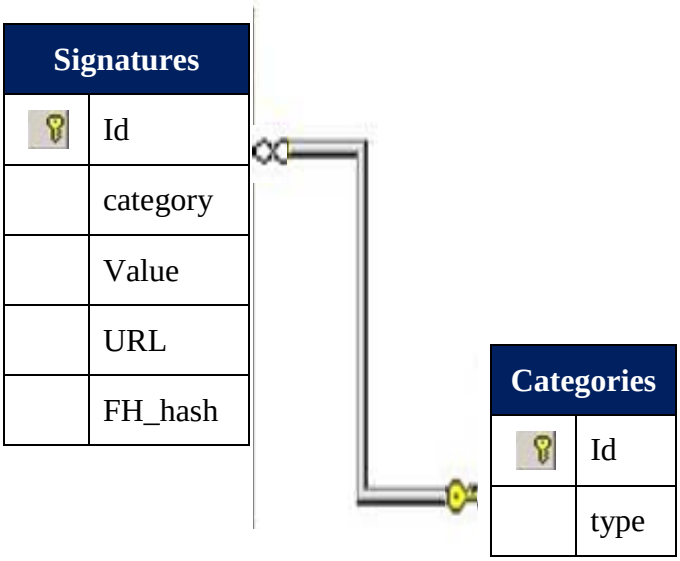


Figura 16. Relación entre tablas de la BBDD de la aplicación

4.2 Interfaz web de la Aplicación

A continuación se explicará brevemente cómo esta implementada la aplicación desarrollada.

En primer lugar nos dirigimos a la página de inicio http://example.org/PFC/signatures, y lo que se puede visualizar al ir a dicha dirección es la siguiente página.

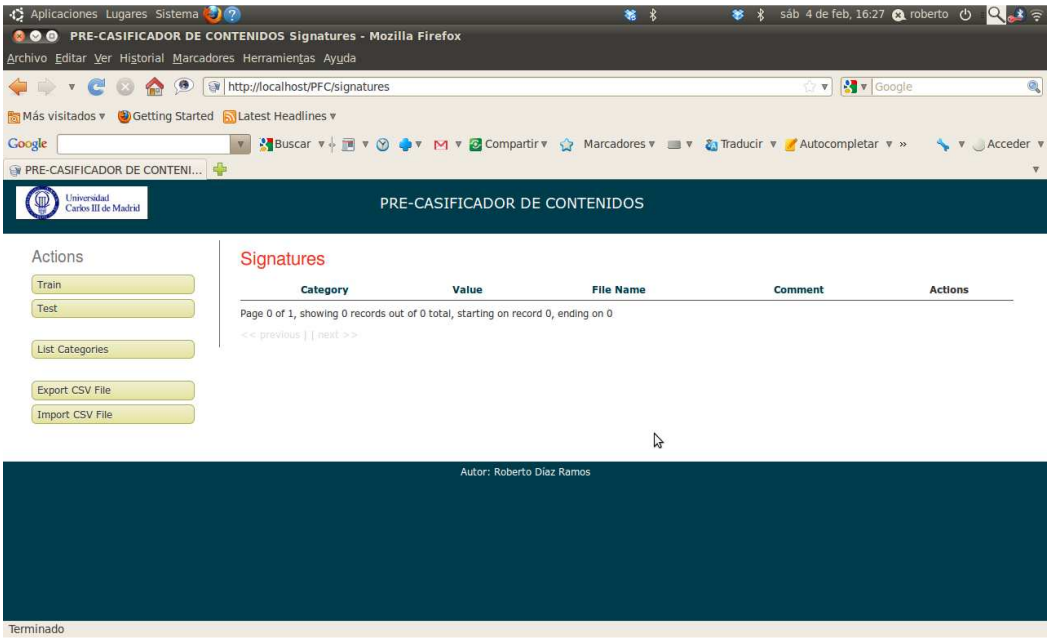


Figura 17. Página principal de la aplicación

En esta primera página se ven las acciones que se pueden realizar con la aplicación (situados a la izquierda de la pantalla) y con los archivos analizados, en este caso el sistema no ha analizado ningún fichero todavía.

Las acciones que se pueden realizar son:

- **Entrenamiento**
  - **Listado de contenido analizado**
  - **Listado de categorías**
- **Evaluación**
  - **Listado de contenido analizado**
  - **Listado de categorías**
- **Listar las categorías Existentes**
  - **Añadir categoría**
  - **Listado de contenido analizado**
- **Exportar a un fichero CSV**
- **Importar de un fichero CSV**
  - **Listado de contenido analizado**

Este es el diagrama de flujo definido para la aplicación, cuando el usuario decide entrenarlo (Train).

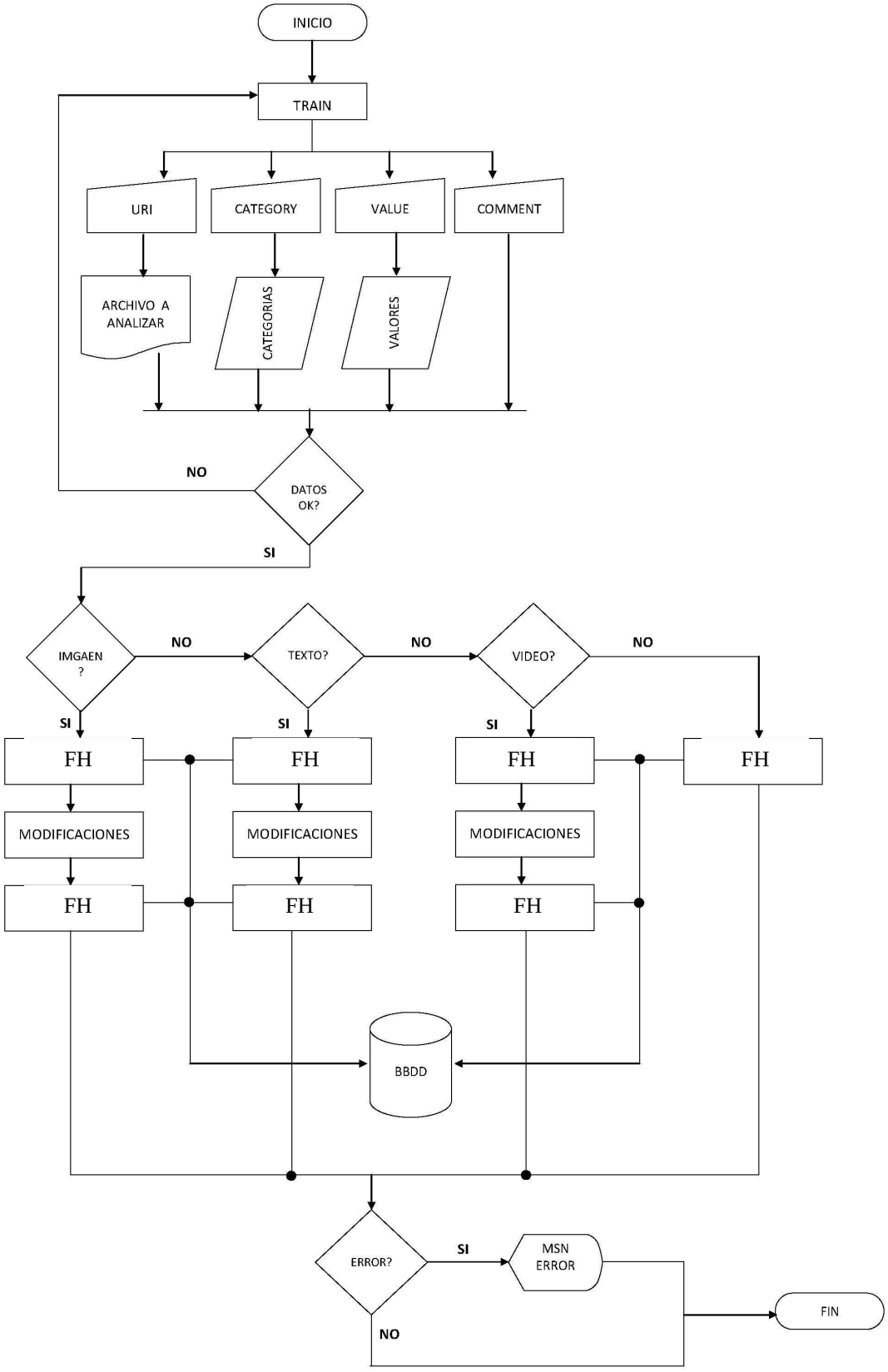


Figura 18. Diagrama de Flujo de Entrenamiento

Una vez visto el diagrama de flujo, se muestra las distintas pantallas que se va encontrando el usuario para entrenar al sistema. En esta primera página hay que indicar la categoría en la cual se quiere entrenar, la importancia que tiene el contenido según el usuario que lo está entrenando e información complementaria.

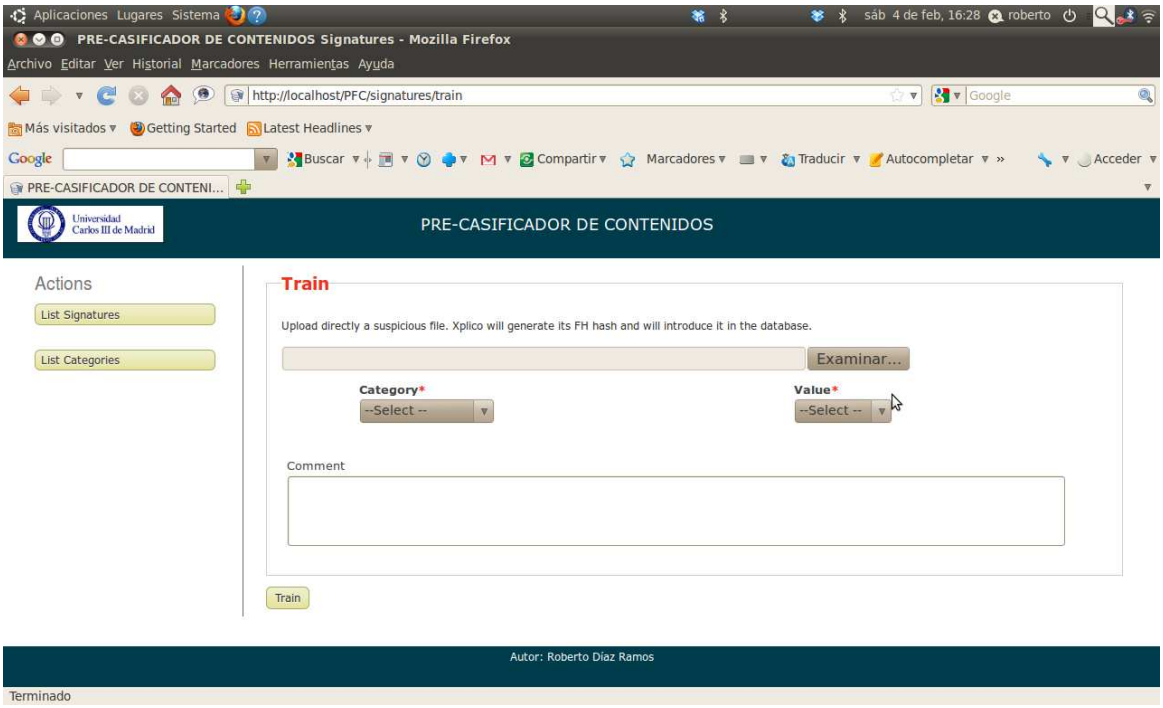


Figura 19. Página de entrenamiento del módulo

Los contenidos que se quieren analizar son subidos al servidor para poder proceder a realizar el cálculo del fuzzy hashing, una vez realizado dicho calculo los contenidos son borrados del servidor quedando únicamente referenciados por su fuzzy hashing en las bases de datos del módulo implementado

Una vez entrenado el sistema con distintos contenidos, en la aplicación aparecerá la información de los mismos.

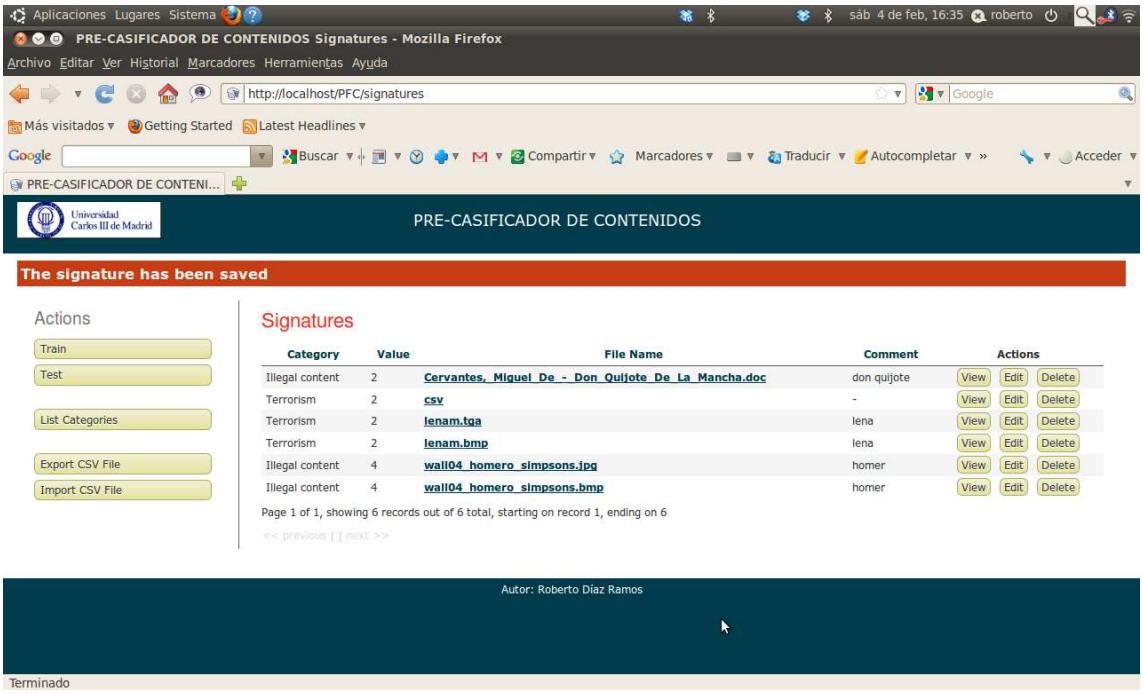


Figura 20. Opciones sobre los contenidos

Este es el diagrama de flujo que tiene la aplicación, cuando el usuario decide realizar algunas de las acciones posibles (View, Edit y Delete) sobre los contenidos con el que se la ha entrenado.

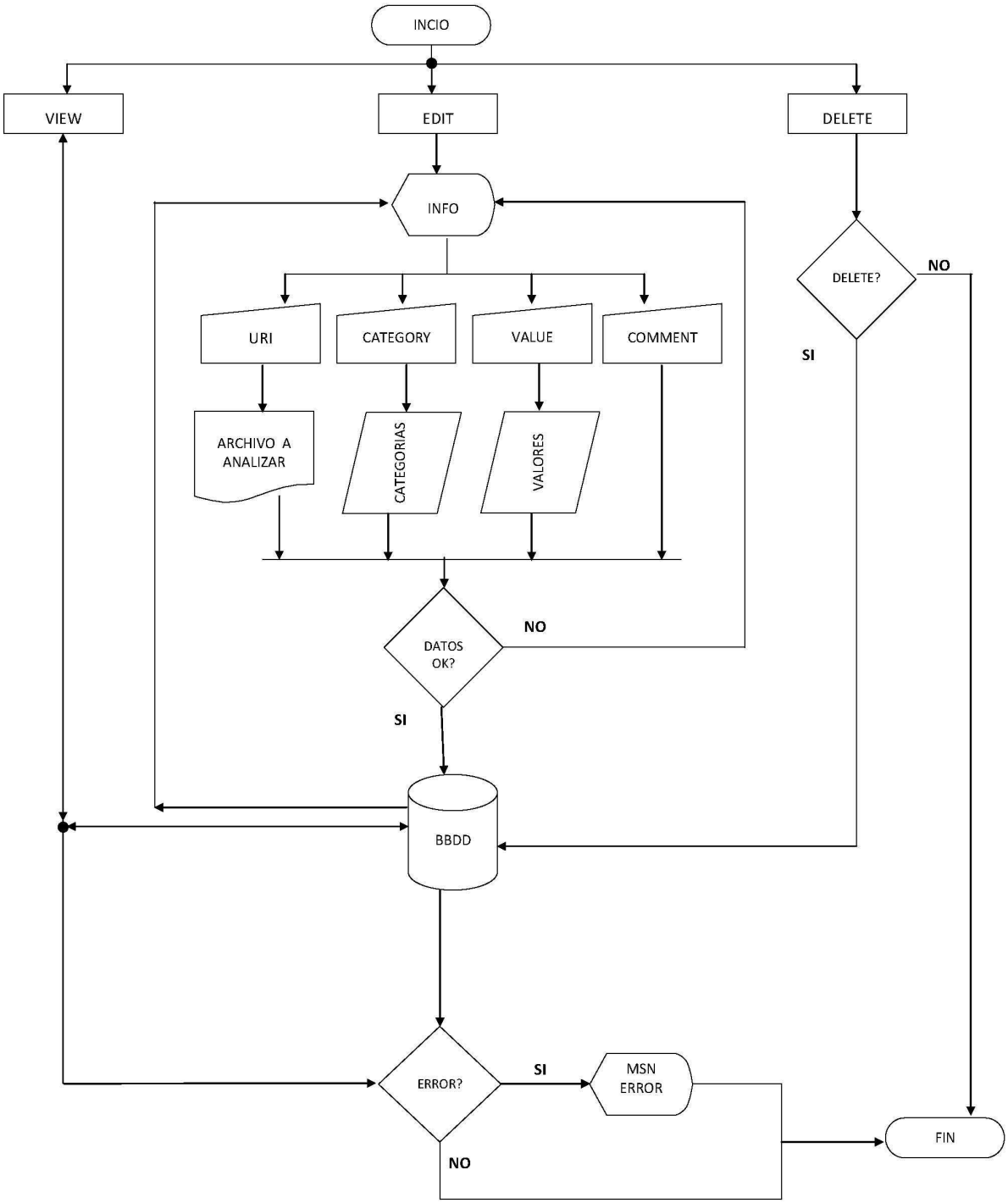


Figura 21. Diagrama de flujo funciones entrenamiento

Las distintas pantallas que van apareciendo función de la acción que se quiera realizar sobre el contenido con el que se ha entrenado la aplicación son las indicadas a continuación.

Se puede visualizar la información detallada de cada uno de los contenidos, pulsando sobre el botón “View”.

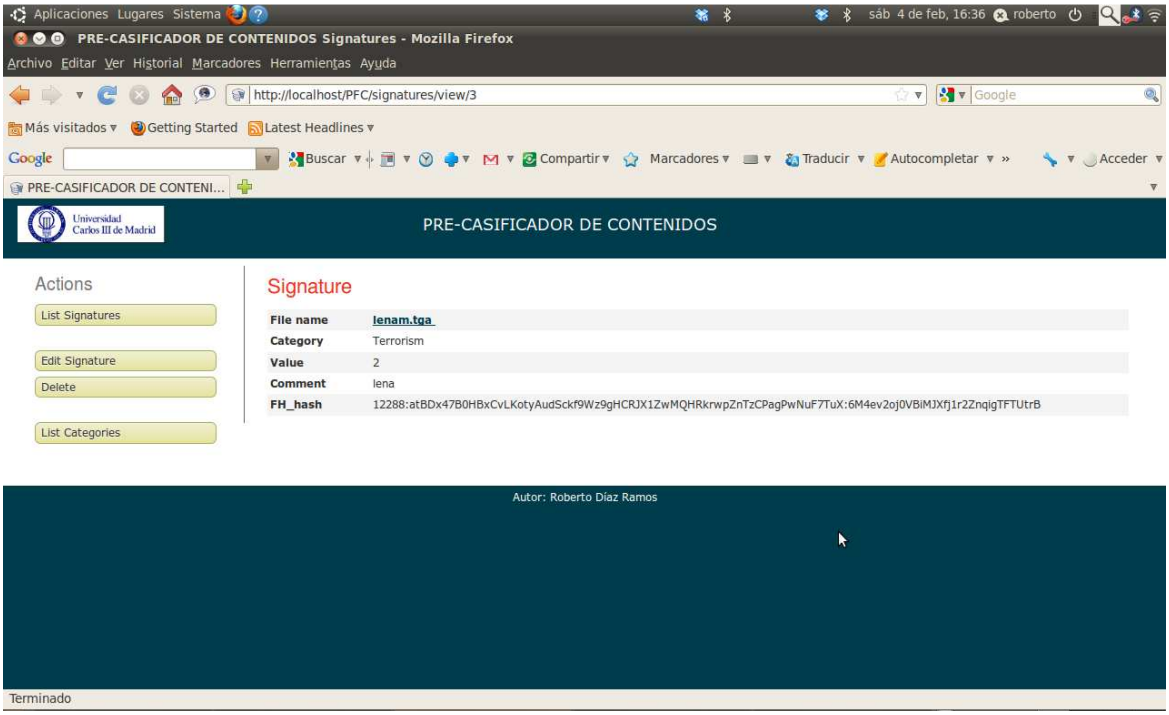


Figura 22. Información detallada del contenido

También se puede editarlo, lo cual permite cambiar la URL, ampliar el comentario del mismo o modificar el valor de su importancia. Mediante el botón “Edit”

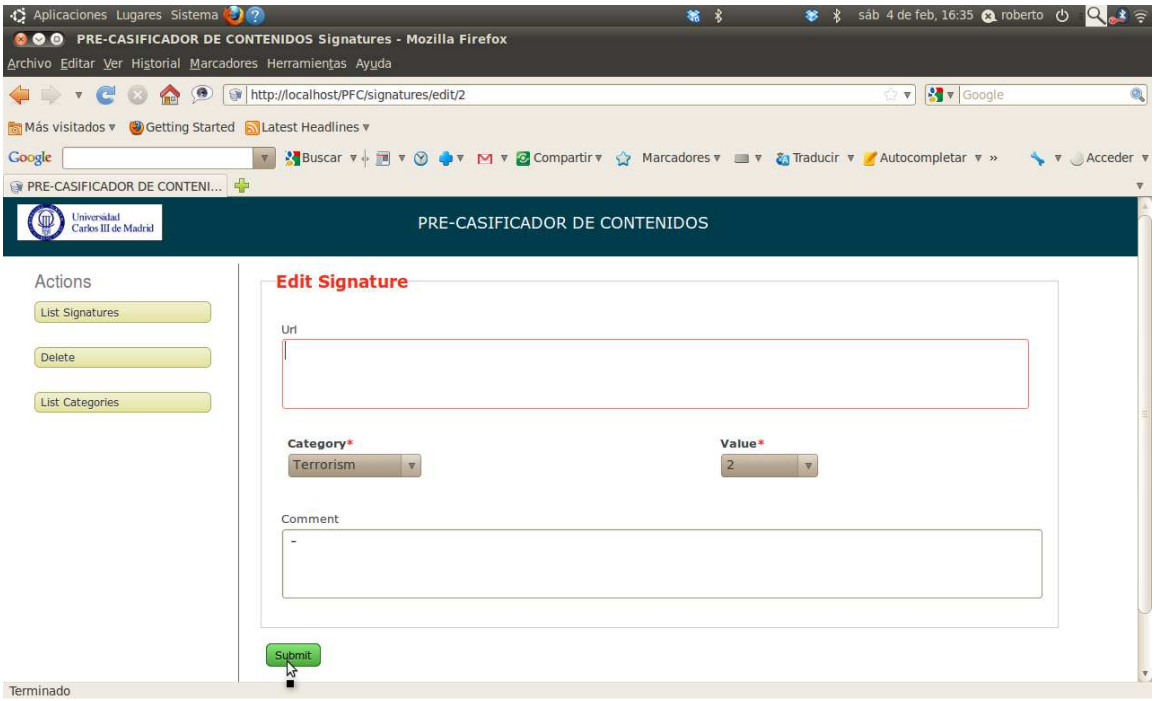


Figura 23. Página de edición de contenidos

O borrarlo de nuestra aplicación, mediante el botón “delete”. Antes de realizar el borrado se pide confirmación, para evitar borrados por error.

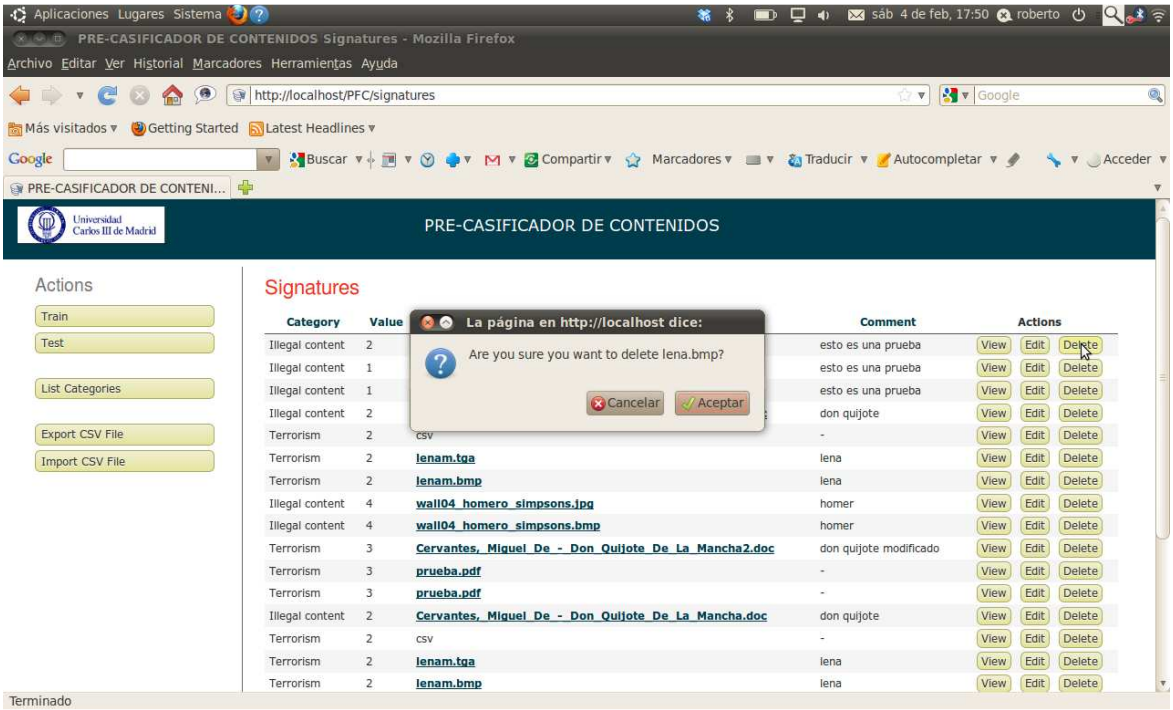


Figura 24. Borrado de contenidos

Este es el diagrama de flujo que tiene la aplicación, cuando el usuario decide evaluar algún contenido (Test).

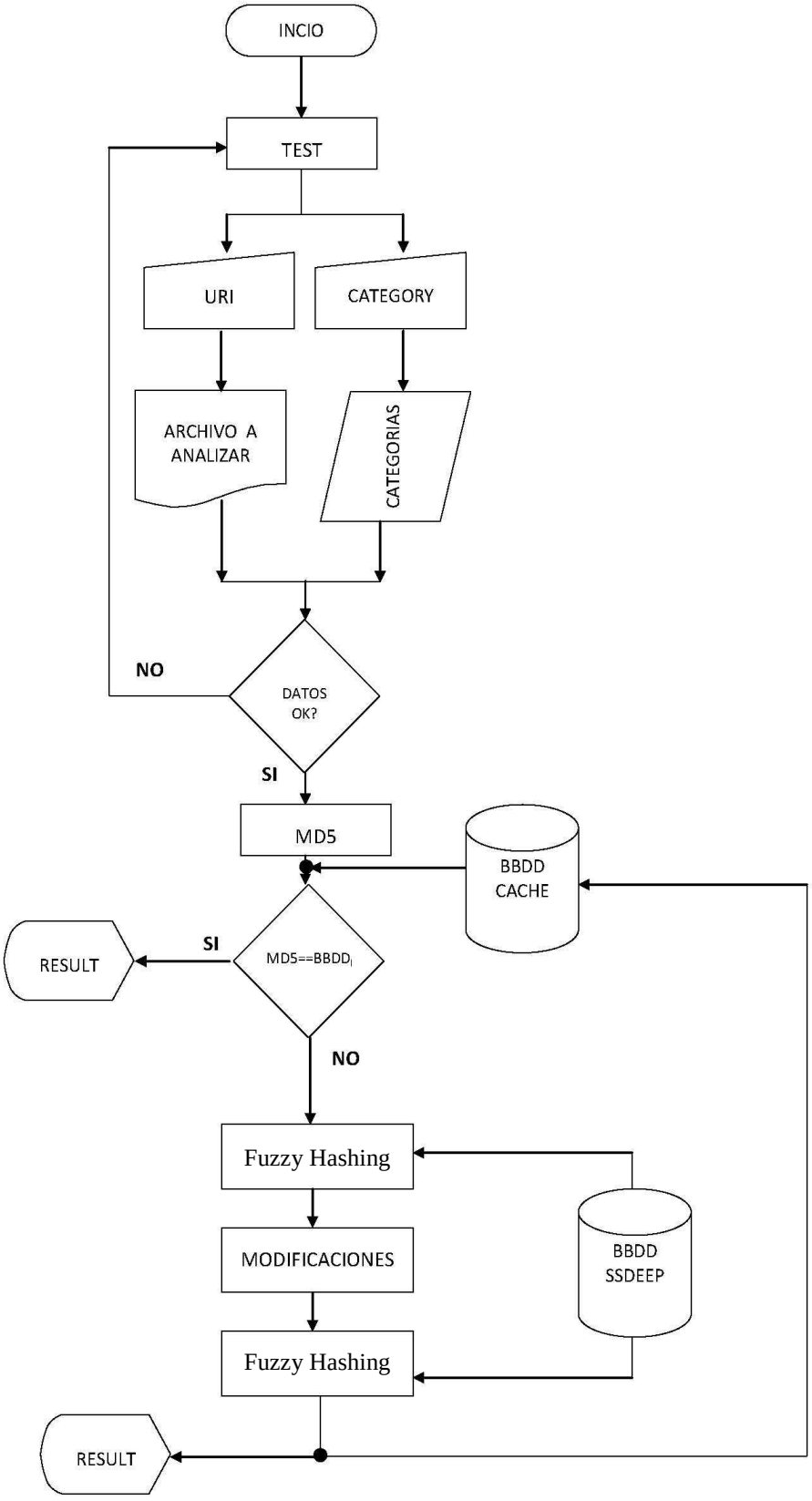


Figura 25. Diagrama de flujo Evaluación

Las distintas pantallas que se van a ir mostrando en cuando el usuario decida evaluar un contenido son las indicadas a continuación.

En esta primera página hay que indicar el contenido y la categoría en la cual se quiere evaluar.

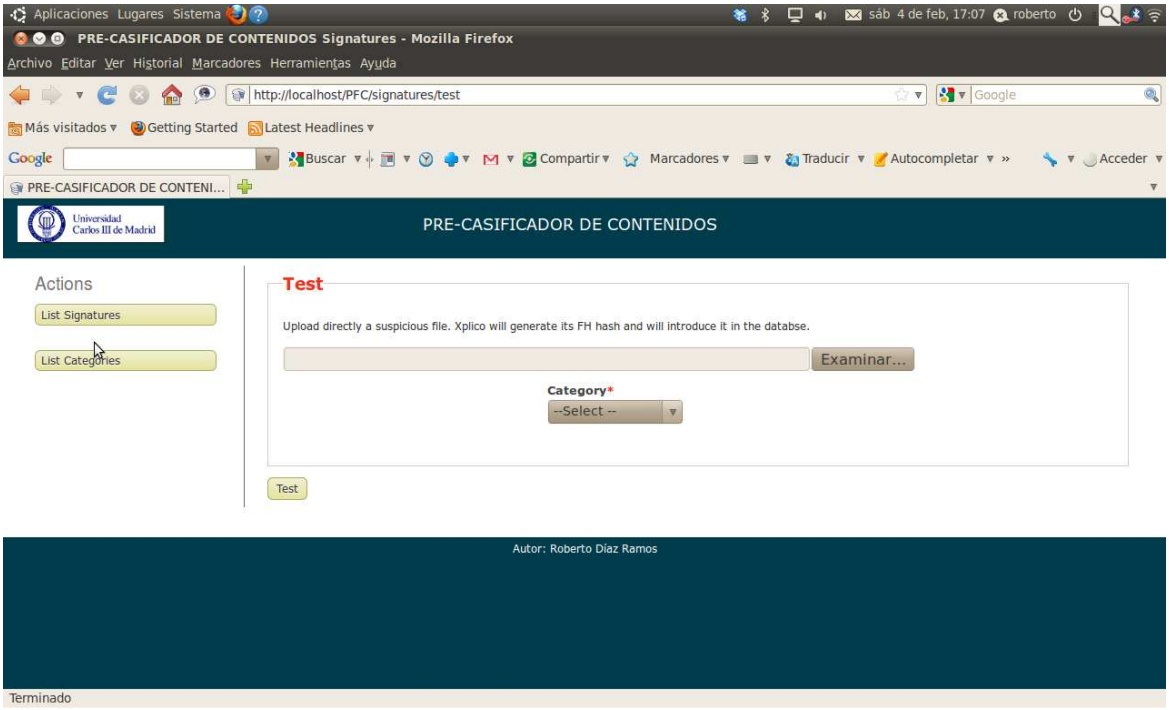


Figura 26. Página de Evaluación del módulo

Antes de realizar ningún cálculo, nuestra aplicación comprueba si el contenido coincide con alguno de los contenidos analizados previamente en la categoría seleccionada. En caso contrario procederá a realizar los cálculos y modificaciones necesarias, para mostrar el resultado de la evaluación. A continuación se muestran los posibles resultados de la aplicación una vez realizada la evaluación de contenidos.

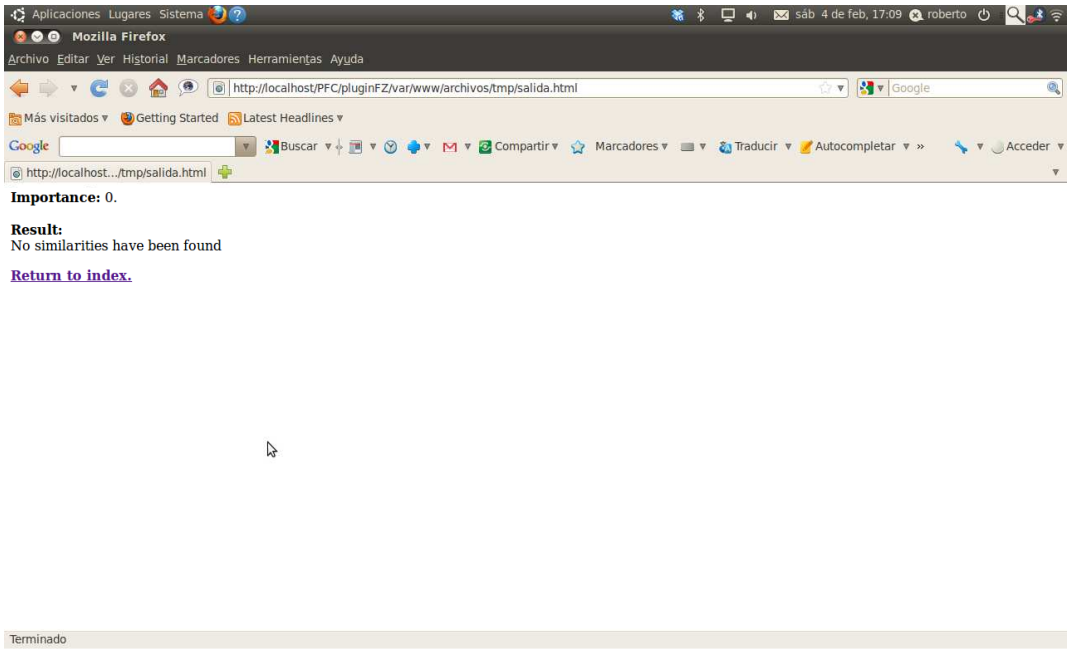


Figura 27. Resultado de la evaluación (No similarities have been found)

La pantalla anterior muestra el mensaje que no se han encontrado semejanzas entre los contenidos existentes

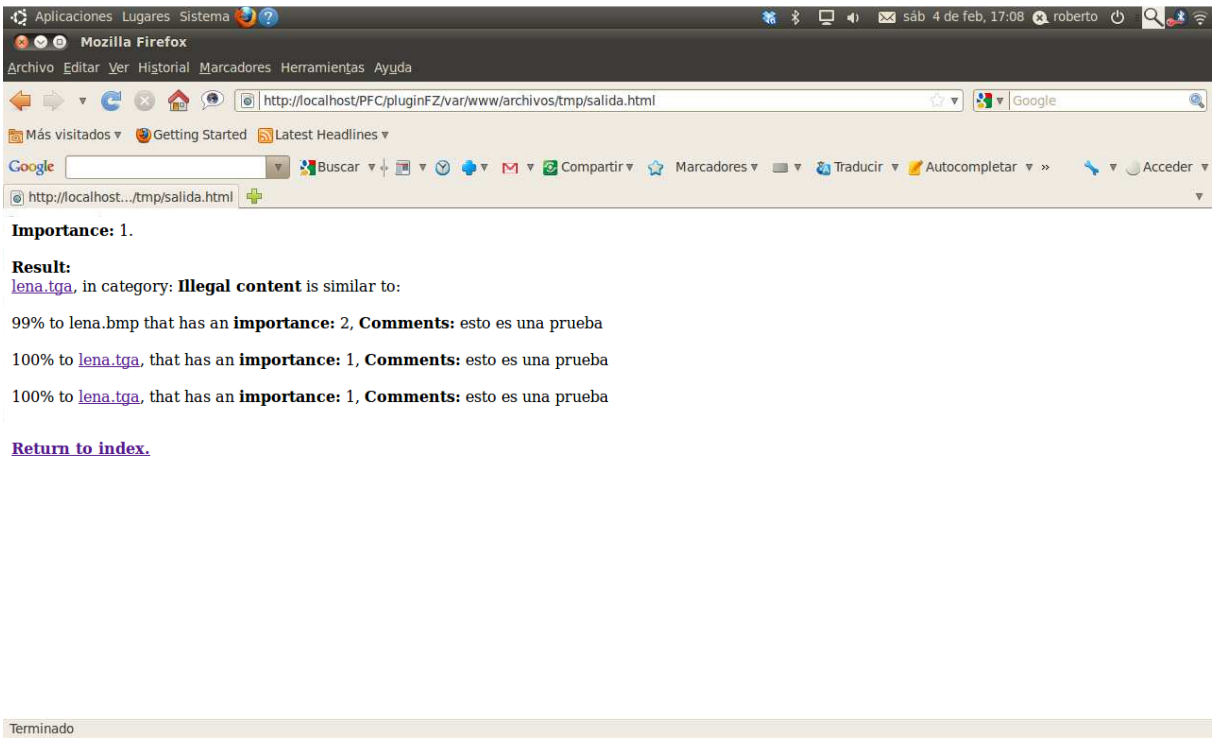


Figura 28. Resultado Test (se encuentran similitudes)

La pantalla anterior muestra el mensaje que se han encontrado semejanzas con los archivos existentes

Este es el diagrama de flujo que tiene nuestra aplicación, cuando el usuario decide actuar en la lista de categorías con las que cuenta el módulo (List categories).

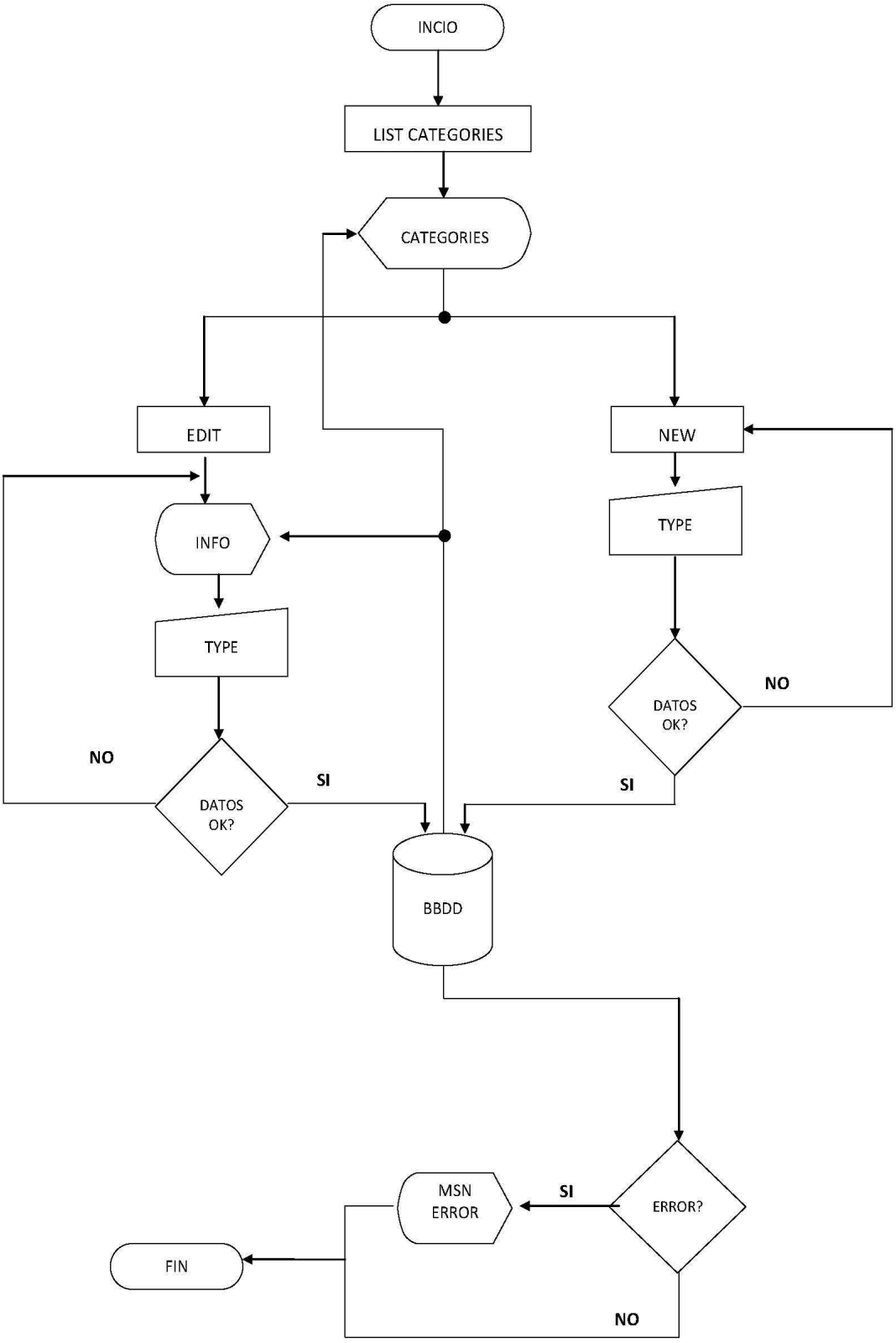


Figura 29. Diagrama de flujo Lista de categorías

A continuación se muestran las distintas pantallas, que se va a ir encontrando el usuario cuando quiera actuar con lista de categorías. En primer lugar se muestran todas las categorías con las que cuenta el módulo. Esta lista de categorías puede ser ampliada y editada por el usuario.

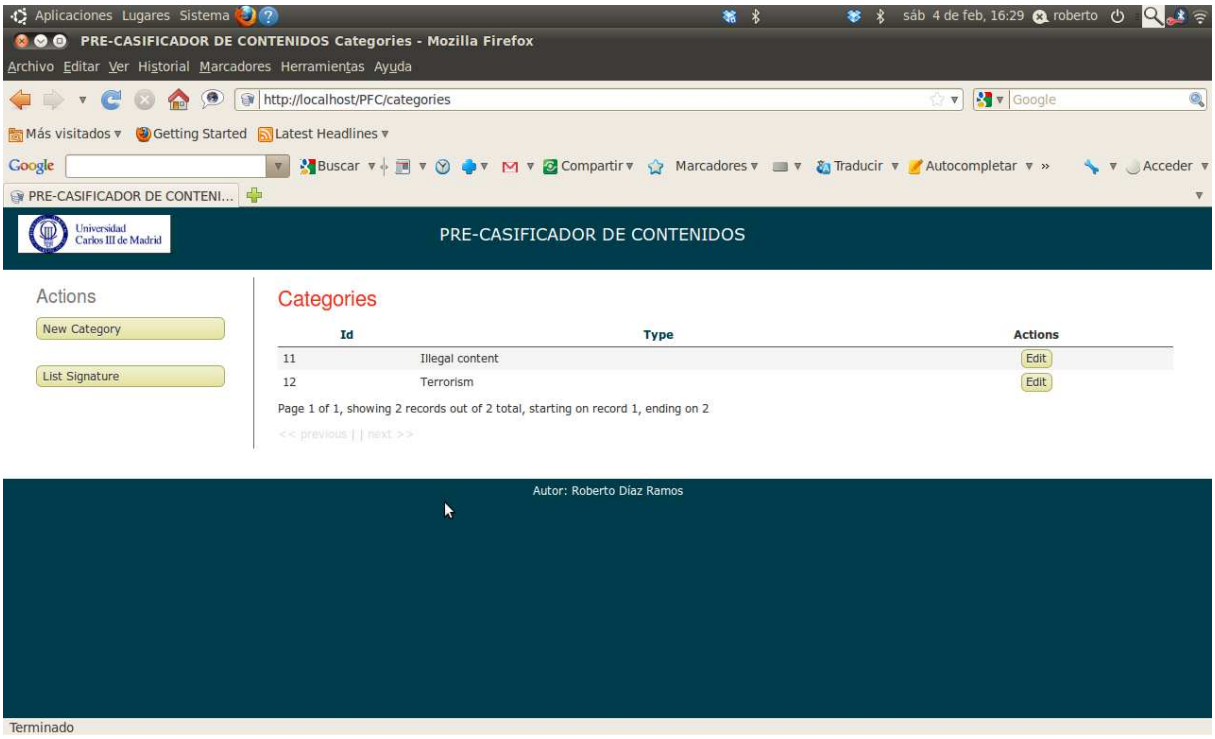


Figura 30. Página de categorías del sistema

Para introducir más categorías en la aplicación basta con pulsar en el botón de "New Category" e introducir el tipo de la nueva categoría que queremos añadir.

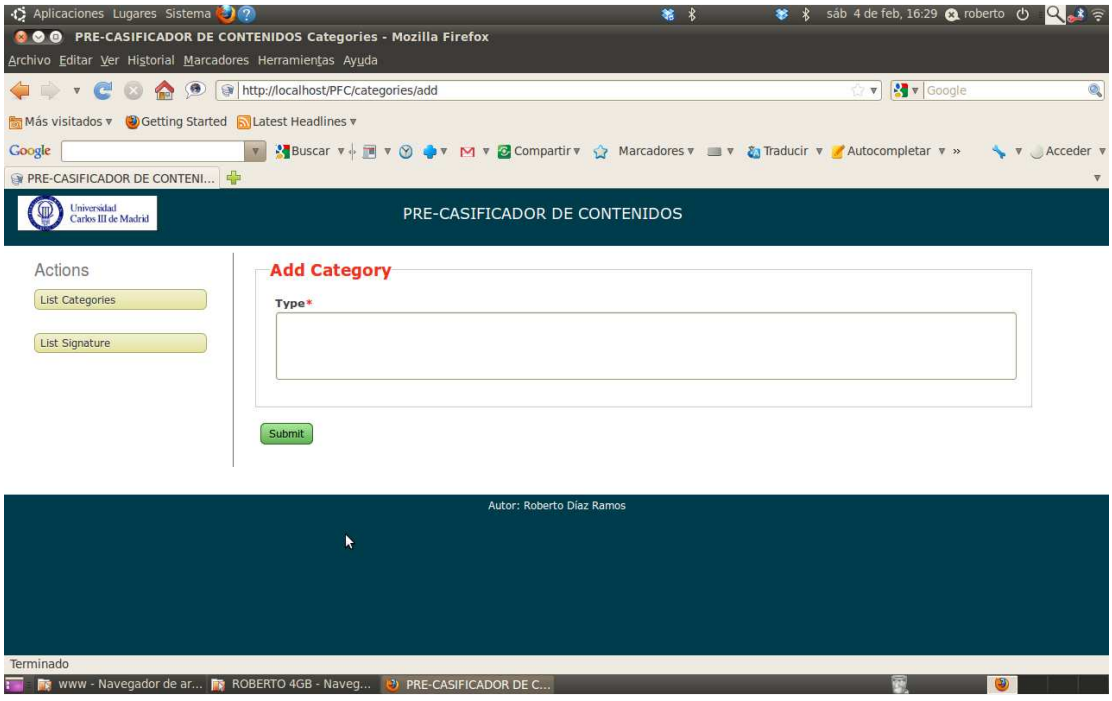


Figura 31. Página de añadir una nueva categoría al sistema

Para modificar el tipo de algunas de las categorías basta con pulsar sobre el botón de "Edit" de la categoría en cuestión, y modificar el nombre de la misma

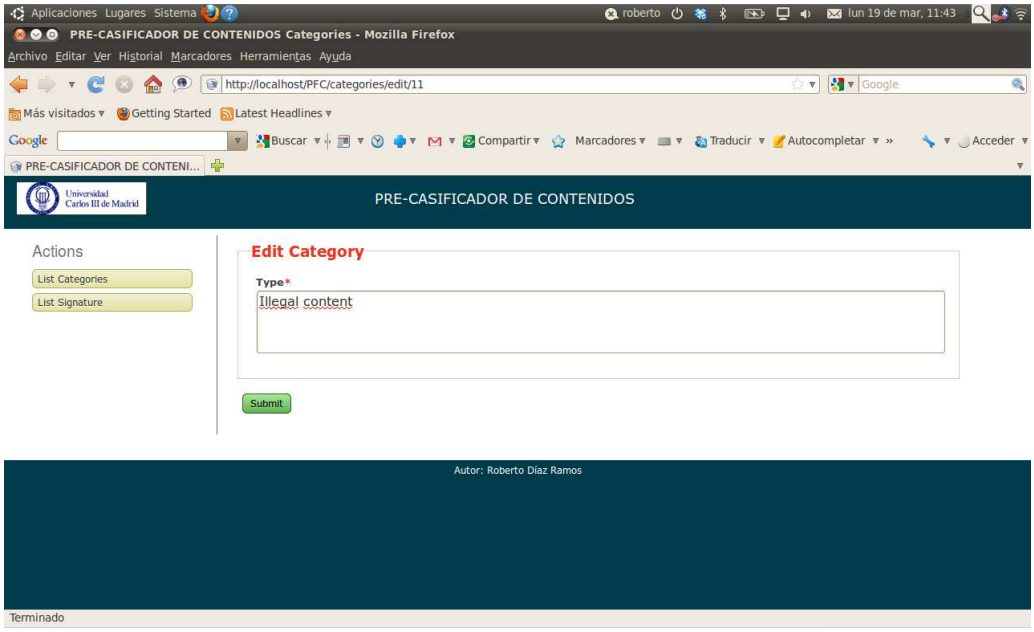


Figura 32. Página de editar una categoría del sistema

Este es el diagrama de flujo de nuestra aplicación, cuando el usuario decide exportar la información del contenido que ha analizado previamente (Export CSV).

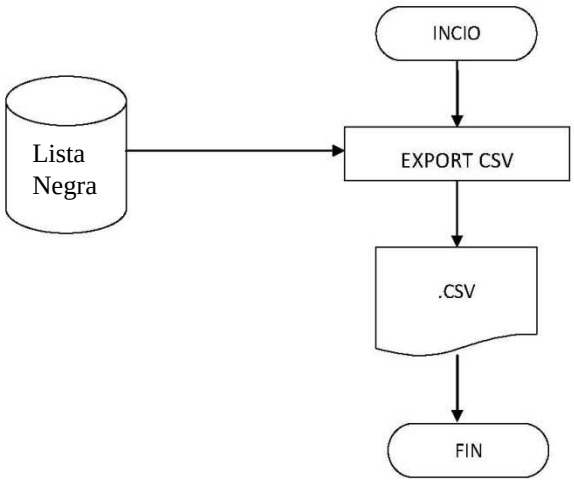


Figura 33. Diagrama de flujo Export CSV

A continuación se muestran las pantallas, que se va a ir encontrando el usuario cuando quiera exportar la información analizada por el módulo a CSV (Comma Separated Values, en inglés).

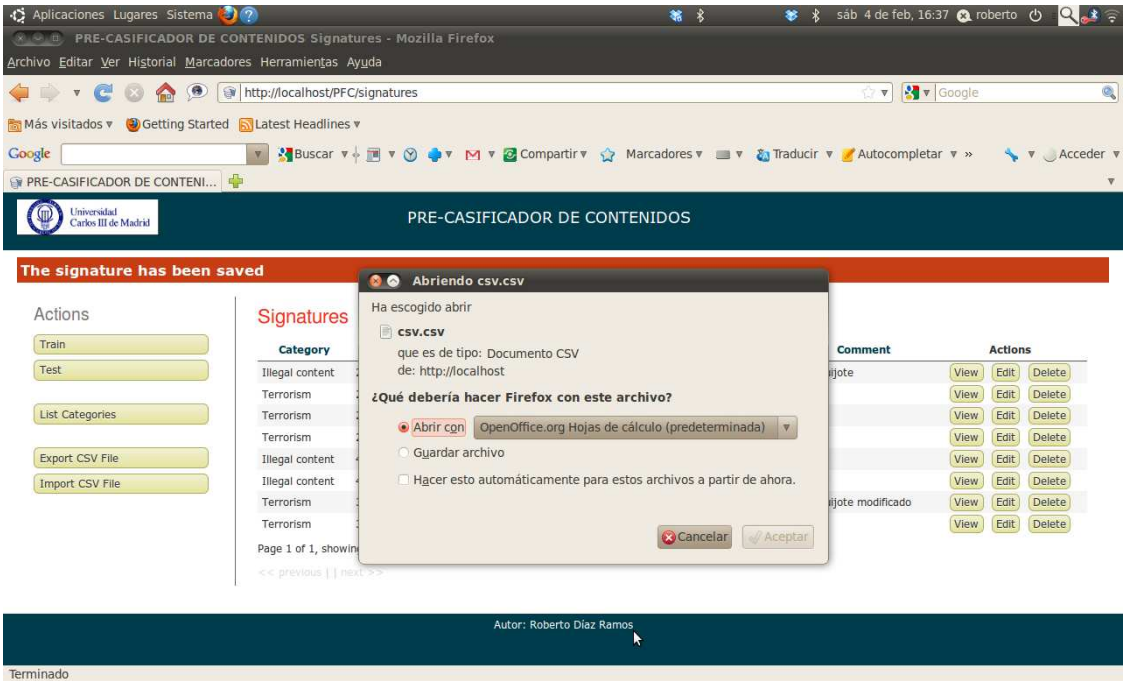


Figura 34. Exportar a CSV la calificación de los contenidos

El formato del fichero generado será el siguiente:

Nombre_del_arcihvo;comentario;categoría;valor; url;firma

Figura 35. Formato fichero CSV

A continuación podemos un ejemplo de un fichero csv generado por nuestra aplicación

lena.bmp;esto es una prueba;Illegal content;2;;12288:QtBDx47B0HBxCvLKotyAudSckf9Wz9gHCRJX1COQHRkrwpZnTzCPagPwNuF7Tuce:UM4ev2oj0VBiMJX51r2ZnqigTFTUtrB

lena.tga;esto es una prueba;Illegal content;1;http://localhost/PFC/pluginFZ/var/www/archivos/train/lena.tga;12288:atBDx47B0HBxCvLKotyAudSckf9Wz9gHCRJX1COQHRkrwpZnTzCPagPwNuF7Tuce:6M4ev2oj0VBiMJX51r2ZnqigTFTUtrB

Cervantes,_Miguel_De_-_Don_Quijote_De_La_Mancha.doc; don quijote;Illegal content;2;http://localhost/PFC/pluginFZ/var/www/archivos/train/Cervantes,_Miguel_De_Don_Quijote_De_La_Mancha.doc;6144:of3hZHzbaH73teCvxNoYUdd6SZAzNmwFvC7yVf0XjqppHbtp63p+qC7XDqtZpQrH:kab9aRUXhho8d

Figura 36. Ejemplo fichero CSV

Este es el diagrama de flujo que tiene la aplicación, cuando el usuario decide importar una lista negra, evitando el entrenamiento de contenidos de forma individual (Import CSV).

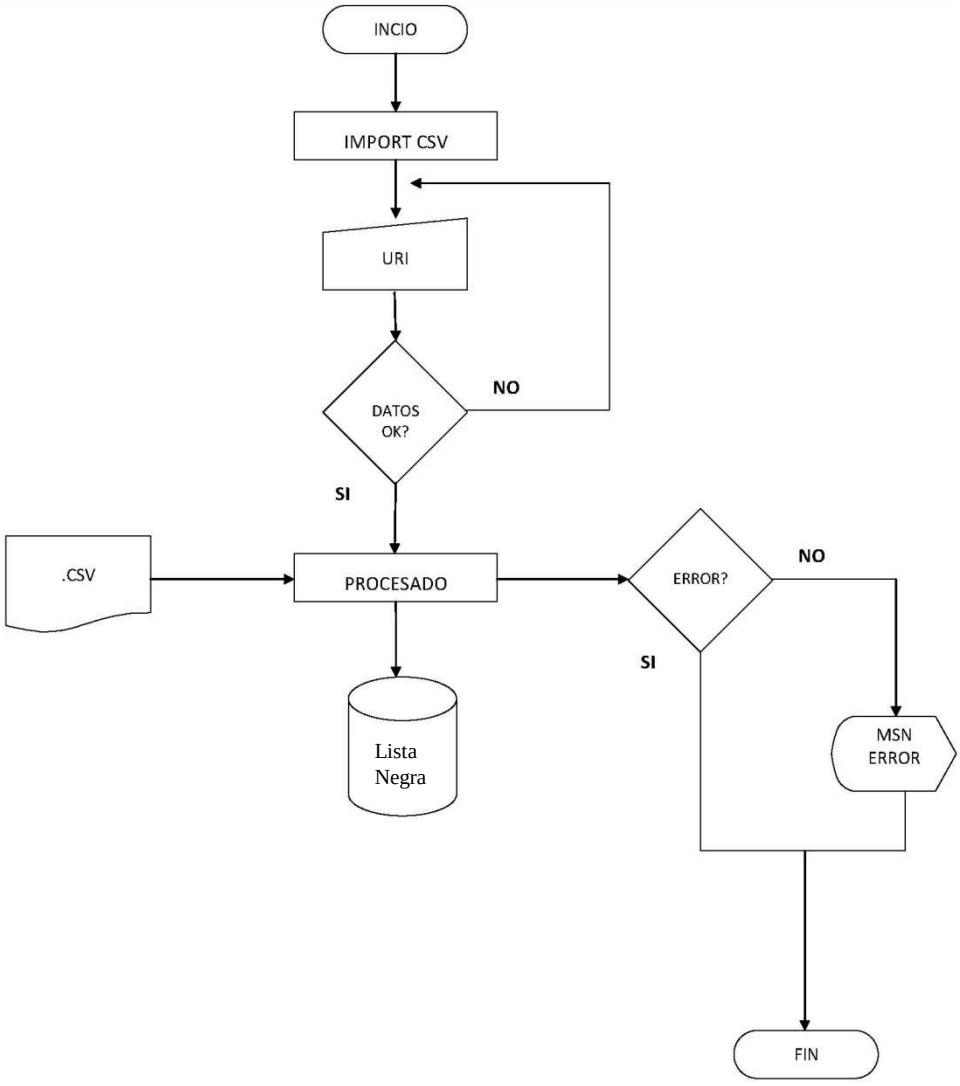


Figura 37. Diagrama Import CSV

A continuación se muestran las pantallas, que se va a encontrar el usuario cuando quiera importar la información de terceros al módulo mediante un fichero CSV.

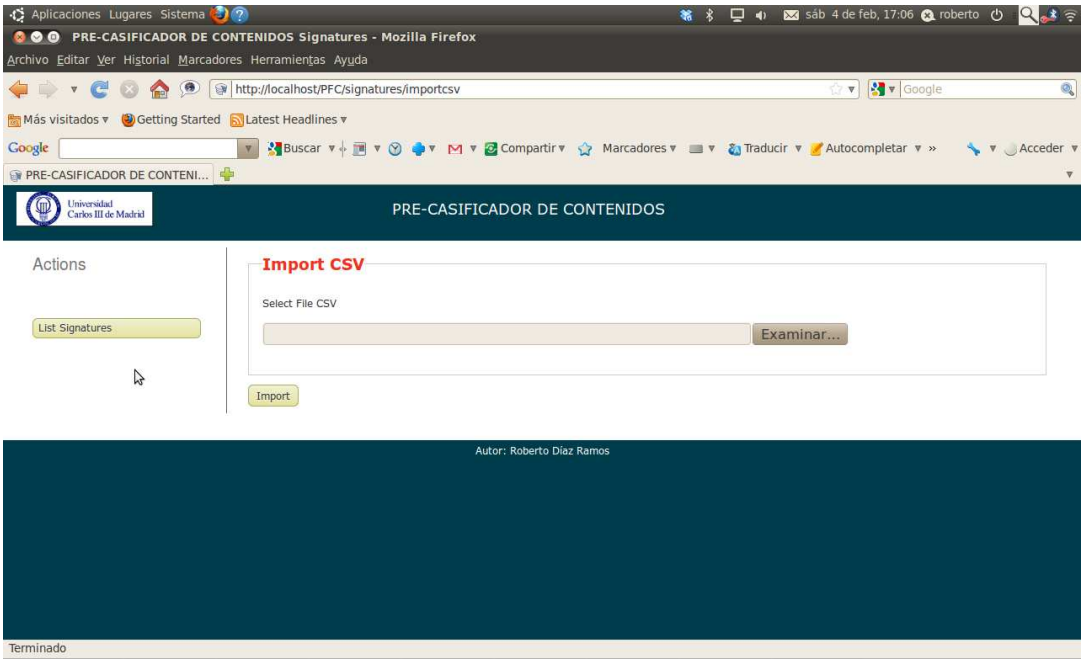


Figura 38. Importar CSV

Este es el diagrama de flujo que tiene la aplicación, cuando el usuario decide visualizar la cache actual de la aplicación.

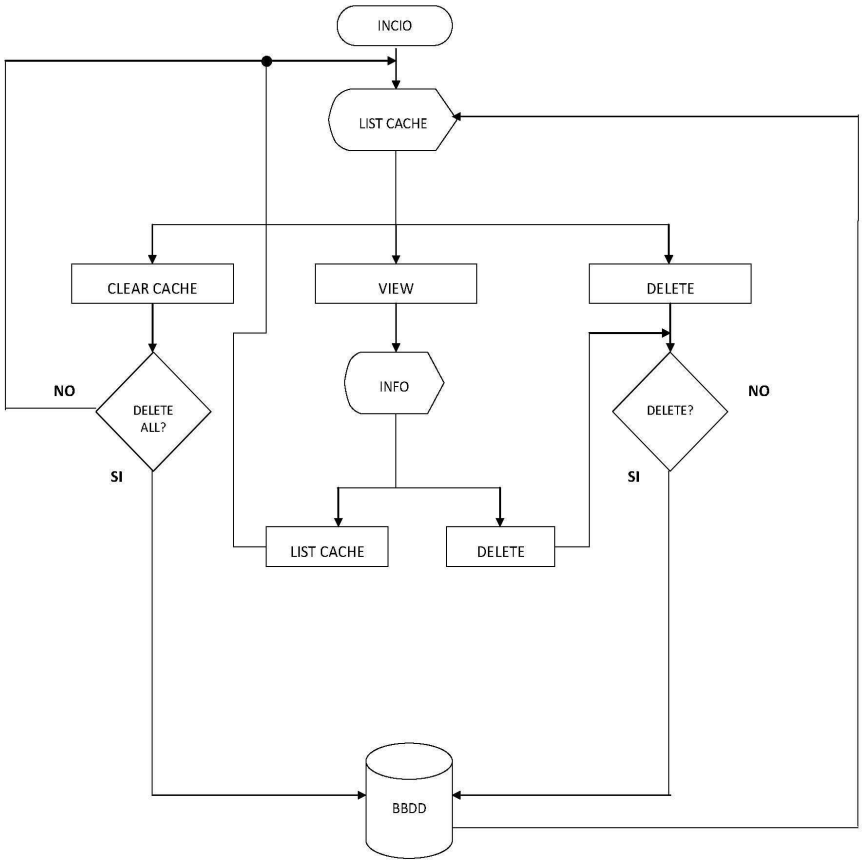


Figura 39. Diagrama de flujo Cache

Además de visualizar la cache del sistema donde se almacena todos los test previos realizados se puede visualizar el contenido con mayor información, se puede borrar una entrada en concreto de la cache y se puede borrar toda la cache.

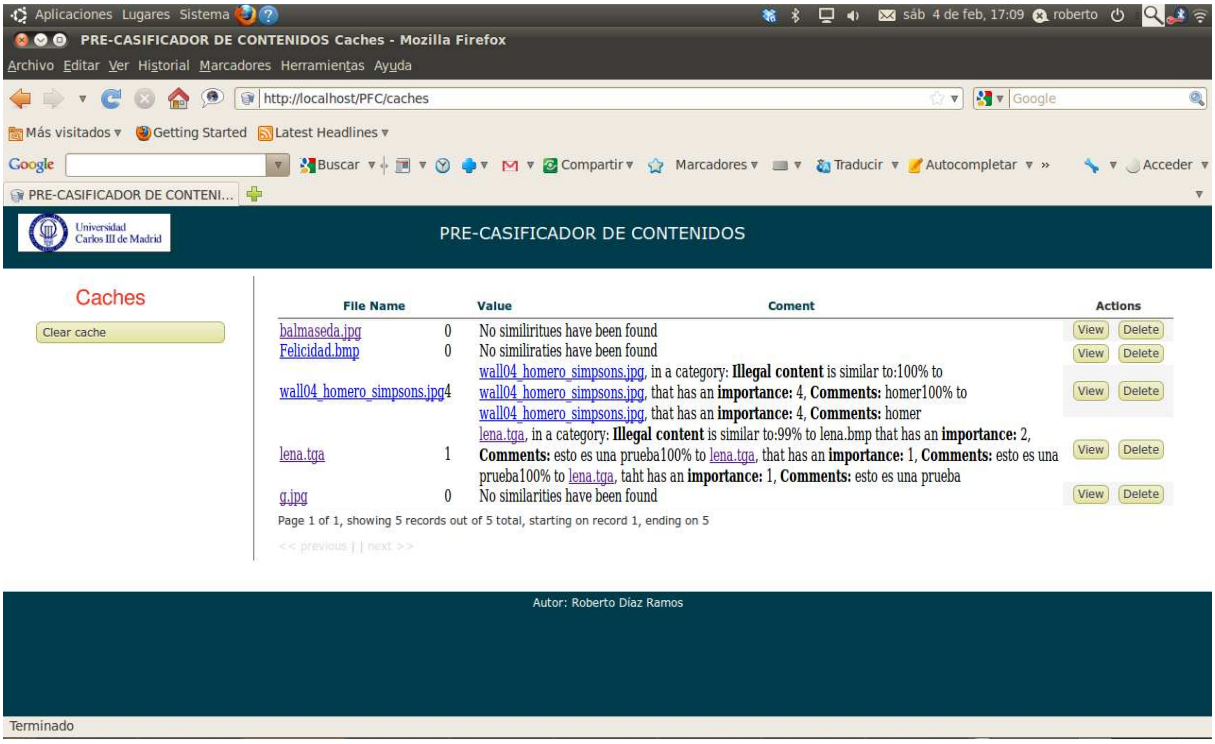


Figura 40. Caché del módulo

La aplicación implementa una caché para agilizar la evaluación del contenido, el único objetivo es no evaluar dos veces el mismo contenido.

4.3 Tecnologías empleadas

En la implementación del sistema de pre-clasificación de contenidos interceptado de forma legal, se han empleado las siguientes tecnologías:

Lenguaje de programación JAVA. Se ha implementado tanto el cálculo de de Fuzzy Hashing, como el sistema de comparación de firmas en este leguaje de programación, con el fin de facilitar la compatibilidad entre el Plugin Manager y los módulos a desarrollar dentro de la plataforma de intercepción legal de comunicaciones que está desarrollando la UC3M. En el desarrollo del cálculo de Fuzzy hash se incluido las modificaciones de seguridad explicadas anteriormente.

Dado que la transformación de contenidos trabaja sobre todo con ficheros se ha decidido hacerlo con scripts, para poder adaptarlos a formatos neutros y poder realizar todas las adaptaciones necesarias para el óptimo funcionamiento de la aplicación.

Y por último se utiliza lenguaje de programación PHP, mediante el framework de CakePHP, para crear el interface Web de la aplicación

Capítulo 5

Evaluación

Para finalizar el trabajo se quiere evaluar la efectividad del fuzzy hashing frente a cambios en tipos de contenidos populares en Internet, en particular imágenes, vídeos, textos. En este capítulo se puede ver cómo afectan distintas modificaciones al cálculo del fuzzy hashing. Y como realizando una serie de modificaciones en los contenidos conseguimos mejorar la efectividad del fuzzy hashing

5.1 Evaluación de Imágenes

Para determinar la eficiencia de fuzzy hashing, a la hora de comprobar ficheros de tipo imagen, se han realizado una serie de experimentos, todas las pruebas parten de la misma imagen de Lena (Ver Figura 41), con un tamaño de 512x512pixeles RGB con una profundidad de 8 bits y representación del color en RGB.

Figura 41. Imagen de ejemplo utilizada en las pruebas.

Para las distintas modificaciones que se realizan a la imagen original, lena.tif se ha usado el programa de edición de imágenes imageMagick en adelante IM, el cual es una potente herramienta de edición de imágenes, que permite realizar modificaciones en las imágenes mediante línea de comandos.

La primera modificación que nos podemos encontrar es una misma imagen en distintos formatos. Para simular esa casuística se procede desde la imagen original a cambiarla de formato mediante el siguiente comando del IM:

convert lena.tif lena.xxx

donde .xxx es la nueva extensión del formato al que se quiere cambiar la imagen original. Con IM se puede cambiar una imagen a multitud de formatos distintos, mediante el siguiente comando:

convert -list format

se obtienen los distintos formatos a los que se puede convertir una imagen con IM:

```
   Format  Module    Mode  Description
------------------------------------------------------------------------------
      3FR  DNG       r--   Hasselblad CFV/H3D39II
       A*  RAW       rw+   Raw alpha samples
       AI  PDF       rw-   Adobe Illustrator CS2
      ART* ART       rw-   PFS: 1st Publisher Clip Art
      ARW  DNG       r--   Sony Alpha Raw Image Format
      AVI* AVI       r--   Microsoft Audio/Visual Interleaved
      AVS* AVS       rw+   AVS X image
       B*  RAW       rw+   Raw blue samples
      BGR* RGB       rw+   Raw blue, green, and red samples
      BMP* BMP       rw-   Microsoft Windows bitmap image
     BMP2* BMP       -w-   Microsoft Windows bitmap image v2
     BMP3* BMP       -w-   Microsoft Windows bitmap image v3
      BRF* BRAILLE   -w-   BRF ASCII Braille format
      BRG* RGB       rw+   Raw blue, red, and green samples
       C*  RAW       rw+   Raw cyan samples
      CAL* CALS      rw-   Continuous Acquisition and Life-cycle Support Type 1 Image
                           Specified in MIL-R-28002 and MIL-PRF-28002
     CALS* CALS      rw-   Continuous Acquisition and Life-cycle Support Type 1 Image
                           Specified in MIL-R-28002 and MIL-PRF-28002
  CAPTION* CAPTION   r--   Image caption
      CIN* CIN       rw+   Cineon Image File
      CIP* CIP       -w-   Cisco IP phone image format
     CLIP* CLIP      -w+   Image Clip Mask
     CMYK* CMYK      rw+   Raw cyan, magenta, yellow, and black samples
```

```
    CMYKA* CMYK      rw+   Raw cyan, magenta, yellow, black, and alpha samples
      CR2  DNG       r--   Canon Digital Camera Raw Image Format
      CRW  DNG       r--   Canon Digital Camera Raw Image Format
      CUR* CUR       rw-   Microsoft icon
      CUT* CUT       r--   DR Halo
      DCM* DCM       r--   Digital Imaging and Communications in Medicine image
                           DICOM is used by the medical community for images like X-
                           rays.  The specification, "Digital Imaging and Communications
                           in      Medicine      (DICOM)",      is      available      at
                           http://medical.nema.org/.   In particular, see part 5 which
                           describes   the   image   encoding   (RLE,   JPEG,   JPEG-LS),
                           and supplement 61 which adds JPEG-2000 encoding.
      DCR  DNG       r--   Kodak Digital Camera Raw Image File
      DCX* PCX       rw+   ZSoft IBM PC multi-page Paintbrush
      DDS* DDS       r--   Microsoft DirectDraw Surface
    DFONT* TTF       r--   Multi-face font package (Freetype 2.3.11)
     DJVU* DJVU      r--   Déjà vu
                  See http://www.djvuzone.org/ for details about the DJVU format.  The
                  DJVU 1.2 specification is available there and at
                  ftp://swrinde.nde.swri.edu/pub/djvu/documents/.
      DNG  DNG       r--   Digital Negative
      DOT  DOT       r--   Graphviz
      DPX* DPX       rw-   SMPTE 268M-2003 (DPX 2.0)
           Digital Moving Picture Exchange Bitmap, Version 2.0.
           See SMPTE 268M-2003 specification at http://www.smtpe.org

     EPDF  PDF       rw-   Encapsulated Portable Document Format
      EPI  PS        rw-   Encapsulated PostScript Interchange format
      EPS  PS        rw-   Encapsulated PostScript
     EPS2* PS2       -w-   Level II Encapsulated PostScript
     EPS3* PS3       -w+   Level III Encapsulated PostScript
     EPSF  PS        rw-   Encapsulated PostScript
     EPSI  PS        rw-   Encapsulated PostScript Interchange format
      EPT  EPT       rw-   Encapsulated PostScript with TIFF preview
     EPT2  EPT       rw-   Encapsulated PostScript Level II with TIFF preview
     EPT3  EPT       rw+   Encapsulated PostScript Level III with TIFF preview
      ERF  DNG       r--   Epson RAW Format
      EXR  EXR       rw-   High Dynamic-range (HDR)
      FAX* FAX       rw+   Group 3 FAX
           FAX machines use non-square pixels which are 1.5 times wider than they
           are tall but computer displays use square pixels, therefore FAX images
           may appear to be narrow unless they are explicitly resized using a
           geometry of "150x100%".

     FITS* FITS      rw-   Flexible Image Transport System
  FRACTAL* PLASMA    r--   Plasma fractal image
      FTS* FTS       rw-   Flexible Image Transport System
        G* RAW       rw+   Raw green samples
       G3* FAX       rw-   Group 3 FAX
      GBR* RGB       rw+   Raw green, blue, and red samples
      GIF* GIF       rw+   CompuServe graphics interchange format
    GIF87* GIF       rw-   CompuServe graphics interchange format (version 87a)
 GRADIENT* GRADIENT  r--   Gradual linear passing from one shade to another
     GRAY* GRAY      rw+   Raw gray samples
      GRB* RGB       rw+   Raw green, red, and blue samples
   GROUP4* TIFF      rw-   Raw CCITT Group4
     HALD*           r--   Identity Hald color lookup table image
HISTOGRAM* HISTOGRAM -w-   Histogram of the image
      HRZ* HRZ       rw-   Slow Scan TeleVision
      HTM* HTML      -w-   Hypertext Markup Language and a client-side image map
     HTML* HTML      -w-   Hypertext Markup Language and a client-side image map
      ICB* TGA       rw+   Truevision Targa image
      ICO* ICON      rw+   Microsoft icon
     ICON* ICON      rw-   Microsoft icon
     INFO  INFO      -w+   The image format and characteristics
   INLINE* INLINE    r--   Base64-encoded inline images
      IPL* IPL       rw+   IPL Image Sequence
   ISOBRL* BRAILLE   -w-   ISO/TR 11548-1 format
      JNG* PNG       rw-   JPEG Network Graphics
           See http://www.libpng.org/pub/mng/ for details about the JNG
           format.
      JP2* JP2       rw-   JPEG-2000 File Format Syntax
      JPC* JPC       rw-   JPEG-2000 Code Stream Syntax
     JPEG* JPEG      rw-   Joint Photographic Experts Group JFIF format (62)
      JPG* JPEG      rw-   Joint Photographic Experts Group JFIF format (62)
      JPX* JPX       rw-   JPEG-2000 File Format Syntax
        K* RAW       rw+   Raw black samples
```

```
      K25  DNG       r--   Kodak Digital Camera Raw Image Format
      KDC  DNG       r--   Kodak Digital Camera Raw Image Format
    LABEL* LABEL     r--   Image label
        M* RAW       rw+   Raw magenta samples
      M2V  MPEG      rw+   MPEG Video Stream
      M4V  MPEG      rw+   Raw MPEG-4 Video
      MAP* MAP       rw-   Colormap intensities and indices
      MAT  MAT       rw+   MATLAB image format
    MATTE* MATTE     -w+   MATTE format
     MIFF* MIFF      rw+   Magick Image File Format
      MNG* PNG       rw+   Multiple-image Network Graphics (libpng 1.2.42)
           See http://www.libpng.org/pub/mng/ for details about the MNG
           format.
     MONO* MONO      rw-   Raw bi-level bitmap
      MOV  MPEG      rw+   MPEG Video Stream
      MP4  MPEG      rw+   MPEG-4 Video Stream
      MPC* MPC       rw+   Magick Persistent Cache image format
     MPEG  MPEG      rw+   MPEG Video Stream
      MPG  MPEG      rw+   MPEG Video Stream
      MRW  DNG       r--   Sony (Minolta) Raw Image File
      MSL* MSL       rw+   Magick Scripting Language
     MSVG  SVG       rw+   ImageMagick's own SVG internal renderer
      MTV* MTV       rw+   MTV Raytracing image format
      MVG* MVG       rw-   Magick Vector Graphics
      NEF  DNG       r--   Nikon Digital SLR Camera Raw Image File
     NULL* NULL      rw-   Constant image of uniform color
        O* RAW       rw+   Raw opacity samples
      ORF  DNG       r--   Olympus Digital Camera Raw Image File
      OTB* OTB       rw-   On-the-air bitmap
      OTF* TTF       r--   Open Type font (Freetype 2.3.11)
      PAL* UYVY      rw-   16bit/pixel interleaved YUV
     PALM* PALM      rw+   Palm pixmap
      PAM* PNM       rw+   Common 2-dimensional bitmap format
  PATTERN* PATTERN   r--   Predefined pattern
      PBM* PNM       rw+   Portable bitmap format (black and white)
      PCD* PCD       rw-   Photo CD
     PCDS* PCD       rw-   Photo CD
      PCL  PCL       rw+   Printer Control Language
      PCT* PICT      rw-   Apple Macintosh QuickDraw/PICT
      PCX* PCX       rw-   ZSoft IBM PC Paintbrush
      PDB* PDB       rw+   Palm Database ImageViewer Format
      PDF  PDF       rw+   Portable Document Format
     PDFA  PDF       rw+   Portable Document Archive Format
      PEF  DNG       r--   Pentax Electronic File
      PFA* TTF       r--   Postscript Type 1 font (ASCII) (Freetype 2.3.11)
      PFB* TTF       r--   Postscript Type 1 font (binary) (Freetype 2.3.11)
      PFM* PFM       rw+   Portable float format
      PGM* PNM       rw+   Portable graymap format (gray scale)
      PGX* PGX       r--   JPEG-2000 VM Format
    PICON* XPM       rw-   Personal Icon
     PICT* PICT      rw-   Apple Macintosh QuickDraw/PICT
      PIX* PIX       r--   Alias/Wavefront RLE image format
    PJPEG* JPEG      rw-   Joint Photographic Experts Group JFIF format (62)
   PLASMA* PLASMA    r--   Plasma fractal image
      PNG* PNG       rw-   Portable Network Graphics (libpng 1.2.42)
           See http://www.libpng.org/ for details about the PNG format.
    PNG24* PNG       rw-   opaque 24-bit RGB (zlib 1.2.3.3)
    PNG32* PNG       rw-   opaque or transparent 32-bit RGBA
     PNG8* PNG       rw-   8-bit indexed with optional binary transparency
      PNM* PNM       rw+   Portable anymap
      PPM* PNM       rw+   Portable pixmap format (color)
  PREVIEW* PREVIEW   -w-   Show a preview an image enhancement, effect, or f/x
        PS  PS       rw+   PostScript
      PS2* PS2       -w+   Level II PostScript
      PS3* PS3       -w+   Level III PostScript
      PSD* PSD       rw+   Adobe Photoshop bitmap
     PTIF* TIFF      rw+   Pyramid encoded TIFF
      PWP* PWP       r--   Seattle Film Works
        R* RAW       rw+   Raw red samples
RADIAL-GRADIENT* GRADIENT  r--   Gradual radial passing from one shade to another
      RAF  DNG       r--   Fuji CCD-RAW Graphic File
      RAS* SUN       rw+   SUN Rasterfile
      RBG* RGB       rw+   Raw red, blue, and green samples
      RGB* RGB       rw+   Raw red, green, and blue samples
     RGBA* RGB       rw+   Raw red, green, blue, and alpha samples
     RGBO* RGB       rw+   Raw red, green, blue, and opacity samples
      RLA* RLA       r--   Alias/Wavefront image
```

```
      RLE* RLE       r--   Utah Run length encoded image
      SCR* SCR       r--   ZX-Spectrum SCREEN$
      SCT* SCT       r--   Scitex HandShake
      SFW* SFW       r--   Seattle Film Works
      SGI* SGI       rw+   Irix RGB image
    SHTML* HTML      -w-   Hypertext Markup Language and a client-side image map
      SR2  DNG       r--   Sony Raw Format 2
      SRF  DNG       r--   Sony Raw Format
  STEGANO* STEGANO   r--   Steganographic image
      SUN* SUN       rw+   SUN Rasterfile
      SVG  SVG       rw+   Scalable Vector Graphics (XML 2.7.6)
     SVGZ  SVG       rw+   Compressed Scalable Vector Graphics (XML 2.7.6)
     TEXT* TXT       rw-   Text
      TGA* TGA       rw+   Truevision Targa image
THUMBNAIL* THUMBNAIL -w+   EXIF Profile Thumbnail
     TIFF* TIFF      rw+   Tagged Image File Format (LIBTIFF, Version 3.9.2)
   TIFF64* TIFF      ---   Tagged Image File Format (64-bit) (LIBTIFF, Version 3.9.2)
     TILE* TILE      r--   Tile image with a texture
      TIM* TIM       r--   PSX TIM
      TTC* TTF       r--   TrueType font collection (Freetype 2.3.11)
      TTF* TTF       r--   TrueType font (Freetype 2.3.11)
      TXT* TXT       rw-   Text
     UBRL* BRAILLE   -w-   Unicode Text format
      UIL* UIL       -w-   X-Motif UIL table
     UYVY* UYVY      rw-   16bit/pixel interleaved YUV
      VDA* TGA       rw+   Truevision Targa image
    VICAR* VICAR     rw-   VICAR rasterfile format
      VID* VID       rw+   Visual Image Directory
     VIFF* VIFF      rw+   Khoros Visualization image
      VST* TGA       rw+   Truevision Targa image
     WBMP* WBMP      rw-   Wireless Bitmap (level 0) image
      WMF* WMF       r--   Windows Meta File
      WMV  MPEG      rw+   Windows Media Video
      WMZ* WMZ       r--   Compressed Windows Meta File
      WPG* WPG       r--   Word Perfect Graphics
        X* X         rw+   X Image
      X3F  DNG       r--   Sigma Camera RAW Picture File
      XBM* XBM       rw-   X Windows system bitmap (black and white)
       XC* XC        r--   Constant image uniform color
      XCF* XCF       r--   GIMP image
      XPM* XPM       rw-   X Windows system pixmap (color)
      XPS  XPS       r--   Microsoft XML Paper Specification
       XV* VIFF      rw+   Khoros Visualization image
      XWD* XWD       rw-   X Windows system window dump (color)
        Y* RAW       rw+   Raw yellow samples
    YCbCr* YCbCr     rw+   Raw Y, Cb, and Cr samples
   YCbCrA* YCbCr     rw+   Raw Y, Cb, Cr, and alpha samples
      YUV* YUV       rw-   CCIR 601 4:1:1 or 4:2:2

* native blob support
r read support
w write support
+ support for multiple images
```

De la población de formatos que ofrece IM para trabajar este estudio se centra en una muestra lo más representativa posible. Los formatos que forman parte de esta muestra son los que se pueden encontrar con mayor frecuencia en la red. La muestra está compuesta por:

- **fli,** (formato de animación Autodesk)
- **ico**, (icon Microsoft)
- **dcm,** (Imagen digital y imagen de comunicaciones en medicina)
- **bmp,** (Microsoft Windows bitmap)
- **psd,** (Adobe Photoshop bitmap)
- **gif,** (CompuServe graphics interchange format)
- **sgi,**(Irix RGB image)
- **jpg,** (Joint Photographic Experts Group JFIF format)
- **pbm,** (Portable bitmap format (black and white)

- **pgm,** (Portable graymap format (gray scale))
- **png,** (Portable Network Graphics (libpng 1.2.42))
- **pnm,** (Portable anymap)
- **ppm,** (Portable pixmap format (color))
- **im1,** (imagen SUM Rasterfile)
- **tga,** (Truevision Targa image)
- **tiff,**(Tagged Image File Format)
- **xbm,** (X Windows system bitmap (black and white)
- **xpm,** (X Windows system pixmap (color))
- **pcx,** (ZSoft IBM PC multi-page Paintbrush)
- **pat,** (Predefined pattern)
- **gih,** (Pincel GIMP animado)
- **pse,** (Proyecto Adobe Photoshop)
- **jpf,** (JPEG-2000 File Format Syntax)
- **raw,** (Raw red samples)
- **pxr,** (Formato Pixar)

Ahora se procede a comparar todos los formatos entre sí con fuzzy hashing, para comprobar si fuzzy hashing es sensible a cambios de formatos de una imagen. Para ello se construye una matriz en la cual tanto las filas como las columnas tienen los distintos formatos que se comparan. Y en las intersecciones entre ellas se indican el porcentaje de similitud de la misma imagen codificada en los dos formatos comparados. Los resultados obtenidos siguiendo este sistema son:

	Fli	icom	dcm	bmp	psd	gif	sgi	jpeg	pbm	pgm	png	pnm	ppm	im1	tga	tiff	xbm	xpm	pcx	pat	gih	pse	jpf	raw	pxr
fli	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
icom	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
dcm	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	97%	0%	0%	0%	99%	0%	0%	0%	86%	99%
bmp	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
psd	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
gif	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
sgi	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
jpg	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
pbm	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
pgm	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
png	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
pnm	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
ppm	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
im1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
tga	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
tiff	0%	0%	97%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	97%	0%	0%	0%	88%	97%
xbm	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
xpm	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
pcx	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
pat	0%	0%	99%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	97%	0%	0%	0%	100%	0%	0%	0%	86%	99%
gih	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
Pse	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	80%	0%	0%	0%
jpf	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
raw	0%	0%	86%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	88%	0%	0%	0%	86%	0%	0%	0%	100%	86%
pxr	0%	0%	99%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	97%	0%	0%	0%	99%	0%	0%	0%	86%	100%

Tabla 5. Comparación de formatos de imagen con fuzzy hashing

Como se puede comprobar en la matriz anterior, fuzzy hashing no detecta las transformaciones del formato de las imágenes, por lo que se puede pensar que cuando se detecte la necesidad de calcular el hash difuso de una imagen con fuzzy hashing, se tendría que ir a un formato neutral, el cual no produzca modificaciones en el contenido del fichero.

Por lo que el objetivo es convertir los distintos formatos a un formato neutral para que se puedan comprara entre sí y el resultado de esta comparación sea positivo.

Las características estructurales de este formato neutral deberían ser:

- Formato sencillo y estructurado, para que la conversión a este no suponga una sobrecarga computacional a la máquina.
- Mayor número de colores posible, (la imagen no perderá información)
- Sin compresión para que no exista perdida de información y los cambios no se propaguen por el fichero

A prior se consideró que los formatos de imagen que podrían ser utilizados como un formato neutral son **bmp**, **pcx, gif, tga, jpg, tiff.** Pero como se puede comprobar en la siguiente tabla, en función de las características que se buscan en el formato ideal de neutralidad, **bmp** el formato idóneo.

FORMATO	VENTAJAS	ICONVENIENTES	Nº MÁXIMO DE COLORES	MÉTODOS DE COMPRESIÓN
PCX	Simplicidad y rapidez	Escasa compresión	16 millones	RLE
GIF	Compresión	256 colores	256 colores	LZW
TGA	Simplicidad y estructuración	Escasa compresión	16 millones	Sin comprimir. RLE
JPG	Impresionante compresión	Modifica la imagen original, y complejidad	16 millones	Jpg
BMP	Sencillez y estructuración	Escasa compresión	16 millones	Sin comprimir RLE8, RLE4
TIF	Polivalencia total	Complejidad y variedad	16 millones	Sin comprimir RLE LZW CCITT G3 u G4 y JPG

Tabla 6. Comparativa de formatos de imagen

Por lo que se procede a convertir todas las imágenes de los distintos formatos a **Windows bitmap** (.bmp).

Dentro de las distintas configuraciones que puede soportar el formato bmp, la conversión es a una imagen con una profundidad de color de 8 bits, o lo que es lo mismo 256 colores (RGB), y sin compresión ni máscara de color.

También se puede observar como hay ciertos formatos que originalmente son compatibles entre sí, como son:

- dcm, tiff, pat, raw y pxr.

El motivo por el cual estos formatos son compatibles entre sí es porque tiene características muy similares, entre ellas se encuentra que estos formatos o no tienen compresión o su compresión es sin perdidas. Además todos ellos tienen la misma profundidad de color

Se procede convertir las imágenes en los distintos formatos a bmp y se comparan entre sí, con fuzzy hashing, siguiendo el mismo método enunciado anteriormente. El resultado de la comparación pasando al formato neutro (.bmp) es:

	fli	icom	dcm	bmp	psd	gif	sgi	jpeg	pbm	pgm	png	pnm	ppm	im1	tga	tiff	xbm	xpm	pcx	pat	gih	pse	jpf	raw	pxr
fli	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
icom	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
dcm	99%	99%	100%	99%	99%	0%	100%	0%			99%	100%	100%	99%	100%	99%		0%	99%	99%	99%	99%	99%	99%	99%
bmp	99%	99%	99%	100%	99%	0%	99%	0%			100%	99%	99%	99%	99%	99%		0%	99%	99%	99%	99%	99%	99%	99%
psd	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
gif	0%	0%	0%	0%	0%	100%	0%	0%			0%	0%	0%	0%	0%	0%		100%	0%	0%	0%	0%	0%	0%	0%
sgi	99%	99%	100%	99%	99%	0%	100%	0%			99%	100%	100%	99%	100%	99%		0%	99%	99%	99%	99%	99%	99%	99%
jpeg	0%	0%	0%	0%	0%	0%	0%	100%			0%	0%	0%	0%	0%	0%		0%	0%	0%	0%	0%	0%	0%	0%
pbm									100%																
pgm										100%															
png	99%	99%	99%	100%	99%	0%	99%	0%			100%	99%	99%	99%	99%	99%		0%	99%	99%	99%	99%	99%	99%	99%
pnm	99%	99%	100%	99%	99%	0%	100%	0%			99%	100%	100%	99%	100%	99%		0%	99%	99%	99%	99%	99%	99%	99%
ppm	99%	99%	100%	99%	99%	0%	100%	0%			99%	100%	100%	99%	100%	99%		0%	99%	99%	99%	99%	99%	99%	99%
im1	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
tga	99%	99%	100%	99%	99%	0%	100%	0%			99%	100%	100%	99%	100%	99%		0%	99%	99%	99%	99%	99%	99%	99%
tiff	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
xbm																	100%								
xpm	0%	100%	0%	0%	0%	100%	0%	0%			0%	0%	0%	0%	0%	0%		100%	0%	0%	0%	0%	0%	0%	0%
pcx	100%	100%	99%	99%	100%	0%	0%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
pat	100%	100%	99%	99%	100%	0%	0%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
gih	100%	100%	99%	99%	100%	0%	0%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
pse	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
jpf	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
raw	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	100%		0%	100%	100%	100%	100%	100%	100%	100%
pxr	100%	100%	99%	99%	100%	0%	99%	0%			99%	99%	99%	100%	99%	1000%		0%	100%	100%	100%	100%	100%	100%	100%

Tabla 7. Comparación formatos bmp con fuzzy hashing

Con esta transformación fuzzy hashing detecta similitudes en la mayoría de los ficheros, con varias excepciones.

Una de ellas proviene de los ficheros .pbm y .xbm que convierten la representación de la imagen a B&N y el formato .pgm, que convierte la representación de la imagen a escala de grises. Estos cambios de formato conllevan un cambio radical en la paleta de colores de las imágenes, con lo que se pierde muchísima información de la misma. Al comparar estos formatos con el resto no se obtiene ninguna similitud.

La otra excepción es en los ficheros:

- gif (CompuServe graphics interchange format)
- jpg (Joint Photographic Experts Group JFIF format)
- xpm (X Windows system pixmap (color))

Estos formatos de ficheros conllevan algoritmos de compresión por lo que nuestro objetivo es realizar transformaciones adicionales en estos ficheros para conseguir que fuzzy hashing identifique la similitud entre ellos.

El primero paso para ver las posibles modificaciones que se tendrían que realizar a los distintos ficheros es analizar sus metadatos, es decir:

- Existencia de comentarios
- Tamaño
- Paleta de colores
- Profundidad de color
- Resolución
- Representación de píxeles

Este análisis se realiza mediante el comando de IM:

identify –verbose fichero

una vez analizados todos los ficheros en los formatos “conflictivos”, obtenemos que todos ellos tienen las mismas características a excepción de la representación de pixel. Mediante la representación de pixel, podemos clasificar los ficheros en los distintos formatos en dos grupos distintos

- Palette

- Truecolor

Los ficheros con formato gif y xpm tienen una representación de pixel Truecolor, mientras que el resto de ficheros tienen una representación de pixel tipo palette. Por lo que el objetivo es realizar una transformación de todos los ficheros a una misma representación de pixel, y que fuzzy hashing pueda dar un valor alto de semejanza entre ficheros. Esta transformación de la representación de pixel la realizo mediante el comando

convert –type palette input output

Con el comando anterior se consigue el objetivo buscado, es decir, cambiar todos los ficheros a una representación de pixel del tipo *palette.* En esta representación los colores se hallan definidos en un fichero de IM (paleta de colores) y cada pixel se refiere a una entrada de la tabla de colores definida.

De este modo se consigue que fuzzy hashing detecte similitudes del 99% - 100% entre los formatos gif y xpm. Y el resto de formatos analizados.

Llegados a este punto, se ha conseguido que fuzzy hashing detecte similitudes entre los distintos formatos analizados con las salvedades de los ficheros .jpg. Ninguna modificación efectuada a estos ficheros hace que fuzzy hashing detecte similitudes. Esto es debido a la compresión con pérdidas que realiza jpg. Esta compresión con pérdidas implica una modificación del histograma, lo que hace imposible que fuzzy hashing detecte similitudes entre los ficheros.

Algunos de los formatos de imagen analizados pueden integrar comentarios o metadatos de la imagen, por lo que sería conveniente eliminar esa información de la imagen para analizarla con fuzzy hashing. Esto se consigue mediante el comando de IM

convert –strip

el cual elimina de los ficheros los metadatos de cualquier perfil que contenga la imagen.

En la siguiente tabla se puede comprobar el grado de similitud que conseguimos obtener con las modificaciones anteriormente propuestas. Como se puede comprobar se consigue una alta eficiencia de fuzzy hashing en el análisis de imágenes, que antes no se tenía.

	fli	icon	dcm	bmp	psd	gif	sgi	jpeg	pbm	pgm	png	pnm	ppm	im1	tga	tiff	xbm	xpm	pcx	pat	gih	pse	jpf	raw	pxr
Fli	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
icon	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
dcm	99%	99%	100%	99%	99%	99%	100%	0%			99%	100%	100%	99%	100%	99%		99%	99%	99%	99%	99%	99%	99%	99%
bmp	99%	99%	99%	100%	99%	99%	99%	0%			100%	99%	99%	99%	99%	99%		99%	99%	99%	99%	99%	99%	99%	99%
Psd	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		99%	100%	100%	100%	100%	100%	100%	100%
Gif	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
sgi	99%	99%	100%	99%	99%	99%	100%	0%			99%	100%	100%	99%	100%	99%		99%	99%	99%	99%	99%	99%	99%	99%
jpeg	0%	0%	0%	0%	0%	0%	0%	100%			0%	0%	0%	0%	0%	0%		0%	0%	0%	0%	0%	0%	0%	0%
pbm									100%																
pgm										100%															
png	99%	99%	99%	100%	99%	99%	99%	0%			100%	99%	99%	99%	99%	99%		99%	99%	99%	99%	99%	99%	99%	99%
pnm	99%	99%	100%	99%	99%	99%	100%	0%			99%	100%	100%	99%	100%	99%		99%	99%	99%	99%	99%	99%	99%	99%
ppm	99%	99%	100%	99%	99%	99%	100%	0%			99%	100%	100%	99%	100%	99%		99%	99%	99%	99%	99%	99%	99%	99%
im1	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		99%	100%	100%	100%	100%	100%	100%	100%
tga	99%	99%	100%	99%	99%	99%	100%	0%			99%	100%	100%	99%	100%	99%		99%	99%	99%	99%	99%	99%	99%	99%
tiff	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		99%	100%	100%	100%	100%	100%	100%	100%
xbm																	100%								
xpm	100%	100%	99%	99%	99%	100%	99%	0%			99%	99%	99%	99%	99%	99%		100%	100%	100%	100%	100%	100%	100%	100%
pcx	100%	100%	99%	99%	100%	100%	100%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
pat	100%	100%	99%	99%	100%	100%	100%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
gih	100%	100%	99%	99%	100%	100%	100%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
pse	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
jpf	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
raw	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	100%		100%	100%	100%	100%	100%	100%	100%	100%
pxr	100%	100%	99%	99%	100%	100%	99%	0%			99%	99%	99%	100%	99%	1000%		100%	100%	100%	100%	100%	100%	100%	100%

Tabla 8. Comparación de formatos tras la transformación

Jpg es el único formato de los analizados que modifica el histograma de la imagen, haciendo casi imposible que fuzzy hashing encuentre semejanzas en los ficheros. Además el único formato analizado que realiza compresión con pérdidas, mientras que el resto de los formatos analizado o no tienen compresión o tienen compresión sin pérdidas del tipo LZW, RLE o ZIP.

Una vez que se ha conseguido que fuzzy hashing detecte similitudes entre la mayoría de formatos de ficheros, el siguiente paso es intentar conseguir que fuzzy hashing detecte similitudes ante modificaciones de brillo, contraste y pequeñas modificaciones, como pueden ser marcas de agua, marcos o alguna modificación en la imagen.

La modificación del brillo o contraste de una imagen supone la modificación de todos los bits de la misma y no de forma lineal, por lo que resulta imposible que fuzzy hashing detecte igualdad entre la imagen original y la modificada. Se realizan distintas modificaciones para intentar que se detecte similitud. Pero todas han sido fallidas, entre ellas se han realizado:

- Cambiar el espacio de colores de la imagen
- Reducir la profundidad de color
- Modificar la representación de pixel
- Convertirla a B&N
- Normalizar la imagen
- Ecualizar el histograma

La más mínima modificación en esta particularidad de la imagen hace que fuzzy hashing falle. Se ha probado con modificaciones de brillo de +5%

Figura 42. Lena original

Figura 43. Lena con el brillo modificado

A simple vista las imágenes parecen idénticas pero una tiene un 5% más de brillo que la otra, este cambio tan poco significativo es suficiente para que fuzzy hashing no encuentre similitud entre ambas imágenes.

Algo similar sucede con el contraste de la imagen:

Figura 44. Lena original

Figura 45. Lena con el contraste modificado

La siguiente modificación planteada, es introducir un marco a la imagen de un grosor de 1 pixel, esta modificación supone un cambio en el 0,78% de los pixeles de la imagen. Con esta pequeña modificación fuzzy hashing no detecta similitud entre las imágenes. Esto es debido al funcionamiento de fuzzy hashing, ya que en todos los bloques en los que divide la imagen, tienen variaciones sobre la original. En las imágenes adjuntas se puede comprobar la modificación que supone la inserción de dicho marco:

Figura 46. Lena original

Figura 47. Lena con marco

La siguiente prueba realizada es la introducción de una marca de agua a forma de firma en la imagen. Esta modificación es la más habitual que nos podemos encontrar en la red. En las imágenes siguientes se puede comprobar el impacto visual que supone dicha modificación.

Figura 48. Lena original

Figura 49. Lena firmada

La modificación realizada en la imagen original es la inserción de una imagen a modo de firma con un tamaño de 50x50 y una transparencia del 70%. Fuzzy hashing encuentra una similitud del 82% entre las dos imágenes. Con independencia en la posición en la que insertemos la firma, fuzzy hashing sigue encontrado similitudes superiores al 80% entre las dos imágenes.

Figura 50. Lena original

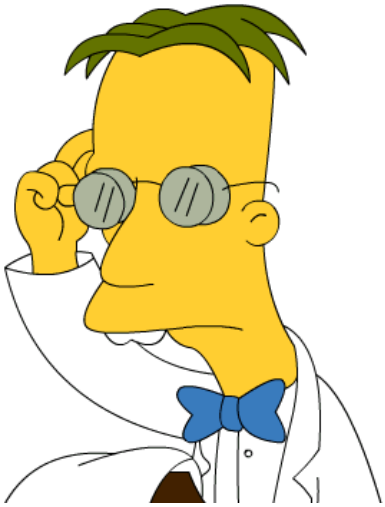


Figura 51. Dr. Frink

Una vez analizados los diferentes formatos de imagen y que se ha llegado a un compromiso de transformaciones en las mismas para que fuzzy hashing detecte similitudes entre la mayoría de ellas, se compara el fichero de Lena con otros ficheros de tamaño idéntico, para comprobar si fuzzy hashing produce falsos positivos. Tras la comparación de Lena con un conjunto de imágenes distintas a esta visualmente, entre la que se encuentra la imagen del Dr. Frink llegamos a la conclusión que a priori fuzzy hashing no produciría falsos positivos.

Tras los estudios realizados anteriormente sobre las distintas modificaciones realizadas sobre la imagen de Lena, podemos confirmar que dos imágenes que tienen una similitud del 70% son la misma imagen con alguna modificación visual. En la aplicación se decide

establecer un umbral algo más restrictivo para asegurarnos la ausencia de falsos positivos. El umbral establecido en la aplicación es del 80%

5.2 Evaluación de Videos

En este apartado se estudia el comportamiento de fuzzy hashing frente a ficheros de video. Actualmente existen diversos tipos de ficheros de video y todos ellos cuenta con distintos sistemas de compresión y códec. Además en los formatos de compresión multimedia se han ido incorporando contendores multimedia, los cuales encapsulan video, audio, subtítulos, capítulos, meta-datos e información de sincronización siguiendo un formato preestablecido en su especificación. Las pistas de vídeo y audio suelen ir comprimidas, siendo distintos los códec utilizados dentro de cada uno de los contenedores, estos códec son los encargados de descomprimir la información en aras a su reproducción.

En la comparativa entre los distintos formatos de video, se han utilizado los más habituales:

- MPEG-4
- REALVIDEO
- WMV
- Xvid
- DivX
- ACC

Y los contenedores multimedia analizados han sido:

- 3GP
- AVI
- FLV
- Matroska
- QuickTime
- RealMedia

En primer lugar se utiliza un fichero de video en formato AVI, este fichero va a ser el que vamos a utilizar como original y de este mismo fichero se van a realizar las transformaciones a los distintos formatos de ficheros enunciados anteriormente. Para la conversión entre los distintos formatos se utilizan distintos programas de edición de videos entre los que se encuentra vitualDub 1.9.11, todos ellos en entorno gráfico. Por lo que en un principio sería imposible la conversión entre los distintos formatos por línea de comandos.

Figura 52. Video original

La siguiente tabla muestra como fuzzy hashing no detecta similitudes entre los distintos formatos. Lo cual era de esperar porque cada formato utiliza un sistema y un índice de compresión distinto, e incorporan al fichero multitud de metadatos en su cabecera.

	MPEG4	REALVIDEO	WMV	Xvid	DivX	ACC	3GP	AVI	FLV	Matroska	QuickTime
MPEG-4	100%	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
REALVIDEO	0 %	100%	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
WMV	0 %	0 %	100%	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Xvid	0 %	0 %	0 %	100%	0 %	0 %	0 %	0 %	0 %	0 %	0 %
DivX	0 %	0 %	0 %	0 %	100%	0 %	0 %	0 %	0 %	0 %	0 %
ACC	0 %	0 %	0 %	0 %	0 %	100%	0 %	0 %	0 %	0 %	0 %
3GP	0 %	0 %	0 %	0 %	0 %	0 %	100%	0 %	0 %	0 %	0 %
AVI	0 %	0 %	0 %	0 %	0 %	0 %	0 %	100%	0 %	0 %	0 %
FLV	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	100%	0 %	0 %
Matroska	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	100%	0 %
QuickTime	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	100%

Tabla 9. Comparativa formatos de video

El siguiente paso es comprobar si fuzzy hashing es capaz de detectar similitudes entre distintos ficheros ante ciertas modificaciones de los mismos, como pueden ser:

- Supresión de partes del video
- Inserción de nuevos frames de video en el fichero
- Supresión de audio
- Inserción de una marca de agua.

Fuzzy hashing detecta similitudes entre dos ficheros de video idénticos a los que a uno de ellos se le han suprimido una serie de frames. Se suprimen frames en distintas posiciones del fichero de video, tanto al inicio, como al final como en partes intermedias del mismo, y los resultados obtenidos han sido muy similares. Como dato representativo, la supresión de un 18% de los frames de un fichero video hace que fuzzy hashing detecte una similitud en torno al 90% respecto al original.

El comportamiento de fuzzy hashing ante la inserción de frames, es análogo a la supresión de los mismos, es decir, se realizan pruebas de inserción de frames en distintas posiciones del fichero. Y para la inserción de un 16 % frames en el fichero original hemos obtenido una similitud del 87%.

Estos dos resultados son muy positivos para la investigación, ya que si un fichero de video puede ser manipulado para eliminar de él cierta información o se le añade información adicional, fuzzy hashing es capaz de detectar las similitudes en él.

La siguiente prueba realizada es la supresión del canal de audio a un fichero de video, y los resultados han sido bastante positivos, ya que fuzzy hashing ha detectado una similitud en torno 64% entre ambos ficheros.

Y por último se comparan el fichero de video original con un fichero de video idéntico a él, al que se le ha añadido una marca de agua con una opacidad del 30%. El resultado de la comparación con fuzzy hashing ha sido positivo ya que ha determinado que entre el fichero original y el modificado con la marca de agua existe una similitud del 54%

Figura 53. Video con marca de agua

Como conclusión podemos se puede determinar que fuzzy hashing es capaz de detectar similitudes entre distintos ficheros de video a los cuales se les han realizado una serie de modificaciones, pero no es capaz de encontrar similitudes entre ficheros de video en distintos formatos, debido a los distintos metadatos y cabeceras que añada cada formato de video a los ficheros.

Fuzzy hashing falla incluso pasando los distintos formatos de video a un formato neutro. En este caso nos decantaríamos por MPEG-4, por ser un algoritmo de compresión con bajas perdidas, frente al resto de formatos analizados que implementas algoritmos de compresión con altas perdidas. El resultado de la comparación una vez convertidos los videos de los distintos formatos a MPEG-4 es análogo al mostrado en la Tabla 9. Comparativa formatos de video de la presente memoria.

5.3 Evaluación de Textos

Ahora procederemos al estudio de la eficiencia de las hashes difusas en documentos de texto, mediante fuzzy hashing. Para determinar la eficiencia de fuzzy hashing se parte de un fragmento de la novela “El ingenioso hidalgo don Quijote de la mancha”, dicho fragmento está compuesto por:

- 9 páginas
- 6.460 Palabras
- 29.473 caracteres
- 72 párrafos
- 567 líneas.

Dicho texto es guardado en distintos formatos de texto, entre estos formatos se encuentran formatos de texto enriquecido tipo ODT, DOC, DOCX, PDF y formatos de texto plano tipo ASCII, UNICODE Y UTF-8. Se procede a comprar entre sí los distintos ficheros con fuzzy hashing.

En la siguiente matriz se observan los resultados obtenidos con la comparación entre los distintos formatos:

	ODT	**DOC**	**DOCX**	**PDF**	**ASCII**	**UTF-16**	**UTF-8**
ODT	100%	0%	0%	0%	0%	0%	0%
DOC	0%	100%	0%	0%	0%	60%	0%
DOCX	0%	0%	100%	0%	0%	0%	0%
PDF	0%	0%	0%	100%	0%	0%	0%
ASCII	0%	0%	0%	0%	100%	0%	100%
UTF-16	0%	60%	0%	0%	0%	100%	0%
UTF-8	0%	0%	0%	0%	100%	0%	100%

Tabla 10. Comparativa formatos texto

Se puede comprobar en la tabla anterior que fuzzy hashing no encuentra similitudes entre los distintos formatos de texto, lo cual esperábamos ya que cada formato tiene cabeceras distintas y distinta codificación de los caracteres. Ante esta situación no podemos hacer nada para que aumentar la eficiencia de fuzzy hashing.

A priori para que fuzzy hashing encontrase similitudes entre los distintos formatos analizados, tendríamos que transformarlos a un formato común, como se hizo en el caso de las imágenes. En este caso la transformación se tendría que realizar a un formato de texto plano, por ejemplo UTF-8, el motivo por el cual se tendría que realizar una transformación a un formato de texto plano, es debido a que los formatos .doc, .docx, etc. incorporan cabeceras e información ajena al propio contenido del documento. Estos metadatos distorsionan la comparación. Otro motivo por el cual se considera que sería conveniente la transformación de los ficheros de texto a un formato de texto plano, es como se podrá ver más adelante, tienen un mejor comportamiento ante posibles modificaciones en su contenido que los formatos enriquecidos.

Actualmente se tiene la problemática de no poder contar con una aplicación que mediante la línea de comandos pueda realizar la transformación de los distintos formatos de fichero a texto plano, como lo realiza IM para las imágenes.

Se procede a analizar si fuzzy hashing es capaz de encontrar similitudes entre ficheros en el mismo formato, pero con distintas modificaciones, como pueden ser:

- Sustitución de una letra por otra.
- Sustitución de palabras.
- Inserción de un párrafo.
- Supresión de párrafos.
- Cambio del tipo de letra
- Cambio del tamaño de la letra

La siguiente prueba que se realiza es sustituir un carácter dentro del contenido del texto por otro. Partiendo del fichero original, se realizan los siguientes cambios.

Se cambian todas las letras “e” por “z”, con este cambio se modifican un total de 1780 caracteres dentro del texto, lo que supone un 12,76 % de los caracteres del texto. Este cambio produce que fuzzy hashing sea incapaz de detectar similitudes entre el fichero original y el modificado, en cualquiera de los formatos analizados.

Viendo que fuzzy hashing no ha sido capaz de encontrar similitudes entre dos ficheros a los que uno de ellos se les ha modificado un 12,6% de los caracteres al azar, se procede a realizar cambios que impliquen un porcentaje menor de lo caracteres modificados.

Se cambian los caracteres “r” por “s”, lo que supone una modificación del 6% de los caracteres del fichero, obteniendo idénticos resultados. En una última prueba se realiza un cambio del 1% de los caracteres del fichero al azar. Ante esta modificación fuzzy hashing tampoco es capaz de encontrar similitudes entre el fichero original y modificado.

En ninguno de los formatos de texto analizados fuzzy hashing es capaz de encontrar similitudes entre ficheros a los cuales se les modifican caracteres al azar.

El motivo por el cual fuzzy hashing no es capaz de encontrar similitudes entres ficheros que han sufrido modificaciones al azar en sus caracteres, es que estas modificaciones se producen en todos o casi todas las partes en las que fuzzy hashing divide al fichero de entrada. Por lo que el hash de cada una de esas partes es distinto al original.

Ahora Se procede a realizar una modificación similar a la anteriormente mencionada, ahora en lugar de cambiar caracteres, procedemos a cambiar palabras enteras en el texto de los ficheros.

Se realizan tres cambios distintos de palabras, para determinar el umbral a partir del cual fuzzy hashing es capaz de detectar similitudes en los ficheros, a los cuales se les ha cambiado una palabra por otra. Los umbrales con los que se han trabajado han supuesto la modificación del 6%, 0.2% y del 0.07% de las palabras del documento. A pesar de trabajar con cambios tan poco representativos, fuzzy hashing no ha sido capaz de encontrar similitudes entre el fichero original y el modificado en los formatos de texto enriquecido .odt, .doc, .docx y .pdf

Sin embargo en los formatos de texto plano, fuzzy hashing ha considerado que los ficheros en los que hemos realizado modificaciones del 0.2% y del 0.07% de las palabras son similares al original en un porcentaje superior al 80%. Sin embargo, para el fichero en el que hemos modificado el 6% de sus palabras, fuzzy hashing no encuentra similitudes con el original

Los datos exactos obtenidos para los formatos de texto plano han sido:

Umbral / Formato	6%	0.2%	0.07%
ANSI	0 %	83 %	97 %
UNICODE	0 %	90 %	99 %
UTF-8	0 %	80 %	99 %

Tabla 11. Modificación de palabras

El motivo por el cual fuzzy hashing no es eficiente ante cambio de palabras a azar a partir de un umbral es análogo al que se ha justificado en el cambio de caracteres, ya que al cambiar el contenido del fichero en distintas partes del mismo, hace que fuzzy hashing obtenga hashes parciales de los distintos trozos analizados distintos al original, formando de esta manera una huella digital del fichero modificado completamente distinta a la original.

El motivo por el cual fuzzy hashing no detecta similitudes por las modificaciones sufridas en los ficheros de texto enriquecido .doc, .docx, .odt, .pdf, es debido a las cabeceras que insertan tanto Open office como Microsoft office a los ficheros.

La siguiente prueba a realizar es la inserción de distintos párrafos al inicio, final y puntos intermedios del texto, al igual que se realizo con la inserción de palabras, se insertan párrafos de distinto tamaño, para ver si fuzzy hashing es capaz de encontrar similitudes entre el fichero modificado y el original.

Creamos 4 párrafos de prueba, de distintos tamaños:

Párrafo 1: 138 palabras lo que supone un 2% del total del texto

Párrafo 2: 322 palabras lo que supone un 4% del total del texto

Párrafo 3: 729 palabras lo que supone un 11% del total del texto

Párrafo 4: 1458 palabras lo que supone un 22% del total del texto

Una vez insertado los distintos párrafos en las distintas posiciones se obtiene que la comparación de firmas no encuentra similitudes para los formatos de texto enriquecido, mientras que para los formatos de texto plano, obtiene similitudes superiores al 80% entre el fichero original y el modificado, con independencia de la posición en el texto en la que se insertan los párrafos de prueba.

A modo estimativo, los resultados obtenidos con la inserción de los distintos párrafos en el texto han sido los indicados en la siguiente tabla, cabe destacar que el punto de inserción del párrafo no afecta al porcentaje de similitud calculado por fuzzy hashing.

Umbral / Formato	2 %	4 %	11 %	22 %
ANSI	99 %	97 %	93 %	97 %
UNICODE	99 %	97 %	93 %	95 %
UTF-8	90 %	85 %	83 %	80 %

Tabla 12. Inserción de párrafos

Con la Extracción de párrafos, los resultados obtenidos por fuzzy hashing por la supresión de párrafos en el texto original son análogos a los obtenidos por la inserción de párrafos de longitudes similares.

Por último se realiza un cambio del tipo de letra y del tamaño de la misma en ficheros de texto enriquecido, hace que fuzzy hashing detecte similitudes superiores al 95%. Las pruebas han sido realizadas con el control de cambios desactivado, y con la utilización de varios estilos en el documento.

Como se ha podido comprobar, los hashes difusos no encuentran similitudes entre ficheros con el mismo contenido y en distinto formato. Esto es debido a las cabeceras que cada programa de edición de texto inserta a los documentos y la forma de codificar los caracteres. Los hashes difusos tienen un comportamiento bastante bueno ante pequeños cambios del contenido del fichero, siempre y cuando sean en una zona concreta del mismo y no se disperse dicho cambio a lo largo de todo el fichero. La modificación de estilos, ya sean tamaños de letra o tipos de letra en editores de texto de alto nivel son despreciables para encontrar similitudes entre hashes difusas, ya que esta información va insertada en la cabecera de este tipo de fichero.

Capítulo 6

Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Roberto Díaz Ramos
2.- Departamento:
Departamento de Ingeniería Telemática
3.- Descripción del Proyecto:
- Titulo: **Desarrollo de un preclasificador de contenidos basado en hashes difusos para una plataforma de intercepción legal de contenidos**
- Duración (meses): **12**
Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
					0,00	
Manuel Urueña		Ingeniero Senior	2	4.289,54	8.579,08	
Roberto Díaz Ramos		Ingeniero	5	2.694,39	13.471,95	
					0,00	
					0,00	
		Hombres mes	**7**	**Total**	22.051,03	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Portatil acer Aspire 5735	700,00	100	7	60	81,67
Portatil hp compact 6730b	500,00	100	5	60	41,67
Microsoft 2007	325,00	100	3	60	16,25
Ubuntu 10.4	0,00	100	7	60	0,00
Java 1.6	0,00	100	7	60	0,00
Squlite 3.7.10	0,00	100	7	60	0,00
Apache 5.0.5	0,00	100	7	60	0,00
PHP 5.4.0	0,00	100	7	60	0,00
CakePHP 1.3	0,00	100	7	60	0,00
					0,00
				Total	139,58

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Reuniones con director de proyecto		
KM		24,00
parking		30,00
Impresión y presentación PFC		170,00
	Total	224,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	22.051
Amortización	140
Subcontratación de tareas	0
Costes de funcionamiento	224
Costes Indirectos	4.483
Total	26.898

Figura 54. Presupuesto PFC

CAPÍTULO 6: PRESUPUESTO

Para la ejecución del proyecto se estima un coste total de 26.898 €, de los cuales un 80% corresponde a las horas de los ingenieros que han intervenido en él. Para este proyecto se ha necesitado la utilización de dos recursos, uno con perfil de ingeniero *senior* encargado de la gestión y coordinación del mismo y otro con perfil de ingeniero o ingeniero junior, encargado del desarrollo y documentación de la aplicación.

A estos ingenieros se les dota el siguiente equipamiento:

- Portátil acer Aspire 5735
- Portátil hp compact 6730b
- Microsoft 2007
- Ubuntu 10.4
- Java 1.6
- Sqlite 3.7.10
- Apache 5.0.5
- PHP 5.4.0
- CakePHP 1.3

Que utilizaran durante la vida del proyecto, este material se ha estimado que tiene un periodo de depreciación de 5 años, por lo que se repercutirá su coste por el periodo de tiempo que se ha utilizado. Esto ha supuesto un coste de 139,58€

Por último para la ejecución de este proyecto también se ha tenido en cuenta los costes que supondrían las diversas reuniones entre los intervinientes en el mismo, así como el coste de encuadernación del proyecto. Lo que suma un total de 224 €

Capítulo 7

Conclusiones y Trabajos futuros

7.1 Conclusiones

Este Proyecto final de carrera, se ha ocupado de desarrollar un sistema de pre-clasificación de contenidos ilícitos para una plataforma de interceptación legal de las comunicaciones. Gracias a la implementación de Fuzzy hashing (hashes difusas) y la modificación de los contenidos a formatos neutros, se ha conseguido que el pre-clasificador implementado sea capaz de identificar contenidos similares, de forma que los sospechosos no puedan eludir fácilmente la detección de los contenidos ilegales intercambiados.

En el estudio de la eficiencia del sistema implementado nos hemos encontrado con que no en todos los casos estudiados esta similitud ha sido obtenida de forma directa. Tras el estudio de las distintas casuísticas, se ha obtenido las transformaciones que se tienen que realizar a los ficheros a analizar para que Fuzzy Hashing detecte dichas similitudes. Y también se ha descubierto qué modificaciones que hacen muy difícil que Fuzzy Hashing pueda encontrar similitudes entre contenido homologo.

Debido a la multitud de formatos existentes, y que para ficheros de video e imagen, los códec y los formatos de los mismos se actualizan diariamente, supondría un trabajo de investigación continuo.

Sin embargo para la detección de similitudes en ficheros de texto Fuzzy Hashing es una herramienta bastante eficaz, capaz de detectar inserción y eliminación de párrafos dentro de un fichero, modificación de estilos, etc.…

En cualquier caso, se considera que el sistema implementado es una aplicación muy válida para los objetivos perseguidos en este Proyecto Final de Carrera, ya que aparte de conseguir detectar similitudes entre distintos contenidos, mediante la transformación a algún formato neutro del fichero analizado. No hemos encontrado ninguna colisión en las firmas de los distintos ficheros analizados.

7.2 Trabajos futuros

Básicamente, el trabajo futuro se puede dividir en dos direcciones. Por un lado, se puede tratar de desarrollar aplicaciones capaces de preparar a los ficheros a analizar, para que la aplicación detecte similitudes de forma más eficiente. Estas aplicaciones tendrían que:

- Convertir de texto a texto plano
- Convertir distintos formatos de video a formato estándar

Y por otro lado sería la integración del modulo desarrollado con Xplico, trabajo que está muy avanzado ya que se ha desarrollado la aplicación con las mismas tecnologías utilizadas por Xplico.

La integración del modulo con Xplico sería similar a la siguiente figura en la que se representa una arquitectura lógica de la integración.

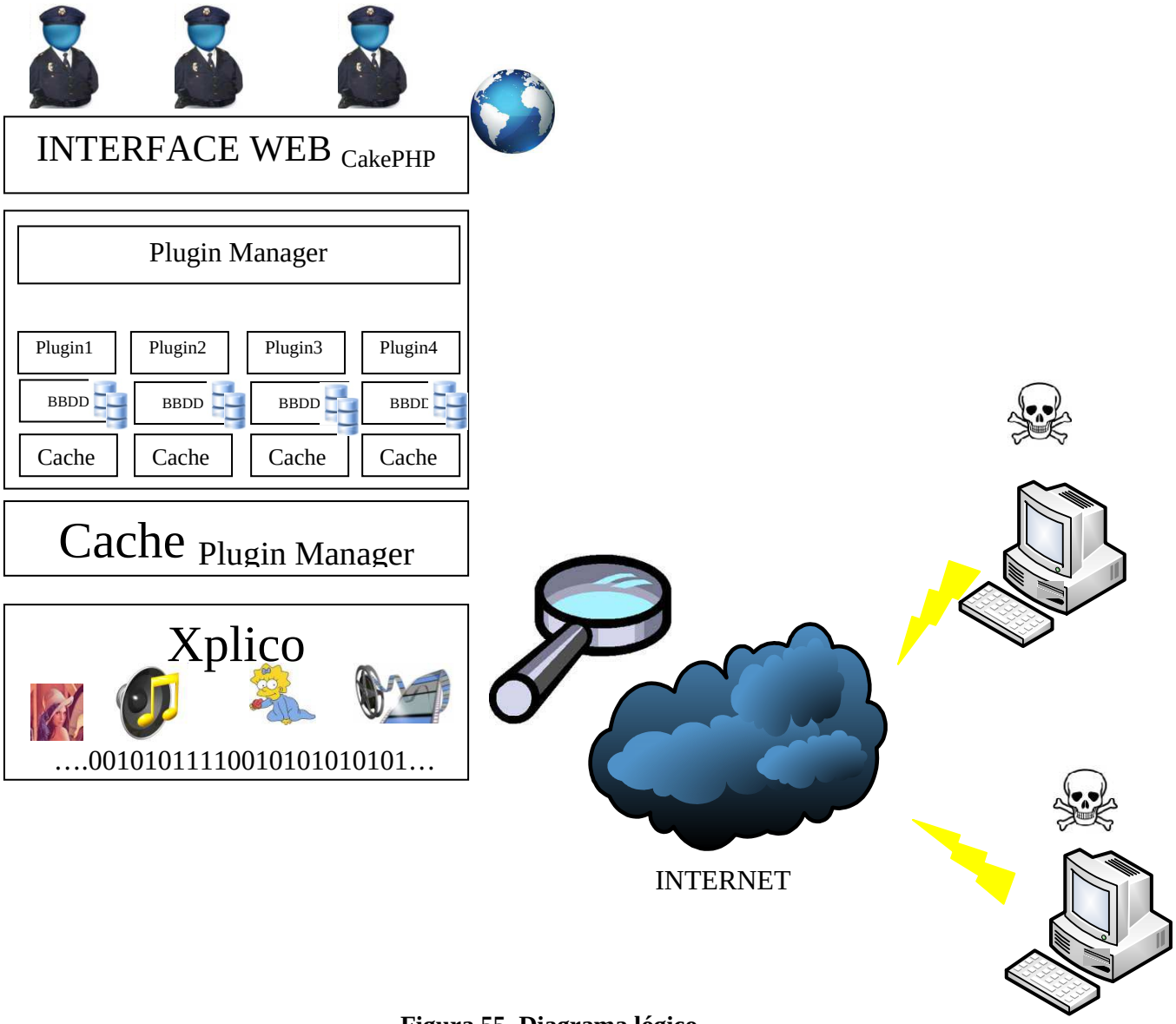


Figura 55. Diagrama lógico

La primera capa sería la encargada de mostrar la aplicación, este interface Web estaría implementado en CakePHP y sería el punto de acceso de los usuarios a la aplicación. El usuario empezaría a escuchar la red con Xplico y el plugin manager que se integraría con este sería el encargado de distribuir el contenido a analizar entre los distintos plugin/complementos que tuviese el sistema implementado, entre ellos el modulo desarrollado en este PFC, para el análisis de información.

Como se puede ver en la representación anterior cada plugin contaría con sus propias bases de datos, donde almacenan sus listas de contenidos y su propia de cache con el fin de ahorrar tiempo de cómputo. De igual forma el plugin manager también debería contar con su propia cache para ahorrar mensajes entre los complementos del sistema y aumentar la eficiencia del sistema.

Otro trabajo futuro que se podría realizar a partir de este PFC es integrar tanto el módulo desarrollado en este, como el resto de módulos desarrollados en paralelo que se engloban dentro del proyecto de investigación de la UC3M, de creación de una plataforma de

intercepción legal de comunicaciones, en una arquitectura física en alta disponibilidad La cual debería tener una estructura similar a la siguiente figura.

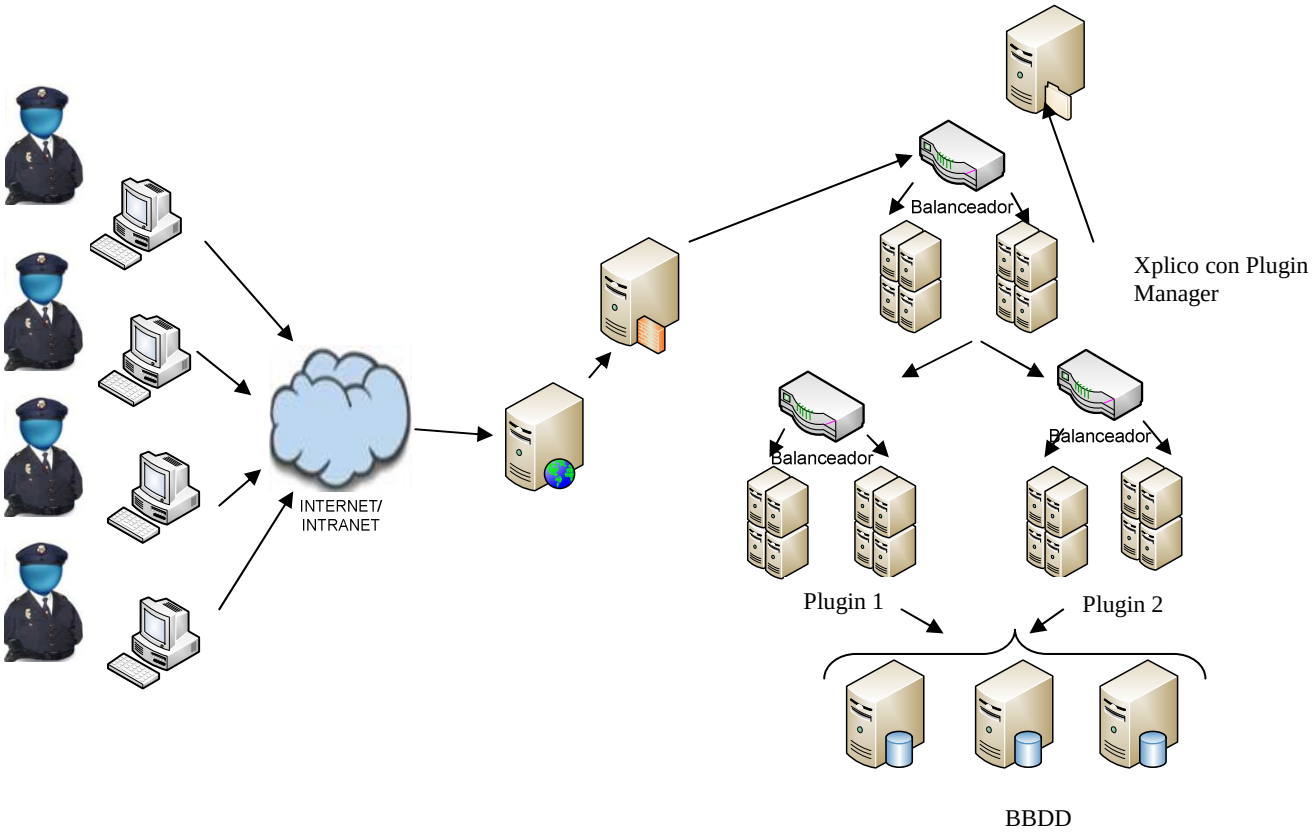


Figura 56. Arquitectura física en alta disponibilidad.

En la figura anterior se puede ver como los analistas de las fuerzas de seguridad podrían acceder desde sus equipos al sistema. Los usuarios accederían a un servidor en frontend el cual haría las funciones de frontal web y en el que se debería implementar la gestión de usuarios mediante LDAP. El frontal web sería la única comunicación del usuario con la aplicación, quedando totalmente asilado en backend los servidores de base de datos, servidor de archivos y servidores de aplicaciones (módulo Manager y demás módulos que se engloban dentro del proyecto de investigación de la UC3M). Se considera necesario que se tuviera replicadas las máquinas para evitar problemas por caídas de sistema y posibles ataques por denegación de servicios (**DoS**). La primera parte de la aplicación sería la granja de servidores de Xplico/plugin Manager. Esta granja de servidores analizaría el tráfico de la red o los paquetes interceptados mientras que el plugin manager sería el encargado de distribuir los distintos contenidos a los distintos plugin con los que contaría el sistema, cada uno especializado en un sistema de análisis.

La granja en la que se encuentra Xplico/plugin Manager debería existir un sistema de intercambio de archivos entre este y el servidor de archivos, para hacer una gestión de los contenidos interceptados.

Referencias

[1] “DFRWS (Digital Forensics Research Conference)”, <http://www.dfrws.org/> Disponible [Internet] [17 de abril de 2012]

[2] “Jesse Kornblum”.: < http://jessekornblum.com/ > [17 de abril de 2012]

[3] “Fuzzy Hashing and ssdeep”. Disponible [Internet]: < http://ssdeep.sourceforge.net/> [17 de abril de 2012]

[4] “DeepToad is a library and a tool to clusterize similar files using fuzzy hashing”. Disponible [Internet]: < http://code.google.com/p/deeptoad/> [17 de abril de 2012]

[5] “CakePHP: the rapid development php framework”. Disponible [Internet]: <http://cakephp.org> [17 de abril de 2012]

[6] “Proyecto Indect”. Disponible [Internet]: <http://www.indect-project.eu/> [17 de abril de 2012]

[7] Gianluca Costa & Andrea De Franceschi. “ Xplico, network forensic analysis tool”. Disponible [Internet]: <http://www.xplico.org/> [17 de abril de 2012]

[8] “Sqlite software library”. Disponible [Internet]: <www.sqlite.org> [17 de abril de 2012]

[9] “ImageMagick: Convert, Edit, Or Compose Bitmap Images”. Disponible [Internet]: <http://www.imagemagick.org> [17 de abril de 2012]

[10] “The Apache Software Foundation”. Disponible [Internet]: <http://www.apache.org/> [17 de abril de 2012]

[11] “JavaTM 2 Platform Standard Edition 5.0 API Specification”. Disponible [Internet]: <http://docs.oracle.com/javase/1.5.0/docs/api/> [17 de abril de 2012]

[12] Jesse Kornblum, “Identifying almost identical files using context triggered piecewise hashing” digital investigation 3S (2006) S91–S97

[13] Benjamin López Pérez. “Formatos gráficos”. Departamento de informática. Universidad de Oviedo. Autores: Curso 2006/2007

[14] “Internet Observatory”. Disponible [Internet]: <http://www.internetobservatory.net/> [17 de abril de 2012]

[15] Dustin Hurlbut – AccessData. “Fuzzy Hashing for Digital Forensic Investigators” [9 de Enero del 2009]

[16] “FNV Hash”.Disponible [Internet]: http://isthe.com/chongo/tech/comp/fnv

[17] “Rolling Hash (Rabin-Karp Algorithm)”. Disponible [Internet]: <http://courses.csail.mit.edu/6.006/spring11/rec/rec06.pdf> [18 de Febrero 2011]

[18] Abdiel E. Cáceres González. “La métrica de Levenshtein”. Universidad Juárez Autónoma de Tabasco, [Diciembre 2008]

Anexo A

Instalación y configuración

Para ejecutar nuestra aplicación necesitamos tener instalado en el servidor Ubuntu 10.4 y las siguientes aplicaciones:

- Java1.6
- Sqlite3.7.10
- Apache5.0.5
- Php5.4.0

InstalaciónJava1.6

Para instalar Java primero nos descargamos el archivo autoinstalable del siguiente link *http://java.com/es/download/* Una vez descargado el archivo, abrimos un terminal y nos vamos a la carpeta donde hemos descargado el fichero **jre-6-linux-i586.bin**:

$cd <carpeta>

Le damos permisos de ejecución al archivo y lo ejecutamos para instalarlo:

$chmod+xjre-6-linux-i586.bin

$sudo./jre-6-linux-i586.bin

Movemos la carpeta creada después de la instalación (llamada jre1.6.0) a una más apropiada:

$sudomvjre1.6.0/usr/lib/jvm

Establecemos el nuevo Java como una de las "alternativas de java":

$sudo update -alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jre1.6.0/bin/java" 1

Ahora establecemos la "nueva_alternativa" como la real de Java. Este paso hace que la versión de java sea la usada por defecto:

$sudo update -alternatives --set java/usr/lib/jvm/jre1.6.0/bin/java

Para comprobar si tenemos la versión 1.6.0, escribimos en la terminal:

```
$java -version

java version "1.6.0"

Java(TM) SERuntime Environment (build1.6.0-b105)

Java Hot Spot (TM) ClientVM( build1.6.0-b105, mixed mode)
```

Instalación de SQLite3 y sus librerías de desarrollo

```
$sudo apt -get install sqlite3 libsqlite3-dev

$sudo apt -get install sqlitebrowser

$sudo apt -get install php5-sqlite
```

Instalación de apache

```
$sudo aptitude install apache2

$sudo a2enmod rewrite

$sudo /etc/init.d/apache2 restart

$sudo gedit /etc/apache2/sites-available/default
```

Instalación de php

```
$sudo apt –get install php5
```

Instalación de ImageMagick

Para instalar ImageMagick primero nos descargamos el archivo autoinstalable del siguiente link *http://www.imagemagick.org* Una vez descargado el archivo, abrimos un terminal y nos vamos a la carpeta donde hemos descargado el fichero **ImgaeMagick.tar.gz**:

```
$tar xvfz ImageMagick.tar.gz

$cd ImageMagick-6.7.6

$./configure
```

$make

$sudo make install

Instalación del pre-clasificador

$tar xvf Roberto_Diaz_Ramos.tar.gz

$sudo mv Roberto_Diaz_Ramos /var/www/

$chmod –R 777 /var/www/Roberto_Diaz_Ramos

Para realizar la configuración de los paquetes instalados se deben seguir los siguientes pasos:

Modificamos el alias de nuestro servidor

Editamos el archivo /etc/apache2/mods_avaliable/alias.conf, modificando la línea que tiene la etiqueta Alias, por la siguiente línea

Alias/PFC/pluginFZ/ "/"

Modificamos el archivo php.ini, para poder trabajar con archivos mayores de 2MB.

Editamos el archivo /etc/php5/apache2/*php.ini* y cambiamos los parámetros deseados, en nuestro caso habilitamos un máximo de 100 Mb de tamaño a subir y un tiempo máximo de 1000 segundos:

```
;;;;;;;;;;;;;;;;;;
;ResourceLimits;
;;;;;;;;;;;;;;;;;;
max_execution_time=1000
max_input_time=1000
;Maximum size of POST data that PHP will accept.
post_max_size=100M
;Maximum allowed size for uploaded files.
upload_max_filesize=100M
```

Configuramos ImageMagick

Configuramos ImageMagick para poder utilizarlo desde cualquier directorio del sistema:

```
$sudo ldconfig /urs/local/lib
```