UOC
Universitat Oberta de Catalunya

Filtratge Web

Cristóbal Velasco Moreno

Enginyeria Tècnica en Informàtica de Sistemes

Consultor: Maria Isabel March Hermo

Gener de 2007

Dedicatòries i agraïments

Voldria agrair a moltes persones el recolzament proporcionat durant el projecte i tots els estudis en general. En especial les següents persones:

- A la meva família, sense la qual mai no hagués pogut arribar fins al final.
- A Pilar, ja que sense ella ni tan sols hauria començat.
- I finalment a tots els companys i professors que han aportat el seu gra de sorra en la realització d'aquest treball.

Gràcies a tots

Resum

Aquest projecte s'engloba dins el Treball de Fi de Carrera de la titulació d'Enginyeria en Informàtica de Sistemes realitzada a la Universitat Oberta de Catalunya.

Avui dia, la utilització de Internet s'ha generalitzat, tant a nivell d'usuari domèstic com a nivell laboral. La franja d'edats avarca pràcticament a tota la població. Aquest últim fet, i la presència d'elements no desitjables per a menors a la xarxa, fa que sorgeixi la necessitat de tenir alguna eina per limitar l'accés. Un altre aspecte és el rendiment en el treball, i la utilització d'Internet només per les tasques relacionades directament amb la feina.

L'objectiu del treball és realitzar un programari que realitzi tasques de filtratge de les planes web que pot explorar qualsevol explorador del sistema. A més, ha d'incloure la capacitat de mantenir un historial de navegació, que pot servir per millorar la tasca de filtratge..

Per la implementació s'ha emprat la base de dades MySQL i l'entorn de programació Eclipse, amb el llenguatge de programació Java. El programari està realitzat en entorn Windows, però es pretén que sigui portable a d'altres sistemes. Per això utilitzem aquests recursos, tots disponibles tant en entorns Windows com Linux. Per la realització de la memòria s'ha optat per un altre programari lliure i gratuït, l'OpenOffice. Per la realització del disseny UML s'han considerat diverses opcions, optant finalment per la utilització del programari Sun Java Enterprise 8, que no és lliure però sí disponible gratuïtament.

Les funcionalitats del programari són les següents:

- Filtratge web: Bloqueig de planes web per domini, ip i contingut.
- Registres: Captura d'informació sobre les connexions, tant les autoritzades com les que no s'arriben a completar.
- Entorn gràfic. Per facilitar l'ús del programari al usuari.

Hi ha d'altres funcionalitats previstes, que de moment no s'implementaran com són les següents:

- autenticació: Per evitar l'accés al programari a personal no autoritzat.
- Generació d'informes: Per extreure informació per modificar tant la política de filtratge com la d'autoritzacions a la empresa. Aquests informes es generaran en un

fitxer de text, possiblement en format pdf. En un principi, les dades obtingudes estan disponibles directament des del SGBD.

Índex

Índex de contingut

Índex d'imatges

1. Introducció

1.1 Justificació i context

En el moment actual ens trobem que l'accés a Internet cada dia resulta més fàcil per a més gent. Degut a això, la quantitat de informació disponible té una àmplia varietat de temes i d'enfocaments. D'altra banda, cada cop hi ha més menors amb accés a Internet, així com més accés a Internet des de les escoles. Tot això fa que hi hagi la necessitat de tenir alguna eina per evitar l'accés a continguts perillosos o inadequats, davant de la impossibilitat de controlar físicament l'accés a tothom.

Un altre entorn on es pot aplicar aquest programa és a l'entorn empresarial, per limitar la utilització d'Internet als empleats, deixant-la només coma eina de treball.

Tot això comporta una reducció de costs, una millora del rendiment laboral i en definitiva, pel que fa a la vessant laboral del programari, més beneficis per a l'empresa. La vessant de control d'accés al menor, proporciona un eina amb la qual augmenta la seguretat del pares i professors respecte als nens, sent una gran ajuda en el cas de no poder tenir presència física en tot moment.

Per tant, aquest treball està enfocat des d'aquesta perspectiva, com una eina per protegir de material no desitjable més que una eina de control del que ja s'ha fet. De totes maneres, hi ha un control de les connexions ja realitzades, tant per a ajudar al funcionament correcte del filtratge com control de la utilització que es fa del sistema.

L'objectiu final d'aquest treball és obtenir un producte lliure i adaptable a les pròpies necessitats que cobreixi els problemes de la utilització d'Internet

1.2 Objectius

L'objectiu del projecte és tenir una eina que permeti el filtratge de planes web, amb la inclusió d'un historial de les planes visitades i dels intents d'accedir a planes no autoritzades. També es vol que sigui una eina portable a diversos sistemes, cosa que es veu facilitada per l'ús del llenguatge Java.

D'altra banda, també es vol assegurar la modularitat del programari, de cara a millorar el manteniment de l'aplicació, ja que es redueixen costs si no es vol fer un canvi gran. Un disseny no modular augmenta els costs de manteniment, ja que qualsevol modificació comporta la modificació de tot el codi.

Tant la modularitat com la portabilitat comporten l'ús del llenguatge Java.

La utilització d'una interfície gràfica també augmenta de facilitat d'ús de l'usuari final, ja que avui en dia molts usuaris no estan acostumats a treballar amb la línia de comandes.

També es pensa en funcionalitats futures, relatives a la presentació d'informes de la informació, per la qual cosa s'ha utilitzat una base de dades per emmagatzemar tota la informació generada pel programari. Això també permetrà en una futura actualització la generació d'una base de dades remota on generar els registres.

1.3 Metodologia

- En primer lloc, s'ha realitzat una recerca sobre la informació relacionada amb el programari. En particular, s'ha cercat informació sobre el funcionament del protocol HTTP en concret i s'han consultat diversos programaris ja existents sobre PROXY i Firewall.

- A partir de la cerca, i tenint en compte els objectius, s'ha decidit la utilització del llenguatge Java per la implementació del programari i de la base de dades MySQL per la base de dades per generar els registres i emmagatzemar les regles.

- A continuació s'ha procedit al anàlisi de les classes Java necessàries per implementar el programari i s'ha dissenyat un model UML amb les classes principals del projecte.

- També s'ha analitzat el disseny de la base de dades, s'ha procedit a la seva creació i s'han generat les taules.

- El pas següent ha estat la implementació de les classes en Java, seguint un cicle de vida iteratiu i incremental. Es divideix la implementació en diverses tasques diferents i es repeteix el cicle en cascada diversos cops, fins a finalitzar el programari.

- A continuació, es procedeix a realitzar les proves utilitzant l'explorador Netscape 8 i el nostre programari com a proxy.

- Finalment, i de forma paral·lela a tots aquests processos, es realitza la documentació del projecte dins aquesta memòria de treball, on es documenta de forma més extensiva cada punt del desenvolupament del producte.

.

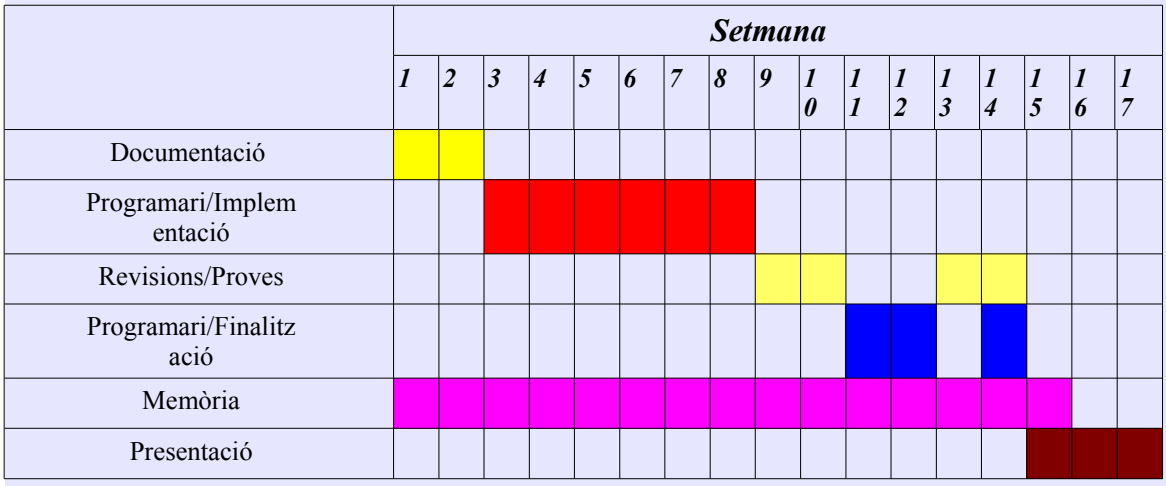
1.4 Planificació

A la següent taula trobem les tasques a realitzar , així com la seva planificació temporal. També s'indica la precedència de tasques, així com les subtasques. La implementació s'ha dividit en dos grups. En primer lloc, s'implementa el nucli del programa i un cop revisat i verificat, es crea l'entorn gràfic d'execució.

Setmana	*Data*	*Id*	*Activitat a realitzar*	*Altres esdevenimen ts*	*Precedents*
1	20/09 - 24/09	0.1	Documentació - Inici de la memòria		_
2	25/09 - 01/10	0.2	Documentació- Pla de treball-	PAC1- Pla de treball	_
3-4	02/10 – 15/10	1.0	Desenvolupament del programari- Captura de connexions des de Java – Redacció del capítol corresponent de la memòria		0.2
5	16/10 – 22/10	1.1	Desenvolupament del programari- Comunicació JDBC-MySQL – Redacció del capítol corresponent de la memòria		1.0
6-7	23/10 – 05/11	1.2	Desenvolupament del programari - Definició de regles e utilització d'aquestes al programa – Redacció del capítol corresponent de la memòria		1.0 1.1
8	06/11 – 12/11	1.3	Desenvolupament del programari – Creació de registres i accés a la informació d'aquests – Redacció del capítol corresponent de la memòria	PAC2	1.0 1.1 1.2
9-10	13/11 – 27/11	2.0	Revisió de les fases anteriors i modificacions en cas necessari – Revisió dels requisits – Proves – Redacció del capítol corresponent de la memòria		1.3
11	28/11 - 03/12	3.0	Desenvolupament del programari – Creació de la interfície amb finestres – Redacció del capítol corresponent de la memòria		2.0
12	04/12 – 10/12	3.1	Desenvolupament del programari – Unificació de tots els mòduls, funcionament global – Redacció del capítol corresponent de la memòria		2.0 3.0
13	11/12 –	4.0	Revisions i proves del programari – Redacció del capítol corresponent de la memòria		3.0 3.1

Setmana	*Data*	*Id*	*Activitat a realitzar*	*Altres esdevenimen ts*	*Precedents*
	17/12				
14	18/12 – 31/12	4.1	Finalització del programari- Proves finals – Redacció del capítol corresponent de la memòria	PAC3	4.0
15	01/01 – 13/01	5.0	Finalització de la memòria i inici de la presentació	Lliurament de la memòria i el producte	4.1
16	14/01 – 19/01	6.0	Preparació presentació		5.0
17	19/01		Presentació final	Lliurament de la presentació	

En el següent diagrama de *Gantt* es pot veure la simultaneïtat de diverses tasques, en especial la memòria. A més, tot i que no estan incloses al gràfic, hi ha una sèrie de revisions quinzenals o punts de control del treball.



L'estructura del pròxims capítols es correspon amb la divisió del treball proposada pel pla de treball detallat anteriorment, tenint un capítol dedicat a cada sub-tasca de desenvolupament i un altre a les revisions.

Tanmateix, s'ha d'indicar que el pla inicial no ha estat possible realitzar-lo de la manera prevista i que s'han produït una sèrie de desviacions del pla de treball original. A continuació detallem breument les desviacions:

- Durant la primera setmana, s'ha endarrerit el moment d'iniciar la programació de la captura de les connexions per efectuar una recerca dels mètodes aplicables en

Java per realitzar-la, així com una recerca general de tot el programari que es fa servir.

- Setmana 4. Problemes per la captura de les dates HTTP, fan que s'endarrereixi el següent punt del pla de treball, la connexió amb la Base de Dades. A partir d'aquest punt, es decideix implementar paral·lelament els punts des del 1.1 fins al 1.3
- Setmana 6. Pràcticament tota la implementació dels punts anteriorment citats s'ha enllestit, amb la qual cosa es va per davant del pla previst. Es realitzen les primeres proves, que determinen un error en la captura, que es procedeix a estudiar.
- Setmana 9. S'inicia paral·lelament a la correcció de l'error de captura la implementació de l'entorn gràfic, davant la dificultat per reparar l'error anteriorment citat i en previsió d'un menor disponibilitat de temps en setmanes posteriors.
- Setmanes 10, 11 i 12. Degut a la falta de disponibilitat de temps anteriorment citada, durant aquestes dues setmanes no s'ha realitzat gairebé cap tasca. S'ha intentat corregir el problema de la captura d'imatges, però no s'ha pogut implementar.
- Setmana 13. S'ha tornat a assolir el ritme indicat pel pla de treball, tot i que apareixen diversos errors encara per solucionar, sent el més important la limitació al protocol HTTP.

1.5 Productes obtinguts

El producte consisteix en dos fitxers, el primer un arxiu executable jar, anomenat ***filtrewebapp.jar*** que executa el servidor encarregats de filtrar les planes web i realitzar la captura de dades sobre les connexions.

L'altra part del producte consisteix en una petita base de dades MySQL anomenada ***filtreweb.sql*** que guarda totes les dades sobre les connexions.

1.6 Descripció dels capítols següents

Els capítols es divideixen en diferents apartats segons els desenvolupament del projecte.

La primera part està dedicada a la a la investigació prèvia, indicant els aspectes principals.

A continuació hi ha l'apartat de disseny del programari, dividit en dues seccions principals dedicades respectivament al disseny de les classes i al disseny de la base de dades.

La secció següent conté la part principal d'aquest document i fa referència a la implementació del programari. Està dividida en diverses seccions relacionades amb les diverses parts de la implementació:

- Comunicació amb l'explorador per capturar les peticions
- Captura de dades i enviament a l'explorador
- Gestió de la base de dades
- Entorn gràfic

L'apartat següent comenta totes les proves realitzades i els resultats obtinguts, dividida en apartats relacionats amb cada secció d'implementació.

La cinquena secció recull els requisits per a la utilització del programari i un manual d'usuari on mostra el funcionament del producte.

A continuació tenim la secció on es mostren les funcionalitats que no han estat implementades però que en futures modificacions del programari es vol implementar.

Per acabar, tenim la secció on es resumeixen les conclusions obtingudes en la realització d'aquest projecte.

2. Projecte

2.1 Recerca d'informació

Per començar, s'ha fet una recerca de dos temes principals:

- La primera recerca ha estat sobre el tractament que es fa en *Java* de les xarxes en general i de la comunicació per *TCP/IP* en concret.

- La segona, sobre el funcionament del protocol *HTTP*, que serveix com a base per la construcció del programari.

A més, s'han fet recerques puntuals sobre aspectes concrets del programari a utilitzar, així com la forma d'aplicar-lo al nostre programari. Aquestes cerques s'han dut a terme paral·lelament a la implementació de les primeres funcionalitats del programari. Entre aquestes cal destacar la comunicació entre *Java* i *MySQL* i la creació de finestres des de *Java*.

Un altre enfocament en la recerca de la informació ha estat l'obtenció d'exemples de codificació, centrada principalment en codi de *Firewalls* i *Proxys*, La principal raó d'aquesta recerca ha estat trobar les classes Java que controlen la captura dels paquets de l'explorador i gestionen la captura de la informació i la transmissió a l'explorador.

Les planes web consultades i els llibres utilitzats es troben indicats a la bibliografia. Aquesta cerca s'ha realitzat al llarg de tota la implementació del producte, especialment a partir de l'error en la captura d'imatges.

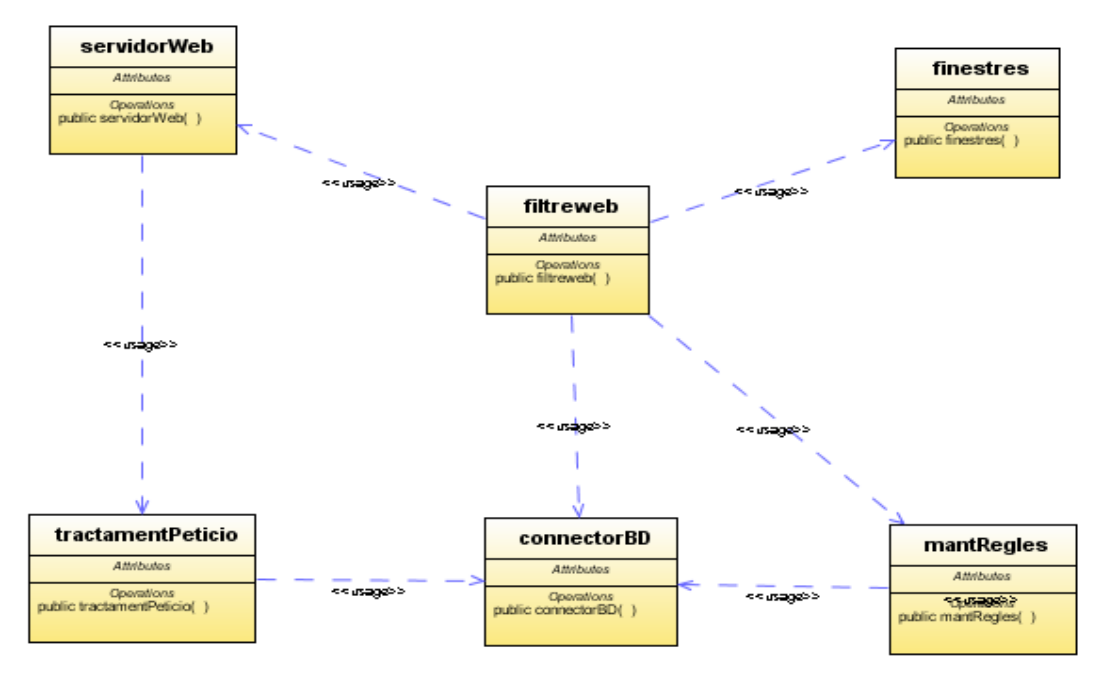
2.2 Disseny general

Donat que a dia d'avui el sistema operatiu més emprat és *Windows*, desenvoluparem el programari en aquest entorn. Com que es pretén que el sistema sigui portable, es vol treballar amb programari lliure, disponible tant en sistemes *Windows* com *Linux*.

En funció dels objectius del projecte, tenint en compte l'elecció de les eines a emprar pels motius expressats al apartat anterior, s'ha creat una estructura de classes i una base de dades específiques.

Com a pas preliminar, s'ha fet un petit disseny UML de les classes emprades al

programari, tenint en compte totes les funcionalitats previstes:



Il·lustració 1: Disseny UML de les classes Java - Disseny inicial

També s'ha d'indicar que ha estat obligat la definició de certes classes auxiliars, com és **ClientHandler**, per realitzar diverses tasques. En particular tenimles següents:

- **ClientHandler** controla el client que es connecta al socket. Estén la classe thread per permetre diversos clients alhora. Tot i estar programada com una classe, es troba definida dins la classe servidorWeb.
- **Utils**: Classe amb mètodes accessoris als procediments de les altres. Implementa, entre d'altres, la gestió de la captura del temps de la connexió, la comprovació de l'autorització per descarregar la IP, etc. Substitueix la classe tractamentPeticio.

Per emmagatzemar tant les regles a seguir com les dades obtingudes a partir del funcionament del programari , es fa servir una base de dades *MySQL*, per millorar tant l'emmagatzematge com la visual·lització de les dades obtingudes. Un altra alternativa per emmagatzemar les dades és la utilització de fitxers de text, però la utilització de una base de dades simplifica el tractament d'aquestes i facilita la posterior generació d'informes.

La utilització de *MySQL* es deguda a dos motius principals: el primer econòmic, ja que és un programari gratuït. El segon és la facilitat d'ús tant a *Windows* com a *Linux*. La utilització d'aquest SGBD en concret és la major informació trobada a Internet sobre la utilització del mateix. Hi ha d'altres possibles, el més significatiu pot ser *PostgreSQL*. La elecció d'aquest sistema per emmagatzemar les dades (un altre podria haver estat amb fitxers de text), comporta la utilització d'un pont MySQL-Java,

2.1.1 Disseny de la estructura de classes

En aquest projecte ens trobem amb diverses tasques molt diferenciades. En primer lloc, es té que crear una estructura de comunicació entre l'explorador web i el programari. D'altra banda, tenim la comunicació entre la base de dades i el programari. Aquí ja trobem una primera divisió de classes.

La comunicació amb l'explorador, ens obliga a tenir una classe que anomenarem **servidorWeb**, encarregada de la part principal d'aquesta comunicació. Aquesta classe és l'encarregada d'obtenir les peticions TCP de l'explorador, encarregar-se de les peticions de connexió a planes,comprovar l'autorització, obtenir-les, descarregar-les del servidor i passar-les a l'explorador.

Part d'aquestes tasques les pot delegar en d'altres classes auxiliars, per evitar sobrecarregar les classes amb molts mètodes i millorar l'orientació a objectes del programari. Concretament, la comprovació de l'autorització es delegarà, així com l'obtenció de les planes del servidor. Per tant, finalment, aquesta classe quedarà només per la comunicació entre l'explorador i el producte. La classe **ClientHandler** genera les peticions de connexió amb el servidor web i obté la plana web del servidor per enviar-la a l'explorador si no està bloquejada mitjançant la classe **servidorWeb.**

L'altra part, corresponent amb la comunicació amb el SGBD, es pot subdividir en dues tasques. La primera correspon al que seria la comunicació amb el SGBD. La segona és la creació de les consultes necessàries per al funcionament del programari.

En l'apartat de consultes hi ha una nova divisió, ja que ens trobem amb consultes a cada connexió per comprovar l'autorització i el manteniment de les regles.

Cadascuna d'aquestes tres tasques es realitzarà per una classe independent. La

primera serà la classe **connectorBD.** la segona, enfocada a les comprovacions durant les connexions, **tractamentPeticions**. I finalment la tercera, dedicada al manteniment de les regles, **mantRegles**.

Finalment, tenim la classe **principal**, que s'anomenarà **filtrewebapp**, que és l'encarregada de cridar la resta de classes i que és el centre de control del programari. A més, hi ha les classes encarregades de l'entorn gràfic.

S'ha afegit la classe **utils**, encarregada de realitzar tasques generals, com és l'obtenció de la IP d'un host, obtenir el codi web des del servidor, etc. És a dir, conté mètodes auxiliars per diverses classes.

2.1.2 Disseny de la base de dades

Després d'un primer anàlisi, es troba que hi ha 2 entitats diferents, cadascuna amb els seus propis atributs. Es tracta dels registres, amb els temps de connexió i resta de dades relacionades amb aquestes, i les regles, amb les condicions per bloquejar planes web. Després d'un segon anàlisi, es veu que aquestes entitats es poden dividir en entitats menors, de forma que identifiquen millor cada entitat.

Especificant les entitats, tenim les següents:

- Registres
  - connexions autoritzades. Aquesta entitat té com a atribut característic la durada de la connexió, no present a la resta.
  - connexions no autoritzades. És la entitat bàsica dels registres. Els seus atributs són el nom d'usuari i l'hora de la connexió.
- Regles
  - Regles IP. El seu atribut són adreces IP.
  - Regles Domini. Només tenen com a atribut el nom del domini. Es diferencien de les regles de text per l'ús de l'atribut. Tot i ser en els dos casos text, en aquest és un atribut amb l'adreça web, mentre que les altres tenen mots no autoritzats
  - ReglesText. Un únic atribut a cada regla, corresponent a un mot no autoritzat.

Així, finalment s'han creat 5 taules amb els atributs especificats anteriorment. A més, s'ha afegit un nou atribut, anomenat id destinat a control pel usuari.

2.3 Implementació

A diferència del plantejament inicial del pla de treball, s'estan implementant paral·lelament diverses seccions del desenvolupament del producte.

Entre aquestes seccions figuren tot l'accés a la base de dades, tant per consultes com per creació de registres. També s'implementen paral·lelament les classes per comprovació de regles, obtenció del codi del web, etc.

La única fase del projecte que s'implementa de forma separada és l'entorn gràfic.

De forma inicial, el filtratge web es volia fer sense haver de manipular l'explorador del sistema, només sobre els ports de destí 80 i 443. Després de cercar informació en Java sobre aquest aspecte, es decideix la creació d'un proxy i enrutar el tràfic dels exploradors cap al port utilitzat pel programari.

Per tant, tot el tràfic dels exploradors passarà pel programari. Això, en principi comportarà una ampliació de la funcionalitat a tots els ports que faci servir l'explorador habitualment. És a dir, serveis com ftp, gopher, si són suportats per l'explorador, també són capturats. Malgrat això, és necessari la correcta captura de les peticions perquè aquests serveis funcionin de forma correcta.

Degut al complexitat de la base de dades necessària per al programari, després d'un breu anàlisi es dissenya i es crea la base de dades, ja que no presenta cap complexitat.

Un cop generada la base de dades, amb els atributs ja ben definits, es procedeix a la implementació de les classes de control de l'explorador com a primer pas i a continuació, en part de forma paral·lela a les primeres, s'implementen les classes d'emmagatzematge, primer de les regles i a continuació de la informació

Després de tenir implementada el bloqueig de planes i la base de dades, es realitzen una sèrie de proves de bloqueig de diverses planes, per verificar el correcte funcionament d'aquesta part del programari.

Un cop completada la part de control i emmagatzematge, es continua amb la programació de l'entorn gràfic de la aplicació. Completada aquesta, es realitzen les proves del conjunt, per verificar el comportament global. A continuació trobem detallats cadascun dels passos de la implementació.

2.3.1 Captura tràfic procedent de l'explorador

En primer lloc, s'ha volgut capturar el tràfic del port 80 de sortida, mitjançant les classes *ServerSocketChannel*, *SocketServer* i *Socket*, creant un canal al port 80. Mitjançant el sniffer *Wireshark* s'ha pogut comprovar l'error del plantejament inicial, ja que cada sol·licitud es realitza per un port diferent, essent només el port de destí el que tenen totes les connexions en comú. L'error estava en que el plantejament que el server socket escolta els paquets destinats al port 80, ja que el port 80 és d'una màquina remota.

El problema que es planteja a continuació és capturar les connexions destinades al port remot 80, amb port local diferent a cadascuna per poder capturar les comandes HTTP, per exemple, GET, HEAD, etc.

La solució a aquest problema ha estat la utilització d'un proxy als exploradors. S'ha fixat el port 1234 com a port per utilitzar el programari. Aquest serà doncs, el port emprat per a la creació dels sockets. Aquesta modificació comporta un canvi en el plantejament dels objectius, ja que ara no limitem el filtratge als serveis HTTP i HTTPS, sinó que totes les peticions efectuades per l'explorador estaran filtrades.

El següent problema que s'ha plantejat ha estat la captura de les dades. Tot i que s'aconsegueix obrir els canals, gràcies a la classe *ServerSocketChannel*, ens trobem amb la dificultat d'obtenir la informació. Les classes I*nputStreamReader* i *Scanner*, que són les que s'han provat per aquesta tasca, provoquen un error de *IllegalBlockingModeException*.

Això fa que es decideixi usar un servidor *Multithread*, amb la qual cosa hem de prescindir del *ServerSocketChannel* i implementar de nou tot el codi del servidor, tot mantenint l'ús del port 1234. En aquest cas, s'usa una classe que estén la classe *Thread*

.

Per realitzar aquesta tasca s'han creat dues classes,**servidorWeb** i **tractamentPeticio**, que s'encarreguen respectivament de la comunicació amb l'explorador d'Internet i l'obtenció (i posterior tractament) de les peticions GET i HEAD rebudes pel programari. La implementació d'aquesta última classe, genera la necessitat de tenir definides les estructures encarregades de realitzar les consultes a la base de dades, per la qual cosa es comença de forma paral·lela el desenvolupament de la següent fase, la comunicació amb el SGBD.

D'altra banda, la classe **servidorWeb**, és l'encarregada de recuperar les planes web des del servidor i passar-les a l'explorador per mitjà del socket. Per això ha de rebre autorització de la classe **tractamentPeticio**, un cop comprovada l'autorització.

La posterior modificació de la classe **servidorWeb**, per separar la funció de servidor de l'explorador, del client que es connecta al servidor web per rebre la pàgina, que ha comportat la creació de la classe **ClientHandler** com a classe pròpia, comporta que la classe **servidorWeb** deixi de recuperar les planes i es realitzi la tasca a la classe **ClientHandler**. També, com es comenta al següent apartat, la classe **tractamentPetició** passa a ser un mètode de la classe **utils**.

2.3.3 Captura planes web i transmissió a l'explorador

El següent apartat tracta de la captura de la web des de Java i el seu enviament per la visualització a l'explorador. En un principi, es vol emprar algun tipus de classe *URL* o *HyperLinkListener* per rebre-la i enviar-la a l'explorador.

Finalment, es decideix implementar la captura de la web amb una nova classe, **obtenirWeb** i un mètode homònim a la classe **ClientHandler**, que forma part de la classe **servidorWeb**, amb dues finalitats ben diferents. En el primer cas, obté el codi de la plana web per procedir a la comparació amb les dades sobre mots no autoritzats. També serveix per obtenir la cadena per enviar a l'explorador en el mètode del mateix nom.En el segon cas, es forma un pont entre el port 1234 que utilitza l'explorador i el port de la plana web, usualment el 80. En aquest cas, s'utilitza les dades obtingudes amb la classe **obtenirWeb** per generar el codi que s'envia a l'explorador.

Tot i que no sembla haver cap error de compilació, l'explorador no és capaç de rebre les pàgines web. Un error en la terminació d'un bucle *do-while*, genera un bucle infinit. Es decideix utilitzar un booleà per indicar la terminació del bucle. Un cop obtingut el codi, i abans de generar un error per *nullPointer*, es confirma la sortida del bucle.

Solucionat aquest problema, ens trobem amb un altre. No es carreguen les planes web. Un error en la consulta a la base de dades generava bloqueig de les planes quan no s'havien de bloquejar, ja que la base de dades de regles encara és buida. Es decideix modificar el mètode **comprovaRegla** de la classe **connectorBD** canviant el mètode per indicar que la cerca ha estat positiva. Considerant que el mètode *resultSet.first()* retorna fals si la posició és buida(i per tant, no s'ha trobat cap coincidència a la base de dades) es fa servir per indicar que la cerca ha estat negativa. D'altres mètodes no donaven el resultat desitjat, com s'indicarà més endavant en l'apartat de les consultes a la base de dades.

Cal remarcar que es produeixen errors en la descàrrega de les pàgines, donat que es modifica el codi com un *String* i això genera problemes de visualització. Es treballa en la modificació de la classe obtenir web per generar un codi que permeti obtenir la web de forma completa. Està en estudi la utilització de la classe *HyperLinkListener* per obtenir el codi.

La substitució de la càrrega de planes per generació d'un *String* amb la codificació es fa necessària per la impossibilitat de carregar cap element multimèdia a l'explorador. A més, la codificació d'aquests elements provoquen errors en les consultes SQL, causats per l'aparició de caràcters que generen confusió, per exemple cometes simples que fan que no es delimiti el valor que es cerca, retornant un error de sintaxi SQL.

Es decideix modificar la captura de les planes substituint els *Strings* per *StringBuffered*, ja que la utilització dels primers és la que genera els errors. Per tant, s'ha de modificar tot el codi per implementar la nova forma de capturar la informació del servidor. S'implementa la captura de les planes de forma que només es realitza la comprovació dels mots del cos de la plana html.

La substitució anteriorment citada no ha resolt el problema. El rendiment del programari és molt baix, ja que tarden molt a carregar les planes. S'ha treballat en diverses implementacions sobre *StringBuffered*, modificant bucles, el sistema de captura de la connexió, etc. De moment sense cap resultat.

S'està buscant una classe per substituir la classe *Scanner* que és la que es fa servir per obtenir el *String* del *stream* d'entrada del servidor web o bé les classes *printerWriter* que envien la plana a l'explorador.

Donat que totes aquestes solucions no donen el resultat desitjat, s'ha optat per una solució alternativa, que si bé no soluciona el problema de la captura de les imatges, sí que aconsegueix enviar la plana a l'explorador. Es modifica el mètode de la classe ClientHandler anomenat obtenirWeb(), que anomenem finalment enviarWeb(), que és el que s'encarrega d'enviar les planes web a l'explorador, per que simplement retorni el flux de dades de l'explorador directament, sense realitzar cap comprovació, que ja ha estat realitzada amb anterioritat.

Aquesta solució ha estat parcialment positiva, ja que es mostren les imatges i la maquetació correcta de les planes, però no s'accedeix a zones que requereixen identificació, ja que no és possible enviar les dades d'usuari, es manté a la mateixa plana, ja que no accepta el protocol HTTPS, ni qualsevol altre a banda del HTTP. Això indica una errada en la captura de les peticions de connexió que fa l'explorador. D'altra banda, encara es bloquegen totes les pàgines tant per ip com per mots filtrats, ja que si una pàgina queda filtrada, no s'accedeix al mètode **enviarWeb()**

D'altres opcions que s'han estudiat per realitzar la tasca de captura ha estat la utilització de les classes *DataInputStream* i *DataOutputStream*. Tot i que semblen les més adequades, ja que capturen les imatges de forma correcta i no fan necessari obtenir el codi novament com en el cas que hem implementat, degut a qüestions de calendari no es modifica la captura de dades i es deixa tal i com es comenta al paràgraf anterior.

Durant les proves ha sorgit un problema de memòria, donat que no es tancava la connexió del client amb el servidor web. Un cop tancada el problema s'ha solucionat sense cap més incident.

La captura del temps de connexió genera un error quan es realitza una connexió al *localhost*, per exemple quan es realitza les entrades de les dades d'identificació en una web, ja que el temps inicial no es captura. Temporalment, s'assigna el mateix temps que el de desconnexió, quedant el resultat com una connexió de 0 mili segons.

Queda per posterior modificació la captura de les peticions, en cas que sigui possible per qüestions de calendari. S'han provat diverses opcions, principalment diferenciant l'accés per l'inici del URI del web.

La classe que sembla més adequada per aquesta tasca és *HttpsURLConnection* tot i que es produeixen errors de compilació ja que la classe no es pot instanciar, al ser una classe abstracta. La definició d'una classe que implementi els mètodes abstractes fa que la tasca sigui més complicada que el que permet el temps disponible. Per tant, es busquen altres opcions.

Un altra solució que s'ha volgut realitzar utilització del constructor *URL(protocol,*

host,fitxer) per crear una connexió amb el protocol HTTPS. Malgrat tot, aquesta solució tampoc funciona. No es carreguen les pàgines segures ni tampoc s'accepten enllaços segurs des de planes no segures. Per tant, els sistemes d'identificació no funcionen correctament.

Es detecta que l'explorador no inicia les connexions HTTPS. És a dir, no demana cap connexió que sigui amb protocol diferent al HTTP. Per comprovar que no sigui problema de l'explorador, s'han pres dues accions:

- Configurar l'explorador firefox per què utilitzi el proxy, donant els mateixos resultats
- Modificar la configuració de l'explorador Netscape Browser 8.1 per què no faci servir el proxy.

Ambdues solucions permeten comprovar que és un error de la implementació del proxy.

A partir de certes comprovacions del funcionament del programari, és detecta l'omissió de la comanda CONNECT, que provoca que no s'inicïin les connexions i errors en la captura del host, retornant dades incorrectes,com l'explorador enlloc del host.

Posteriors modificacions del codi de captura de les comandes HTTP, provoquen errors en la creació de la URL o retorna documents buits. El problema és troba en la creació de la *url* en el cas de les planes segures. Es treballa per capturar les dades de forma correcta, ja que es produeixen errors en la captura de les dades.

Finalment, s'aconsegueix la captura de les dades de connexió de forma correcta, però es produeix un error en la connexió de les planes segures, que impedeix descarregar-les. Aquest error no es pot corregir a temps, per la qual cosa aquesta funcionalitat no s'ha pogut implementar.

D'altra banda, hi ha certes planes web que tarden molt a carregar-se, degut a totes les comprovacions. La eficiència del programari és bastant baixa, tot i abans d'acceptar connexions HTTPS l'efectivitat era major. Malgrat això, es manté tota la implementació per acceptar aquestes connexions de cara a modificacions futures del programari.

També es produeixen certs errors en les cerques a Google per problemes en el format de la URL.

2.3.4 Base de dades

Un cop decidit el disseny de la base de dades, tal com s'ha indicat abans, passem al disseny del programari. Hi ha tres aspectes diferenciats per la utilització de la base de

dades. A continuació es detallen cada un d'aquests aspectes.

2.3.4.1 Comunicació amb la base de dades

En aquest cas el procediment per assolir la connexió ha estat bastant senzill. Només s'ha d'importar al projecte Java el connector proporcionat a la plana web del fabricant del programari. En aquest cas, la utilització de la base de dades és bastant senzilla. S'han creat una sèrie de taules, però s'ha fet directament amb el programari de control del SGBD, per eliminar un ús excessiu del SQL.

En un principi, es volia deixar de forma estàtica, inclosa dins el codi del programa, les claus d'accés a la base de dades. Un cop realitzades diverses proves, i donat un error en la configuració de la base de dades en *Linux*, s'opta per la introducció de les dades per part de l'usuari. Així també s'impedeix que qualsevol usuari pugui manipular les dades, ja que només el que tinguin accés a la base de dades podran realitzar l'execució del programa.

D'altra banda, això comporta el problema que sense aquest accés, el programari no és utilitzable. En una versió comercial, l'opció és realitzar la introducció de dades durant la instal·lació i la utilització posterior sense la necessitat d'introducció de les dades.

2.3.4.2 Preparació de seqüències SQL i comprovacions

La part fonamental de la comunicació amb el SGBD són les sentències SQL, Per tant, hem de preparar una sèrie de seqüències de *strings* amb els camps a utilitzar en les cerques, insercions, etc. La preparació d'aquestes sentències es treballa conjuntament amb el desenvolupament de la classe **tractamentPetició**, ja que aquesta és l'encarregada tant de processar les peticions del servidors com realitzar la comunicació amb la base de dades, tant per determinar bloqueigs com per realitzar el registre.

Hi ha 4 sentències per definir, la primera relacionada amb la connexió amb la base de dades i la resta amb les operacions bàsiques de *Insert*, *Delete* i *Select*. Totes tres sentències han d'estar capacitades per realitzar l'operació a qualsevol de les cinc taules de la base de dades. Per tant, han d'ésser bastant flexibles, deixant pràcticament tota la informació de la sentència per ser rebuda com a paràmetre.

Posteriors simplificacions del codi, han fet que es prescindeixi de la classe t**ractamentPetició**, ja que aquesta havia quedat reduïda a un sol mètode per la utilització de la classe **utils** en la realització de les tasques comunes a vàries classes. Per tant, es crea un mètode **tractamentPeticio** a la classe útils que substitueix la classe eliminada

.

Les consultes SQL que es fa a cada classe són les següents:

- **connectorBD**:
  - Fa les insercions, consultes i esborra segons els paràmetres que rep d'altres classes.
  - Fa les consultes als registres
  - Esborra els registres
- **servidorWeb**:
  - Envia les dades del registre a la base de dades.
- **Utils**:
  - Comprova la base de dades de regles.

Les últimes classes utilitzen els mètodes de la classe **connectorBD** per accedir a la base de dades. El que fan aquestes classes és crear les consultes que s'envien.

La creació de les consultes ha generat diversos errors de sintaxi. En particular, l'error més important ha estat a la generació de consultes del mètode **comprovarRegla** de la classe **connectorBD**. El motiu sembla que és la problemàtica a l'introduir la codificació dels elements multimèdia en la consulta com a *String*. Aquests *Strings* generats provoquen errors per problemes de codificació, ja que generen símbols no reconeguts que provoquen l'error.

Entre els diversos mètodes que s'han intentat emprar per solucionar aquest problema, han estat la col·locació addicional de dobles cometes en la consulta, que tampoc ha solucionat res, ja que posteriorment les consultes de mots normals genera errors per l'aparició de dues dobles cometes. Un altre sistema que s'ha estudiat ha estat la modificació de la codificació dels caràcters dels *strings*. Aquest error estava provocat per l'afegiment de les cometes dobles en dos mètodes, que ha estat solucionat amb la supressió de les cometes en un d'ells

La consulta de mots filtrats també provoca errors, ja que els caràcters no alfanumèrics, com són els signes de puntuació, en particular el punt i coma, les cometes dobles i el signe = provocaven errors en la construcció de la consulta. Finalment, es fa servir una expressió regular per eliminar tots els caràcters que no siguin alfanumèrics, com signes de puntuació, signes = , + , etcètera, que no són rellevants de cara a l'exploració de mots.

El sistema per determinar si una classe compleix una norma de bloqueig consisteix en realitzar una consulta a la base de dades. Si aquesta consulta no és buida, llavors es bloqueja la plana. S'han fet servir diversos mètodes per determinar si una consulta no és buida, ja que la classe *ResultSet* no conté cap mètode explícitament per aquesta tasca. En d'altres, el mètode *isNull()*, que provocava un error de *nullPointerException*, el mètode *isBeforeFirst()* que no bloquejava les planes, etc. Finalment, ha estat el mètode *First()*, l'idoni per aquesta tasca, ja que retorna cert si es pot col·locar el cursor a la primera fila i fals si no es pot perquè aquesta fila no existeix i per tant, és buit.

2.3.4.3 Introducció de regles

Per a la introducció de les regles farem servir la classe **mantRegles**. Aquesta classe es recolzarà com la classe **tractamentPeticio,** en la classe **connectorBD** per realitzar insercions, consultes i supressions. Els tipus de regles són tres, com indica la estructura de la base de dades. El cas de les regles de domini és especial, ja que només es fa servir per tenir constància de l'existència dels dominis bloquejats, ja que el bloqueig efectiu es realitza per IP. El procediment consisteix en obtenir la IP del host bloquejar-la, sent el procés d'esborrat l'invers d'aquest.

Un cop eliminada la classe **tractamentPeticio**, com s'ha comentat anteriorment, aquesta és substituïda per la classe **utils**, fent servir un mètode, anomenat com la classe eliminada, per realitzar la tasca, en lloc d'una classe complerta.

2.3.4.4 Creació i visualització dels registres

Aquesta és la part encarregada les dades que es guarden sobre les connexions i de recuperar-les per a la visualització pel usuari. Part de la funcionalitat d'aquest apartat es crea a les classes anteriorment utilitzades, com per exemple la classe **utils**, que genera els valors de temps indicats als registres.

La creació dels registres es realitza a la classe **servidorWe**b. En aquesta classe es generen els *Strings* que formaran la consulta per la inserció del registre. En aquesta classe es determina en quin moment es realitza el *timestamp* per generar la entrada al registre. A la classe **servidorWeb** es genera el tipus d'entrada, si és una entrada d'inici de connexió, de final de connexió o de connexió no autoritzada. Això es fa especificant el moment en que es realitza la captura. Per explicar-ho més gràficament, la captura del *timestamp* es produeix en el moment en que s**ervidorWeb** determina que la web està bloquejada, tot i que la connexió es realitza moments abans.

Tot i que la classe **servidorWeb** s'encarrega d'indicar quan es realitza la captura de les dades, aquestes es generen a la classe **utils**. Aquesta classe conté diversos mètodes, que implementen des de la captura de l'objecte *Date*, fins a calcular el temps de connexió. També genera altres dades, com per exemple l'usuari del sistema.

La visualització dels registres s'ha optat per implementar-la per l'explorador. El resultat de la consulta s'envia a l'explorador per la seva posterior visualització. Degut a problemes d'implementació d'aquesta funcionalitat, es decideix mostrar la informació mitjançant una finestra de l'aplicació enlloc de per l'explorador. Per tant, s'endarrereix la implementació d'aquesta funcionalitat fins la implementació de la interfície gràfica.

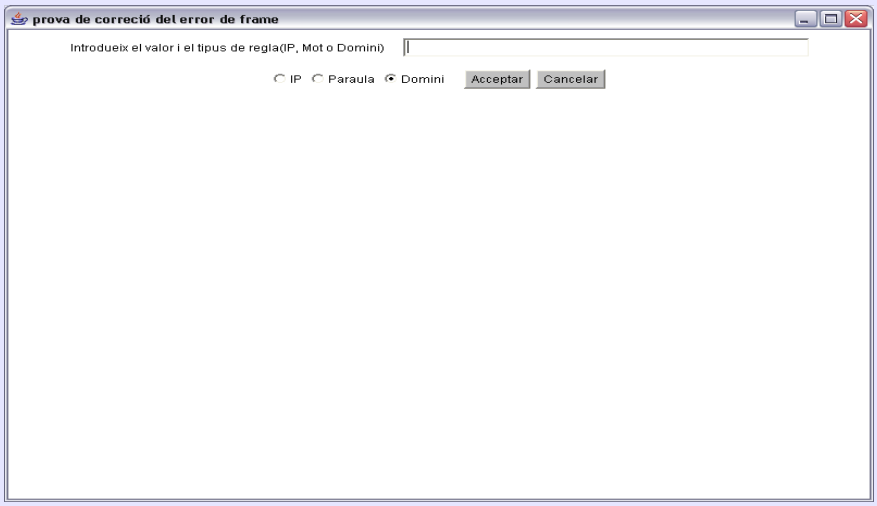
Un cop implementada la interfície gràfica, ens trobem amb un problema d'eficiència en l'accés a les dades, tot i que només es produeix en entorns *Windows*. En

principi pot semblar un problema degut al volum de les dades, però també es produeix sense cap o amb poques dades a la base. En principi, s'estudia un possible problema de disseny de la base, però no s'ha trobat. Degut a que el problema només es presenta en *Window*s i a que no es troba cap diferència en la implementació tant de la base, amb els mateixos tipus de dades, com en el mètode, que és el mateix en tots els casos, es determina que és un error provocat per l'entorn del programari i que no s'ha de produir en altres entorns.

2.3.5 Entorn gràfic

L'entorn gràfic de l'aplicació es vol obtenir amb la utilització de les classes *AWT*. Bàsicament es fan servir *Frames*, *Panels*, i totes les classes relacionades amb components *AWT*. També s'han d'implementar els *Events* i comunicar-los amb la resta del programari.

La primera dificultat es troba en la col·locació dels botons en determinades pantalles, ja que no es generen al lloc on es proposen, per causes que en principi no es poden determinar. La captura següent mostra el error, ja que no apareixen tots els botons del *fram*e inferior, ja que estan tots col·locats en el mateix *frame*.



Il·lustració 2: Error en la creació del frame inserirRegles

L'error, en principi semblava generat per la inclusió de diversos elements en la mateixa zona del *frame*. La solució, després de consultar el API de Java ha estat la utilització de diversos *frames* per incloure els elements a la zona corresponent. Malauradament, això no soluciona el problema original, tot i que es corregeix per evitar altres errors. Finalment, la utilització del *GridLayou*t soluciona el problema. En un principi ja s'havia utilitzat, però creava uns botons adaptats a la mida del *frame*, ocupant tot l'espai disponible. Aquest error s'ha solucionat deixant el *Layout* per defecte a la finestra principal i afegint el *GridLayout* només als panells posteriors. Finalment, el disseny de les pantalles no ha donat més problemes.

A continuació, comencem a realitzar la creació dels *events*, que comença de forma problemàtica ja que no reconeix les classes relacionades amb *ActionListener*, tot i que la importació del *java.awt* i del *java.awt.event* s'ha realitzat forma correcta. Un cop afegit

un *ActionEvent* correcte, el problema desapareix.

La utilització dels *events* fa que alguns components *Swing* s'hagin de declarar com a finals per poder utilitzar-los dins els mètodes definits dins els *ActionEvents* i similars.

La inclusió de diversos panells situats a l'interior de panells creats a pantalles superiors, com per exemple les pantalles relacionades amb la inclusió i esborrat de regles que se situen dins el panell secundari de la pantalla principal, fa que tots aquest panells es defineixen de forma general per fer-los accessibles a totes les classes.

2.4 Proves

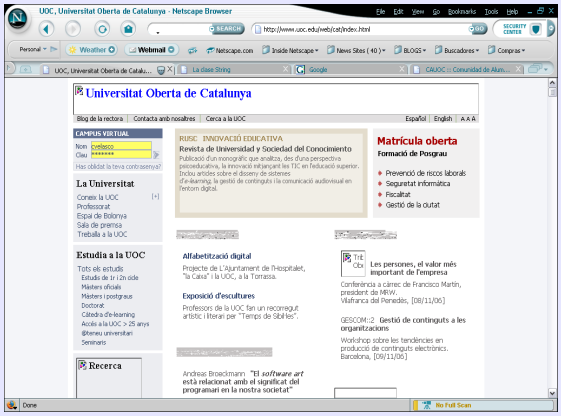
Per la realització de les proves, s'han utilitzat l'explorador *Netscape Browser 8.1* i s'han consultat, entre d'altres, les següents planes:

- http://www.uoc.edu/web/cat/index.html
- http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/string.htm
- http://www.cauoc.com
- http://www.elsentidodelavida.net

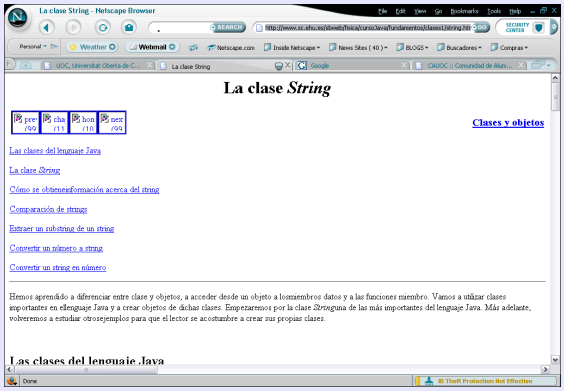
2.4.1 Proves de obtenció de la pàgina web

S'han realitzat dos tipus de proves. El primer tipus, dins l'IDE *Eclipse,* emprant les seves capacitats de execució i debug. El segon, mitjançant un fitxer jar i la comanda de la consola *java -jar jarfilename*, per l'execució des de l'entorn *DOS*.

Els resultats en ambdues proves son idèntics. I no podem dir que en un primer moment siguin gaire satisfactoris. A continuació hi ha una sèrie de captures on es mostren els resultats:



Il·lustració 3: Captura web uoc



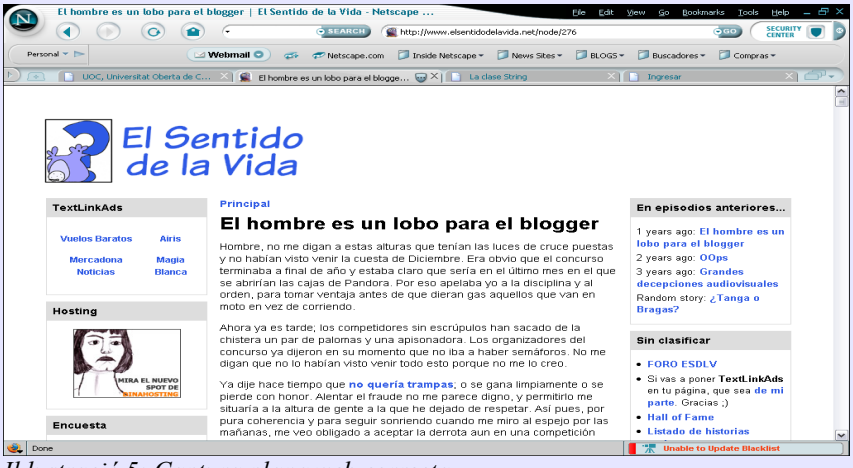
Il·lustració 4: Captura web basada en text

Com es pot apreciar a les captures, els elements multimèdia, en aquests casos imatges, no són capturades. Pel que fa a la resta de factors, la comunicació entre l'explorador i el **servidorWeb**, i la d'aquest amb els servidors de les pàgines web funciona correctament. L'únic a destacar és l'error amb la codificació ja mencionat.

Les solucions a adoptar ja han estat explicades a l'apartat de disseny, però a mode de resum, les dues direccions que es prenen principalment són la modificació del codi del *String* per la codificació UTF-8 o la utilització d'algun altra classe.

Finalment, s'ha optat per obtenir el codi directament del servidor i enviar-lo directament a l'explorador, un cop determinat que la plana no està filtrada per algun motiu fent servir el sistema original de capturar el codi en un *String*.

El resultat es mostra a continuació:

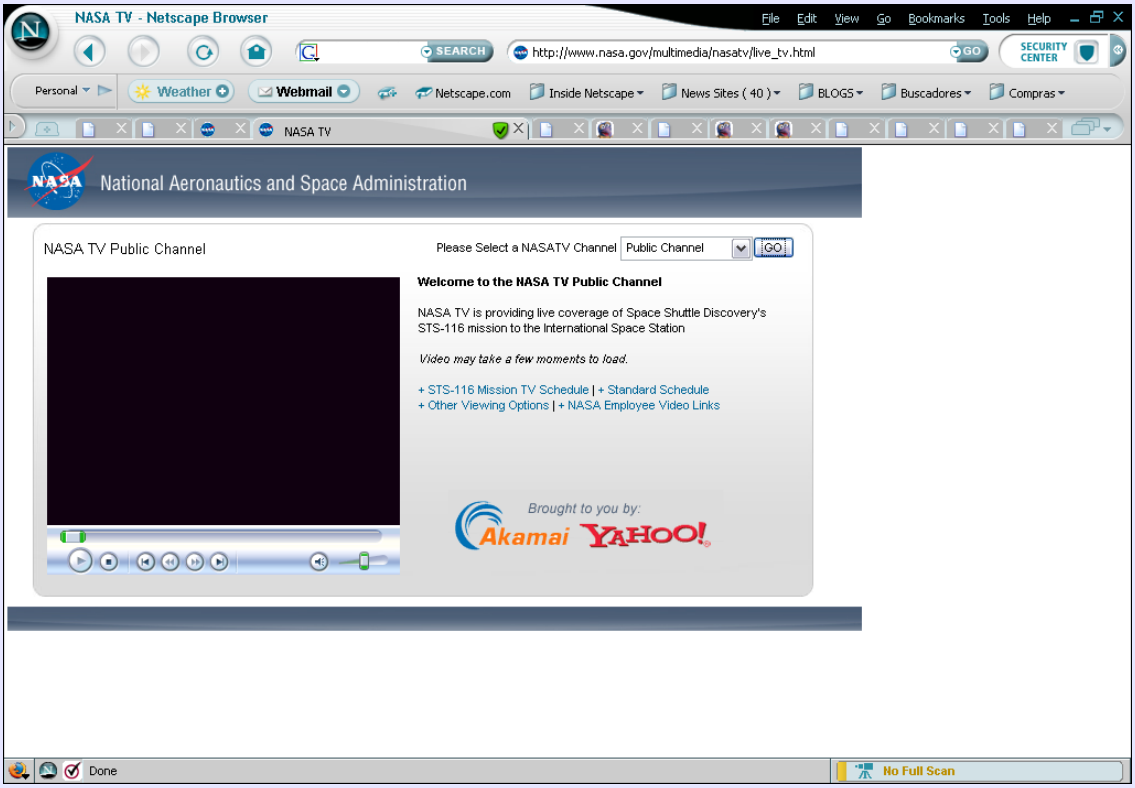


Il·lustració 5: Captura plana web correcta

Com es pot apreciar, ara sí es mostren les imatges, tot i que apareix un nou problema. No s'accepta la introducció de les dades d'identificació, quedant-se l'explorador a la mateixa plana. També es produeix un problema de memòria del *Java Heap Space* quan s'han consultat diverses planes.

També s'han realitzat les proves en entorn Linux, utilitzant l'explorador *Mozilla*. El resultat és el mateix que en entorn Windows. A continuació es mostra la captura de la pantalla en *Ubuntu*. .

Fins i tot, es mostren pàgines amb vídeo incrustat, com per exemple la següent, capturada de la pàgina de la *NASA*:



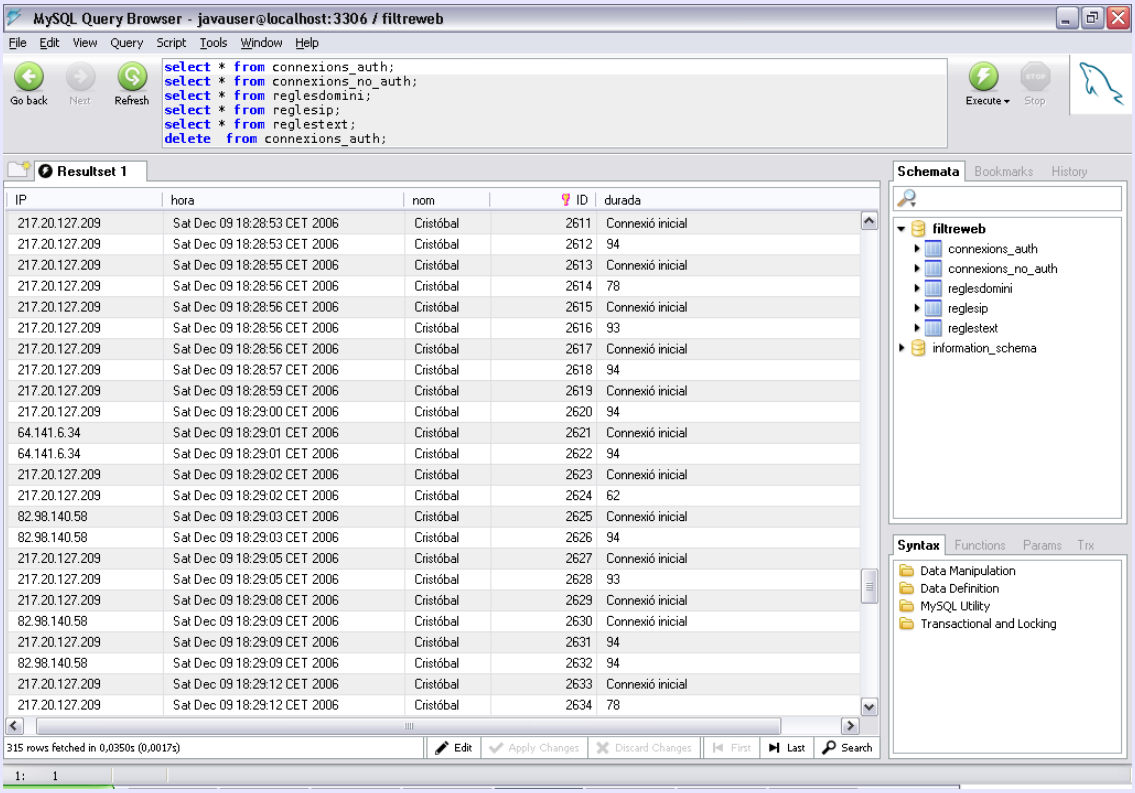
Il·lustració 6: Captura pàgina amb vídeo incrustat

2.4.2 Proves d'accés a la base SQL

Les proves s'han realitzar amb posterioritat a les proves d'accés web. En principi, utilitzant el mode *debug* de l'*Eclipse* es va provocar la impossibilitat de descarregar les planes. Això era degut a errors en la sintaxi, provocats per la inserció dels valors generats pel programari.

Les insercions, esborraments i consultes en si funcionen, però no es pot generar de manera automàtica pel programari, a causa d'aquests errors.

Les insercions al registre de connexions es produeixen de forma correcta i automàtica. A continuació es mostra una captura de la finestra del *MySQL Query Browser* amb les entrades del registre deconnexions autoritzades.



Il·lustració 7: Consulta de les dades sobre les connexions amb MySQL Query Browser

El programari entra automàticament les dades referents a la ip de destí, l'hora i data en que es produeix la connexió, la id de la connexió i finalment el temps que ha durat, indicant connexió inicial si és el cas.

La visualització en l'explorador presenta algun problema de comunicació. Per tant, es modificarà aquest apartat en el moment que estigui implementat l'entorn gràfic, per mostrar el registre al programari, no a l'explorador.

D'altra banda, es produeixen errors de sintaxi SQL quan es consulta la base de dades per comprovar els mots. Un error en la implementació de la sentència provocava aquests errors. Com s'ha indicat a l'apartat d'implementació (punt 2.3.4.2), s'eliminen els caràcters no alfanumèrics.

Les proves en entorn *Linux* han donat errades, degut a problemes amb la instal·lació de *MySQL* i errors de sintaxi en la versió instal·lada. El problema està originat per problemes en la configuració que impedeixen la actualització dels usuaris i la utilització de l'entorn d'administració *MySQL Administrator*. No es creen els registres ni es bloquegen les pàgines.

2.4.3 Proves entorn gràfic

Durant la implementació de l'entorn gràfic s'han realitzat diverses execucions de la classe finestres per intentar obtenir el resultat gràfic. Excepte les primeres

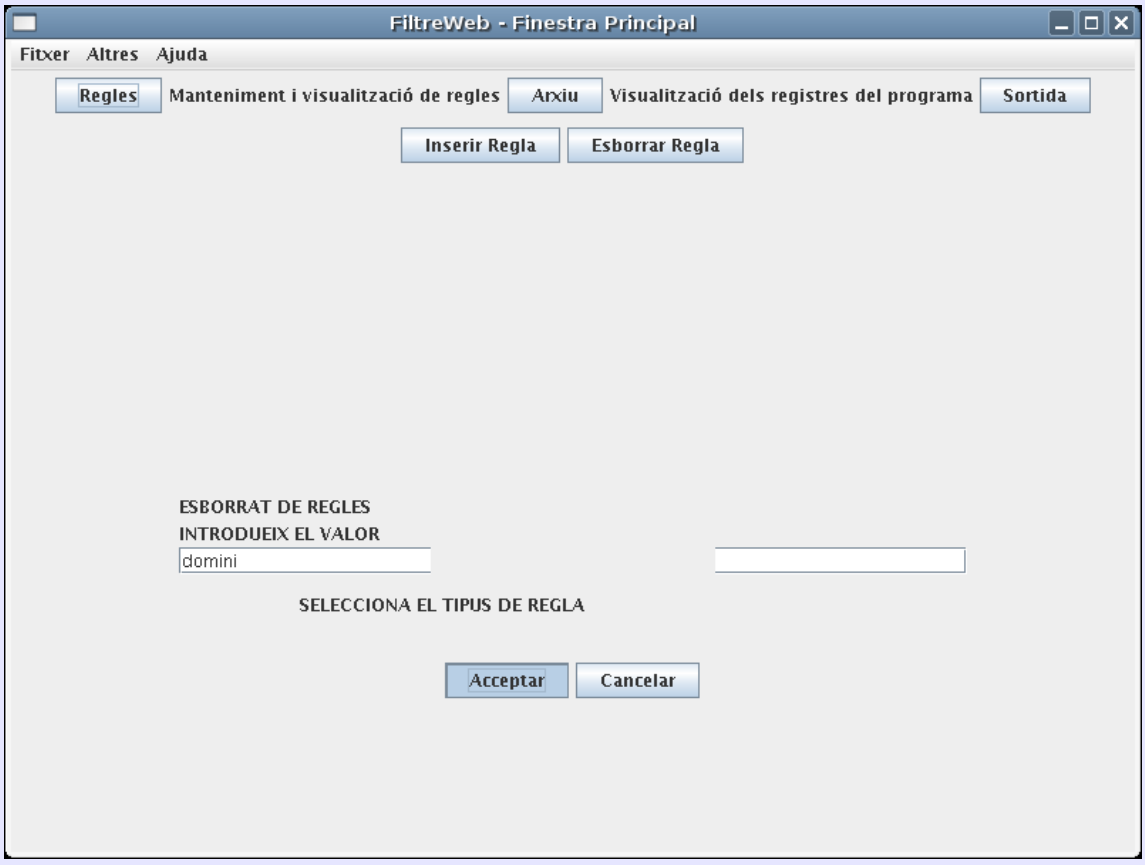
implementacions, que no van quedar capturades en imatge, el resultat ha estat positiu.

Un cop implementada tots els *events* s'ha continuat amb les proves del programari. Aquí s'han detectat els següents errors.

En primer lloc, molts errors de col·locació dels elements en pantalla, que s'ha resolt mitjançant la utilització de diversos *LayoutManagers* en cadascun dels *Panels*.

En segon lloc, es produeixen problemes d'actualització de les pantalles, on queden restes de la pantalla prèviament seleccionada.

El funcionament dels *events* es produeix sense errors. Cada element realitza la funció a la que està destinat sense cap problema de funcionalitat. Es produeixen alguns errors quan es passa d'una funció a un altra perquè no s'esborren part dels caràcters introduïts abans o quan es mostra una finestra **popUp** a sobre la pantalla principal, que al tancar la finestra no mostra el contingut que es trobava a sota del **popU**p:

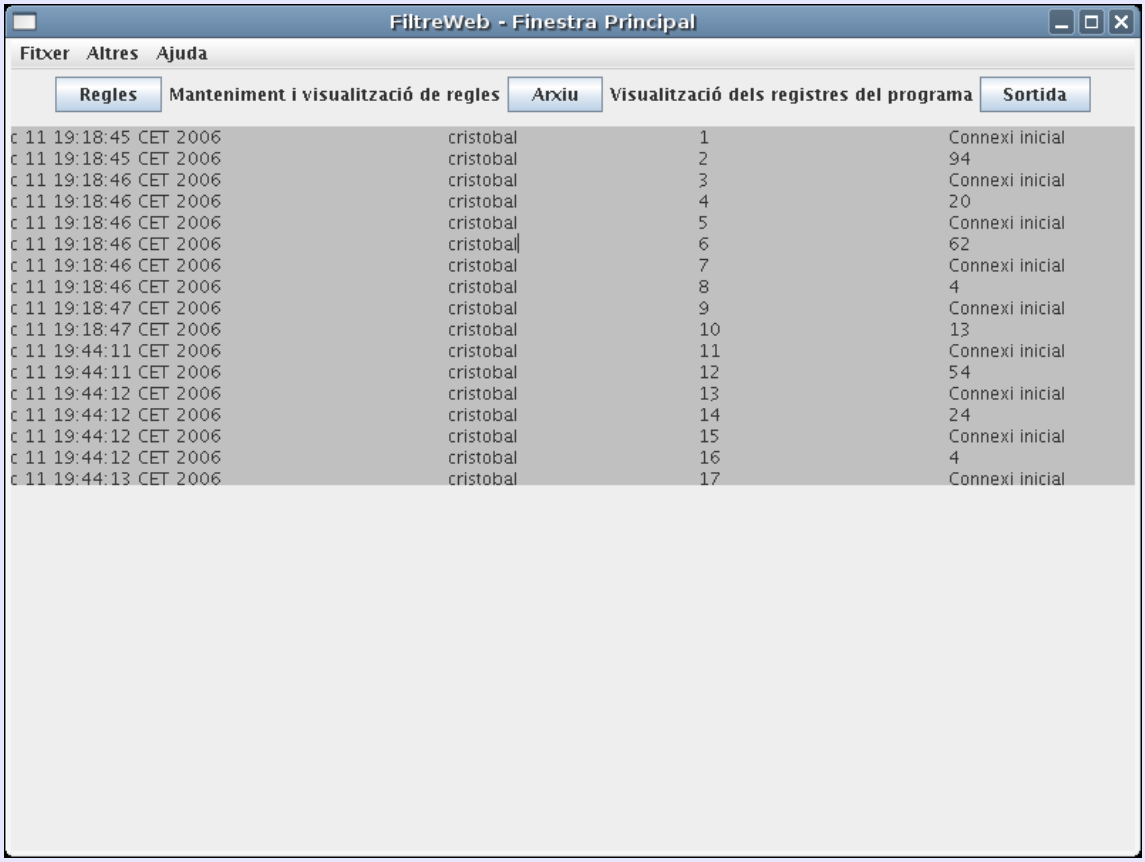


Donat que aquests errors es produeixen de forma diferent segons sigui *Windows* (els primers) o *Linux* (aquest últim), es creu que no és error de la implementació sinó problemes en els entorns d'execució de les *JVM*.

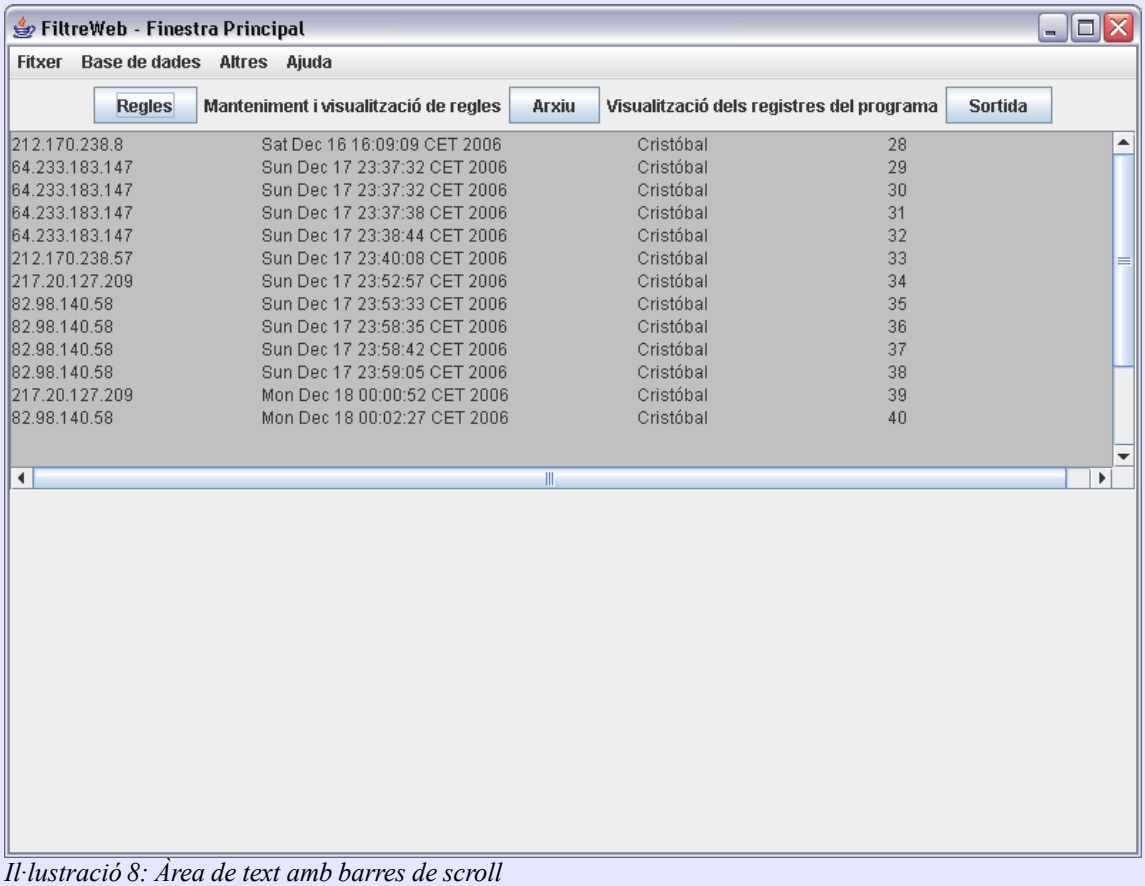
Relacionat amb aquest error anterior, en el cas de pressionar el botó cancel·lar en

entorn *Linux*, no s'esborra el contingut del *frame*, mentre que en *Windows* si que ho fa.

Un altre problema detectat durant les proves, ha estat el fet de que no apareixen les barres de navegació a la pantalla que mostra el resultat de les consultes, fent impossible accedir només a les primeres línies de dades. Aquest problema és comú al dos entorns, tant *Linux* com *Windows*



Això s'ha solucionat afegint l'àrea de text emprada per mostrar les dades dins un *JScrollPane*, en lloc del *JPanel* utilitzat fins al moment.



Il·lustració 8: Àrea de text amb barres de scroll

2.4.4 Proves generals

En aquest apartat estudiarem els problemes generals tant en entorn Windows com en entorns Linux.

2.4.4.1 Entorn Windows

Totes les funcions del programari funcionen correctament excepte alguna gràfica, com s'ha comentat al apartat anterior de proves de l'entorn gràfic.

El principal problema es troba en la memòria del sistema, ja que per algun motiu no detectat a vegades es crea un *OutOfMemoryError* del *java.heap.space*. Aquest error es produeix en diverses ocasions, però no sempre que es donen aquestes circumstàncies.

Després d'augmentar la persistència d'aquests error, i davant un possible error d'implementació, s'ha tornat a estudiar tot el codi i s'ha trobat un error en el tancament d'un *Resultset* ,on no s'executava l'ordre *resultset.close()* que provocava l'error de memòria. De totes maneres, la eficiència del programa encara és baixa, donat que tarda bastant temps en realitzar determinadestasques, com alguna consulta a la base de dades.

Els dos entorns en que es produeix aquest error són, per una banda, realitzant la consulta de connexions autoritzades, que es realitza amb molta lentitud i elevat nombre de requeriments , i per l'altra quan es troben obertes diverses pàgines web a l'hora.

Tal i com s'ha indicat, el segon estava relacionat amb l'esgotament de la memòria per l'omissió d'un ordre *close()* mentre que el primer no s'ha pogut detectar el motiu. Com s'indica al apartat de proves generals en *Linux*, aquests errors es produeix només en *Windows*, cosa que indica més un problema d'entorn que de implementació.

Durant les proves finals, a aparegut un error que no s'havia donat fins al moment. Les pàgines web no es bloquegen per cap motiu. Durant la càrrega de les planes inicials, entre les que es troba la plana de la *UOC*, que en un principi sí havia estat bloquejada, aquesta s'ha carregat i a partir , d'aquest moment no s'ha bloquejat cap pàgina, ni per ip, ni per mot, ni per nom. A partir d'aquest moment, es centra tot el treball en la reparació d'aquest error. Després de modificar la excepció que indica que la pàgina web s'ha de bloquejar, els bloqueigs per IP i domini funcionen correctament.

L'error finalment ha estat un error en la sintaxi de la consulta formulada a la base de dades, on s'utilitzaven cometes simples enlloc de dobles per la qual cosa no detectava els mots filtrats.

Per acabar, l'accés a la base de dades quan el volum d'aquestes és elevat provoca que el programa es quedi col·lapsat per causes que no s'han pogut determinar, ja que l'accés directe a la base de dades és correcte. Una possible causa pot ser el connector amb la base de dades, però no s'ha pogut comprovar.

2.4.4.2 Entorn Linux

Apareixen els erros gràfics comentats anteriorment, tot i que el funcionament per la resta és correcte. Esporàdicament, segons les planes web consultades, apareixen errors de streaming, quan es produeixen errors en l'enviament de la plana a l'explorador. Aquests errors queden corregits al repetir la petició l'explorador.

Tot i que el bloqueig de llocs web pateix dels mateixos problemes que en l'entorn Windows, la consulta a la base de dades es realitza de forma correcta, sense que es produeixin retards. De la mateixa manera que s'ha indicat en l'apartat anterior, els errors de bloqueig han estat solucionats.

2.5 Requisits previs i utilització del producte

2.5.1 Requisits previs i instal·lació

En la execució del programa necessitem la utilització dels següents programes:

- *MySQL*, SGBD, amb la utilització de la base de dades adjunta al producte.
- Utilització del programari *MySQL Administrator* i el *MySQL QUERY BROWSER* per agilitzar la instal·lació. Tot i que no és imprescindible, sí és recomanable.
- Connector *Java-MySQL mysql-connector-java-5.0.3*, que es troba inclòs dins el producte.
- Entorn d'execució de Java, preferentment el *jre1.5.0_06*, que és l'emprat en la realització del producte. Versions anteriors a la *1.5.0* no són compatibles donat la utilització de certes classes no implementades fins aquesta versió. La versió posterior, la *1.6.0*, publicada recentment, provoca problemes amb el connector de la base de dades. Per tant, fins que no es solucioni aquest problema, tampoc és possible la utilització de la versió més recent.

Per instal·lar el programari, en primer lloc hem de crear la base de dades, des de l'*Administrator* ,afegint un catàleg anomenat filtreweb. A continuació, amb la importació la taula base de dades adjunta al producte amb la opció *restor*e, importarem les taules i finalment, hem de donar accés al usuari que volem fer servir per l'accés a la base de dades **filtreweb**, des de la pestanya d'usuaris afegim tots els privilegis disponibles a l'usuari triat.

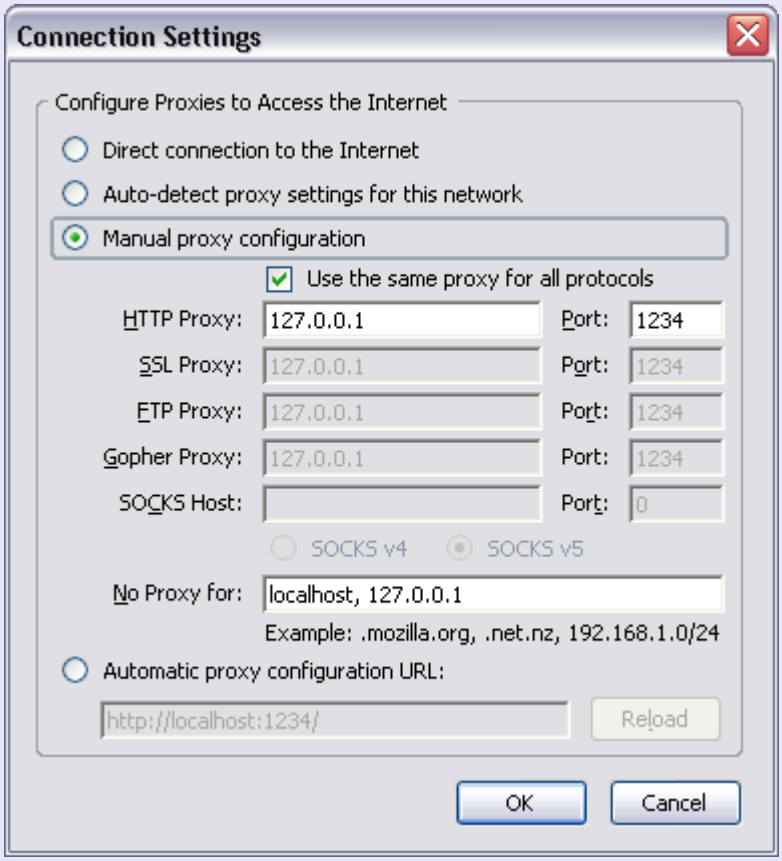
Totes aquestes operacions es fan com a usuari *root* o amb privilegis equivalents.

Per més informació sobre la instal·lació de *MySQL*, consultar la següent pàgina web. http://dev.mysql.com/doc/refman/5.1/en/windows-installation.html i sobre la instal·lació del *MySQLAdministrator* consultar http://dev.mysql.com/doc/administrator/en/ch02s02s01.html

Després, s'ha de configurar l'explorador perquè utilitzi el proxy localhost:1234. Per això, cal consultar les instruccions de l'explorador. Per realitzar les proves, s'ha utilitzat el *Netscape Browser 8.1*, del qual mostrem les instruccions a continuació:

- menú *tools*
  - *options*
- clicar a *General*
- clicar a *Connections Settings*

i posar la configuració com es mostra a la següent captura:



Il·lustració 9: Configuració del proxy a l'explorador Netscape Browser 8.1

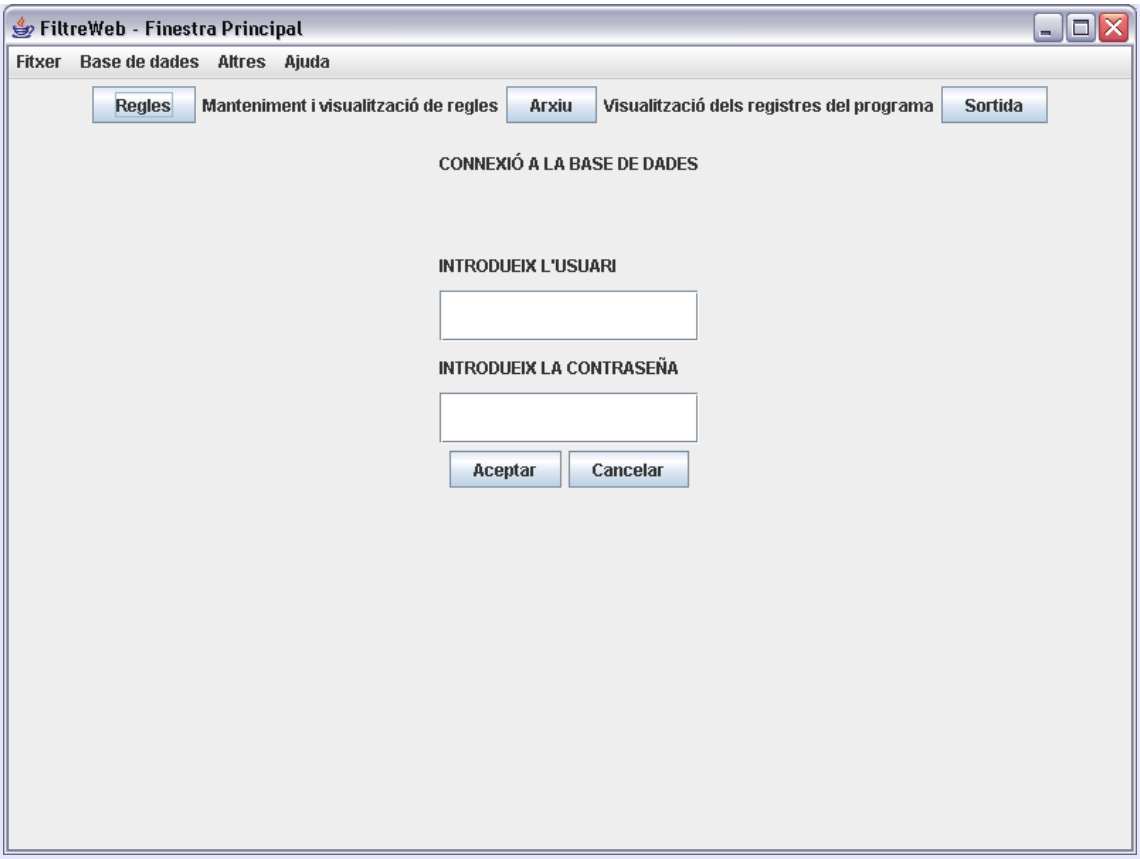
A continuació només cal executar a la línia de comandes, si és que no s'obren automàticament els *jar*, posant la comanda j*ava, java -jar filtrewebapp.jar* , que obrirà el programari de control i iniciarà el servei del proxy.

2.5.2 Manual d'usuari

Donat l'entorn gràfic el funcionament és realment senzill. Un cop llençat el programari mitjançant la comanda de consola *java -jar filtrewebapp.jar*, o executant el fitxer *jar* si està disponible tant en entorn *Windows* com *Linux*, tenim la possibilitat de realitzar gairebé totes les funcions tant per pantalles com per menú. Només hi ha algunes funcionalitats que no estan implementades amb pantalla, tot i que no són les habituals.

A continuació explicarem totes les funcionalitats del programari i el mecanisme d'execució, tant per menú com per pantalles.

En primer lloc, a mode introductori, tenim una captura de la pantalla principal:

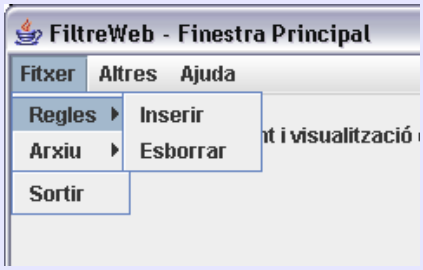


Il·lustració 10: Pantalla Principal

Com es pot apreciar, aquesta conté el menú i els botons principals: *Regles* per la manipulació de les regles de filtratge, *Arxiu*, on es pot consultar tant regles com els registres de connexió i el botó de *Sortida*, que tanca el programa, en la part superior de la finestra. La part central conté el sistema d'accés a la base de dades, que és la primera operació a realitzar, i sense la qual no funciona correctament. Un cop introduïdes les dades, pressionem el botó *Acceptar* per enviar les dades o *Cancel·lar* si volem tornar a introduir les dades. Tot i que és possible accedir a d'altres opcions, fins que no s'ha realitzat la connexió amb la base de dades la resta de funcions no treballen de forma correcta, ja que totes necessiten accés a la base de dades.

2.5.2.1 Manipulació de regles

La manipulació de regles consta de dos funcions principals, la inserció i l'esborrat de les mateixes. El menú *fitxer*, per mitja del menú regles, ens permet accedir a les dues, mostrant-nos la pantalla adequada per realitzar la funció.



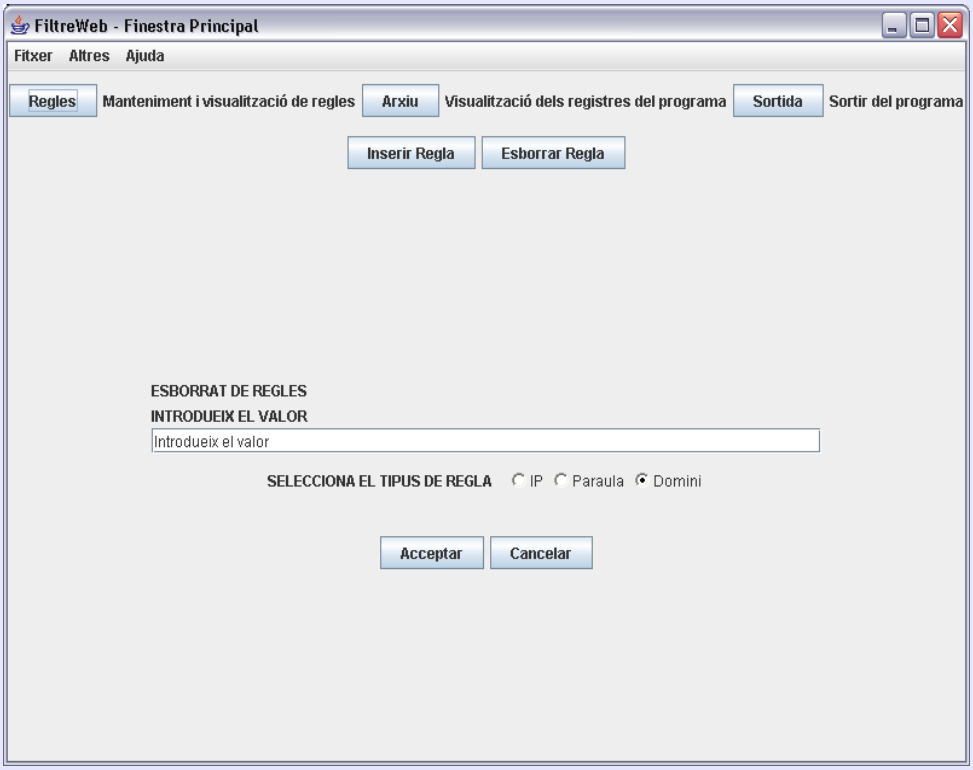
Il·lustració 11: Opcions del menú per la manipulació de regles

Quan seleccionem la opció apropiada, ens apareix la pantalla relacionada

- Insercions

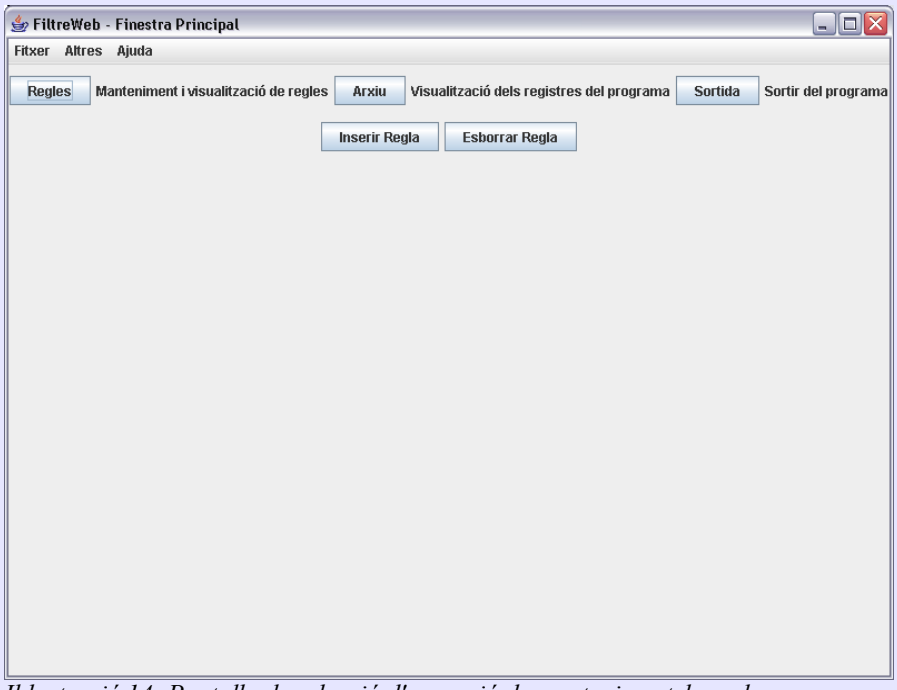
Il·lustració 12: Pantalla de inserció de regles

- Esborrat:



Il·lustració 13: Pantalla per esborrar regles

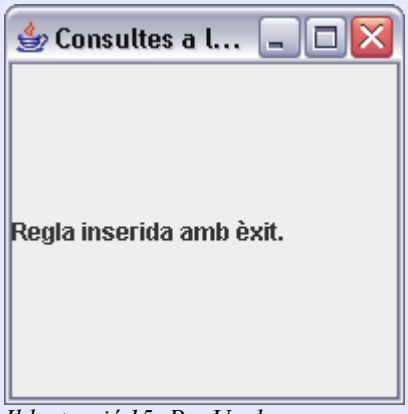
El mecanisme de manipulació mitjançant els botons de pantalla funciona de forma lleugerament diferent, ja que al seleccionar el botó *Regles*, apareix la pantalla de selecció d'operació a realitzar, sense cap mètode d'entrada de dades:



Il·lustració 14: Pantalla de selecció d'operació de manteniment de regles

Els mecanisme per introduir les dades consisteix en la introducció del valor al

camp de text, la selecció del tipus de regla i finalment pressionar el botó acceptar. En aquest cas, s'introdueix en la base de dades, torna a la pantalla de selecció d'operació i ens mostra una finestra de confirmació:



Il·lustració 15: PopUp de confirmació

El mecanisme d'eliminació funciona de forma anàloga, canviant només el missatge de confirmació. Cal remarcar també que en cas d'error es mostra un missatge dins la finestra de confirmació, del tipus següent:

Il·lustració 16: PopUp d'avís

2.5.2.2 Consulta de dades

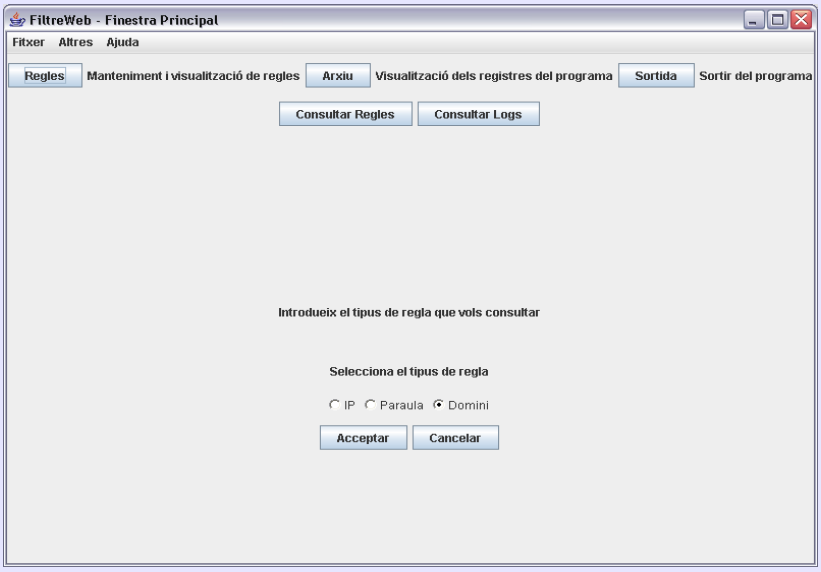
Aquest apartat cobreix la consulta tant del registres de connexions com de les regles existents a la nostra base de dades. De la mateixa forma que el manteniment de regles, es pot realitzar per menú i per la pantalla principal.

El menú és el següent:

Il·lustració 17: Opcions del menú per accedir a les dades

En aquest apartat, només veurem les dues primeres opcions, deixant la tercera per l'apartat següent sobre altres funcions.

El menú *regles*, i el botó *Regles* de la pantalla principal,ens porten a la pantalla de selecció de tipus de regles:



Il·lustració 18: Pantalla de consulta de regles

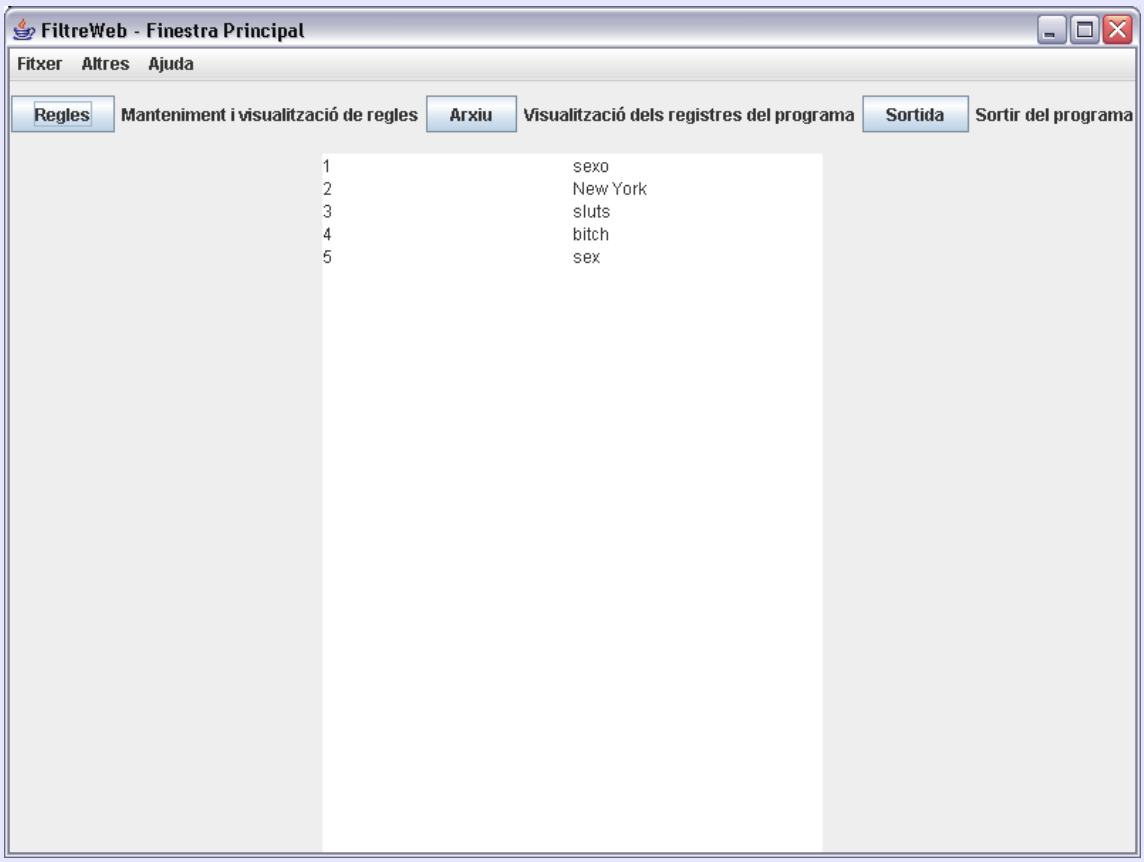
La pantalla de selecció de registres es mostra a continuació:

Il·lustració 19: Pantalla de consulta de connexions

També en aquest cas el funcionament del botó *Arxiu* té el comportament anterior, mostrant la pantalla de selecció de tipus de consulta sense cap mètode d'entrada:

Il·lustració 20: Pantalla de selecció de tipus de consulta de dades

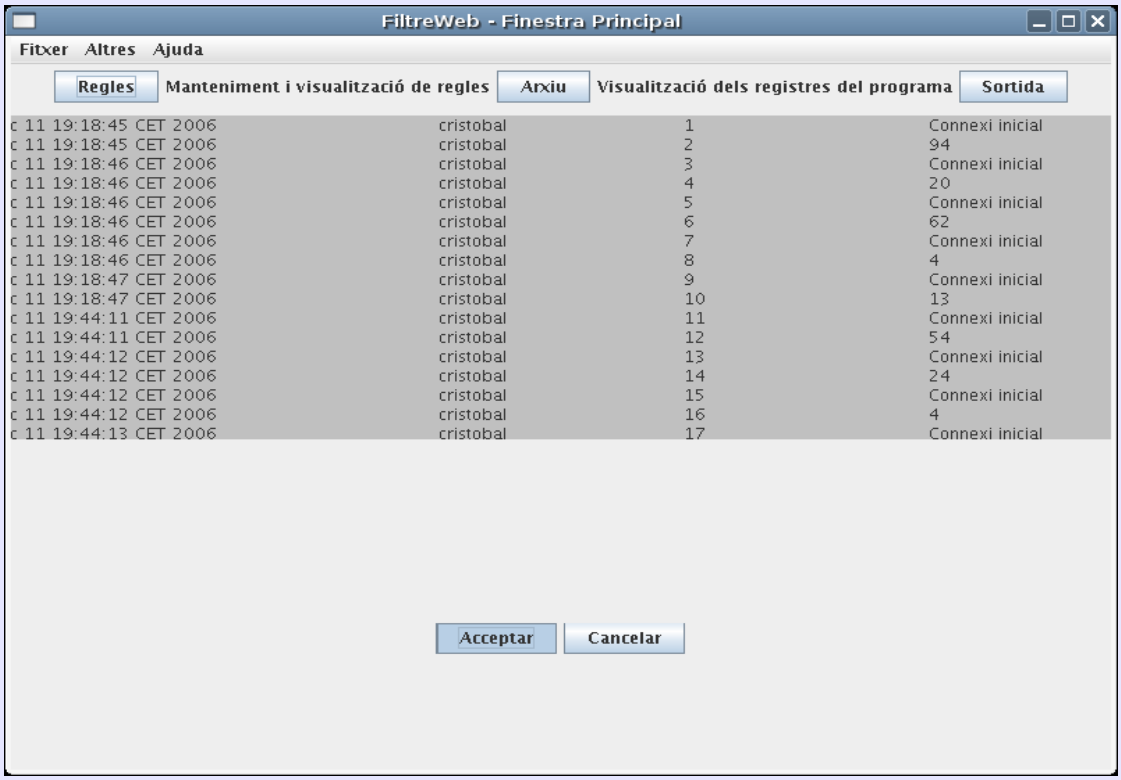
Un exemple de les captures és el següent, que mostra tots els mots filtrats:



Il·lustració 21: Resulta de la consulta dels mots filtrats

La resta de pantalles es mostren de forma anàloga. Es mostra un altra a mode d'exemple.

Il·lustració 22: Pantalla amb la consulta sobre connexions autoritzades



2.5.2.3 Sortida i altres funcions

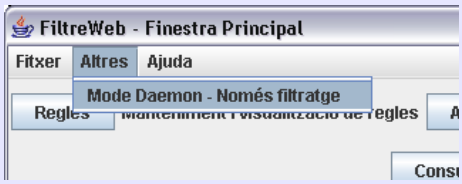
Aquest apartat cobreix la resta de funcions del programa. En primer lloc, tractarem l'opció del menú *esborrar logs* , que s'encarrega d'eliminar totes les dades de la base de dades sobre les connexions:

Il·lustració 23: Opció del menú per esborrar els registres de connexions

Aquesta opció permet esborrar tots els registres, tant el de connexions autoritzades com el de connexions bloquejades.

El menú *Altres*, conté la opció *Mode Daemon*. Aquesta opció tanca la finestra principal sense tancar el programari. Aquest només es tancarà quan tanquem l'ordinador. Serveix per mantenir el programari amb la funció de proxy sense poder manipular les

regles ni consultar els registres permitjà del programari.



Il·lustració 24: Selecció del mode Daemon

Una altra funció, afegida amb posterioritat ha estat la possibilitat de connectar o desconnectar amb la base de dades des del menú. Aquesta funcionalitat ha estat afegida per poder recuperar-se d'errors de connexió amb la base de dades. Està disponible des del menú *Base de dades*.

Finalment tenim el menú Ajuda, que conté les opcions Ajuda i Sobre. La primera està destinada a incorporar la ajuda en línia i la segona informació sobre l'autor del programari. La ajuda estarà implementada en una nova finestra. En principi, donat que l'ajuda no es troba disponible, només s'obre una finestra per informar d'aquest fet.

La sortida del programari es realitza, sempre que no funcioni en mode Daemon, mitjançant les opcions:

- *Menú Fitxer* : Opció sortir.
- *Finestra*: Tancant la finestra el programari s'atura. Si es vol tancar la finestra però que el programari es continuï executant, s'ha de triar la opció *mode Daemon* que es troba al menú *Altres* .
- *Botó Sortida*: Pressionant el botó Sortida, es deté la execució del programari.

2.6 Funcionalitats futures.

Tal i com s'ha comentat al resum , hi ha una sèrie de funcionalitats previstes en la implementació del programari que no s'han realitzat, principalment per ajustar la implementació al calendari disponible. A continuació es detallen els conceptes més importants de cadascuna d'aquestes funcions

- *autenticació*: El nostre programari té un sistema d'identificació bàsic relacionat

amb la gestió de la base de dades. Aquesta funció ampliaria i independitzaria el concepte a la execució del programa, deixant la manipulació de la base de dades a un usuari concret i permeten la utilització per tots els usuaris i la manipulació de la base de dades només als usuaris autoritzats. El funcionament d'aquesta funció serà el següent:

- *Identificació de l'usuari al programari*: Segons l'usuari permet realitzar modificacions o no.
- *Accés a la base de dades*: Si l'usuari està autoritzat, permet afegir i modificar regles, consultar els registres, etc. En aquest cas,l'usuari del programari és independent de l'usuari de la base de dades. Tots els usuaris autoritzats fan servir el mateix usuari de la base de dades, tot i que també es registra en aquest cas quin usuari del sistema ha afegit la regla.

- *Generació d'informes*: Aquesta funció permet generar informes amb funcions més avançades, agrupant els registres, amb consultes SQL complexes, etc. De la mateixa manera, també ha de permetre generar documentació en format PDF, ja que és un format totalment portable, amb el resultat d'aquestes consultes. Les funcions, per tant, són dos:
  - *Realització de consultes:* Permet accedir a les dades de diverses formes i no com un simple llistat. Ha de permetre consultes amb les clàusules *GROUP BY* i *WHERE* així com permetre consultes a diverses taules alhora.
  - *Creació de documents:* A partir de les consultes realitzades amb anterioritat, s'han de generar arxius pdf amb els resultats.
- *Ampliació de protocols*. Aquesta funcionalitat apareix a partir dels problemes d'implementació en la captura de les dades que comporten la limitació dels protocols al protocol HTTP. La funcionalitat és ben senzilla, permetre tots els protocols implementats als exploradors.

3. Conclusions

Per començar les conclusions, cal indicar els punts crucials de la implementació del programari així com el problemes que s'han resolt i els que hanquedat per resoldre.

- En primer lloc, cal remarcar la importància de la captura de les dades del servidor pel seu posterior tractament i la transmissió a l'explorador. Aquesta tasca és la que permet realitzar totes les posteriors i és la que d'alguna manera marca la implementació de tot el programari. Durant la implementació ha provocat molts problemes, sobretot en la captura d'elements multimèdia, que finalment ha estat resolta, tot i que d'una forma poc elegant, ja que es realitzen dues connexions al servidor web per cada petició, la primera per realitzar les comprovacions i la segona per poder transmetre les dades directament del servidor a l'explorador.

- De la mateixa manera, els problemes per capturar el tràfic HTTPS redueixen de forma considerable l'efectivitat del programari. Pot ser la mancança més important del producte. Un cop solucionat aquest problema s'ha d'incloure més protocols, com poden ser el FTP, TFTP i Telnet, per exemple.

- En un desenvolupament real del programari, s'ha de reformar tot el tema de la captura. En el cas del present treball, s'ha volgut implementar des de la base, incloent la captura de dades. Hi ha d'altres opcions, com poden ser la utilització de la eina jpcap per la captura de paquets, que no s'han tingut en compte degut a la voluntat d'implementar tot l'entorn de captura de paquets, a més de garantir la portabilitat del producte final.

- Cal remarcar que la portabilitat del producte és total. El comportament del producte és pràcticament idèntic en tots els entorns que ha estat testat, podent considerar que les diferències es deuen més a l'equip utilitzat que a raons de sistema operatiu. La base d'aquesta portabilitat són, per una banda, la implementació del programari a nivell bàsic, inclosa la captura de les dades de l'explorador, sense la utilització de cap eina addicional, i d'altra banda, un requeriments de programari molt específics però que alhora són suportats per diversos sistemes operatius.

- S'han decidit diversos canvis d'enfocament en el disseny del programari. El primer, en etapes inicials del desenvolupament ha estat la utilització del programari com a proxy enlloc de controlar tots els paquets que surten del sistema. Tot i que és un canvi molt important, ja que afecta a tot el desenvolupament del programari a partir d'aquesta decisió, el fet de prendre'l

durant les etapes inicials mostra la flexibilitat del projecte i l'adaptació als problemes que sorgeixen durant el projecte.

Hi ha altres aspectes ha tenir en compte, relacionats tant amb la implementació com amb decisions de disseny i funcionalitats que s'han afegit amb posterioritat, que cal remarcar.

La part que pot ser és la més feixuga d'implementar ha estat l'entorn gràfic, degut principalment a la repetició de codi en cada ActionEvent i la dificultat de detectar alguns errors, ja que l'actualització de les pantalles a vegades era errònia, però més aviat relacionada amb l'entorn de maquinari i programari emprat que amb la pròpia programació.

Algunes funcionalitats han estat detectades després de realitzar les proves, com a conseqüència d'errors detectats, com és el cas de la connexió a la base de dades per menú, mentre que d'altres han estat detectades amb posterioritat, com per exemple la necessitat d'oferir un funcionamentsense la presència de la finestra principal.

Hi ha alguns canvis menors, com poden ser:

- El sistema de mostrar els registres de connexions, inicialment pensat per fer-ho mitjançant una web i finalment dins el programari mateix.

- El sistema de càlcul del temps de connexió. En principi es volia determinar quan temps es mostrava una plana, però al final s'ha implementat un control de connexions per cada petició realitzada.

Aquests canvis han estat motivats per dificultats en la implementació, i han sorgit per la necessitat de solucionar diversos problemes.

Resumint, s'ha aconseguit un programari que realitza la tasca bàsica, amb alguna mancança per dificultats en la implementació. Altres aspectes s'han millorat de la idea inicial, com pot ser la portabilitat, ja que la idea inicial era que fos fàcilment adaptable a entorn *Linux*, i el resultat final ha estat totalment funcional als dos sistemes sense necessitat de modificacions.

4. Glossari

SGBD:

Un **S**istema de **G**estió de **B**ases de **D**ades és un programa informàtic dissenyat per facilitar la gestió i dur a terme les tasques necessàries pels programes d'aplicació.

Aquestes tasques van des de facilitar el control de la redundància de les dades,la seva integració, la seva integritat fins a la seva seguretat i recuperació.

SQL

SQL (*Structured Query Language* o *Llenguatge de consulta estructurat*) és un llenguatge estàndar de comunicació amb base de dades relacionals. És a dir, un llenguatge normalitzat que permet treballar amb la majoria de bases de dades relacionals. L'SQL es pot hostatjar (es pot utilitzar) dins d'altres llenguatges de programació. La principal característica d'aquest llenguatge és la seva simplicitat, ja que amb pocs coneixements es poden fer consultes bàsiques sobre una base de dades, encara que no per això deixa de ser un llenguatge complet, tant relacionalment com computacionalment (a partir de la versió SQL3 publicada el 1999).

Daemon

Procés, usualment en un sistema operatiu UNIX, que corre en segon pla, enlloc de amb control directe de l'usuari. Generalment arranquen al iniciar el sistema i no tenen cap procés pare sinó el init.

Proxy

És un ordinador o programari que permet realitzar connexions a serveis de xarxa de forma indirecta. Els altres ordinadors o programes es connecten amb el proxy que és qui s'encarrega d'aconseguir el servei i retornar els resultats al programari que demanava el servei

Sniffer

És un programari que permet la captura dels paquets que s'envien per mitjà d'una interfície de connexió a la xarxa, habitualment de tipus Ethernet, que ens permet veure el funcionament dels diversos serveix que s'executen.

HTTP

El protocol HTTP estableix el mecanisme per a l'intercanvi de documents d'hipermèdia al web. Es basa en un sistema de servidor i client on el client efectua

les peticions que el servidor s'encarrega de respondre.

HTTPS

El protocol HTTPS implementa la comunicació segura per al protocol HTTP. Utilitza el xifrat SSL per impedir la recuperació no autoritzada de les dades.

FTP

El protocol FTP estableix el mecanisme per a l'intercanvi de fitxers entre qualsevol sistema operatiu.

5 Bibliografia

Llibres

- Redes de Computadoras. Andrew S. Tanenbaum. Prentice Hall ISBN 968-880-958-6
- Java Network Programming. Elliot Rusty Harold. O'reilly ISBN 1-56592-227-1
- An Introduction to network programming in Java. J Graba. Springer ISBN 978-1-84628-380-2
- Java para Estudiantes Douglas Bell Prentice Hall ISBN 970-26-0144-4

Apunts

- Mòduls Xarxes. Jordi Iñigo Griera. ISBN 84-9707-207-3
- Mòduls Base de Dades I. Jaume Sistacs Planes ISBN 84-8429-586-9
- Mòduls Base de Dades II. Jaume Sistacs Planes . ISBN 84-9707-426-2

Internet

- http://www.ibiblio.org/javafaq/
- http://www.javahispano.org Informació general sobre Java i exemples
- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/V_2.htm Sockets en Java
- http://adrformacion.com/cursos/javaser/leccion2/tutorial3.html Servlets HTTP
- http://www.lfcia.org/openprojects/camllets/doc/html/node19.html#SECTION00733200000000000000 Informació sobre el protocol HTTP
- http://www.rfc-es.org/ . Documentació de referència sobre WWW
- http://dev.mysql.com/doc/refman/5.0/en/connector-j-usagenotes-basic.html#connector-j-examples-connection-drivermanager Crear la connexió amb la base de dades.
- http://www.tutorial-enlace.net/tutorial-Obtener_IP_de_una_URL-2118.html Obtenir la IP d'un host
- http://www9.org/w9cdrom/122/122.html Captura d'imatges
- http://www.idevelopment.info/ Exemples de codificació en Java
- http://www.programacion.com Exemples de codificació en Java
- http://www.pfad.demon.co.uk/free.htm#Proxy Exemple de codificació d'un proxy en Java per TCP
- http://www.nsftools.com/tips/JavaTips.htm Codificació d'un proxy en Java
- http://muffin.doit.org/ Proxy en Java completament funcional.
- http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html Programari per la captura de paquets en Java.
- http://ca.wikipedia.org/wiki/Portada Definicions de termes.

6 Annexos

6.1 Programari

El programari utilitzat per la realització d'aquesta aplicació haestat el següent:

- Eclipse. IDE en Java, per la realització de les classes en Java
- Sun Java Studio Enterprise 8. Utilitzat per la realització del diagrama UML
- MySQL 5.0. SGBD encarregat de l'emmagatzematge de tota la informació
- MySQL Connector/J. Pont per la comunicació Java-SQL
- Netscape Browser 8.1 explorador de Internet.
- Mozilla Browser 1.7.12
- Opera 9.10

6.2 Maquinari

El producte ha estat realitzat i provat en aquest dos sistemes:

- Compaq Presario
  - Pentium M745 a 1,8Ghz
  - 1GB Ram
  - Ethernet 10/100
  - SO Windows XP Home Edition
- Hp Pavilion
  - Athlon 2000+ a 1,67Ghz
  - 750MB RAM
  - Ethernet 10/100
  - S.O.:
    - Ubuntu Dapper
    - Solaris 10

6.3 El protocol HTTP

En aquest apartat farem un petit resum del funcionament del protocol http, que és la base del funcionament del WWW.

El funcionament es basa en la comunicació entre el client i el servidor. El primer envia peticions HTTP que ha de ser respostes pel servidor, que posteriorment tanca la connexió. En el cas del protocol HTTP/1.0, per cada petició ha de ser realitzada amb una connexió, mentre que en el cas del protocol HTTP/1.1, es permeten diverses peticions dins la

mateixa connexió.

Les peticions es poden produir amb diversos mètodes, sent aquests, en el cas del HTTP/1.0, el mètode GET, el mètode HEAD i el mètode POST. En la versió HTTP/1.1 s'han afegit diversos més, però el funcionament bàsic es realitza a partir d'aquests tres mètodes. A continuació s'expliquen les diferències entre els tres mètodes:

- Mètode GET. És el mètode que es fa servir per recuperar el fitxer que es vol obtenir del servidor, indicat en l'URI especificat a la petició. Es pot obtenir aquest fitxer o, en el cas de ser un programa, s'obtindrà el resultat de l'execució d'aquest.
- Mètode HEAD. En aquest cas, només s'obté la capçalera del missatge HTTP. Serveix per comprovar que el URL és vàlid.
- Mètode POST. En aquest cas, s'envien dades que s'han de fer servir pel recurs que es troba indicat al URI.

Totes aquestes peticions generen codis de resposta, tan en el cas d'èxit com en el cas d'error. El codis, en el cas del HTTP/1.0, són els següents:

- 1xx Missatges d'informació

N°	Descripció
100	Continua
101	Canvi de protocol

- 2xx Operació amb èxit

N°	Descripció
200	OK
201	Creat
202	Acceptat
203	Informació no oficial
204	Sense Contingut
205	Contingut per tornar a carregar
206	Contingut parcial

- 3xx Redirecció cap a un altre URL

N°	Descripció
300	Múltiples possibilitats
301	Mudat permanentment
302	Trobat
303	Vegi altres
304	No modificat
305	Utilitzi un proxy
307	Redirecció temporal

- 4xx Error per part del client

N°	Descripció
400	Sol·licitud incorrecta
401	No autoritzat
402	Pagament requerit
403	Prohibit

404 No trobat
405 Mètode no permès
406 No acceptable
407 Proxy requerit
408 Temps d'espera exhaurit
409 Conflicte
410 Ja no disponible
411 Requereix longitud
412 Errada precondició
413 Entitat de sol·licitud massa llarga
414 URL de sol·licitud massa llarg
415 Tipus de medi no suportat
416 Rang sol·licitat no disponible
417 Errada expectativa

- 5xx Error por part del servidor

N°	Descripció
500	Error intern
501	No implementat
502	Passarel·la incorrecta
503	Servei no disponible
504	Temps d'espera de la passarel·la exhaurit
505	Versió de HTTP no suportada

En resposta a aquest codis, el client respon amb noves peticions o generant errors.

6.4 Llenguatge SQL

En l'accés a la base de dades s'utilitza de forma continuada el llenguatge SQL. Aquest llenguatge es basa en consultes que retornen les dades o produeixen canvis en les dades enregistrades a la base de dades. El llenguatge SQL és força complicat, però a continuació farem un petit resum de les principals consultes emprades en la realització d'aquest treball No s'han fet servir cap sentències de definició, ja que la base de dades ha estat creada amb l'assistència d'un programari de control, però sí que s'han utilitzat àmpliament en la implementació del programari consultes de manipulació-

- SELECT. Aquesta consulta és l'encarregada de realitzar les consultes per retornar els valors que estem cercant. La seva sintaxi bàsica és la següent, tenim en compte que hi ha camps optatius, posats entre [].
  - select <camps a retornar> from <nom de la taula> [where <camp >=valor];
- DELETE. Es fa servir aquest consulta per eliminar dades de la base. Pot eliminar dades de tota la taula o seleccionar entre les entrades les que s'esborren al complir certa condició:
  - delete from <nom_taula> [where <camp>= valor];
- INSERT. Amb aquesta consulta es realitzen les insercions a la base de dades. Serveix per inserir noves dades a la base.

- Insert into <nom_taula> (columnes) VALUES (valors);