

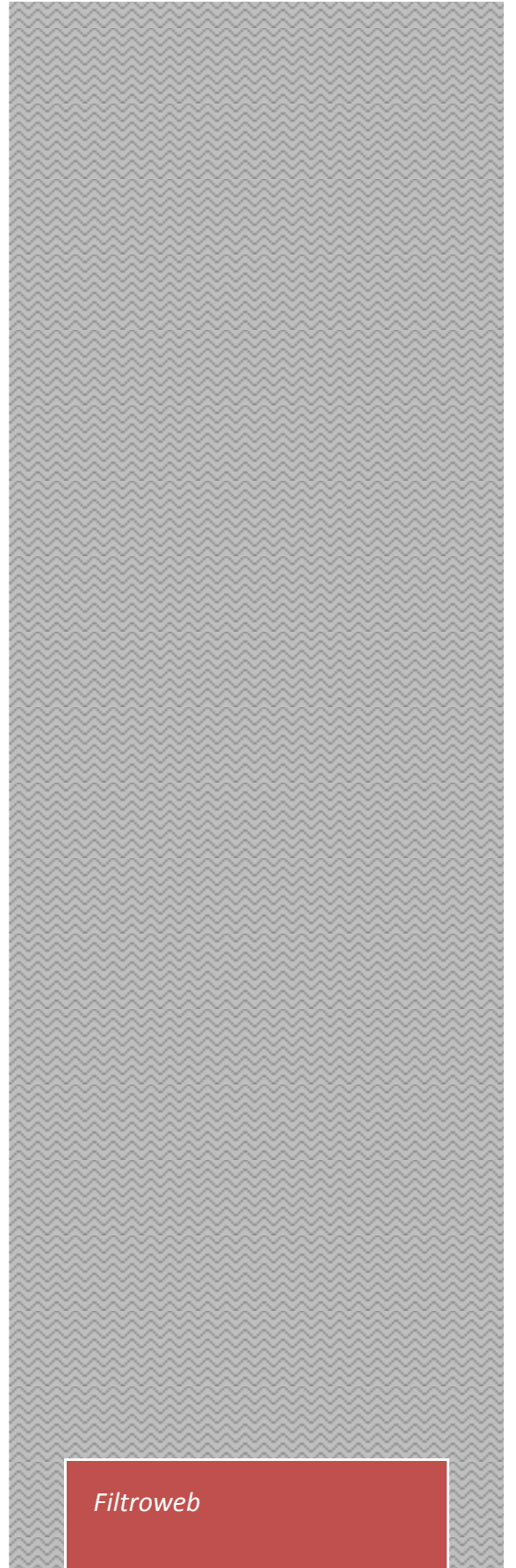
2008

TFC – Redes de computadores

Filtro de páginas web

Este trabajo final de carrera consiste en la creación de un complemento que añade nuevas características a un navegador conocido como Firefox producido y proporcionado por el Proyecto Mozilla. Dicho proyecto desarrolla, implementa y promueve el software libre. El complemento consiste en un filtro de páginas web a nivel de contenido.





Dedicatoria y agradecimientos

A mi adorada esposa Florencia por su gran amor, comprensión y apoyo incondicional.

Agradezco el apoyo inconmensurable de mis padres quienes siempre han sido personas admirables y me han brindado mucho. Con sus sabios consejos orientaron mis pasos por el camino recto de la vida, convirtiéndose por sus virtudes en mis mejores amigos. Por esto mi familia entera merece hoy, mañana y siempre todos mis honores, mi cariño y respeto.

A mi hermana Carolina quien con su apoyo moral me ayudó a no abandonar la batalla contra la adversidad en los días en que se me acumularon compromisos de estudio, trabajo y atención a la familia.

A María Isabel por todo el soporte brindado de consultoría y por compartir sus conocimientos y consejos logrando así una superación continua en la elaboración de este trabajo final de carrera.

A todos, mi fraternal agradecimiento.

Resumen

El trabajo final de carrera escogido consiste en crear un complemento que añade nuevas características a un navegador conocido como Firefox producido y proporcionado por el Proyecto Mozilla. Dicho proyecto desarrolla, implementa y promueve el software libre. El complemento consiste en un filtrado de páginas webs a nivel de contenido. No obstante, se podrán desarrollar más funcionalidades que se irán describiendo a lo largo de la memoria.

Se espera que este trabajo cubra las necesidades de aquel usuario que busca un potente filtro web para un navegador de Internet muy conocido.

En este proyecto se emplearán varios lenguajes de programación que servirán para desarrollar una aplicación útil con el fin de navegar controlando los contenidos que se quieren visualizar.

Índice

Contenido

Capítulo 1 – Introducción	8
1.1 Justificación del TFC y contexto en el cual se desarrolla: punto de partida y aportación del TFC.....	8
1.2. Objetivos del TFC.....	9
1.3. Enfoque y método seguido	10
1.4. Planificación del proyecto	11
1.4.1 Primera actividad: análisis.....	11
1.4.2 Segunda actividad: diseño.....	11
1.4.3 Tercera actividad: programación y pruebas.....	11
1.4.4 Cuarta actividad: documentación	12
1.4.5 Quinta actividad: entrega y presentación	12
1.4.6 Desviaciones temporales	12
1.5. Productos obtenidos	12
1.6. Breve descripción del resto de capítulos de la memoria	13
Capítulo 2 – Análisis	14
2.1 ¿Qué es Mozilla Firefox?	14
2.2 ¿Por qué Firefox?	14
2.3 Funcionalidades.....	14
2.3.1 ¿Qué puede realizar Filtroweb?.....	14
2.3.2 ¿Qué no puede realizar Filtroweb?.....	15
2.3.3 ¿Cómo funciona?.....	15
2.3.4 Filtrado	16
2.4 Extensiones	17
Capítulo 3 – Diseño y Programación	18
3.1 Diseño.....	18
3.2 Creación del manifest de la instalación.....	19
3.2.1 Propiedades.....	19
3.2.2 Versiones.....	22
3.2.3 Montaje del archivo install.rdf	22
3.3 Creación del Chrome manifest.....	26
3.4 Interfaz de usuario	27

3.4.1 XML.....	27
3.4.2 filtroweb.xml	28
3.4.3 settings.xml	28
3.4.4 XUL	28
3.4.5 filtroweb.xul	30
3.4.6 addfilterdialog.xul	31
3.4.7 filterall.xul.....	31
3.4.8 settings.xul	32
3.4.9 em-override.xul	32
3.4.10 contents.rdf.....	33
3.5 Hojas de estilo (CSS)	33
3.5.1 filtroweb.css	34
3.5.2 settings.css	34
3.5.3 filterall.css	35
3.6 Archivos por defecto	35
3.6.1 filtroweb.js	35
3.7 Creación del instalador.....	36
3.7.1 install.js.....	36
3.8 Componente XPCOM	37
3.8.1 nsFiltroweb.js	37
3.8.2 .autoreg	38
3.9 Implementación de funcionalidades mediante Java Script.....	38
3.9.1 addfilterdialog.js.....	38
3.9.2 settings.js.....	38
3.9.3 component.js.....	38
3.9.4 ajax-intercept.js.....	39
3.9.5 component-defaultfilters.js	40
3.9.6 filterall.js.....	40
3.9.7 global.js	40
3.9.8 filtroweb.js	40
3.9.9 uninstall.js	41
Capítulo 4 – Manual de Instalación.....	42
4.1 Instalación de Mozilla Firefox.....	42
4.2 Instalación de Filtroweb.....	49

4.2.1 Instalación desde fichero filtroweb.xpi.....	49
4.2.2 Desinstalación de Filtroweb.....	51
Capítulo 5 – Manual de Usuario.....	53
5.1 El icono Filtroweb.....	53
5.2 El menú contextual.....	53
5.3 La lista de elementos filtrables.....	54
5.4 El menú principal: Preferencias.....	55
Capítulo 6 – Conclusiones.....	57
Glosario.....	58
Bibliografía.....	59
Anexo 1 – Incidencias.....	60
Anexo 2 – Fichero con extensión .jar.....	60

Índice de ilustraciones

Ilustración 1 XPFE	9
Ilustración 2 Metodología	10
Ilustración 3 Planificación	12
Ilustración 4 Versión de Firefox.....	14
Ilustración 5 Arquitectura Mozilla.....	18
Ilustración 6 Diagrama de secuencias	18
Ilustración 7 Ubicación de iconoURL dentro de fichero jar	21
Ilustración 8 Directorio de trabajo	23
Ilustración 9 GUID de la aplicación Filtroweb	23
Ilustración 10 Propiedades del manifest de la instalación.....	24
Ilustración 11 Filtroweb como complemento de Mozilla Firefox	25
Ilustración 12 Acerca de Filtroweb.....	25
Ilustración 13 Página web	26
Ilustración 14 Fichero chrome.manifest	27
Ilustración 15 Esquema XUL.....	28
Ilustración 16 Barra de estado	29
Ilustración 17 Menú en barra de estado.....	30
Ilustración 18 Nuevo filtro.....	31
Ilustración 19 Diseño del menú principal.....	32
Ilustración 20 Fichero contents.rdf	33
Ilustración 21 filtroweb.js de defaults\preferencias.....	35
Ilustración 22 about:config	36
Ilustración 23 Componentes	37
Ilustración 24 Preferencias de Filtroweb	38
Ilustración 26 Introducción de expresiones regulares	40
Ilustración 25 Arquitectura AJAX	40
Ilustración 27 Menú Filtroweb en barra de estado.....	53
Ilustración 28 Elementos filtrables.....	54
Ilustración 29 Menú principal	55
Ilustración 30 Montaje fichero JAR	60

Cuerpo de la memoria

Capítulo 1 – Introducción

1.1 Justificación del TFC y contexto en el cual se desarrolla: punto de partida y aportación del TFC

Para empezar, existen muchas formas de enfocar la implementación de un filtro de páginas webs ya que son varios niveles los que intervienen cuando se navega por Internet. El concepto de Internet, paradigma de la comunicación por conmutación de paquetes, es una red de alcance planetario. De hecho, para hablar con más precisión, se ha de decir que realmente es un conjunto de redes interconectadas sobre las cuales cabalgan los protocolos de Internet TCP/IP.

La comunicación se establece siguiendo un modelo cliente – servidor. El cliente típicamente es el ordenador del usuario que pide algún tipo de servicio a otro ordenador llamado servidor. Será en este momento cuando intervendrá el filtro web. En un momento determinado, múltiples clientes pueden estar accediendo al mismo servidor para utilizar sus servicios. De hecho, la comunicación puede ser con cualquier tipo de ordenador, sin importar la marca, hardware o sistema operativo utilizado. Eso sí, la única condición es que los equipos utilicen el mismo protocolo de comunicaciones, en este caso, el estándar TCP/IP.

El nivel más bajo es el del hardware, el segundo nivel lo conforma el sistema operativo y, finalmente, el tercer nivel, el más alto, son los programas de aplicación. En dicho nivel de aplicación, se encuentra un navegador de páginas web (Mozilla Firefox). Además, el sistema operativo proporcionará, los módulos TCP/IP necesarios para navegar por Internet.

El punto de partida es la utilización de Mozilla Firefox, del cual más tarde, en el capítulo 2, se detallarán sus ventajas y mejoras, comparado con sus rivales. Uno de los fuertes que tiene es la posibilidad de incorporar complementos, y de esta base se parte para implementar un filtro web que permita útiles funcionalidades de filtrado de contenidos, contenidos como imágenes, objetos, applets, iFrames, presentaciones flash, etc. Incluso hasta mejorará la carga de las páginas web.

Afrontar este proyecto supone un gran reto a nivel de codificación ya que demanda varios conocimientos de diversos lenguajes de programación. Aplicar todo lo adquirido en la etapa académica y profesional permitirá obtener un producto atractivo y útil para la navegación por la red de redes, el cuál será reflejo todo el esfuerzo volcado en este trabajo.

Esta herramienta cuya implementación se desarrollará durante todo este proyecto, pretende aportar al usuario un control en lo que a navegación se refiere para poder discernir entre lo que se quiere visualizar y lo que se quiere filtrar para evitar ver contenidos no deseados. Cabe mencionar la existencia de otros productos comerciales como *Naomi*, *Filtrar.com*, *FoxyProxy*, etc. que realizan funciones similares. Inicialmente, el tipo de usuario objetivo de la aplicación es un usuario doméstico. No obstante, no se descarta una extensión del público objetivo en un futuro.

El proyecto “Filtro de páginas web” no solo aportará beneficios al usuario sino que también permitirá aprender nuevos conocimientos del mundo de la programación de aplicaciones.

1.2. Objetivos del TFC

El objetivo de este trabajo final de carrera es sumergirse en el mundo de la programación en el que intervienen diversos lenguajes combinados con un navegador de Internet que cada día toma protagonismo en la comunidad internauta: Mozilla Firefox.

Firefox incluye un sistema propio de extensiones, que pueden instalarse por sus usuarios para personalizar el aspecto y comportamiento del navegador. Estas extensiones, al igual que el propio navegador, se escriben en XUL¹, por lo que pueden modificar cualquier aspecto del interfaz y gran parte de la presentación de las páginas, así como agregar funcionalidades. Y en esto consiste el objetivo de este TFC, en crear un complemento que brinde la posibilidad de filtrar contenido web mientras se navega por la red de redes.

Para la programación de una aplicación basada en Mozilla se han de conocer técnicas como XUL, Java Script y CSS – técnicas que diseñadores de páginas web conocen perfectamente – y en ello se profundizará durante el transcurso de la elaboración de este trabajo. Así el desarrollo de aplicaciones es muy parecido al desarrollo de páginas web. XPFE (*Cross Platform Front End*) es una combinación de diferentes técnicas relacionadas con el mundo de la web. (Welcome to Software Development the Mozilla Way) Las ventajas de separar el contenido del diseño y del funcionamiento son múltiples. Son una base de la programación moderna que se conoce bajo el nombre *Model-View-Controller (MVC)*². Las siguientes técnicas – conocidas bajo el nombre XPFE – permiten la implementación del MVC para crear una aplicación basada en Mozilla:

- XUL para definir el interfaz, la estructura y el contenido la aplicación (similar a HTML)
- Java Script para programar la lógica y el funcionamiento de la aplicación
- CSS para definir el aspecto de la aplicación

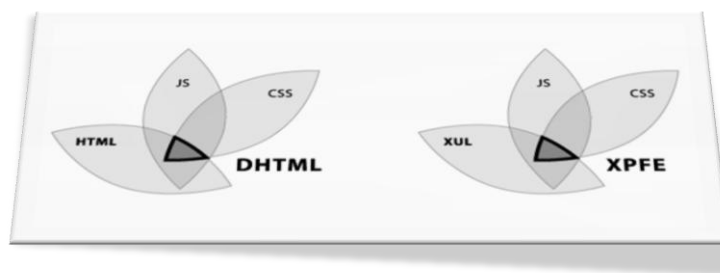


Ilustración 1 XPFE

Lograr una interacción de las técnicas mencionadas junto con los conocimientos adquiridos en asignaturas de diseño, estructura de la información y programación supondrá un gran desafío

¹ XUL acrónimo de **X**ML **U**ser-interface **L**anguage. Más detalles en el capítulo 3.

² Modelo, Vista y Controlador, un patrón arquitectónico MVC que es en definitiva el patrón habitual en la mayoría de los sistemas para el Web. (MetaEmotion, 2006)

y permitirá obtener un potente producto que filtre contenido web cumpliendo con las necesidades del usuario.

Y por último, una meta que se pretende alcanzar es crear un filtro web fácil de usar e intuitivo para el usuario que lo gestione.

1.3. Enfoque y método seguido

Todo proyecto de ingeniería tiene unos fines ligados a la obtención de un producto que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de las fases empleadas se le denomina “ciclo de vida”.

El ciclo de vida se compone de fases sucesivas compuestas por tareas planificables. Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de realimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones de los resultados intermedios que se van produciendo (realimentación).

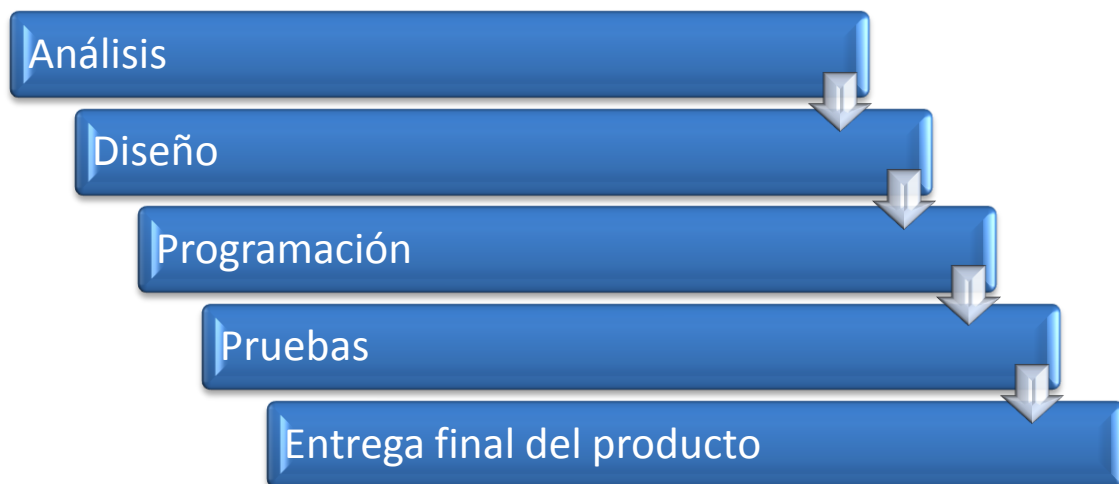


Ilustración 2 Metodología

En la primera etapa de **análisis** se recopilará toda la información necesaria para el desarrollo del complemento web. Se analizarán los requisitos indispensables y los conocimientos necesarios para desempeñar con éxito el trabajo. Debido a la gran capacidad de personalización que tiene el navegador web Firefox, el proyecto es totalmente **viable** siguiendo los métodos adecuados.

Seguidamente en la etapa de **diseño** se ha de identificar las soluciones tecnológicas para cada una de las funciones del sistema. Se ajustarán las especificaciones del producto a fin de que sea un filtro intuitivo y fácil de usar para el usuario final. Si en el análisis se especifica el problema o qué ha de realizar el complemento web, el diseño especifica una solución a este problema o cómo el complemento web ha de llevar a cabo su función.

La etapa de **codificación** se traducirá todo el diseño a código procesable por un ordenador. Se probará el funcionamiento del complemento desde diversos puntos de vista de una manera

ordenada y planificada para localizar y corregir dentro del programa y su documentación las incidencias que se puedan detectar.

Vale la pena destacar, que todo este proceso será de forma **cíclica** por lo que se repetirán las etapas de análisis, diseño y programación.

Cuando los resultados de la etapa de **pruebas** sean satisfactorios se entregará el **producto final**.

1.4. Planificación del proyecto

La definición de un ciclo de vida facilita el control sobre los tiempos en que es necesario aplicar recursos de todo tipo al proyecto. Todas las tareas del proceso de desarrollo de software deben ser planificadas, es decir, para cada una de ellas se debe establecer una fecha aproximada de inicio y otra de fin. Además, todas las tareas deben ser controladas a lo largo de todo el proceso de producción, esto es, se debe realizar un seguimiento continuo del proyecto informático.

El trabajo se define con una duración de 4 meses y a continuación veremos el plan de trabajo.

1.4.1 Primera actividad: análisis

La primera actividad implica la recopilación, clasificación y análisis de toda la información relevante para llevar a cabo el proyecto.

Temporización: 1 mes (24/09/2007 hasta el 24/10/2007)³

Objetivos: Profundizar sobre la metodología de programación utilizada para trabajar con complementos del navegador Firefox, como Java Script, XML, capas XUL, etc.

1.4.2 Segunda actividad: diseño

Diseño de la solución a implementar.

Temporización: 1 semana (25/10/2007 hasta el 01/11/2007)

Objetivos: Establecer de forma detallada funcionalidades del complemento web, para realizar filtro de páginas webs. Además obtener el análisis y el diseño de la aplicación a desarrollar. En una primera instancia, se había pensado que esta fase fuese un poco más extensa, pero nos requerirá más dedicación la actividad a continuación.

1.4.3 Tercera actividad: programación y pruebas

Implementación del complemento web y juegos de pruebas. Codificación y pruebas.

Temporización: 1 mes y 3 semanas (02/11/2007 hasta el 26/12/2007)

Objetivos: Implementar la aplicación en base al diseño creado y generar juegos de pruebas que garanticen el correcto funcionamiento del filtro web y de esta manera demostrar que se cumplen con los requisitos establecidos.

³ Cabe destacar que las primeras 3 semanas consistieron en la tramitación de cambio de TFC junto a tutor, secretaría y rectorado, lo que implicó un retraso en el comienzo del trabajo.

1.4.4 Cuarta actividad: documentación

Documentación del producto desarrollado.

Temporización: 1 semana (27/12/2007 hasta el 03/01/2008)

Objetivos: Documentar la instalación de la aplicación y el uso de las diferentes opciones de configuración del filtro web.

1.4.5 Quinta actividad: entrega y presentación

Memoria, producto y presentación. Es importante mencionar que la memoria se irá desarrollando a lo largo de todo el trabajo, por lo que en estas 2 semanas se revisará para la presentación al tribunal.

Temporización: 2 semanas (04/01/2007 hasta el 19/01/2007)

Objetivos: Síntesis de la memoria creada a lo largo de todo el proyecto y plasmarlo en una presentación.

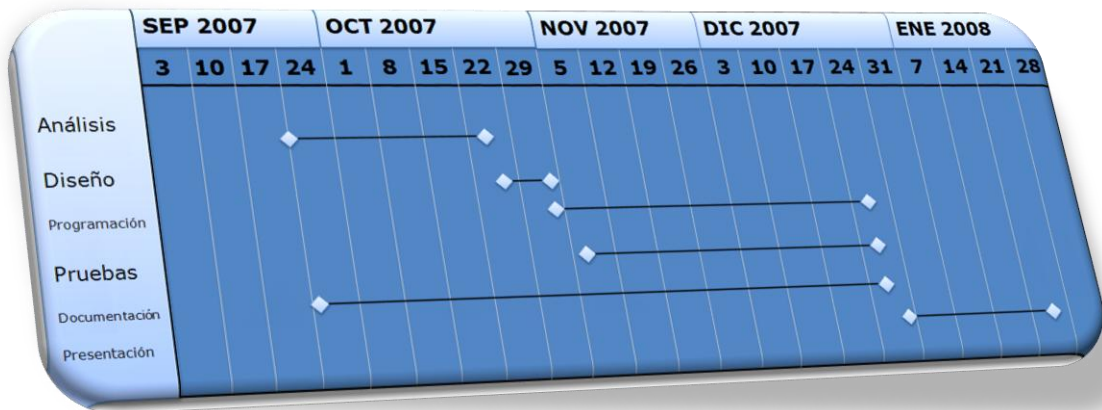


Ilustración 3 Planificación

1.4.6 Desviaciones temporales

La fase de codificación se ha visto afectada debido a que se ha tenido que mejorar aún más el código para optimizar al máximo la aplicación y por tanto dicha fase de programación se ha alargado una semana más. Esa semana adicional ha permitido otorgarle más calidad al trabajo, siempre orientándose al máximo beneficio para el usuario final (Ver [Anexo 1](#)).

1.5. Productos obtenidos

El filtro web dispone de una **página web** (<http://filtroweb-uoc.blogspot.com/>) la cual permite al usuario consultarla para ver información adicional sobre el filtro y descargar utilidades para que el filtro se adecue a sus necesidades específicas (listas de filtros⁴, por ejemplo).

Otro producto obtenido, el cual contiene una poderosa **guía de desarrollo y manual de usuario e instalación**, entre otros temas, es la propia **memoria**. Contiene varios capítulos

⁴ El concepto de lista de filtros se detallará en el apartado de funcionalidades. Se puede interpretar como una *Blacklist*.

estructurados según las fases del proyecto y pretende situar al lector en todo lo relacionado al trabajo de desarrollo del filtro web.

Obviamente se entregará el propio **filtro web** con sus **ficheros** de código que lo conforman, cuyos principales archivos se listan a continuación:

- chrome <directorio>
- chrome.manifest
- components <directorio>
- nsFiltroweb.js
- defaults <directorio>
- preferences <directorio>
- filtroweb.js
- install.js
- install.rdf
- addfilterdialog.js
- addfilterdialog.xul
- component.js
- contents.rdf
- em-override.xul
- filtroweb.css
- filtroweb.js
- filtroweb.xml
- filtroweb.xul
- filtroweb32x32.png
- global.js
- nsFiltroweb-extras.js
- settings.css
- settings.js
- settings.xml
- settings.xul
- uninstall.js

Se unificarán los archivos mencionados en un fichero llamado **Filtroweb.xpi**.

Adicionalmente se proporcionará una versión oficial del navegador **Mozilla Firefox**.

1.6. Breve descripción del resto de capítulos de la memoria

Los capítulos siguientes están estructurados según las fases del proyecto, definidos como capítulo que trata sobre el **análisis** de requisitos y funcionalidades y otro capítulo que aborda el **diseño** y la **programación** utilizada para llevar a cabo el filtro web.

Por otra parte, se incluirán capítulos sobre manuales de **instalación** y de **utilización** de la aplicación.

Se terminará con un capítulo de cierre con las **conclusiones** del trabajo y posibles **mejoras** aplicables al producto.

Capítulo 2 – Análisis

2.1 ¿Qué es Mozilla Firefox?

Mozilla Firefox es el navegador más seguro en la actualidad, gratuito y de código libre. Cuenta con gestor de descargas, navegación por pestañas, bloqueo de ventanas emergentes, búsquedas integradas para cualquier buscador e interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios externos. El programa es multiplataforma y está disponible en versiones para varios sistemas operativos. Su código fuente está disponible libremente bajo la triple licencia de Mozilla como un programa libre y de código abierto. Se partirá de la versión 2.0.0.7. La versión actual es la 2.0.0.9, que fue publicada recientemente y es la que se utilizará para todo el desarrollo del complemento. No obstante, no se descarta que se vaya actualizando de versión a medida que se avanza en este proyecto.



Ilustración 4 Versión de Firefox

2.2 ¿Por qué Firefox?

Porque es el navegador más personalizable, y por lo tanto el que más productividad ofrece. Cada uno se construye su propio Firefox añadiendo las funcionalidades que necesite. No existen dos usuarios de Firefox que tengan exactamente el mismo Firefox, porque cada uno tiene sus propias extensiones, y su tema favorito.

La característica de 100% modificable por los usuarios, mediante las llamadas extensiones (que se explicará en el próximo apartado), hacen que el navegador posea cualquier utilidad que se desee, desde controlar un reproductor multimedia a modificar la página web que se esté visualizando mediante scripts (*Greasemonkey*).

2.3 Funcionalidades

Las necesidades del usuario son filtrar contenido web no deseado para navegar por Internet visualizando lo que realmente se pretende visitar.

A continuación se expondrá un análisis de funcionalidades que se consideran vitales para este filtro web.

2.3.1 ¿Qué puede realizar Filtroweb?

Filtroweb bloquea elementos de páginas webs usando una configuración de **reglas de filtros** definidas por el usuario. Pueden ser filtrados desde objetos como imágenes hasta ficheros que concuerden con determinada extensión. El modo en que la aplicación filtra, consiste en **comparar las reglas de filtrado con la localización de un elemento**. Por ejemplo: el servidor web desde el cual una imagen es distribuida.

Los tipos de elementos que pueden ser filtrados son los siguientes:

- Imágenes (incluso fondos)
- Iframes
- Scripts vinculados externamente
- Objetos plugin (flash, java, vídeos, etc.)

Existe una regla simple aplicable a los elementos que pueden ser filtrados: *Todo elemento que aparezca en la lista de elementos bloqueables/filtrables puede ser filtrado.*

El usuario debe disponer de la posibilidad de añadir sitios, páginas web o elementos a una **lista blanca**, es decir, a una lista segura en la que se confía. Elementos que pertenezcan a la lista blanca no serán filtrados. Además, el usuario también puede consultar una lista de elementos a filtrar – bloquear, para que de esta manera pueda crear la **lista de filtros/reglas de filtrado** que se puede alimentar de varias maneras.

Una de las formas de crear una lista de filtros es descargando del vínculo <http://filtroweb-uoc.blogspot.com/2007/12/lista-de-filtros.html> una lista con reglas típicas de filtrado e importarla a la aplicación. Otro modo consiste en alimentar dicha lista de forma manual con el empleo de expresiones regulares que bloqueen por ejemplo todas las imágenes que contengan el texto *ad* o *publi* en su nombre.

Elementos que no pertenezcan a ninguna de las listas mencionadas por defecto no serán filtrados y si se desea, pueden ser incluidos en cualquiera de las listas en el momento que se desee. La modificación de elementos de las listas, ya sea exclusión o intercambio de elementos entre listas, también son permitidos.

Es muy importante que la aplicación sea ágil e intuitiva y para cumplir con dicha expectativa existe un **indicador en la barra de estado** con el fin de brindarle al usuario un acceso rápido a las opciones de configuración del filtro web (botón izquierdo del ratón) y a las preferencias (botón del medio del ratón). Abre también una lista de elementos bloqueables (scripts, etc).



El filtro web también da la posibilidad de clicar directamente sobre **elementos bloqueables** para que sean filtrados la próxima vez que se cargue la página web que está siendo visitada.



Objetos de una página web son etiquetados con una pestaña flotante para bloquear de manera rápida y directa.

2.3.2 ¿Qué no puede realizar Filtroweb?

Filtroweb no puede ser utilizado para filtrar imágenes en relación con sus dimensiones. No se puede realizar ninguna modificación de código fuente. Esto significa que texto (ej.: publicidad en formato texto) integrado en el código fuente HTML de las páginas no puede ser filtrado. Se debe tener en cuenta que Filtroweb solo filtra por localización de un elemento.

2.3.3 ¿Cómo funciona?

Cuando una página web es cargada, Filtroweb compila una **lista de elementos filtrables** de la página que se está visitando. Dicha lista es comparada con todas las entradas de la **lista de**

filtros. Tan pronto un elemento bloqueable coincide con una de las reglas de filtrado, el elemento es bloqueado.

Debido a que el algoritmo de filtrado compara cada elemento de una página web con cada entrada en las reglas de filtrado, el **tiempo de procesamiento** necesario puede aumentar y ser percibido cuando la lista es grande y la ejecución se realiza en una máquina con recursos limitados. Por lo tanto, se aconseja mantener la lista de filtrado tan compacta como sea posible, evitando reglas demasiado específicas.

Por ejemplo, si la dirección web del elemento que se pretende filtrar es `http://www.publicidad.com/banners/banner1.gif` se podrá modificar por `*publicidad.com*` para que de este modo bloquee todas las imágenes publicitarias de esta empresa de publicidad.

2.3.4 Filtrado

Cada línea de las reglas de filtrado puede tener varios **formatos** como:

- Reglas simples (posibilidad del uso del carácter *)
- Expresiones regulares

Reglas simples

La sintaxis se explicará a continuación. Asumiendo que una página embebe publicidad de `http://adserv.publi.bar/...` Una regla de filtro posible puede ser la línea `adserv`. Filtroweb percibirá que la entrada corresponde con `http://adserv.publi.bar/...` y filtrará el elemento correspondiente.

Sin embargo, el siguiente publicitario puede distribuir su contenido desde `http://advertising.publicidad.baz/...` siendo la regla anterior inútil. Por tanto, aquí el * puede ser utilizado, significando “cero o más caracteres arbitrarios”. Cambiando la regla a `ad*v`, filtrará `adserv.publi.bar/...` como `advertising.publicidad.baz/...`

Incluir * (como en `*advert*`) es innecesario y solo incrementa el tamaño del conjunto de reglas de filtrado.

Empleando filtros simples, se pueden construir reglas útiles de filtrado. Sin embargo, se ha de procurar evitar redundancia de reglas.

La lista de reglas conforma una lista negra (*blacklist*). No obstante, Filtroweb también ofrece la opción de invertir este comportamiento. Una regla puede ser invertida precediéndola con @@. Una entrada en la lista blanca precede a una entrada en la lista negra. En el caso de que existan reglas contradictorias, ganará la que indique mostrar el elemento.

Expresiones regulares

Mucho más poderosas, son las llamadas expresiones regulares. Utilizándolas, uno puede definir clases de caracteres (como dígitos o caracteres especiales), operaciones lógicas y otros caracteres con el objetivo de definir **reglas eficientes** y al mismo tiempo **compactas**.

Una línea de filtro se declara como expresión regular incluyéndole el carácter /. Dicho carácter / le indica a Filtroweb que trate la línea como una expresión regular.

En lugar de definir reglas simples como: *adserver*, *adcode*, *adsize*, *adlayer*, etc., una simple línea conteniendo */ad(server|code|size|layer)/* tiene exactamente el mismo efecto.

Filtrado rápido

La funcionalidad de filtrado rápido es activada con Ctrl+Shift+K. Si el filtrado rápido se encuentra activo, elementos individuales en una página web pueden ser removidos temporalmente usando Shift + doble clic. Esto es muy útil para imprimir, ya que los elementos también desaparecen de la página de impresión de la web. Elementos filtrados siguiendo este método, reaparecen en la carga siguiente y las reglas de filtrado permanecen intactas.

2.4 Extensiones

Las **extensiones** (*add-ons*), también llamados complementos o agregados, añaden nuevas funcionalidades a las aplicaciones Mozilla, tales como Firefox. Las extensiones permiten añadir a dichas aplicaciones desde un botón para una barra de herramientas hasta características/funcionalidades totalmente nuevas. Permiten personalizar completamente la aplicación para ajustarla a las necesidades de cada usuario. En nuestro caso, nos ayudará a implementar el filtrado de páginas webs. (Nukeador, Psanz, Arteadonis, Gbulfon, & otros, 2007)

Las extensiones se distribuyen en archivos comprimidos en formato ZIP, o en paquetes, con extensión **xpi**. Los archivos XPI contienen el siguiente código:

```
extension.xpi:
    /install.rdf
    /components/*
    /components/cmdline.js
    /defaults/
    /defaults/preferences/*.js
    /plugins/*
    /chrome.manifest
    /chrome/icons/default/*
    /chrome/
    /chrome/content/
```

Un nuevo fichero XPI requiere la siguiente infraestructura:

```
PORTNAME=          filtroweb
PORTVERSION=       1.0
CATEGORIES=        www
MAINTAINER=        Nicolás Rodríguez Sánchez
COMMENT=           Filtra contenido web
XPI_ID=            {cf1682d8-4a58-4fb7-8db6-20d8b897c2a1}
XPI_FILES=         chrome/filtroweb.jar chrome.manifest
                   components/.autoreg components/nsFiltroweb.js
                   defaults/preferences/filtroweb.js install.js
                   install.rdf
XPI_DIRS=          chrome/filtroweb chrome components
                   defaults/preferences defaults
```

Capítulo 3 – Diseño y Programación

3.1 Diseño

A continuación se muestra la arquitectura de Mozilla y sus conceptos:

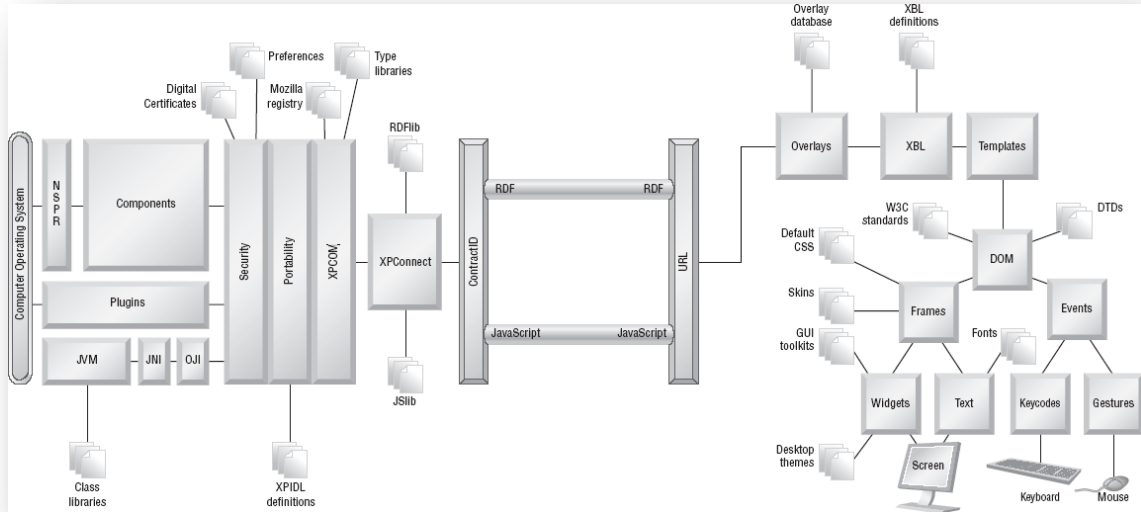


Ilustración 5 Arquitectura Mozilla

El **diagrama de secuencias** diseñado que dará un pantallazo general del diseño de la aplicación *Filterweb* se muestra en el siguiente gráfico:

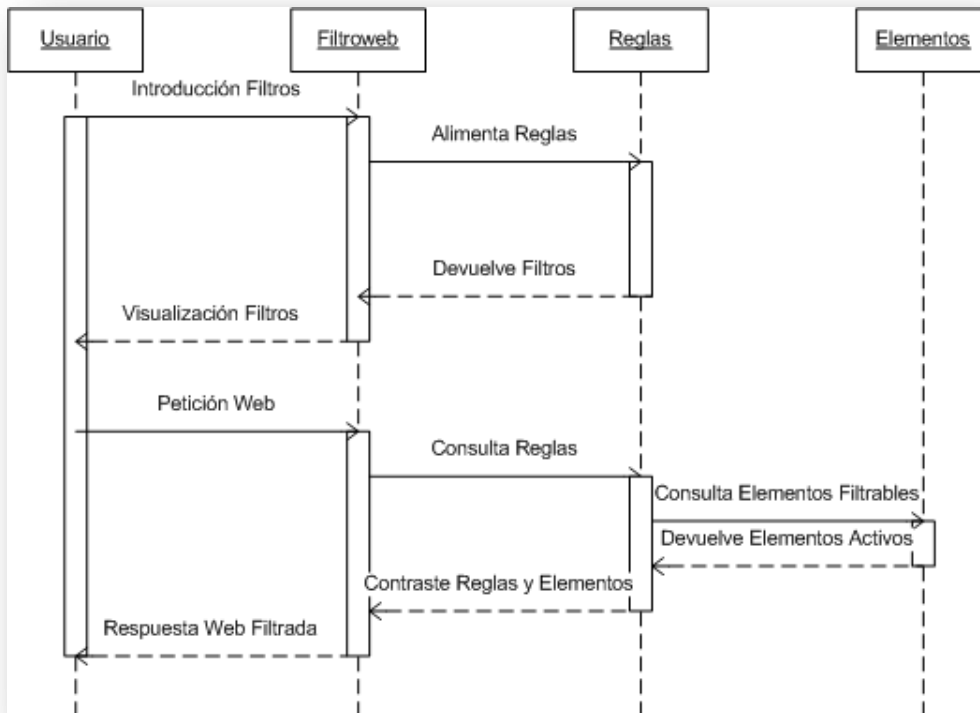


Ilustración 6 Diagrama de secuencias

En el diagrama se representa explícitamente el orden en el tiempo y la duración de los mensajes y de las diferentes operaciones (funcionalidades). Está estructurado en **dos dimensiones**: el tiempo (verticalmente) y los distintos papeles de clasificadores que participan en la interacción (horizontalmente). La línea de vida simboliza la existencia del papel en un cierto período de tiempo, representada por la línea punteada en vertical. Una activación (representada por rectángulos verticales) es una parte de la línea de vida durante la cual el papel en cuestión ejecuta una acción, o bien otros papeles ejecutan otras acciones, a consecuencia de una acción ejecutada por el papel. Los mensajes se indican con flechas que comienzan en una activación y acaban en otra. (Rumbaugh, Jacobson, & Booch, 2000)

3.2 Creación del manifest de la instalación

Entrando ya en la codificación, a continuación se describirán conceptos importantes que se han de tener en cuenta para el desarrollo de la aplicación.

Un instalador "**Manifest**" es el archivo Administrador-Activador de la extensión (complemento) que una aplicación XUL utiliza para determinar la información acerca de un complemento como es inicialmente instalado. Este contiene **metadatos** que identifican el complemento, proporcionando información acerca de quién lo ha creado, dónde se puede encontrar más información acerca del complemento, con cuáles versiones de las aplicaciones es compatible, cómo debe ser actualizado y así sucesivamente. El formato de un instalador "Manifest" es RDF/XML. El archivo debe ser nombrado `install.rdf` y ubicarse en el nivel superior de un complemento de un archivo XPI. (Nukeador, Psanz, Arteadonis, Gbulfon, & otros, 2007)

<i>Formato:</i>	<i>RDF/XML</i>
<i>Nombre:</i>	<i>install.rdf</i>
<i>Ubicación:</i>	<i>nivel superior del fichero xpi</i>

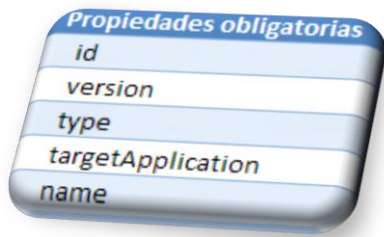
La **estructura** básica de un instalador "Manifest" es la siguiente:

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:em="http://www.mozilla.org/2004/em-rdf#"
      <Description about="urn:mozilla:install-manifest">
        <!-- propiedades -->
      </Description>
</RDF>
```

3.2.1 Propiedades

Existen **propiedades** obligatorias y otras opcionales. Algunas contienen simples cadenas, otras son recursos complejos.

Id: es un GUID (*Firefox 1.0*) o cadena de texto (*Firefox 1.5* en adelante) como: `nombreextension@organizacion.tld`



Version: versión del complemento (ver http://developer.mozilla.org/en/docs/Extension_Versioning%2C_Update_and_Compatibility)

Type: valor de tipo entero. (2-Extensiones, 4-Temas, 8-Local, 16-Plugin, 32-Multiple Item Package)

targetApplication: Un objeto que especifica la aplicación a la cual se dirige este complemento. Contiene cadenas de versión son formateadas del mismo modo que la propiedad versión, antes mencionada. Las extensiones compatibles con *Firefox 2* deben especificar una *maxVersion* de 2.0.0.*

Name: El nombre del complemento previsto para mostrarlo en el UI (interfaz del usuario).

Description: Una breve descripción (pequeña línea) del complemento - prevista para mostrar en el UI.



Creator: Nombre del creador/desarrollador principal - prevista para mostrar en la interfaz del usuario.

Developer: El nombre de los co-desarrolladores.

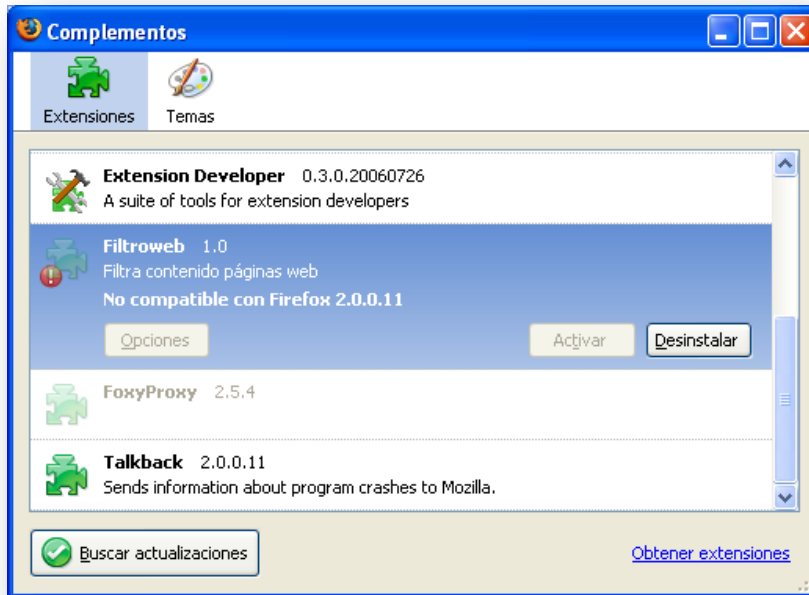
Translator: Lo(s) nombre(s) de los traductores.

Contributor: Lo(s) nombre(s) de contribuidores adicionales.

HomepageURL: Enlace a la pagina del complemento

UpdateURL: Enlace a una actualización personalizada del archivo "Manifest" que especifica actualizaciones viables del complemento.

OptionsURL: La chrome:// URL del cuadro de diálogo de las opciones de la extensión. Esto es solo utilizable para extensiones. Si esta propiedad es especificada, cuando la extensión es seleccionada en la lista de extensiones, los botones de opción son habilitados



AboutURL: La chrome:// URL del cuadro de diálogo acerca de la extensión. Esto es solamente utilizable por las extensiones. Si esta propiedad esta especificada, cuando la extensión es seleccionada en la lista de extensiones, el enlace acerca de... el menú contextual mostrará este cuadro de dialogo

IconURL: Una chrome:// URL de icono 32x32 para mostrar en la lista de complementos. Por ejemplo: `<em:iconURL>chrome://extensiondev/skin/extensiondev.png</em:iconURL>`

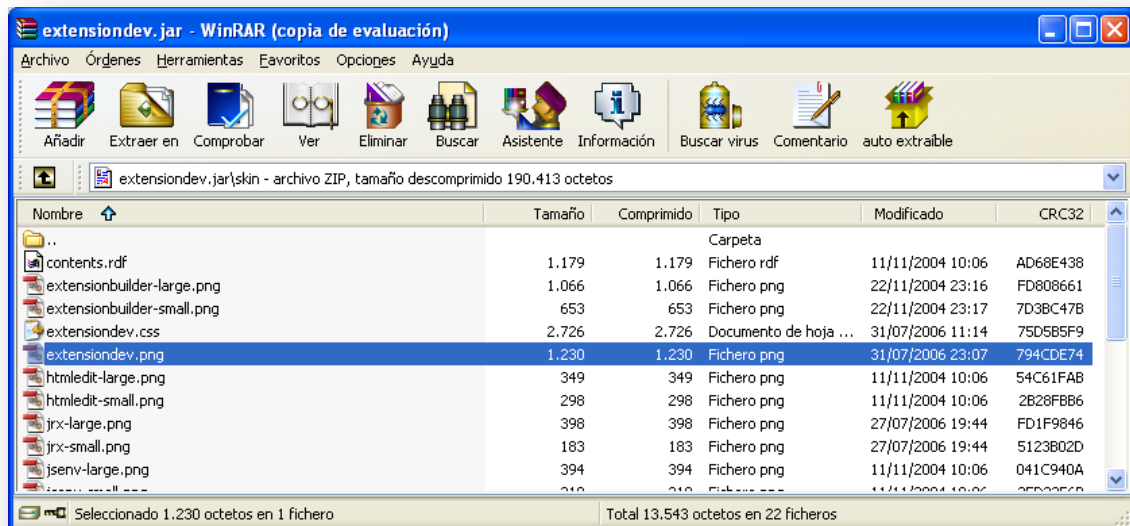


Ilustración 7 Ubicación de iconoURL dentro de fichero jar

Hidden: Esta propiedad tiene un valor booleano que cuando es true hace que el complemento no se muestre en la lista de complementos

targetPlatform: Es una cadena que especifica la plataforma que el complemento soporta. (Wesdorp, 2007)⁵

3.2.2 Versiones

A continuación se enumerarán las **versiones** válidas que se pueden especificar en el archivo install.rdf.



Firefox

(<https://addons.mozilla.org/es-ES/firefox/pages/appversions>)

- GUID: {ec8030f7-c20a-464f-9b0e-13a3a9e97384}
- Versiones: 0.3, 0.6, 0.7, 0.7+, 0.8, 0.8+, 0.9.x, 0.9, 0.9.0+, 0.9.1+, 0.9.2+, 0.9.3, 0.9.3+, 0.9+, 0.10, 0.10.1, 0.10+, 1.0, 1.0.1, 1.0.2, 1.0.3, 1.0.4, 1.0.5, 1.0.6, 1.0.7, 1.0.8, 1.0+, 1.4, 1.4.0, 1.4.1, 1.5b1, 1.5b2, 1.5, 1.5.0.4, 1.5.0.*, 2.0a1, 2.0a2, 2.0a3, 2.0b1, 2.0b2, 2.0, **2.0.0.***, 3.0a1, 3.0a2, 3.0a3, 3.0a4, 3.0a5, 3.0a6, 3.0a7, 3.0a8pre, 3.0a8, 3.0a9

3.2.3 Montaje del archivo install.rdf

Existe un complemento llamado **Extension Builder**, el cual ayudará a crear el fichero install.rdf y a estructurar el complemento a desarrollar. Para la utilización de dicha herramienta se trabaja con un directorio específico, llamado directorio de trabajo (*Working Directory*), donde se alojarán los diferentes ficheros que conforman la aplicación.

⁵ El Centro de Desarrollo Mozilla (Devmo) es un proyecto el cual se ha diseñado para ser un recurso usable, exacto, y valioso para todo desarrollador web o de aplicaciones.

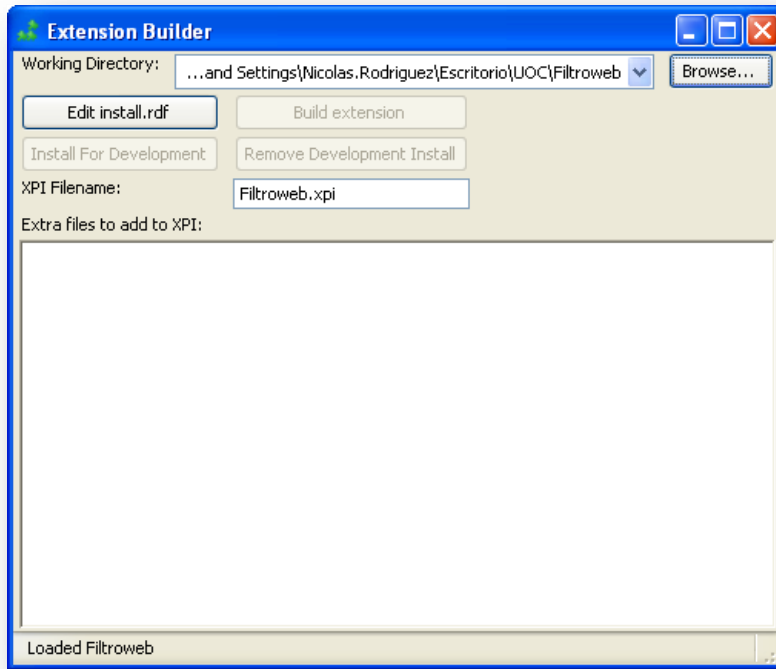


Ilustración 8 Directorio de trabajo

Extension Builder permite al fichero **install.rdf** gestionarlo de una manera visualmente más esquemática. Al crearlo se asigna un GUID, un identificador:

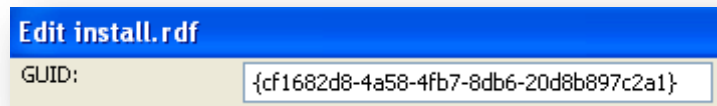


Ilustración 9 GUID de la aplicación Filtroweb

Las propiedades descritas anteriormente se les asignan sus valores en este paso:

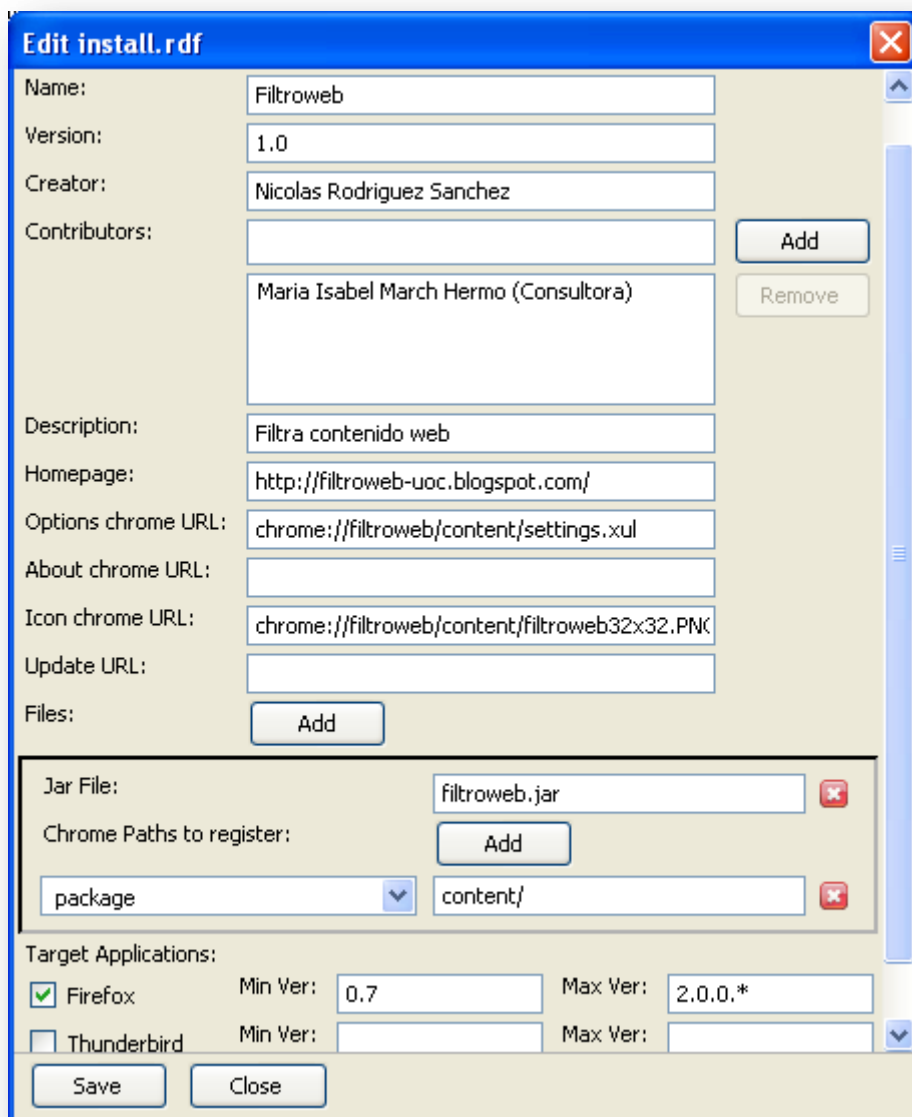


Ilustración 10 Propiedades del manifest de la instalación

Los valores informados de dichas propiedades se comprueban consultando el menú herramientas del navegador Firefox, apartado complementos:

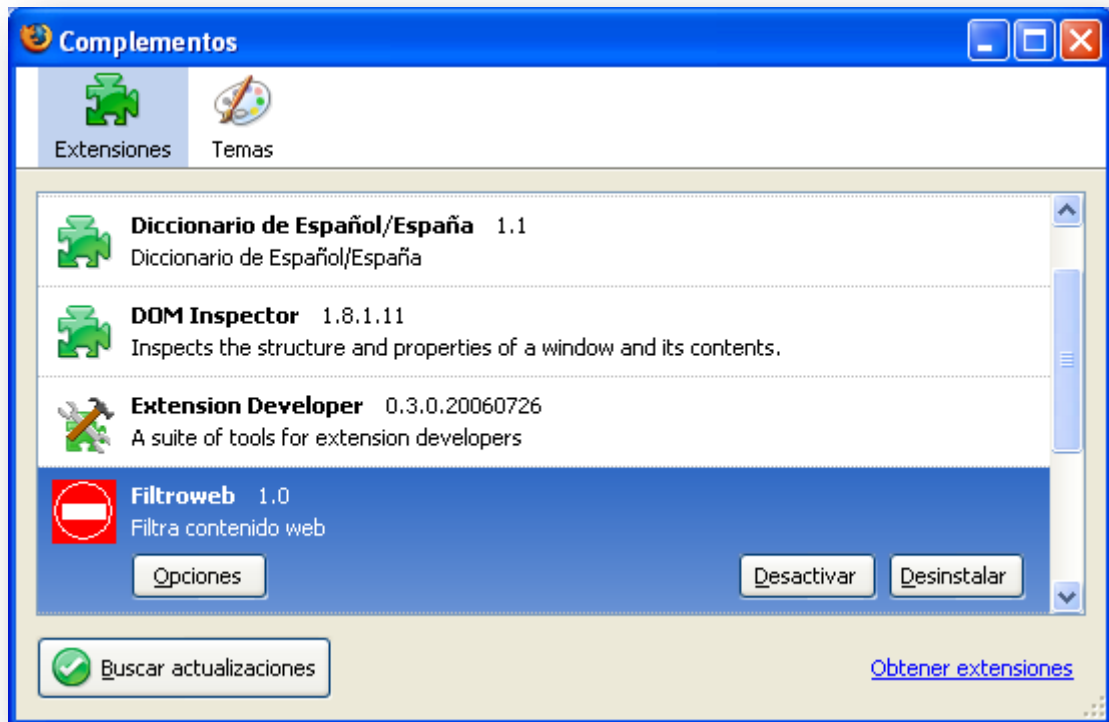


Ilustración 11 Filtroweb como complemento de Mozilla Firefox

Si se quiere conocer más información general acerca del filtro web, aquí se muestra:

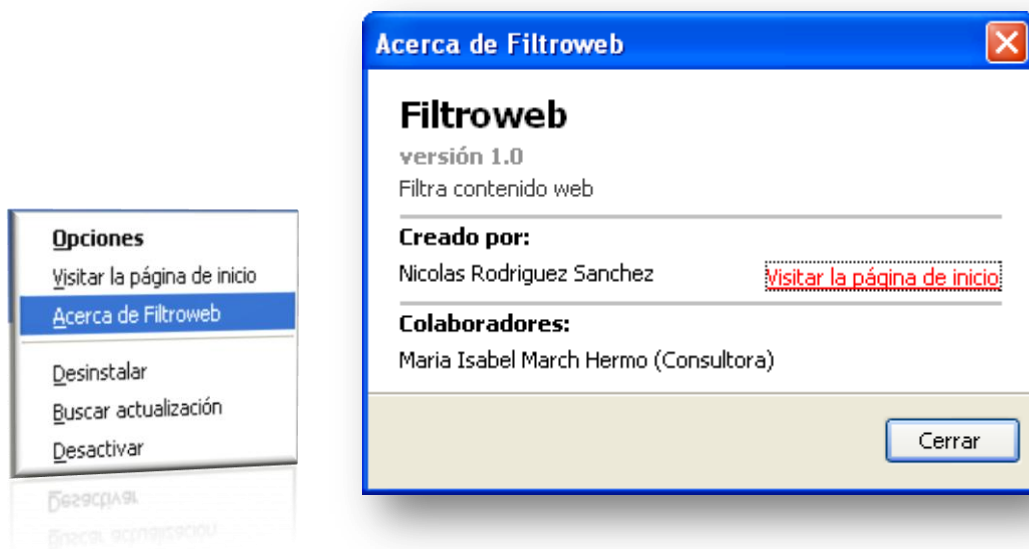


Ilustración 12 Acerca de Filtroweb

Adicionalmente, si se desea visitar la página web de inicio del programa de forma rápida se puede acceder clicando en “Visitar la página de inicio” o a través de <http://filtroweb-uoc.blogspot.com>.

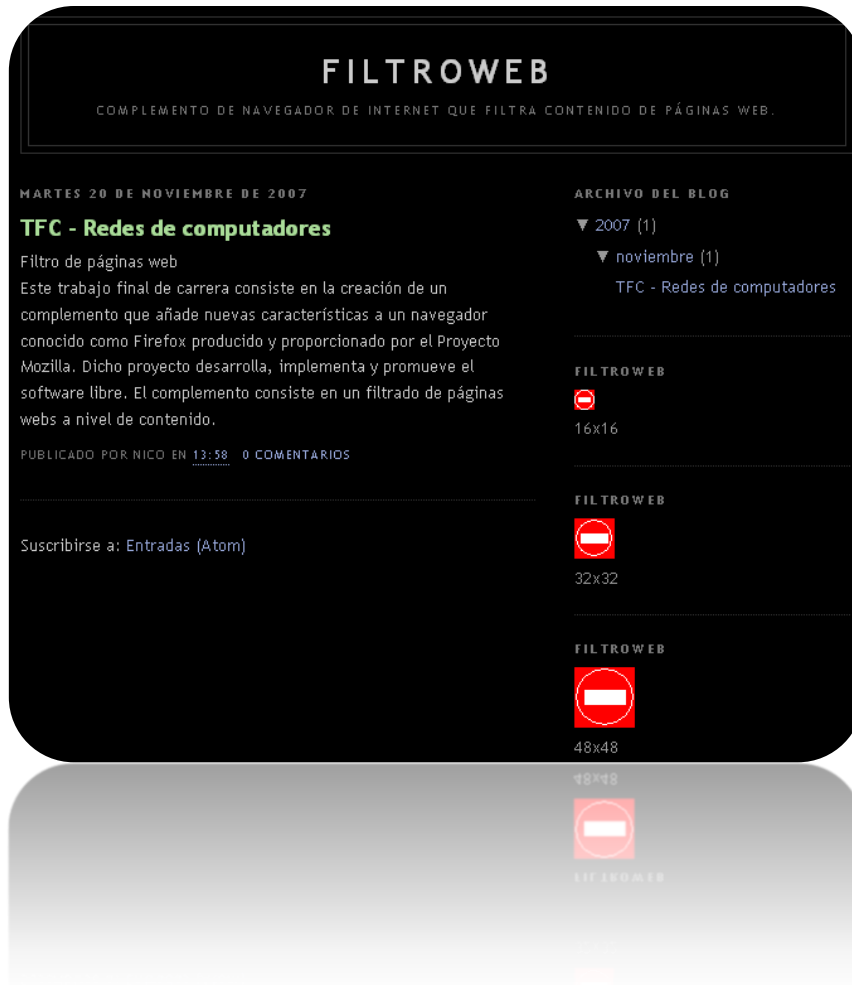


Ilustración 13 Página web

3.3 Creación del Chrome manifest

El fichero **chrome.manifest** especifica lo siguiente:



1. Tipo de material dentro de un paquete chrome
2. Nombre del paquete chrome
3. Localización de los archivos del paquete chrome

La línea *“content filtroweb jar:chrome/filtroweb.jar!/content”* dice que para obtener una muestra del paquete chrome, debemos encontrar los archivos de content en la ruta chrome/content, la cual es relativa a la ruta de chrome.manifest.

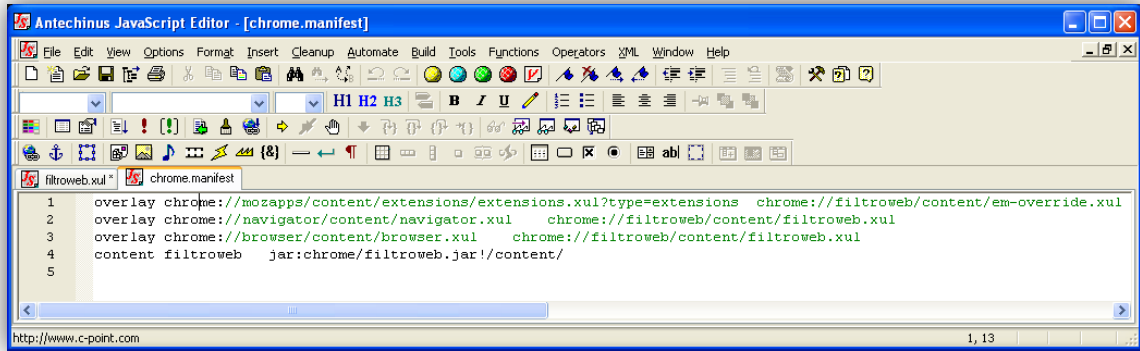


Ilustración 14 Fichero chrome.manifest

Para registrar un Overlay, es necesario que Firefox fusione el overlay con la ventana del explorador (browser) cada vez que se muestra, por lo que se ha de añadir la línea *“overlay chrome://browser/content/browser.xul chrome://tema/sección/archivo.xul”* al archivo chrome.manifest.

La línea *“overlay chrome://browser/content/browser.xul chrome://filtroweb/content/filtroweb.xul”* le indica a Firefox que fusione filtroweb.xul con browser.xul cuando browser.xul se cargue.

3.4 Interfaz de usuario

3.4.1 XML

El lenguaje de Internet para la creación de documentos electrónicos más conocido es el **HTML**, que es con el que se define la estructura y contenido de una página web. Consta de una serie de etiquetas (markups) predefinidas que permiten construir documentos que contienen títulos, párrafos y listas de texto, tablas, imágenes y otro elementos para la presentación de información. Como mejora del lenguaje HTML original surge el lenguaje **XML** que además de pasar a ser "ascendiente" del HTML.

XML, **eXtensible Markup Language** es un lenguaje más flexible y adaptable que el HTML y está diseñado para mejorar la funcionalidad de la Web. XML es un lenguaje de lenguajes (un **metalenguaje**), porque es utilizado para describir otros lenguajes que permiten construir documentos electrónicos. (Pitts, 1999)

XML describe la estructura del documento, pero deja a libre elección el nombre que recibirá cada una de las etiquetas. El número de sub-lenguajes que pueden crearse a partir del metalenguaje XML es prácticamente ilimitado.

Uno de esos sub-lenguajes del XML es por ejemplo: **XHTML**, diseñado para sustituir el limitado lenguaje HTML. Por tanto XHTML es la conversión del lenguaje HTML al mismo lenguaje pero basado en XML, de modo que XHTML es básicamente el HTML conocido hasta ahora pero adaptado a la estructuración de un documento XML.

Los ficheros XML del proyecto son:

 filtroweb.xml	4 KB	Documento XML
 settings.xml	26 KB	Documento XML

3.4.2 filtroweb.xml

La idea es cumplir con uno de los objetivos de simplicidad y agilidad de acceso a determinadas acciones. Acciones como emerger un menú contextual con botón derecho que permita deshabilitar/habilitar *Filtroweb* sin abrir preferencias de la aplicación. Botón izquierdo abrirá *Elementos filtrables* (filterall.xul) y botón central ejecutará las *Preferencias* (settings.xul).

Fichero accesible a través de: *chrome://filtroweb/content/filtroweb.xml*.

3.4.3 settings.xml

Como hoja de estilo asociada se tendrá a *chrome://global/skin/listbox.css*. Se ha **reutilizado** código de *chrome://global/content/bindings/general.xml*.

Implementa a:

- *nsIDOMXULMultiSelectControlElement*
- *nsIDOMXULSelectControlElement*
- *nsIAccessibleProvider*

Además de definir diferentes **métodos** (como anexar elementos, insertar, eliminar, obtener, agregar, seleccionar, etc.) y **propiedades** (como elemento seleccionado, valor, elemento actual, etiqueta, etc.). Ruta de acceso al fichero: *chrome://filtroweb/content/settings.xml*.

3.4.4 XUL

La interfaz de usuario de Firefox está escrita en XUL y Java Script. **XUL (XML-based User-interface Language)** es un lenguaje basado en XML para el desarrollo de interfaces de usuario multiplataforma. XUL es una implementación XML que proporciona elementos de la interfaz de usuario como botones, menús, barras de botones etc. Por otra parte, las acciones del usuario se controlan mediante **Java Script**. Para ampliar el navegador se modifican o se añaden elementos de la interfaz de usuario. Se añaden nuevos elementos mediante la inserción de nuevos elementos XUL DOM dentro de la ventana del navegador, y se modifican usando scripts e incluyendo manejadores de eventos.

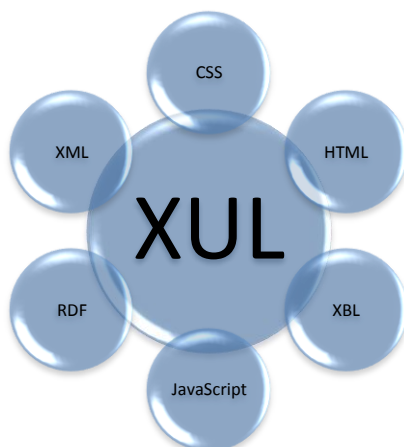


Ilustración 15 Esquema XUL

En resumen, XUL es un lenguaje de especificación de interfaces gráficas y comportamiento basado en XML, es decir, es una especificación de XML que permite diseñar aplicaciones web pero con la filosofía de diseño de ventanas tradicional. Además soporta **CSS**⁶ para cada uno de sus elementos y la interacción entre los mismos se realiza mediante Java Script.

El navegador está definido en un archivo XUL llamado **browser.xul**

⁶ CSS (Cascading Style Sheets) se verá en el siguiente apartado.

(*chrome://browser/content/browser.xul*). En el archivo *browser.xul* podemos encontrar la barra de estado, donde `<statusbar id="status-bar">` es un "punto de anclaje" para una capa XUL.

Las **capas XUL** son una forma de añadir un elemento a la interfaz de usuario de un documento XUL durante el tiempo de ejecución. Una capa XUL es un archivo con extensión *.xul* que marca elementos XUL para insertar en puntos de anclaje específicos dentro del "documento maestro". Estos fragmentos pueden indicar que los elementos sean añadidos, modificados o eliminados.

El **punto de anclaje** dentro de la ventana del navegador dónde se incluirá el complemento de filtro web es la **barra de estado**.

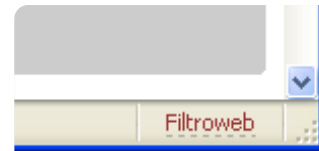


Ilustración 16 Barra de estado

Además de añadir elementos de la interfaz de usuario en el punto de anclaje, se pueden emplear fragmentos XUL dentro de las capas para:

- **Cambiar atributos** en el punto de anclaje, ejemplo: `<statusbar id="status-bar" hidden="true"/>` (esconde la barra de estado)
- **Eliminar** el punto de anclaje del documento maestro, ejemplo: `<statusbar id="status-bar" removeelement="true"/>`
- **Controlar la posición** de los elementos insertados:
 - `<statusbarpanel position="1" .../>`
 - `<statusbarpanel insertbefore="other-id" .../>`
 - `<statusbarpanel insertafter="other-id" .../>`

Es posible crear nuevos **componentes de interfaz de usuario** como: ventanas propias y cajas de diálogo separadas de los archivos *.xul*, proporcionar funcionalidad colocando acciones del usuario en ejecución en archivos *.js*, usar métodos DOM para manipular *UI widgets*, usar las reglas de estilos *.css*, adjuntar imágenes, cambiar colores, etc.

Los ficheros XUL forman parte de "Chrome Packages", paquetes de componentes de interfaz del usuario, los cuales se cargan a través de la dirección *chrome:// URIs*⁷. Debido a que la ubicación de Firefox en un sistema puede variar de una plataforma a otra y de un sistema a otro, en vez de cargar el navegador desde el disco utilizando un (archivo) *file:// URI*. Los desarrolladores de Mozilla se decantaron por una solución para crear URI al contexto de XUL que la aplicación instalada ya conoce.

Los componentes de los Chrome URIs son los siguientes:





1. La *URI scheme* Esquema URI (**chrome**) informa a la librería de red de Firefox de que es un 'Chrome URI' y que el contenido que se cargue debe ser manejado de manera especial.
2. Un **nombre de paquete** (ejemplo *browser*), que identifica la extensión en la interfaz del usuario. Este nombre debe ser, único, para evitar conflicto entre extensiones.

⁷ Uniform Resource Identifier, identificador uniforme de recurso, definido en RFC 2396 (Uniform Resource Identifiers: Generic Syntax). Algunos URI pueden ser URL, URN o ambos. (Inc, 2007)

3. El **tipo de información** que ofrece el archivo. Hay tres tipos:
 - **content** (XUL, JavaScript, XBLs, etc. que forman la estructura y el comportamiento de una aplicación UI)
 - **locale** (DTD, archivos .properties, etc. que contienen cadenas para la localización del UI)
 - **skin** (CSS e imágenes que forman el theme del UI)
4. La **ruta del archivo** a cargar.

Por lo tanto, queda de esta manera **chrome://tema/sección/archivo**. Cuando se carga contenido usando un Chrome URI, Firefox usa el Chrome Registry para traducir esos URIs en los archivos fuentes en el disco (o en paquetes JAR).

Los **ficheros XUL** del proyecto son los siguientes:

	addfilterdialog.xul	1 KB	Archivo XUL
	em-override.xul	2 KB	Archivo XUL
	filterall.xul	4 KB	Archivo XUL
	filtroweb.xul	10 KB	Archivo XUL

3.4.5 filtroweb.xul

Implementa menús a nivel de barra de estado ("status-bar"), barra de herramientas ("menu_ToolsPopup") y menú contextual ("contentAreaContextMenu").

El menú a nivel de **barra de estado** consiste en:

- "Agregar pagina actual a lista blanca"
 - "filtrowebPrefObserver.addWhite([content.location+',0].join(' '),true)"
- "Agregar este sitio a lista blanca"
 - "filtrowebPrefObserver.addWhite([content.location+',1].join(' '),true)"
- "Ver elementos filtrables"
 - "filterAllDialog(true);"
- "Bloquear Flash"
 - "toggleObjectOverride();"
- "Habilitar Filtroweb" / "Deshabilitar Filtroweb"
 - "Components.classes['@mozilla.org/preferences-service;1'].getService(Components.interfaces.nsIPrefService).getBranch('filtroweb.').setBoolPref('enabled', true); Components.classes['@mozilla.org/observer-service;1'].getService(Components.interfaces.nsIObserverService).notifyObservers(null, 'Filtroweb-PrefChange', 'filtroweb');"
 - "Components.classes['@mozilla.org/preferences-service;1'].getService(Components.interfaces.nsIPrefService).getBranch('filtroweb.').setBoolPref('enabled', false); Components.classes['@mozilla.org/observer-service;1'].getService(Components.interfaces.nsIObserverService).notifyObservers(null, 'Filtroweb-PrefChange', 'filtroweb');"
- "Preferencias"
 - "filtrowebSettings();"

Ilustración 17 Menú en barra de estado



A nivel de **barra de herramientas**:

- "Agregar esta pagina a lista blanca"

- "filtrowebPrefObserver.addWhite([content.location+",0].join(' '),true)"
- "Agregar este sitio a lista blanca"
 - "filtrowebPrefObserver.addWhite([content.location+",1].join(' '),true)"
- "Abrir elementos bloqueables"
 - "filterAllDialog(true);"
- "Bloquear Flash"
 - "toggleObjectOverride();"
- "Preferencias"
 - "filtrowebSettings();"

A nivel de **menú contextual**:

- | | |
|---|--|
| <ul style="list-style-type: none"> ● "Filtroweb: Bloquear imagen..." <ul style="list-style-type: none"> ○ "imgFilterDialog();" ● "Filtroweb: Bloquear embedido..." <ul style="list-style-type: none"> ○ "itemFilterDialog();" ● "Filtroweb: Bloquear objeto..." <ul style="list-style-type: none"> ○ "itemFilterDialog();" ● "Filtroweb: Bloquear applet..." <ul style="list-style-type: none"> ○ "itemFilterDialog();" | <ul style="list-style-type: none"> ● "Filtroweb: Bloquear iFrame..." <ul style="list-style-type: none"> ○ "itemFilterDialog();" ● "Filtroweb: Eliminar map" <ul style="list-style-type: none"> ○ "disableMapArea();" ● "Filtroweb: Bloquear fondo..." <ul style="list-style-type: none"> ○ "imgFilterDialog(true);" ● "Filtroweb: Importar Filtros" <ul style="list-style-type: none"> ○ "filtrowebImportSelected();" |
|---|--|

Su código es accesible a través de: `chrome://filtroweb/content/filtroweb.xul`. Utiliza la hoja de estilo: `filtroweb.css` e invoca a los scripts: `filtroweb.js` y `global.js`.

3.4.6 addfilterdialog.xul

Esta rutina hace posible la opción de agregar un nuevo filtro. En la caja de texto que se enseña se puede utilizar el carácter *. Dicho carácter es llamado “wildcard”, lo que significa que puede ser usado para sustituir cualquier otro carácter en una cadena.

Utiliza la hoja de estilo `chrome://global/skin/global.css` e invoca a los scripts `addfilterdialog.js`, `filterall.js` y `global.js`.

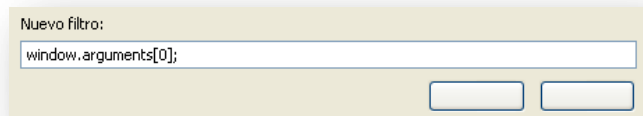
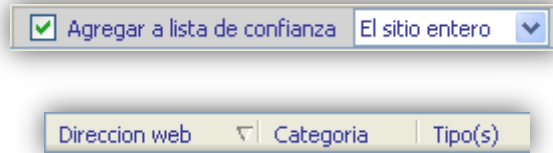


Ilustración 18 Nuevo filtro

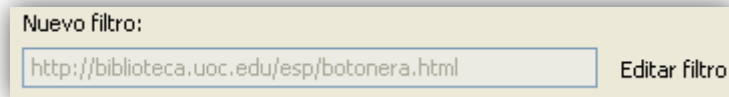
3.4.7 filterall.xul

Especifica la configuración visual de la ventana titulada: **Elementos filtrables**. Además, se establecen los nombres de los campos y menús que se muestran en esta interfaz, la cual permite añadir nuevos filtros y pudiendo incluso ser editados.

Menús como “Agregar a lista de confianza” (página o sitio) y campos de la lista de elementos como se muestran a continuación.



Si ya se tiene un elemento filtrado de la lista que se muestra, al seleccionarlo permite editarlo y *filterall.xul* es quien lo controla:



3.4.8 settings.xul

Este código invoca a scripts como *settings.js*, *fitroweb.js* y *global.js*, y es accesible mediante la ruta *chrome://fitroweb/content/settings.xul*. Implementa todos los menús de las preferencias de Fitroweb y abarca todas las opciones del **menú principal** de la aplicación.

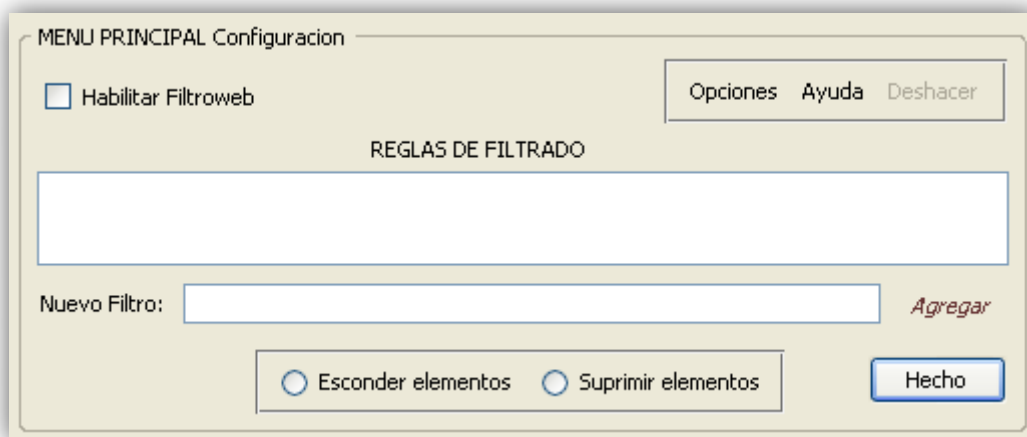
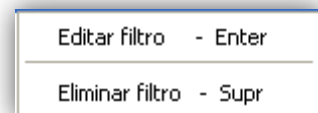


Ilustración 19 Diseño del menú principal

Como menú contextual dentro de la lista de reglas se implementa el siguiente menú:

```
<!-- list context-menu -->
<popup id="listitem-contextmenu" xposition="at_pointer">
  <menumitem id="listitem-contextmenu-modify" label="Editar filtro" - Enter" oncommand="activateMod
  <menuseparator id="listitem-contextmenu-separator.1"/>
  <menumitem id="listitem-contextmenu-delete" label="Eliminar filtro" - Supr" oncommand="removeFilter
</popup>
<!-- /list context-menu -->
```



3.4.9 em-override.xul

Este código contiene una sección CDATA (*Character Data*), el cual es un nodo especial de XML que permite especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado XML. En este caso, la razón de esta construcción llamada CDATA es para contenido de lenguaje de secuencia. Cuando el analizador XML encuentra la `<![CDATA[` inicial, notifica el contenido que sigue como caracteres sin intentar interpretarlos como elemento o marcado de identidad. Las referencias de carácter no funcionan con las secciones CDATA. Cuando encuentra la `]]>` final, el analizador deja de notificar y vuelve al análisis normal.

3.4.10 contents.rdf

El directorio *content* incluye un subdirectorio con el nombre de la aplicación (*filtroweb*) que a su vez contiene todos los archivos XUL, además de los archivos JS con el código Java Script de la aplicación y un fichero *contents.rdf* que indica entre otras cosas cual es el archivo *.XUL* principal. Es un fichero de configuración de contenido. (Sacristán, 2006)

La ruta de acceso al archivo *contents.rdf* es: *chrome://filtroweb/content/contents.rdf*.

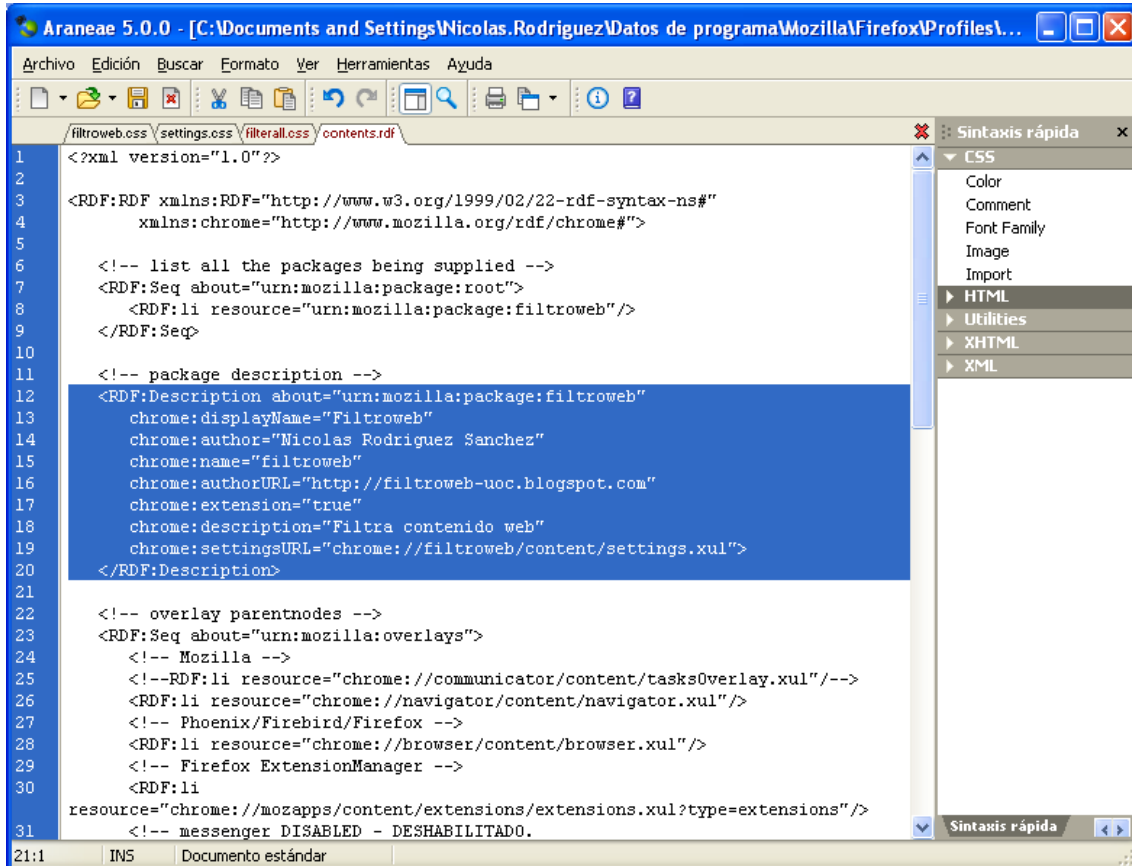


Ilustración 20 Fichero contents.rdf

3.5 Hojas de estilo (CSS)

Hojas de Estilo en Cascada (**Cascading Style Sheets**), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

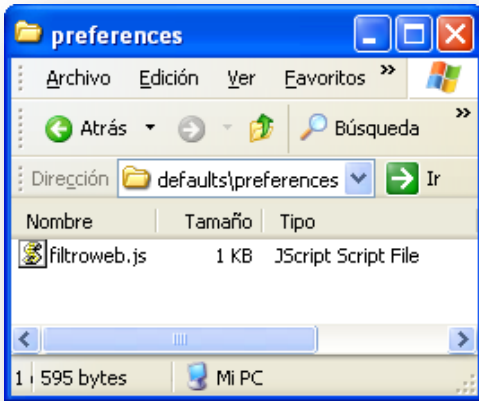
CSS se utiliza para dar estilo a documentos HTML y XML (por consiguiente XUL), separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos XML y XUL. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web o aplicaciones al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas o aplicaciones vinculadas a esa CSS en las que aparezca ese elemento. (W3C, MIT, ERCIM, & Keio, 2007)

Se utilizarán 3 documentos de hoja de estilos en cascada:

3.5.3 filterall.css

El fichero filterall.css tiene como ruta: *chrome://filtroweb/content/filterall.css*. Y trata todos los formatos que hacen referencia al funcionamiento visual de la lista de confianza, lista blanca.

3.6 Archivos por defecto



Los archivos por defecto son utilizados para crear un perfil de usuario y se crean en la carpeta defaults/ que se encuentra dentro de la carpeta raíz de la extensión. Los archivos .js se deben almacenar dentro de *defaults/preferences/* - al ser almacenados aquí serán cargados automáticamente por el sistema de preferencias de Firefox de modo que se pueda tener acceso mediante las *Preferences API*.

3.6.1 filtroweb.js

La ruta del archivo filtroweb.js es: *defaults\preferences\filtroweb.js* y su contenido es el siguiente:

 A screenshot of the Antechinus JavaScript Editor window. The title bar reads 'Antechinus JavaScript Editor - [filtroweb.js]'. The editor displays the following JavaScript code:


```

1  pref("filtroweb.enabled", true);
2  pref("filtroweb.frameobjects", false);
3  pref("filtroweb.fastcollapse", false);
4  pref("filtroweb.frameobjects", false);
5  pref("filtroweb.hide", false);
6  pref("filtroweb.linkcheck", true);
7  pref("filtroweb.listsort", false);
8  pref("filtroweb.pageblock", false);
9  pref("filtroweb.quickblock", "partial"); //"partial", "off", "full"
10 pref("filtroweb.warnregexp", true);
11 pref("filtroweb.ajaxintercept", false);
12
13 /*
14 pref("filtroweb.patterns", "");
15 pref("filtroweb.quickblockbackground", false);
16 pref("filtroweb.observer", "-"); //"-", ""
17 pref("filtroweb.statusdrop", "");
18 */
19
  
```

Ilustración 21 filtroweb.js de defaults\preferences

Si se accede desde la barra de direcciones del navegador Firefox a **about:config** se comprueban las preferencias:

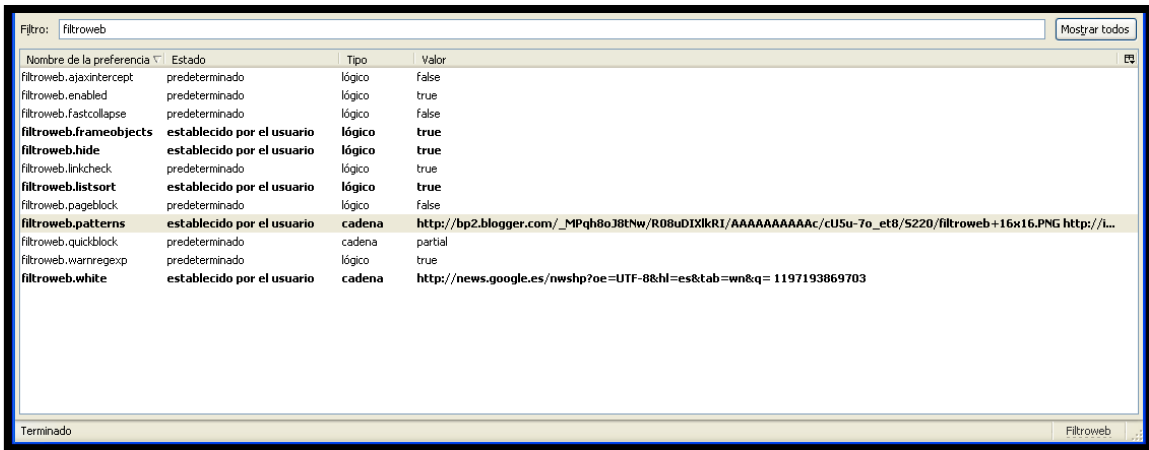


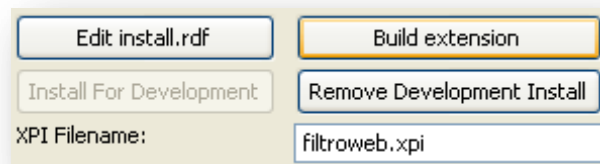
Ilustración 22 about:config

Los filtros se guardan en la preferencia llamada **filterroweb.patterns**, la cual es de tipo cadena y cuyo estado es establecido por el usuario.

3.7 Creación del instalador

Mozilla proporciona un mecanismo que se puede utilizar para empaquetar ventanas de XUL, scripts y otros archivos en un solo instalador. Se puede colocar este instalador en alguna parte para que los usuarios lo puedan descargar. Un script simple puede ser usado para tener descargado el paquete e instalarlo. Este mecanismo se llama XPIInstall (plataforma cruzada de instalación).

La herramienta *Extension Builder* permite crear de forma rápida el instalador mediante el comando *Build extension*.



Una vez que la extensión funciona, también se la puede empaquetar para su distribución e instalación de forma manual y el procedimiento consiste en comprimir en formato ZIP los contenidos de los directorios y cambiar la extensión del archivo de .ZIP a .XPI. Esto es debido a que el archivo XPI es un archivo tipo ZIP. El nombre del fichero XPI será **filterroweb.xpi**.

Se requiere que el archivo XPI contenga un archivo llamado **install.js** que es un archivo JavaScript que se ejecuta durante la instalación. Los archivos restantes son los archivos que se instalarán. Estos archivos típicamente están colocados dentro de un directorio dentro del archivo pero no es obligatorio. Los archivos cromo (chrome), deben estar estructurados como el directorio chrome.

A menudo, los únicos archivos colocados en un archivo XPI serán el script de instalación (install.js) y un archivo JAR. Este archivo JAR contiene todos los archivos usados por la aplicación. Los componentes proporcionados con Mozilla se almacenan de esta manera. (Deakin, 2007)

3.7.1 install.js

Los instaladores deben contener un script de instalación (un archivo llamado **install.js**) que se pueda utilizar para escribir el proceso de la instalación. Este script tiene acceso a varias

funciones de instalación que se pueden utilizar para instalar archivos y componentes. Es una rutina de instalación que crea e instala constantes como fichero de registro de aplicación (components/.autoreg) y fichero componente de la aplicación (components/nsFiltroweb.js).

Hay varios pasos implicados en abrir un instalador e instalar los componentes. Éstos se describen gradualmente.

1. Crea una página web desde la cual el usuario pueda descargar el software que se instalará. Esta página contendrá un 'disparador de la instalación' que es una pequeña parte del script que abre la instalación.
2. Se le presenta al usuario un diálogo indicando que el paquete está siendo instalado. Es posible que el 'disparador de la instalación' abra múltiples instaladores. En este caso, serán presentados en una lista. El usuario puede elegir continuar o cancelar.
3. Si el usuario elige continuar, se descargará el instalador del archivo XPI. Una barra de progreso se exhibe al usuario durante este proceso.
4. El archivo **install.js** es extraído del archivo de la instalación y se ejecutará. Este script abrirá las funciones del instalador que indicarán qué partes del archivo deben ser instalados.
5. Una vez que el script esté completo, el nuevo paquete ya habrá sido instalado. Si los paquetes múltiples están siendo instalados, sus scripts funcionarán en secuencia.

3.8 Componente XPCOM

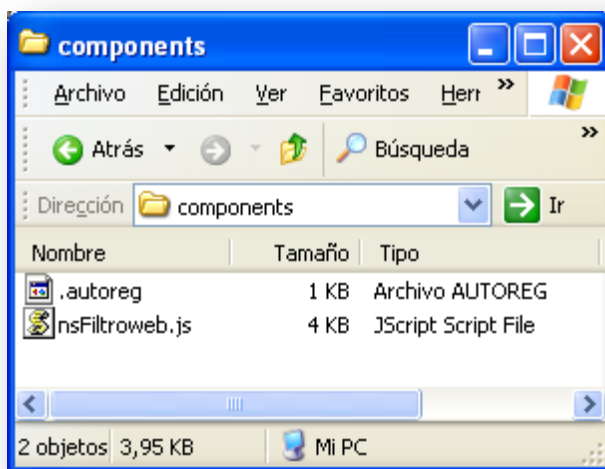


Ilustración 23 Componentes

Firefox soporta el uso del componente XPCOM en extensiones. Se pueden crear componentes propios fácilmente usando Java Script o C++. (Mgjobot, Jorolo, & Nukeador, 2006)

Colocando todos los archivos .js o .dll en el directorio components/-serán automáticamente registrados la primera vez que inicie Firefox después de instalada la extensión.

El *Cross Platform Component Object Model* (XPCOM) o Modelo de Objeto de Componentes Multiplataforma es un modelo de componente multi-plataforma. Posee, múltiples vínculos idiomáticos y descripciones IDL con las que los programadores pueden usar sus funcionalidades propias en el marco de trabajo y conectarlo con otros componentes.

3.8.1 nsFiltroweb.js

Dentro del directorio *components* se tendrá el fichero Java Script **nsFiltroweb.js**. Define las constantes:

- `_FILTROWEB_CONTRACTID = "@mozilla.org/filtroweb;1"`

- `_FILTROWEB_CID = Components.ID('{cf1682d8-4a58-4fb7-8db6-20d8b897c2a1}')`
- `_CATMAN_CONTRACTID = "@mozilla.org/categorymanager;1"`.

Por otra parte, registra componentes y carga al fichero `components.js` ("`chrome://filtroweb/content/component.js`").

3.8.2 .autoreg

Un fichero vacío llamado `.autoreg` se localiza en el directorio `components`. Si se realizan cambios al fichero `nsFiltroweb.js` será necesario actualizar el fichero `.autoreg` con el objetivo de tener el navegador con el componente re-registrado. (LouCypher, Edanuff, & Asqueella, 2005)

3.9 Implementación de funcionalidades mediante Java Script

Se exponen las diferentes **funciones** que dispone la aplicación Filtroweb y las explicaciones de los **scripts** que lo conforman.

3.9.1 addfilterdialog.js

Contempla las acciones del usuario (aceptación, cancelación) de los **filtros** solicitados. Adicionalmente este código es quien se encarga de verificar la existencia de los caracteres `@@` para invertir reglas de filtrado, tal como se ha visto en el apartado de funcionalidades de este trabajo. (Issi Camy, 2002)

```
(textbox.value.charAt(0) == "@" && textbox.value.charAt(1) == "@")
```

3.9.2 settings.js

Se encarga de cargar las **preferencias** en la interfaz de usuario. Además, permite introducir nuevas reglas de filtrado en el menú principal de la aplicación. Controla todas las acciones del **menú principal**:

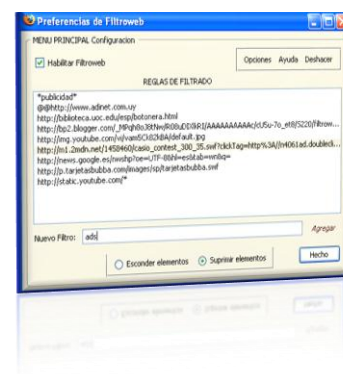
- Introducción de reglas.
- Alerta al borrar todas las reglas de filtrado.
- Importación de lista de filtros `importList()`.
- Exportación de filtros a un fichero de texto.
- **Gestión de las reglas de filtrado** (modificación, menús contextuales, etc.).
- Convierte la lista de reglas en una única cadena.
- Gestiona el menú deshacer.
- Enlaza con las webs de ayuda.

3.9.3 component.js

El fichero de código `component.js` se encarga de varias tareas.

- Es quien controla los sitios que están bloqueados enseñando un mensaje en el navegador como el

Ilustración 24 Preferencias de Filtroweb



que se muestra a continuación: (línea 500)

- **Chequea las URLs** a las que se acceden desde el navegador. Contrasta la URL con la lista de confianza y las reglas de filtrado. También verifica con enlaces contenidos.(línea 600)
- Administración de filtros y preferencias.
- Permite importar las reglas externas de filtrado desde el navegador. Función llamada *importFiltersFromString(input)* (línea 800)
- Gestiona los tipos de objetos filtrables:
 - SCRIPT
 - IMAGE
 - OBJECT
 - DOCUMENT
 - SUBDOCUMENT
 - XMLHttpRequest (Ajax)
- Crea los frames alrededor de los objetos filtrables y manipula eventos, por ejemplo, cuando se clic en la pestaña flotante de un objeto filtrable.



3.9.4 ajax-intercept.js

Fichero accesible mediante la ruta: *chrome://filtroweb/content/ajax-intercept.js*. Carga las preferencias y los filtros.

AJAX, acrónimo de *Asynchronous JavaScript* (asíncrono) *And XML*, es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- *Document Object Model* (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como Java Script, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor.

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente. Es más que Java Script, incluye el envío – por debajo – de tramas en formato XML que permiten ser pintadas al cliente, sin necesidad de refrescar la página web.

Se puede decir que es un aprovechamiento del lenguaje JavaScript para realizar llamadas **asíncronas** a recursos del servidor. Cuando se dice asíncrona, significa que la aplicación puede hacer un **pedido al servidor** sin necesidad de que la página completa “se detenga”, “se blanquee”. Un pedido al servidor es por ejemplo una consulta al listado de usuarios, es decir, uno puede presionar un botón y con esto traer información del usuario sin necesidad de blanquear toda la página que se está visualizando.

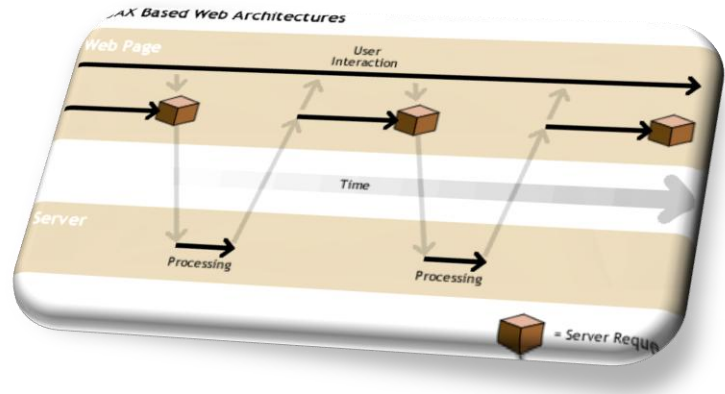


Ilustración 25 Arquitectura AJAX

3.9.5 component-defaultfilters.js

Contiene una variable utilizada llamada *defaultFilterString* cuyo valor es una cadena de filtros por defecto cargada por el fichero de código *component.js*.

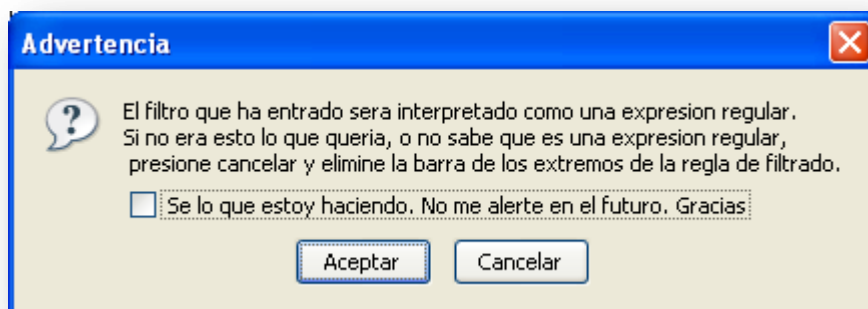
3.9.6 filterall.js

Esta rutina mantiene la lista de confianza (lista blanca) y por tanto las reglas de filtrado. A dicho fichero se puede acceder mediante la ruta: *chrome://filtroweb/content/filterall.js*.

3.9.7 global.js

Chequea presencia de expresiones regulares en la introducción de reglas de filtrado y advierte al usuario.

Ilustración 26 Introducción de expresiones regulares



3.9.8 filtroweb.js

Indica el estado de Filtroweb:

- this.statusText.value = "NoCargado"
 - this.statusElement.setAttribute("tooltiptext", "Filtroweb no se pudo cargar")
 - Estado: No ejecución
- this.statusText.value = "Deshabilitado"
 - this.statusElement.setAttribute("tooltiptext", "Filtroweb esta deshabilitado")

- Estado: En ejecución pero no escuchando. Deshabilitado.
- `this.statusText.value = "Filtroweb"`
 - `this.statusElement.setAttribute("tooltiptext", "Filtroweb esta habilitado")`
 - Estado: Habilitado, escuchando.

Entre las principales acciones que realiza el script `filtroweb.js` se encuentran:

- Contempla fallos en la carga de la aplicación.
- Chequea ficheros necesarios en Mozilla.
- Controla la activación/desactivación de la aplicación.
- Administra el filtrado de los elementos.
- En base a los tipos de elementos a filtrar muestra un menú u otro (contextual).
- Intercepta clics de ratón y esconde o suprime elementos filtrables.
- Eliminación de objetos que no se quiere visualizar.
- Chequea duplicidades de registros.
- Lanza las ventanas de preferencias y de elementos filtrables.

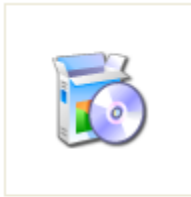
3.9.9 `uninstall.js`

Se ha adaptado código del proyecto *Optimoz* (<http://optimoz.mozdev.org/>). Prepara las variables de aplicación para la inicialización de la instalación. Desinstalará fichero *jar*, *XUL* caché, componentes y *chrome* instalado. Contempla todas las posibles excepciones. Las funciones principales son:

- `removeChromeAndOverlayEntries()`
- `removeInstalledChrome()`
- `deleteJar()`
- `deleteXULCache()`
- `deleteComponent()`
- `removeCompregDat()`

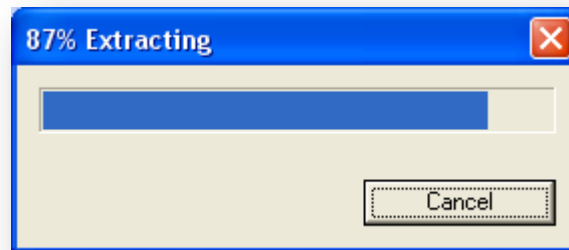
Capítulo 4 – Manual de Instalación

4.1 Instalación de Mozilla Firefox



Firefox Setup
2.0.0.7.exe

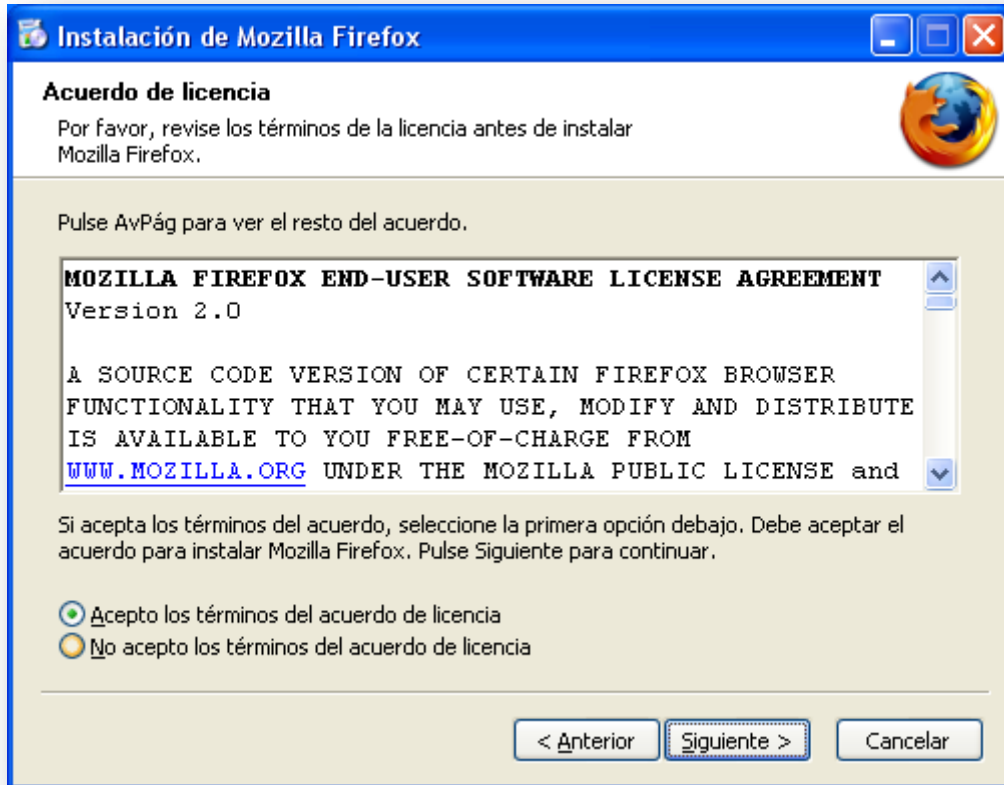
Para empezar se ha de instalar el navegador Mozilla Firefox. Ejecutar el instalador llamado Firefox Setup 2.0.0.7.exe. Al ejecutarlo comenzará la extracción de ficheros y se mostrará una barra de proceso como la siguiente:



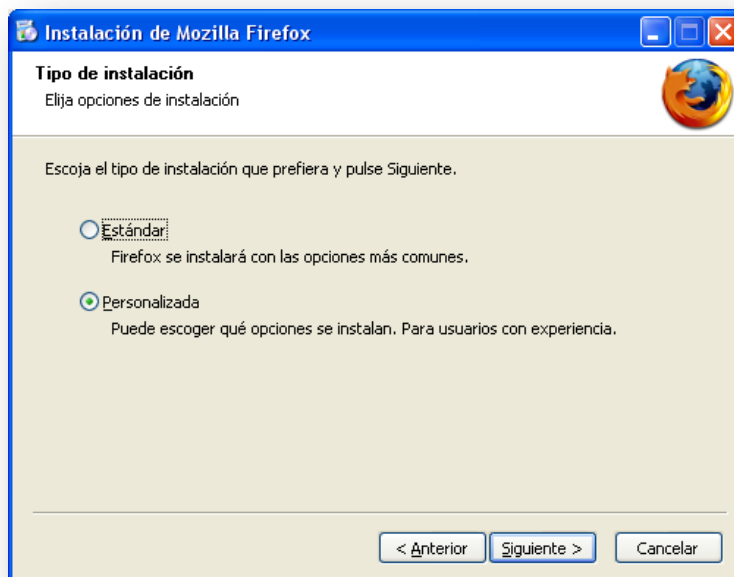
Se mostrará la bienvenida al asistente de instalación Mozilla Firefox:



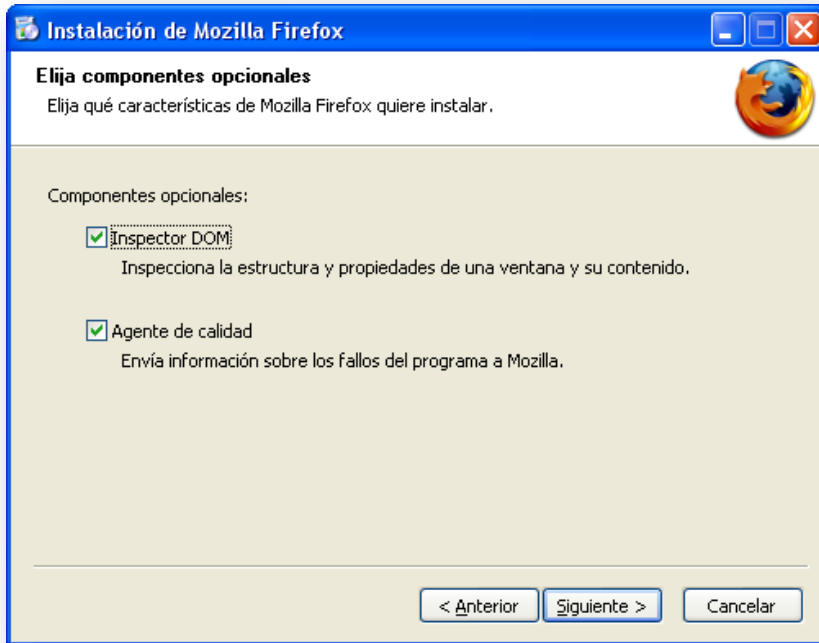
Aceptando los términos de acuerdo de licencia se puede continuar con el proceso de instalación:



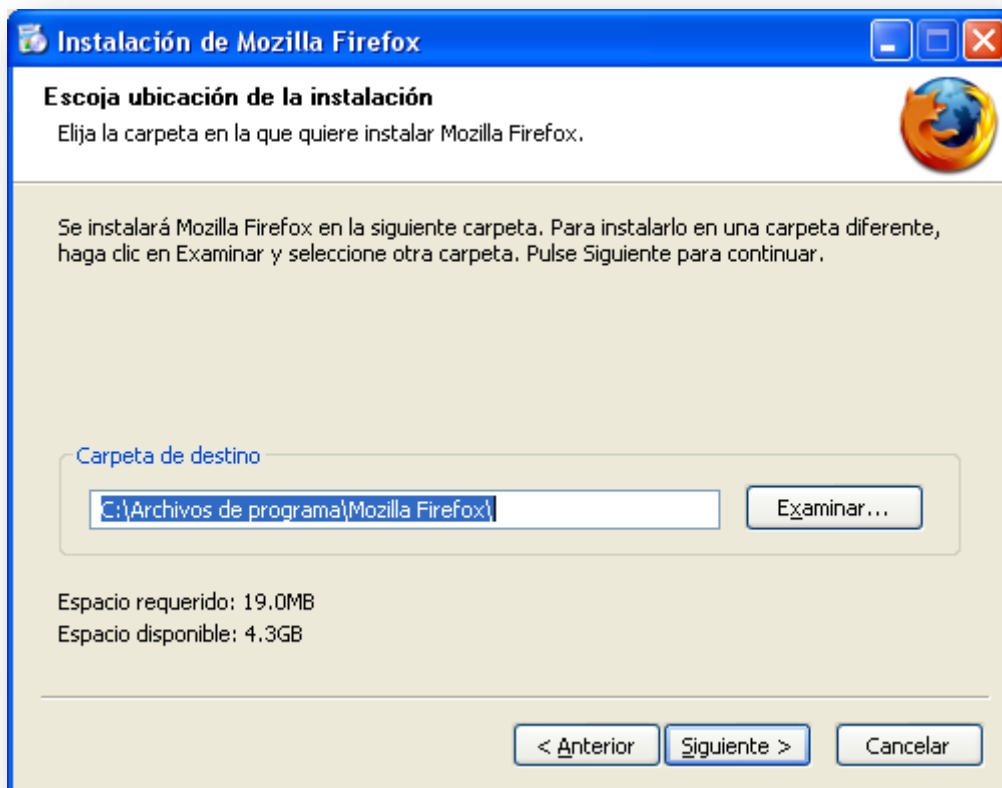
Se instalará la aplicación de forma personalizada:



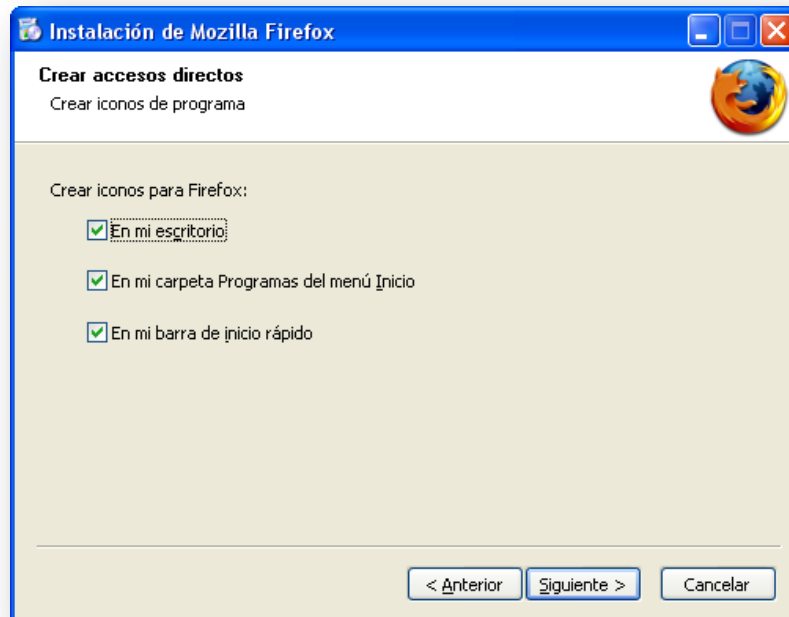
Seleccionar los componentes Inspector DOM y agente de calidad:



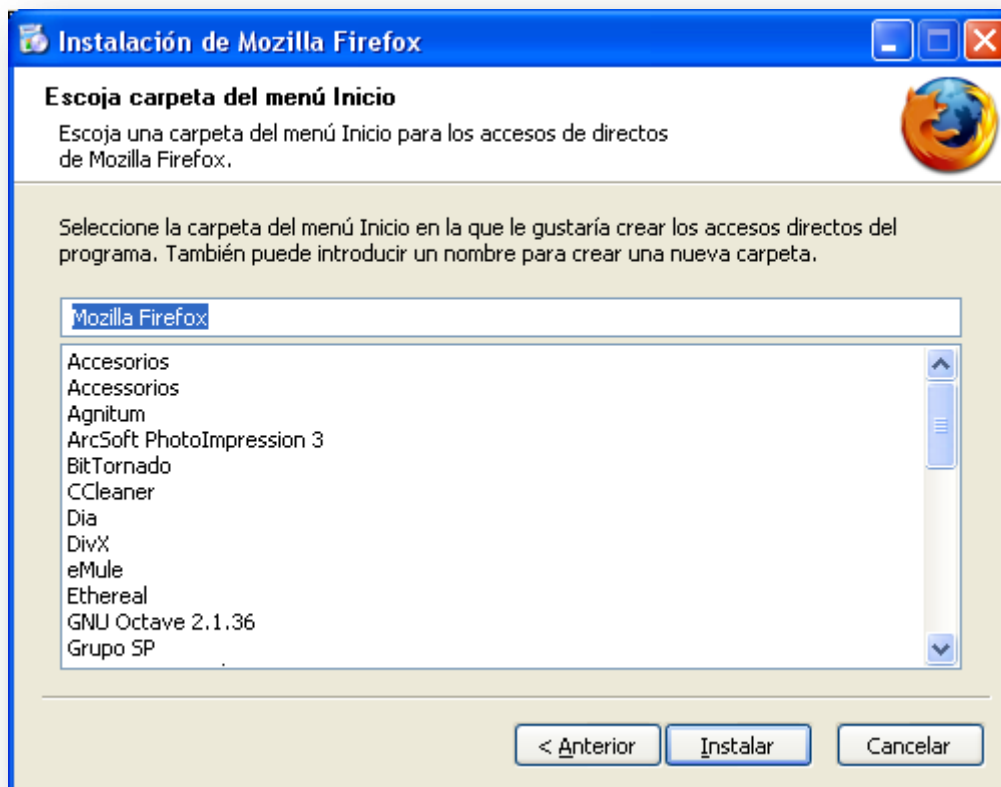
Propondrá una carpeta de destino de instalación:



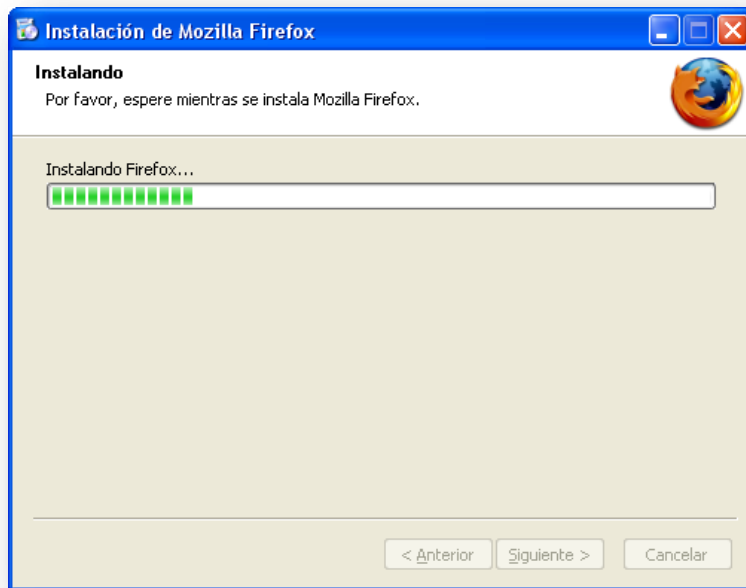
Crear iconos de acceso en escritorio, carpeta Programas del menú Inicio y en la barra de inicio rápido:



Especificar el nombre de la carpeta del menú inicio:



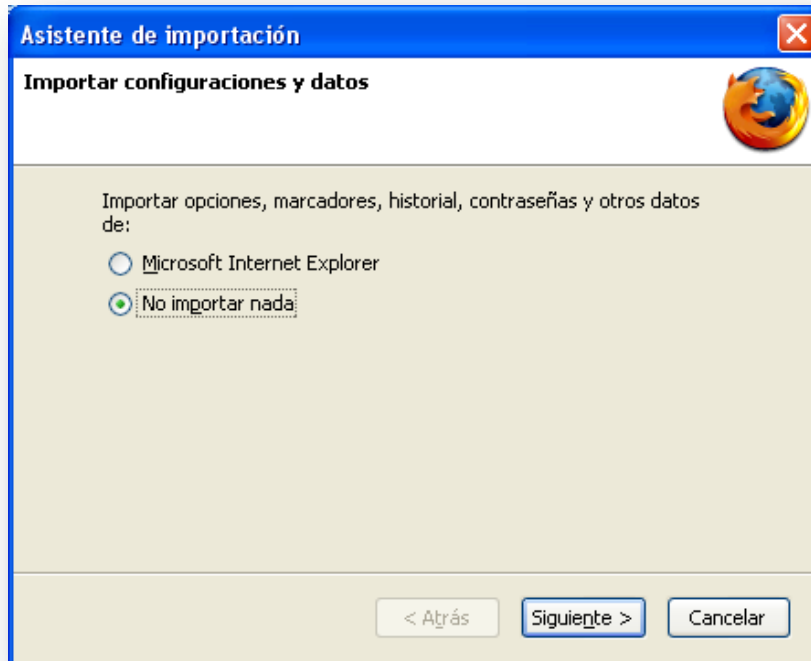
Barra de proceso de instalación:



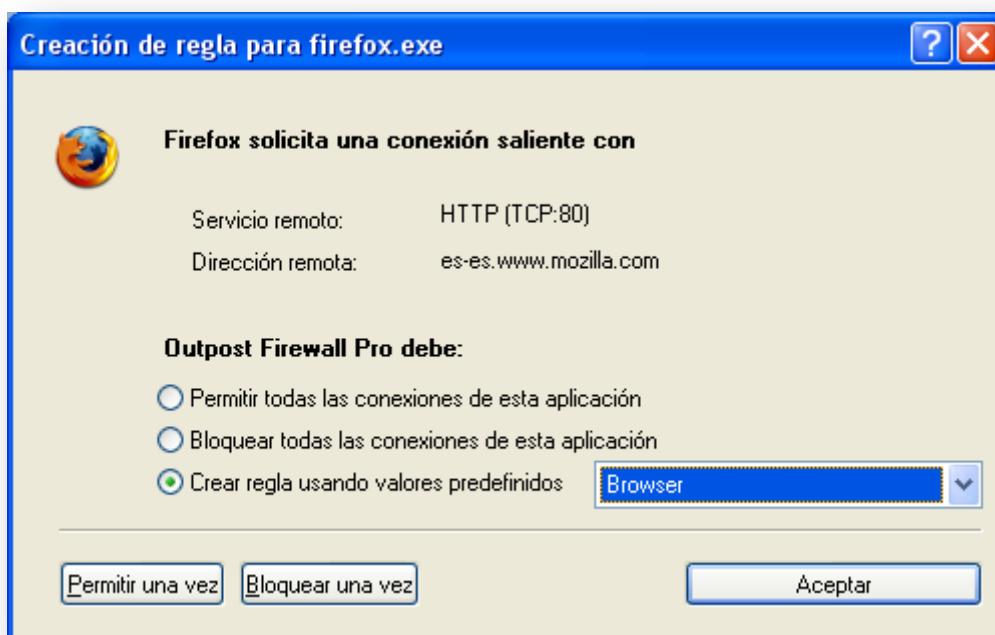
Proceso de instalación terminado:



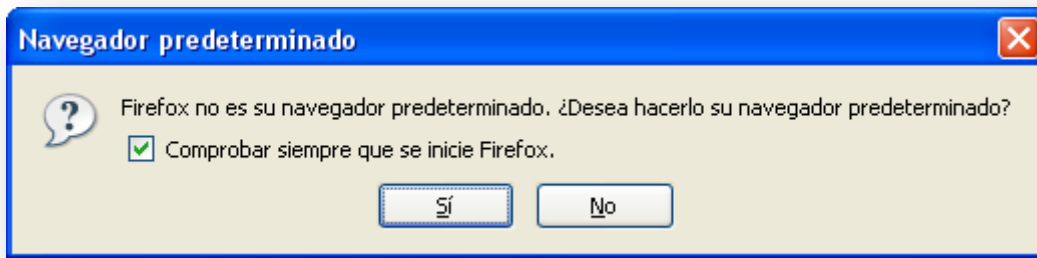
Es posible importar opciones de configuración de Microsoft Internet Explorer. Sin embargo se optará por no importar nada:



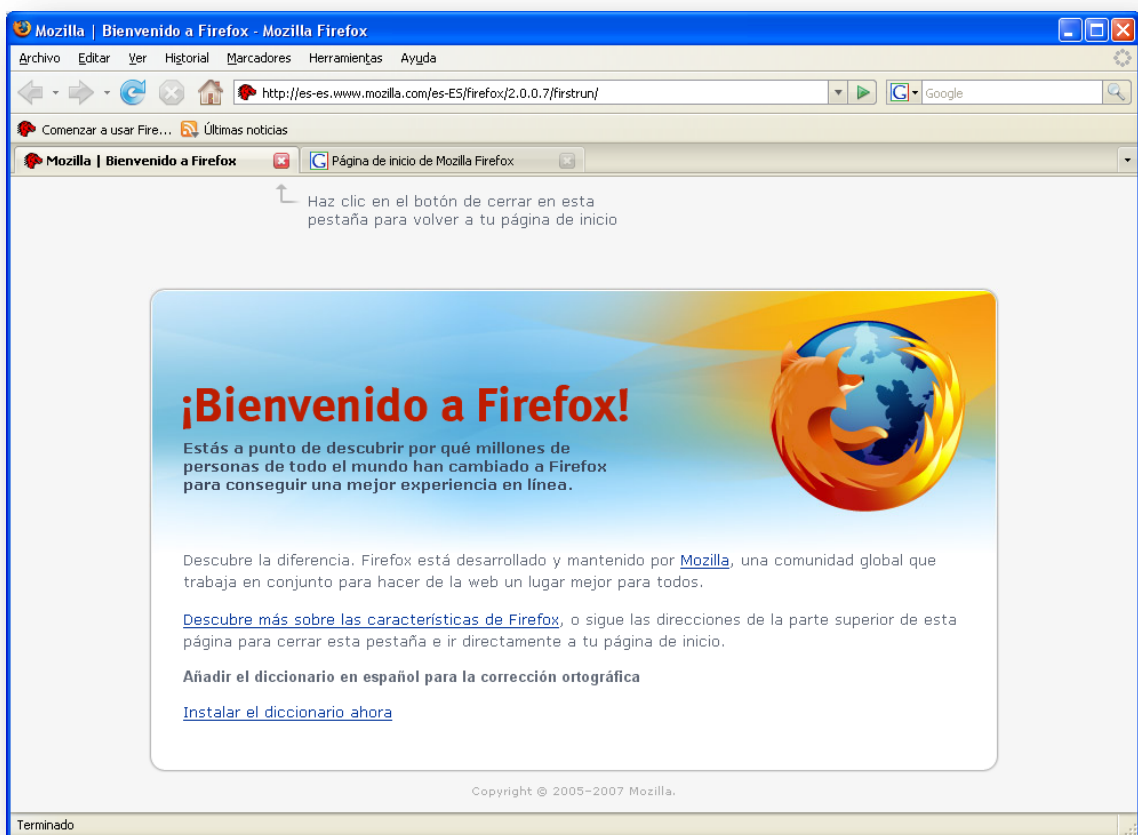
Firefox solicita conexión saliente y si se tiene un cortafuegos instalado puede que aparezca alguna advertencia. En este caso, se aceptará dicha petición de conexión saliente:



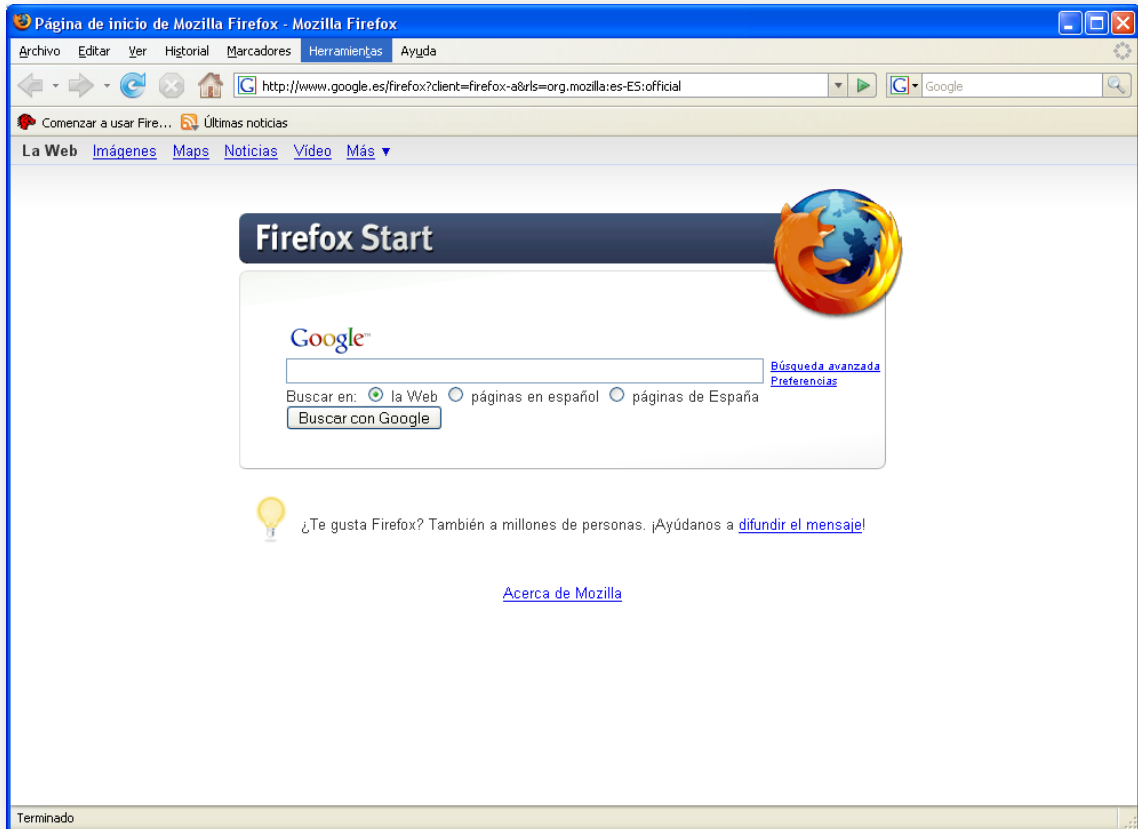
Propone a Mozilla Firefox como navegador predeterminado.



Se inicia el navegador con una página de bienvenida:



Y ya se puede disfrutar del navegador instalado Mozilla Firefox:

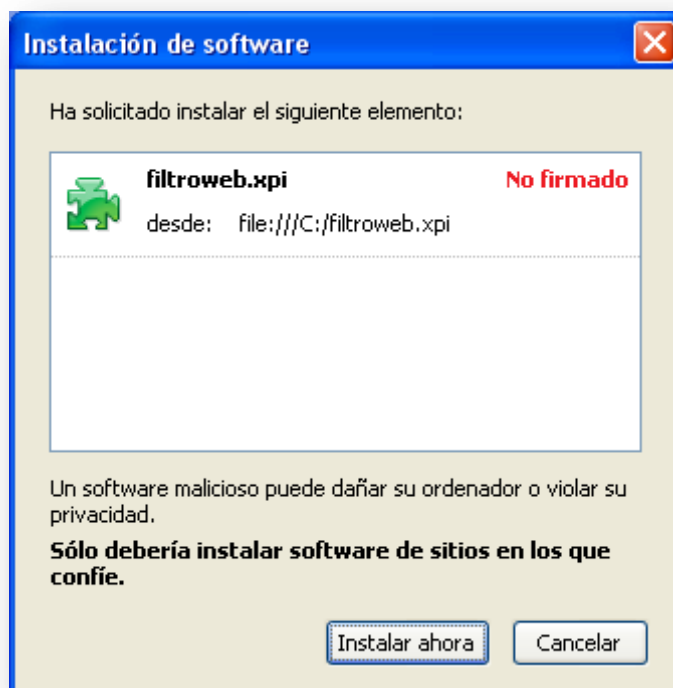


4.2 Instalación de Filtroweb

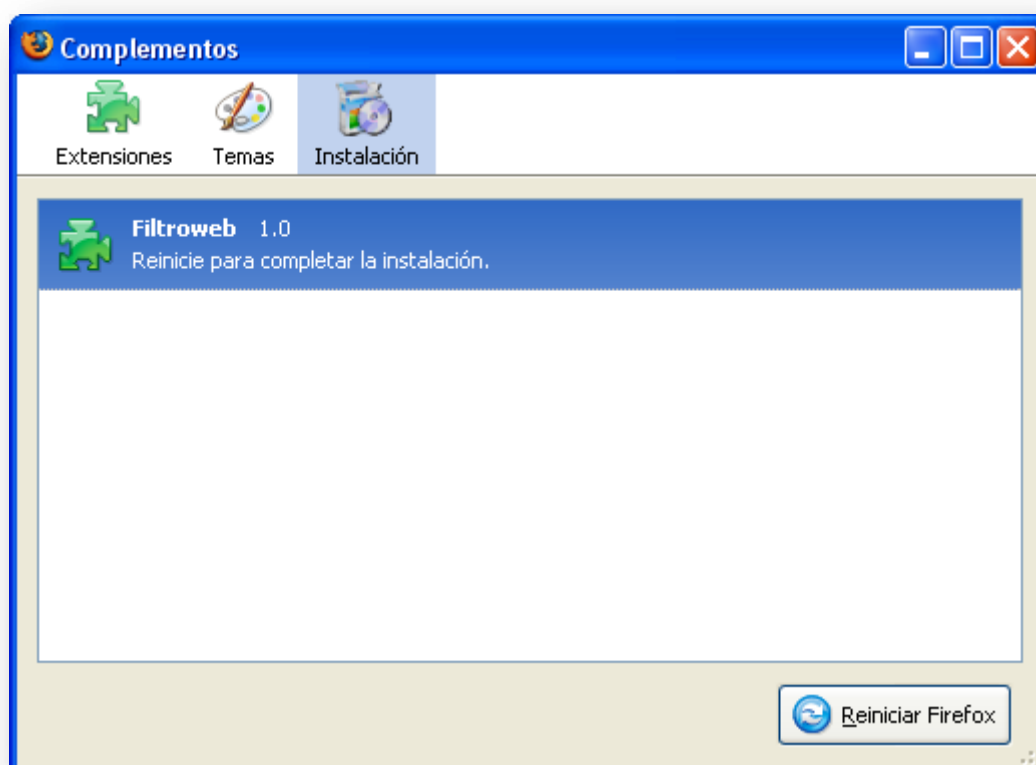
A continuación se describirán los procesos de instalación/desinstalación de la aplicación Filtroweb, los cuales son muy ágiles y sencillos.

4.2.1 Instalación desde fichero filtroweb.xpi

Se ejecuta el fichero filtroweb.xpi con la aplicación Mozilla Firefox. Una forma rápida de realizarlo es abrir Firefox y arrastrar filtroweb.xpi al navegador y se mostrará la ventana siguiente:



Al clicar en *Instalar ahora*, se ejecutará el proceso de instalación:

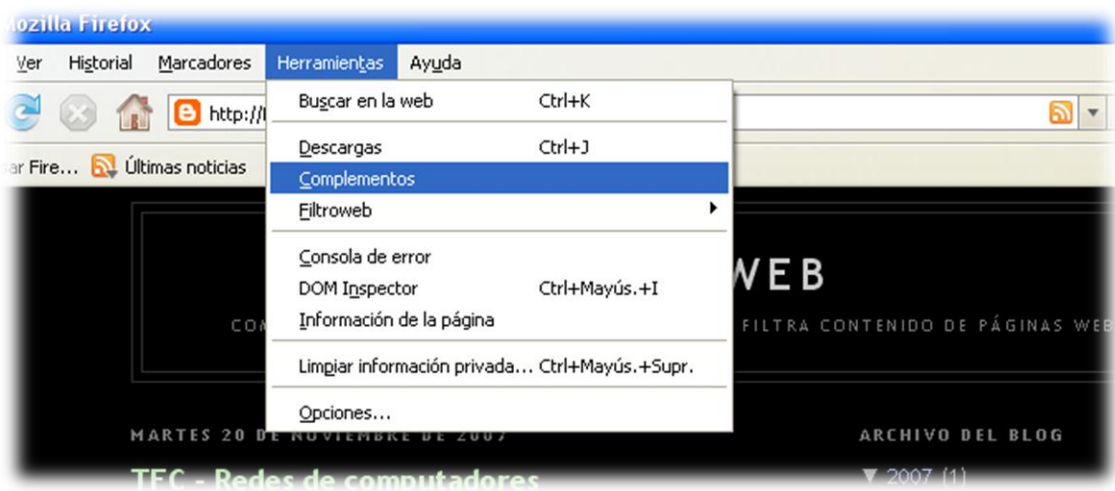


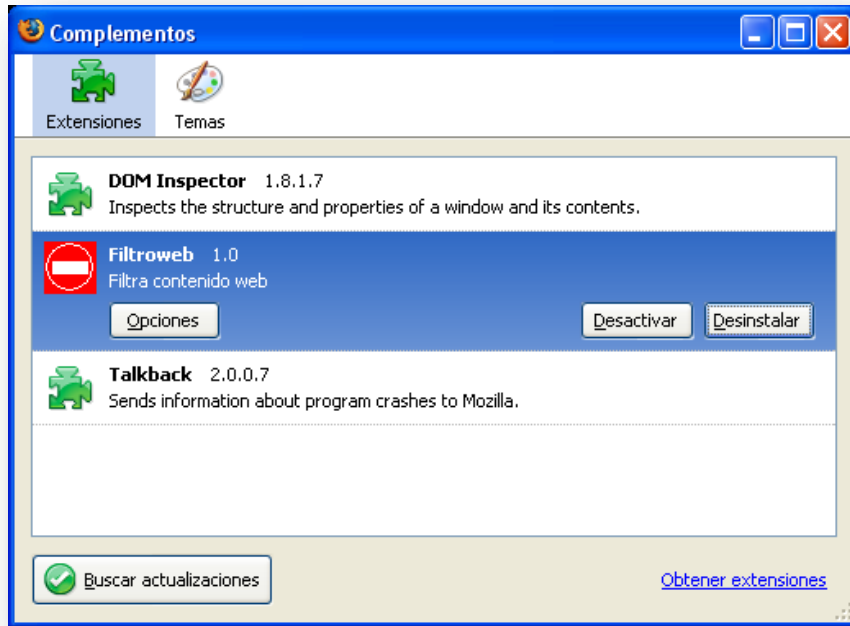
Pedirá reiniciar Firefox y al reiniciarlo se verá en la barra de estado (sector inferior derecho) que la aplicación Filtroweb ya se encuentra en ejecución:



4.2.2 Desinstalación de Filtroweb

Se ha de entrar en el menú **Herramientas** del navegador seguido de **Complementos**:

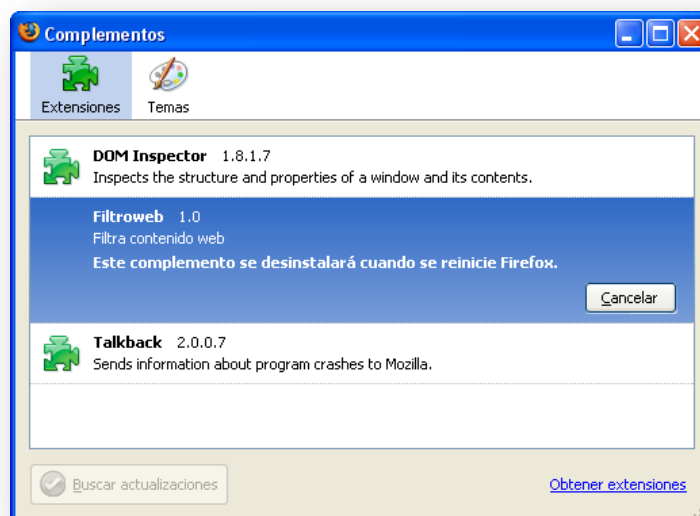




Si se clica en **Desinstalar**:

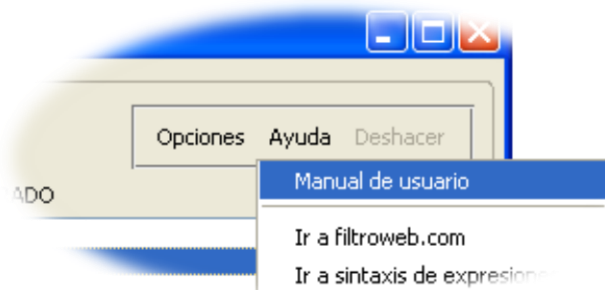


Al confirmarlo:



Capítulo 5 – Manual de Usuario

A continuación se describen las funcionalidades básicas de la aplicación Filtroweb. Se puede acceder a una versión más ampliada desde las preferencias del propio programa en el menú ayuda.



5.1 El icono Filtroweb

Una vez instalada la aplicación Filtroweb, se visualiza en la barra de estado – en el extremo inferior derecho de la pantalla – el icono Filtroweb. Clicando sobre él abre la lista de elementos filtrables (Ver [apartado 5.3](#)).

Adicionalmente, clicando con botón derecho sobre dicho icono se muestra el siguiente menú:

Agregar página actual a lista blanca: Añade la página web que se está visualizando a la lista de confianza.

Agregar este sitio a lista blanca: Añade el sitio entero a la lista segura en la que se confía y por tanto no se aplica ningún tipo de filtro.

Ver elementos filtrables: Muestra los elementos bloqueables.

Bloquear Flash: Filtra objetos flash.

Deshabilitar Filtroweb: Deja inactiva la aplicación. Si está deshabilitada, este menú muestra **Habilitar Filtroweb** para reactivar el programa.

Preferencias: Abre el menú principal de Filtroweb (Ver [apartado 5.4](#)).

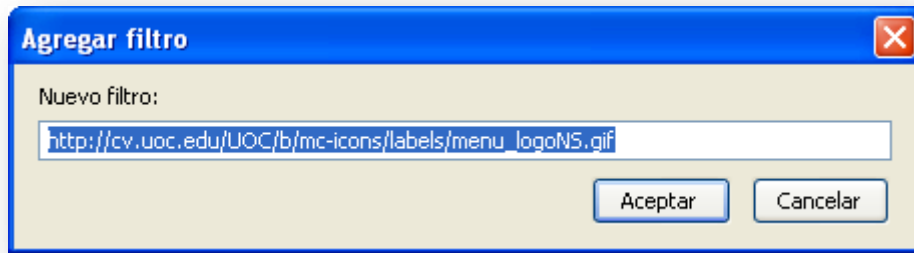


Ilustración 27 Menú Filtroweb en barra de estado

5.2 El menú contextual

El menú contextual mediante el botón derecho del ratón sobre la mayoría de elementos en una página web contiene un elemento de menú para filtrarlo, es decir, si se trata de un elemento de tipo imagen mostrará el elemento de menú “**Filtroweb: Bloquear imagen...**”. Si se trata de un elemento de tipo *frame*, el menú será “**Filtroweb: Bloquear marco...**”, y así sucesivamente para cada tipo de elemento filtrable. Clicando sobre el elemento de menú correspondiente aparecerá una ventana emergente para introducir un nuevo filtro, con dicho

campo ya rellenado con la dirección del elemento para la edición por si fuera necesario modificar la regla:



5.3 La lista de elementos filtrables

La lista de elementos filtrables (accesible también mediante botón izquierdo del ratón en la barra de estado) es la herramienta más poderosa para **identificar** los **elementos** de una página web que se pueden filtrar. Elementos que pueden ser de varios tipos: imágenes, fondos, marcos, scripts, objetos embebidos, etc.

Cada línea de dicha lista representa un elemento filtrable por Filtroweb, mostrando su **dirección** y sus **propiedades**. Dicha dirección es la cadena usada para contrastar con las reglas de filtrado. Clicando una línea en esta lista hace que un cuadro rojo (flash) marque al elemento correspondiente en el sitio web. Adicionalmente, la dirección es mostrada en el campo de nuevo filtro para utilizarlo y aceptándolo, tal cual es o editándolo, se añade a la lista de filtros.

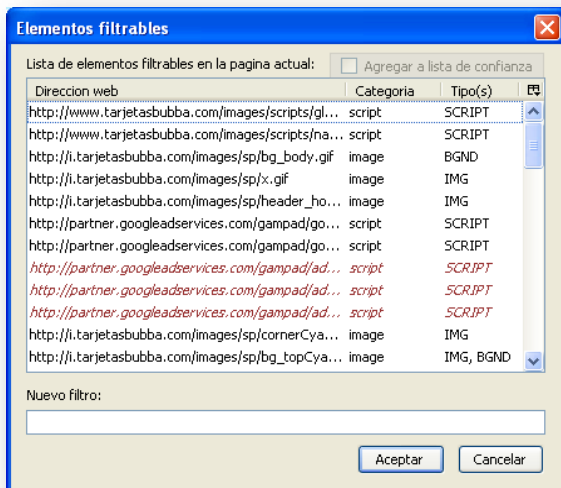
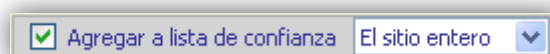


Ilustración 28 Elementos filtrables

y eligiendo del desplegable: esta página o el sitio entero.

Los elementos ya filtrados son mostrados en *rojo y cursiva*. Clicando sobre una de estas líneas muestra la regla de filtrado responsable del bloqueo del elemento en el campo debajo y permite obtener una idea general sobre el funcionamiento de las reglas que estructuran a la aplicación Filtroweb.

Por si fuera poco, en este punto también es posible agregar páginas o sitios a la lista de confianza. El procedimiento es muy intuitivo, seleccionando el elemento filtrable se ha de activar la casilla de verificación “Agregar a lista de confianza”



5.4 El menú principal: Preferencias

Menú accesible pulsando el botón central del ratón sobre la barra de estado o desde el navegador menú Herramientas – Filtroweb – Preferencias. El gran campo central en la ventana **Preferencias** muestra las **Reglas de Filtrado**, las cuales pueden ser editadas incluso aquí clicando con botón derecho sobre ellas y eligiendo: **Editar filtro** (o pulsando Enter). Incluso, existen

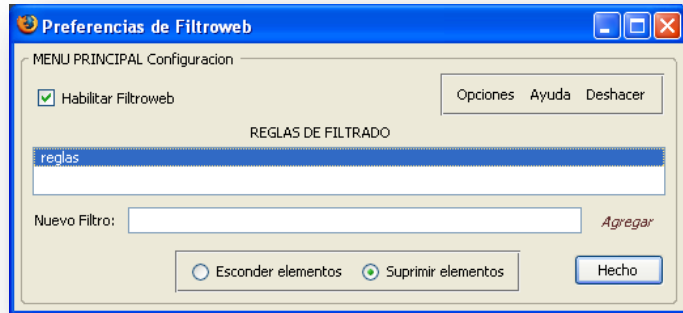



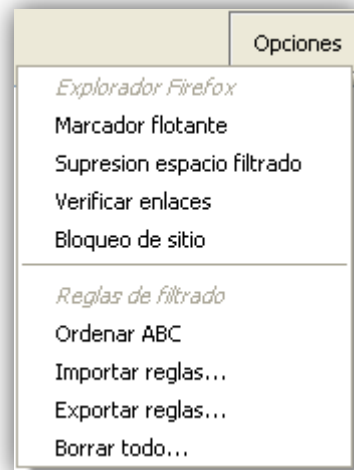
Ilustración 29 Menú principal

otros elementos de menús que se describirán a continuación (en sentido horario desde arriba a la izquierda).

Habilitar Filtroweb: Otra forma de activar/desactivar Filtroweb.

Opciones: Es el ítem de menú más importante. Se describirán sus sub-menús agrupados en “*Explorador Firefox*” y “*Reglas de filtrado*” y se explicarán sus comportamientos.


- **Marcador flotante:** Activando esta opción añade una pestaña flotante a cada objeto *plugin*, facilitando un acceso directo a su localización. 
- **Supresión espacio filtrado:** Indica si el espacio ocupado por un elemento bloqueado debería ser suprimido o no. Suprimiendo el espacio hace que la posición ocupada por el elemento filtrado desaparezca completamente, liberando el estado real de la pantalla para lo que realmente lo que se quiere ver.
- **Verificar enlaces:** Si se activa esta opción no solo chequeará la localización del elemento, sino que también a donde enlaza. Esto permite captar banners servidos desde una localización inofensiva que enlaza a un conocido sitio de publicidad, por ejemplo.
- **Bloqueo de sitio:** Permite bloquear un sitio completo. Filtroweb mostrará una página de información indicando que el sitio ha sido bloqueado por completo. Precediendo un dominio con @@ desactiva completamente el bloqueo del sitio para dicho dominio.
- **Ordenar ABC:** Activando esta opción Filtroweb ordena la lista de reglas de filtrado alfabéticamente.
- **Importar reglas:** Permite importar configuración de filtros desde un fichero local. El fichero debe contener en la primera línea: **[Filtroweb]**.
- **Exportar reglas:** Exporta reglas de filtrado a un fichero local.



- **Borrar todo:** Elimina todos los filtros, por consiguiente, Filtroweb no bloqueará ningún elemento.

Ayuda: Permite consultar información relacionada con la aplicación Filtroweb.

- **Manual de usuario:** Abre el manual de usuario en formato PDF.
- **Ir a filtroweb.com:** Accede a la página web del programa.
- **Ir a sintaxis de expresiones:** Accede a la página web que explica la sintaxis de expresiones regulares.

Deshacer: Revierte todos los cambios realizados durante la sesión mientras preferencias ha estado activo. Cerrando la ventana utilizando el botón  en lugar del botón **Hecho** tiene el mismo efecto.

Esconder elementos: Deja en blanco el espacio ocupado por el elemento filtrado.

Suprimir elementos: Elimina el espacio que ocupa el elemento filtrado.

Tal como se ha comentado anteriormente, si un sitio ha sido bloqueado por completo, cuando se intente acceder al mismo, se mostrará una página informativa como esta:



Capítulo 6 – Conclusiones

El trabajo final de carrera ha consistido en la creación de un complemento que añade nuevas y potentes características al navegador conocido como Mozilla Firefox. El complemento es un filtro de páginas webs a nivel de contenido con el fin de poder navegar controlando los elementos que se quieren visualizar y que cumpla con las expectativas, no solo de un usuario doméstico, sino también de un **usuario avanzado**. La aplicación ha sido bautizada: **Filtroweb**.

En este proyecto se emplearon varios lenguajes de programación y técnicas de diseño y codificación lo cual me ha permitido **aplicar** muchos de los **conocimientos** adquiridos durante mi carrera universitaria y etapa profesional y al mismo tiempo **aprender** muchísimo de todo el entorno de la aplicación.

A nivel personal me he sentido muy **involucrado** con el trabajo final de carrera ya que cada paso que daba en la implementación del mismo era una meta marcada que se iba cumpliendo y eso me ha **motivado** enormemente a la hora de trabajar con el proyecto a lo largo de todos estos meses.

Considero que todos los objetivos se han cumplido, aunque el proyecto haya sufrido alguna desviación temporal. La causa siempre ha tenido un fundamento: la aspiración a la **superación continúa** en la evolución del desarrollo de este trabajo.

Una meta alcanzada, ha sido también el logro de crear una aplicación de **fácil manejo** para el usuario, ya que es vital para que un programa de esta magnitud tenga éxito.

No obstante, se pueden añadir más funcionalidades a Filtroweb, siempre orientándose al usuario final, y permaneciendo alineados con los objetivos marcados. Una mejora posible sería que la aplicación fuera **multi-idioma**, permitiendo así llegar a un público más amplio.

Una siguiente fase podría ser estudiar la posibilidad de mantener una lista de reglas de filtrado continuamente actualizada mediante el protocolo HTTP, otorgando así al usuario un **mantenimiento** de la aplicación **desatendido**.

Espero que el usuario que utilice la aplicación se encuentre a gusto y conforme con el rendimiento y la funcionalidad de este filtro de páginas web.

Vocabulario utilizado en la memoria

Glosario

BE: Acrónimo de *Back End*, la parte de networking de Mozilla.

FE: Acrónimo *Front End*, la parte gráfica (de *display*) de Mozilla.

Lista blanca: Sinónimo de lista de sitios de confianza. Todo elemento que pertenezca a esta lista no será filtrado por la aplicación Filtroweb.

Lista de filtros: Reglas de filtrado.

URI: Acrónimo de Uniform Resource Identifier. Un URI es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc). Normalmente estos recursos son accesibles en una red o sistema.

XBL: XML Binding Language (XBL, algunas veces simplemente denominado Extensible Bindings Language) es un lenguaje para describir vinculaciones que pueden ser adjuntadas a elementos en otros documentos. El elemento al cual el vinculado es añadido, llamado elemento vinculado, adquiere el nuevo comportamiento especificado por el vínculo.

XP: *Cross platform*, significando portable, como en XPCOM, XPFE, XPInstall.

Bibliografía

Bibliografía

BijuGC, Mjibot, Ptak82, Nickolay, Ldrhcp, & Readams. (21 de Octubre de 2007). *CSS:-moz-binding*. Recuperado el 23 de Octubre de 2007, de CSS:-moz-binding: <http://developer.mozilla.org/en/docs/CSS:-moz-binding>

Deakin, N. (28 de Julio de 2007). *Tutorial de XUL*. Recuperado el 20 de Noviembre de 2007, de Tutorial de XUL: http://developer.mozilla.org/es/docs/Tutorial_de_XUL

Inc, W. F. (7 de Diciembre de 2007). *Wikipedia*. Recuperado el 10 de Diciembre de 2007, de Wikipedia: <http://es.wikipedia.org/wiki/Portada>

Issi Camy, L. (2002). *La Biblia de JavaScript*. Madrid: Anaya Multimedia.

LouCypher, Edanuff, & Asqueella. (15 de Mayo de 2005). *Dev : Extending the Chrome Protocol*. Recuperado el 21 de Octubre de 2007, de Dev : Extending the Chrome Protocol: http://kb.mozillazine.org/Dev_-_Extending_the_Chrome_Protocol#Installation

MetaEmotion. (2006). *XUL: una alternativa viable para el desarrollo Web*. Recuperado el 9 de Octubre de 2007, de XUL: una alternativa viable para el desarrollo Web.: <http://blog.metaemotion.com/2007/01/11/xul-una-alternativa-viable-para-el-desarrollo-web/>

Mjibot, Jorolo, & Nukeador. (8 de Diciembre de 2006). *XPCOM*. Recuperado el 11 de Octubre de 2007, de XPCOM: <http://developer.mozilla.org/es/docs/XPCOM>

Nukeador, Psanz, Arteadonis, Gbulfon, & otros. (9 de Agosto de 2007). *Extensiones*. Recuperado el 25 de Septiembre de 2007, de Extensiones: <http://developer.mozilla.org/es/docs/Extensiones>

Pitts, N. (1999). *XML / Natanya Pitts*. Madrid: Anaya Multimedia.

Rumbaugh, J., Jacobson, I., & Booch, G. (2000). *UML. El lenguaje de modelado unificado. Manual de referencia*. Addison Wesley.

Sacristán, L. (2006). *FireMarker: cómo crear una extensión para Firefox*. Recuperado el 18 de Noviembre de 2007, de FireMarker: cómo crear una extensión para Firefox: <http://sentidoweb.com/2007/03/02/firemarker-como-crear-una-extension-para-firefox.php>

W3C, MIT, ERCIM, & Keio. (2 de Noviembre de 2007). *Guía Breve de CSS*. Recuperado el 5 de Noviembre de 2007, de Guía Breve de CSS: <http://www.w3c.es/Divulgacion/Guiasbreves/HojasEstilo>

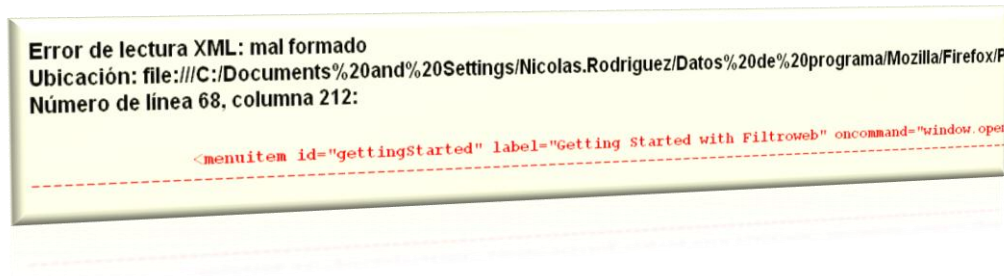
Welcome to Software Development the Mozilla Way. (s.f.). Recuperado el 17 de Noviembre de 2007, de Welcome to Software Development the Mozilla Way: <http://mb.eschew.org/intro>

Wesdorp, J. (6 de Febrero de 2007). *Creating a Firefox sidebar*, English. Recuperado el 26 de Septiembre de 2007, de Creating a Firefox sidebar:
<http://occidopagus.nl/firefox/emptysidebar/>

Anexos

Anexo 1 – Incidencias

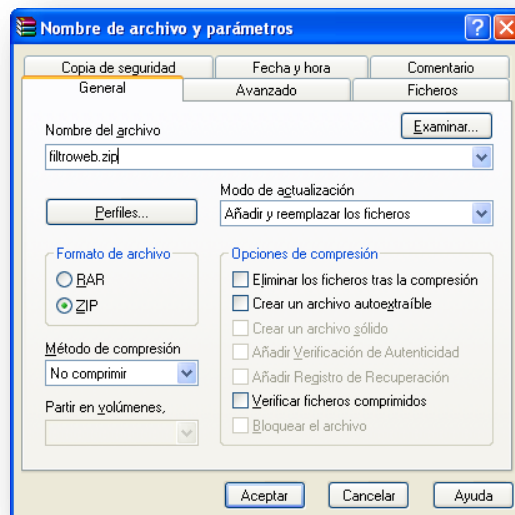
Ejemplo de algunas de las **incidencias** en la codificación lo que ha implicado una aplicación de **mejora continua** al trabajo y por tanto una mayor dedicación y aumento de una semana más en la planificación inicial.



Anexo 2 – Fichero con extensión .jar

Para montar el fichero JAR (*filtroweb.jar*) primero se ha de crear el fichero ZIP mediante la siguiente herramienta (sin ningún método de compresión):

Ilustración 30 Montaje fichero JAR



Y seguidamente renombrar de *filtroweb.zip* a *filtroweb.jar*.