UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

etsinf

Desarrollo de un filtro de contenidos con diccionario semántico para el navegador Web Firefox

Autor:
Domingo Morello Ostos

Director:

Josep Silva Galiana

"En primer lugar me gustaría agradecer a Josep la oportunidad de realizar el proyecto bajo su dirección.

A mis padres y mi hermana por no tener en cuenta los cambios de humor derivados del estrés al que estaba sometido en determinadas épocas del año.

A los compañeros de clase y profesores que comprendieron mi situación y me dieron su confianza.

A David, Juan, Sergio, Furió y a ti compañero del club de ajedrez por ayudar a desintoxicarme de los estudios.

Y en especial a Sandra porque sin su apoyo incondicional durante toda la carrera nada de esto hubiera sido posible".

Contenido

Índice de ilustraciones

1 Objetivos

Con la realización de este proyecto se pretende ampliar las funcionalidades de una herramienta ya existente. La herramienta en concreto tiene como nombre "*Webfiltering toolbar*" y se trata de un plugin para el navegador web Firefox cuya funcionalidad principal permite aplicar filtros sobre el contenido de la página que se visualiza en cada momento, permitiendo al usuario final centrar su atención únicamente en los conceptos que se introducen en el campo de búsqueda y que aparecen en la web sobre la que se realiza el filtrado. La funcionalidad sobre la que se centrará el proyecto pretende dotar a "*Webfiltering toolbar*" de un mecanismo por el cual además de filtrar por la palabra o palabras introducidas en el campo de búsqueda, aplicará el filtrado sobre conceptos relacionados semánticamente, como por ejemplo sinónimos, con los introducidos en el campo de búsqueda.

Durante el desarrollo del proyecto aparecen otros objetivos derivados del objetivo principal que consisten en comprender la estructura general de los plugins de Firefox, así como entender las funcionalidades de la última versión estable de la barra "*Webfiltering toolbar*" y por último conseguir dominar los lenguajes de programación utilizados en este tipo de aplicaciones (**XUL, Javascript**) y el modelo de Objetos de Documento **DOM**.

El presente documento pretende detallar el proceso de creación de la aplicación que sigue todos los pasos de todo proceso de ingeniería: Especificación de requisitos, análisis y diseño del producto, implementación del producto y pruebas.

La documentación que se presenta a continuación se estructura en dos bloques. El primero está dedicado al estudio de las tecnologías que emplearemos durante la etapa de implementación del proyecto. El segundo bloque es el que incluye el análisis del problema, y la especificación de requisitos, basada en el documento de Especificación de Requisitos Software basado en el estándar IEEE. 830-1998.

Por último es también en el segundo bloque donde se detalla la implementación de la aplicación, entrando en detalle sobre las funciones que proporcionan nuevas funcionalidades a "*Webfiltering toolbar*". En esta sección se comenta la estrategia general de implementación adoptada. Se ilustran con los propios fragmentos de código **XUL** (Tonymec [Mozilla Firefox], 2011), las técnicas seguidas con el fin de obtener la funcionalidad requerida para la aplicación.

Seguidamente se muestran pruebas del producto que demuestran que se han alcanzado los objetivos propuestos de la especificación de requisitos.

La documentación se cierra con un apartado de conclusiones y la relación bibliográfica consultada.

2 Descripción de la tecnología empleada

2.1 Mozilla XPCOM

"La arquitectura XPCOM es un entorno de desarrollo que permite a los programadores de aplicaciones desligarse de los modelos monolíticos de producción de software y llevar el desarrollo modular un paso más allá. Hasta el punto de interconectar objetos escritos en distintos lenguajes de programación". (Monzonís, 2008, p. 22).

XPCOM (hack.augusto [Mozilla Firefox], 2011) ofrece muchas de las funcionalidades útiles para el desarrollo de aplicaciones, como pueda ser Gestión de memoria, de componentes, Protocolos de red, etc. Nosotros nos centraremos en la funcionalidad que permite una interacción entre nuestra aplicación y el sistema de ficheros de la máquina local, que utilizaremos para el manejo del fichero que contiene las palabras y sus conceptos relacionados semánticamente, en el proceso de búsqueda.

2.1.1 Gecko

Gecko (Akilaa [Mozilla Firefox], 2011) es el motor de las aplicaciones Mozilla, que se encarga sobre todo de realizar el renderizado web, es decir, recorre el código no visible, (**HTML** (W3SCHOOLS, 2011) y **Javascript** (W3SCHOOLS, 2011)) de la web para proporcionar al usuario una representación visual de los elementos utilizados por el desarrollador web. Aunque se trata de un motor de renderizado web, también proporciona un framework para el desarrollo de aplicaciones basadas en **XPCOM**, que utilizará el sistema de componentes e interfaces para extender la estructura de **Gecko** y su funcionalidad.

Gecko además incluye un motor gráfico que renderiza los elementos y las ventanas de forma similar a como renderiza el HTML: Se utilizan documentos XML para especificar el contenido de la ventana y lo muestra.

2.1.2 Interfaces proporcionadas por XPCOM

2.1.2.1 nsIFile

Esta interfaz proporciona un método de representación del sistema de ficheros independiente de la plataforma utilizada por el usuario final.

Utilizando **nsIFile** (Madarche [Mozilla Firefox], 2011)podemos navegar a través del sistema de ficheros sin necesidad de utilizar los separadores de directorios específicos de cada sistema operativo, consultar el estado de cualquier fichero o directorio y crear, mover o copiar elementos en el sistema de ficheros.

Podremos recuperar un fichero **nsIFile** mediante la instanciación de un objeto (Trevorh [Mozilla Firefox], 2010)al que le pasaremos el path específico de una plataforma o utilizando localizaciones recuperadas por el componente "directory service" (usaremos la segunda).

Para realizar esta acción procederemos de la siguiente forma:

```
var file =
Components.classes["@mozilla.org/file/directory_service;1"].getService(Components.interfaces.nsIProperties).get("
ProfD", Components.interfaces.nsIFile);
```

Una vez obtenido el objeto nsIFile sobre el que realizaremos la búsqueda, utilizaremos los siguientes métodos (Madarche [Mozilla Firefox], 2011):

1. **append**: Este método es utilizado para definir el fichero al que se referirá el objeto nsIFile que creamos. (Este método no devuelve una copia del fichero, si no que realizará los cambios sobre el fichero referenciado).
   a) **Definición**: La estructura del método es la siguiente: **"void append(in AString node);"**
   b) **Parametros de entrada:**
      - **node:** nombre que estableceremos como referencia al fichero que pretendemos leer.
   c) **Ejemplo de utilización:**

```
file.append("index_M2.sense");
```

2.1.2.2 nsIScriptableUnescapeHTML

La interfaz **nsIScriptableUnescapeHTML** (Trevorh [Mozilla Firefox], 2010) proporciona la funcionalidad de transformar una cadena en texto plano a un objeto **HTML**.

En la implementación de nuestro proyecto nos será de utilidad para transformar el resultado proporcionado por la web de **Wordnet** (Princeton University, 2011)a un objeto HTML que podremos manejar mediante el uso de sentencias DOM.

El método en el que utilizamos el objeto es el siguiente:

```
function HTMLParser(aHTMLString){
  var html = document.implementation.createDocument("http://www.w3.org/1999/xhtml", "html", null);
    body = document.createElementNS("http://www.w3.org/1999/xhtml", "body");
  html.documentElement.appendChild(body);

  body.appendChild(Components.classes["@mozilla.org/feed-unescapehtml;1"]
  .getService(Components.interfaces.nsIScriptableUnescapeHTML).parseFragment(aHTMLString, false, null, body));

  return body;
}
```

Ilustración 1: Código HTMLParser

Una vez se obtiene el objeto **nsIScriptableUnescapeHTML** hemos utilizado el siguiente método:

1. **parseFragment**: Añade la cadena que se le pasa como parámetro a un objeto DOM existente.
   a) **Definición:**
      La estructura del método es la siguiente:

      "nsIDOMDocumentFragment parseFragment(in AString fragment, in PRBool isXML, in nsIURI baseURI, in nsIDOMElement element);"

   b) **Parámetros de entrada:**
      - **Fragment:** Puntero a la cadena que se añadirá al elemento.
      - **isXML:** Si el parámetro fragment se trata de un XML, se establece a true, en caso contrario a false.
      - **baseURI:** Referencia a la URI base que apuntan todas los enlaces incluidos en el fragment. Este fragmento se ignora si el parámetro isXML es false
      - **element:** Referencia al objeto DOM al que se añade el parámetro fragment.

   c) **Ejemplo de utilización:**
      En el fragmento de código anterior se podrá encontrar el ejemplo de utilización en nuestro proyecto.

2.1.2.3 *nsIFileOutputStream*

Esta interfaz (Sheppy [Mozilla Firefox], 2011) nos permite definir un objeto con el cual podremos escribir contenido en un fichero.

Para instanciar un objeto de este tipo procederemos de la siguiente forma:

```
var outputStream = Components.classes["@mozilla.org/network/file-output-stream;1"].createInstance(
Components.interfaces.nsIFileOutputStream );
```

Una vez hecho esto ya tenemos disponible el objeto del tipo **nsIFileOutputStream**, lo siguiente será especificar el fichero sobre el que escribiremos utilizando los distintos métodos.

1. **init**: Este es el método que enlaza el objeto actual con el nsIFile que le pasemos como parámetro, junto a los diferentes "flags" de control.
   a. **Definición:**

      La definición de este método es la siguiente:

      ```
      "void init(in nsIFile file, in long ioFlags, in long perm, in long behaviorFlags);"
      ```

   b. **Parámetros de entrada:**
      - **file:**Fichero que se leerá.
      - **ioFlags:** Define el modo de acceso del fichero.**Ilustración 2.**

Name	Value	Description
PR_RDONLY	0x01	Open for reading only.
PR_WRONLY	0x02	Open for writing only.
PR_RDWR	0x04	Open for reading and writing.
PR_CREATE_FILE	0x08	If the file does not exist, the file is created. If the file exists, this flag has no effect.
PR_APPEND	0x10	The file pointer is set to the end of the file prior to each write.
PR_TRUNCATE	0x20	If the file exists, its length is truncated to 0.
PR_SYNC	0x40	If set, each write will wait for both the file data and file status to be physically updated.
PR_EXCL	0x80	With PR_CREATE_FILE, if the file does not exist, the file is created. If the file already exists, no action and NULL is returned.

Ilustración 2: Modo de acceso a fichero

- **perm:** Permite seleccionar uno de los modos de acceso. **Ilustración 3**.

Name	Value	Description
PR_IRWXU	0700	read, write, execute/search by owner.
PR_IRUSR	0400	read permission, owner.
PR_IWUSR	0200	write permission, owner.
PR_IXUSR	0100	execute/search permission, owner.
PR_IRWXG	0070	read, write, execute/search by group
PR_IRGRP	0040	read permission, group
PR_IWGRP	0020	write permission, group
PR_IXGRP	0010	execute/search permission, group
PR_IRWXO	0007	read, write, execute/search by others
PR_IROTH	0004	read permission, others
PR_IWOTH	0002	write permission, others
PR_IXOTH	0001	execute/search permission, others

Ilustración 3: Permisos de acceso a fichero

- **behaviorFlags:** Especifican diferentes comportamientos de la clase.

c. **Ejemplo de utilización:**

```
outputStream.init( file, 0x04 | 0x10, 420, 0 );
```

2. **write:** Este es el método que realiza la escritura en el fichero. Copia los datos desde un buffer hacia el fichero.
   a. **Definición:**

```
unsigned long write(
in string aBuf,
in unsigned long aCount);
```

   b. **Parámetros de entrada:**

      - **aBuf:** Es el buffer que contiene los datos que se escribirán en el fichero.
      - **aCount:** Define el tamaño del buffer, o el máximo de bytes que se copiarán desde el buffer.

   c. **Parámetros de salida:**

      El método devuelve el número de bytes que han sido copiados.(debe de ser menor que el parámetro aCount).

   d. **Ejemplo de utilización:**

```
var result = outputStream.write( content, content.length );
```

3. **close:** Cierra el Stream, además limpia cualquier dato que pudiera quedar en el Stream de salida.
   a. **Definición:**

```
close();
```

   b. **Ejemplo de utilización:**

```
outputStream.close();
```

2.1.2.4 *nsIFileInputStream*

Esta interfaz (Philikon [Mozilla Firefox], 2011) nos permite definir un objeto con el cual podremos leer el contenido de un fichero. Para instanciar un objeto de este tipo procederemos de la siguiente forma:

```
var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
              createInstance(Components.interfaces.nsIFileInputStream);
```

Una vez hecho esto ya tenemos disponible el objeto del tipo **nsIFileInputStream**, con lo que el siguiente paso es utilizarlo sobre un fichero que hayamos definido. Con este cometido utilizaremos los siguientes métodos:

1. ***init**: Este es el método que enlaza el objeto actual con el **nsIFile** que le pasemos como parámetro, junto a los diferentes "flags" de control.*
   - ***a.* Definición*:***
     La definición de este método es la siguiente:
     "void init(in nsIFile file, in long ioFlags, in long perm, in long behaviorFlags);"
   - **b. Parámetros de entrada:**
     - **file:** Fichero que se leerá.
     - **ioFlags:** Define el modo de acceso del fichero. **Ilustración 4**.

Name	Value	Description
PR_RDONLY	0x01	Open for reading only.
PR_WRONLY	0x02	Open for writing only.
PR_RDWR	0x04	Open for reading and writing.
PR_CREATE_FILE	0x08	If the file does not exist, the file is created. If the file exists, this flag has no effect.
PR_APPEND	0x10	The file pointer is set to the end of the file prior to each write.
PR_TRUNCATE	0x20	If the file exists, its length is truncated to 0.
PR_SYNC	0x40	If set, each write will wait for both the file data and file status to be physically updated.
PR_EXCL	0x80	With PR_CREATE_FILE, if the file does not exist, the file is created. If the file already exists, no action and NULL is returned.

Ilustración 4: Modo de acceso a fichero

     - **perm:** Permite seleccionar uno de los modos de acceso especificados en la siguiente tabla. Ver **Ilustración 5**.

Name	Value	Description
PR_IRWXU	0700	read, write, execute/search by owner.
PR_IRUSR	0400	read permission, owner.
PR_IWUSR	0200	write permission, owner.
PR_IXUSR	0100	execute/search permission, owner.
PR_IRWXG	0070	read, write, execute/search by group
PR_IRGRP	0040	read permission, group
PR_IWGRP	0020	write permission, group
PR_IXGRP	0010	execute/search permission, group
PR_IRWXO	0007	read, write, execute/search by others
PR_IROTH	0004	read permission, others
PR_IWOTH	0002	write permission, others
PR_IXOTH	0001	execute/search permission, others

Ilustración 5: Permisos de acceso a fichero

     - **behaviorFlags:** Especifican diferentes comportamientos de la clase.
   - **c. Ejemplo de utilización:**

```
istream.init(file, 0x01, 0444, 0);
```

2. **readLine:** Lee hasta el siguiente salto de línea. Para ejecutar este método con éxito hay que realizar primero una consulta sobre el inputStream acerca de un objeto **nsILineInputStream** que permitirá realizar la lectura de una línea del inputStream.
   a. **Definición:**
      Boolean readLine(String aBuf);
   b. **Parámetros de Entrada:**
      - **aBuf:** Variable en la que se almacenará la línea leida.
   c. **valor de Salida:**
      Devuelve true si el fichero tiene más líneas.
   d. **Ejemplo de utilización:**
      hasmore = istream.readLine(line);
3. **close:** Cierra el Stream, además limpia cualquier dato que pudiera quedar en el Stream de salida.
   a. **Definición:**
      void close();

   b. **Ejemplo de utilización:**

      outputStream.close();

2.1.3 XUL

2.1.3.1 ¿Qué es XUL y porque fue creado?

XUL (Tonymec [Mozilla Firefox], 2011), fue creado con la intención de proporcionar una herramienta que permitiera un desarrollo mucho más rápido del navegador Mozilla. Es un lenguaje basado en XML, por lo que encontraremos características de este lenguaje entre las propias del lenguaje XUL.

Este lenguaje fue creado con la intención de realizar desarrollos de aplicaciones soportadas por diferentes plataformas de forma mucho más rápida y fácil.

2.1.3.2 Diseño de interfaces de usuario con XUL

XUL permite la utilización de los distintos elementos que aparecen en las Interfaces de usuario actuales.

Algunos elementos que podemos encontrar en aplicaciones desarrolladas con XUL son los siguientes:

- Controles de entrada (Cuadros de texto y cajas de chequeo).
- Barra de herramientas.
- Menús desplegables o emergentes.
- Cuadros de dialogo.
- Árbol de información Jerárquica o Tabulada.
- Teclas de accesos directos.

En el proyecto, el contenido mostrado será cargado desde el contenido de un archivo XUL.

Existen distintos tipos de aplicaciones XUL, como pueden ser:

- **Extensión de Firefox**: una extensión amplía las características y funciones que encontramos en el navegador, como pueden ser la barra de herramientas suplementaria, menús contextuales, etc. Para conseguir eso se utiliza una funcionalidad XUL llamada *overlay*. Esta característica hace posible la integración de las interfaces capaces de mostrar la extensión, con el propio navegador Mozilla Firefox. Las extensiones podrían llegar a ser instaladas en otros productos de Mozilla, tal como el gestor de correo Thunderbird.
- **Aplicación XULRunner autónoma**: XULRunner es una versión empaquetada de una plataforma Mozilla que permite la creación de aplicaciones en lenguaje XUL, por lo que podrán ser ejecutadas en multiples plataformas de forma independiente a cualquier navegador web.
- **Paquete de XUL**: Al igual que las aplicaciones XULRunner no es necesario un navegador que las contenga, ejecutándose de forma separada al navegador, pero utilizando el mismo método de desarrollo que las extensiones.
- **Aplicación XUL remota**: Al igual que hacemos cuando desarrollamos una página web, se pueden ubicar en la web ficheros con código XUL que posteriormente se podrá visualizar utilizando el navegador. Sin embargo este método es limitado, por razones de seguridad sobre que acciones se permiten, como la apertura de otras ventanas.

Los tres primeros tipos necesitan cada uno su instalación en la máquina del usuario. Por lo cual, estos tipos de aplicaciones no tienen restricciones de seguridad, pueden acceder a los archivos locales y leer y escribir las preferencias, por ejemplo. El contenido XUL es normalmente cargado desde un paquete instalado dentro de Mozilla. Los archivos XUL, "scripts" asociados e imágenes de una aplicación podrán ser empaquetados dentro de un simple archivo, descargados e instalados por el usuario. Mozilla proporciona una manera

de tener estos paquetes instalados y registrados sin tener que escribir un montón de códigos complejos. Además, estos paquetes pueden incorporarse al navegador u otras aplicaciones para adicionarles características; esta es la forma en que trabajan las extensiones de Firefox.

También es posible abrir directamente archivos XUL desde el sistema de archivos o desde un sitio web remoto, sin embargo ellos estarán restringidos en el tipo de operaciones que puedan hacer, y algunas características de XUL no trabajarán. En cambio, si Ud. quiere cargar el contenido XUL desde un sitio remoto, el servidor web debe estar configurado para enviar archivos XUL con contenidos de tipo 'application/vnd.mozilla.xul+xml'. El XUL es usualmente almacenado en archivos con una extensión .xul. Ud. puede abrir un archivo XUL con Mozilla tal como abre otros archivos, usando el comando Abrir Archivo desde el menú de Archivo o escribiendo la URL en la barra de dirección.

2.1.3.3 Estructura XUL

Habitualmente existen tres partes distintas en un paquete chrome, aunque son opcionales. Cada parte está almacenada en una carpeta distinta. Estas tres partes son el contenido, apariencia (skin) y la configuración regional, explicados más abajo. Un paquete de instalación debe proporcionar uno o más apariencias y configuraciones locales, pero un usuario puede reemplazarlos con los suyos propios. El sistema de paquetes es suficientemente flexible por lo que puedes incluir todas las partes que necesites y permitir que otras partes, como el texto para diferentes idiomas, sean descargadas de forma separada.

Los tres tipos de paquetes chrome son:

- **Paquetes de Contenido:** Entre sus directorios se incluyen los formularios que se mostraran al usuario y ficheros que contienen todas las funciones **Javascript** necesarias para el funcionamiento de la aplicación. Los formularios o ventanas de la interfaz de usuario se almacenan en archivos XUL, que tienen extensión .XUL. Un paquete de contenido puede almacenar múltiples ficheros XUL, pero la ventana principal debe tener el mismo nombre de archivo que el nombre del paquete. Por ejemplo, el paquete 'slicetoolbar' tendrá un archivo que se llamará slicetoolbar.xul. Por otra parte las funciones **Javascript** se encontrarán incluidas en ficheros con extensión .js y normalmente aparecerán junto a los ficheros que definen la estructura de los formularios y ventanas que serán mostradas al usuario.
- **Aspecto (skin)**: Este tipo de paquete incluye Hojas de estilo, imágenes y otros archivos específicos al tema. Las hojas de estilo se especifican en ficheros **CSS** y son las que contienen información sobre el diseño o apariencia de una ventana. Se almacenan de forma separada a los archivos **XUL** para facilitar modificar el aspecto (skin o tema) de una aplicación. Algunas imágenes usadas también se almacenan aquí.
- **Configuración regional:** Son archivos de configuración regional. En estos ficheros se incluyen, para cada idioma soportado por la aplicación, el texto que debe aparecer en los formularios y ventanas, con un identificador único en la aplicación. De esta forma se puede hacer referencia a ese identificador desde los ficheros .XUL. De esta forma en función del idioma predefinido en el navegador cliente se utilizará una configuración local u otra.

2.1.3.3.1 Paquetes de contenido

El nombre de un archivo JAR debe describir lo que contiene, pero no se puede asegurar el contenido si no se mira dentro. Como ejemplo para hacer una descripción más visual vamos a utilizar el paquete incluido en nuestro proyecto. Si extraemos los archivos de slicetoolbar.jar, nos encontraremos con una jerarquía de directorios entre los que se encuentra el directorio content. **Ilustración 6**.

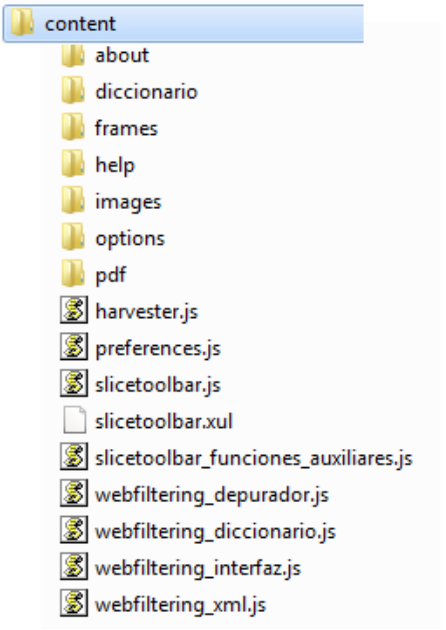


Ilustración 6: Directorio content.

La carpeta content contiene un número de archivos con extensiones .xul y .js. Los archivos XUL son los que tienen la extensión .xul. Los archivos con extensiones .js son archivos JavaScript que contienen los scripts que se encargan de la funcionalidad de una ventana. Muchos archivos XUL tienen un archivo script asociado con ellos, y muchos deben tener más de uno.

En el listado superior, dos archivos han sido vistos. De hecho hay otros, pero para simplificar no serán vistos. El archivo slicetoolbar.xul es el archivo XUL que describe la ventana principal de la extensión Firefox que estamos desarrollando. La ventana principal para un paquete de contenido debe tener el mismo nombre que el paquete con una extensión .xul. En este caso, el nombre del paquete es "slicetoolbar.jar", por eso esperaremos encontrarnos con slicetoolbar.xul.

Otros archivos XUL que encontraremos en subdirectorios de content describen ventanas distintas. Por ejemplo, el archivo diccionario.xul contenido en el subdirectorio “content\diccionario\“ describe el cuadro de dialogo de configuración del diccionario semántico que integraremos con la implementación existente.

Muchos paquetes incluirán un archivo contents.rdf que describe el paquete, su autor y el revestimiento que usa. Sin embargo, este archivo está obsoleto y ha sido reemplazado con un mecanismo más simple. Este nuevo método es el archivo manifest mencionado anteriormente, y encontraremos estos archivos con la extensión .manifest en el directorio chrome. Por ejemplo, browser.manifest describe el paquete del navegador.

2.1.3.3.2 Aspectos (skins) o Temas

El siguiente directorio que encontramos en el paquete slicetoolbar.jar es el que incluye información sobre el aspecto (skin) de la extensión, en este directorio se incluyen todas las imágenes que se muestran en la aplicación. Ilustración 7: Directorio Skin.

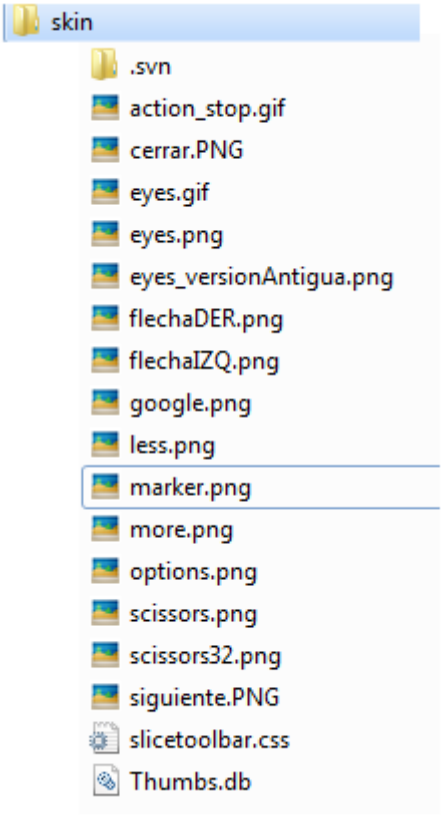


Ilustración 7: Directorio Skin.

Una piel (skin) se compone de archivos CSS y de un número de imágenes usadas para definir el aspecto y la interfaz. El archivo slicetoolbar.css es usado por slicetoolbar.xul y contiene estilos que definen el aspecto de varias partes de la interfaz del navegador. Nuevamente, nótese como el archivo **slicetoolbar.css** tiene el mismo nombre que el paquete. Cambiando los archivos CSS, puedes ajustar el aspecto de una ventana sin cambiar su función. De esta forma puedes crear un nuevo tema. La parte **XUL** continúa igual pero la parte de la piel (el skin) cambia independientemente.

2.1.3.3.3 Configuración regional

El directorio “locale” contiene la información para cada uno de los idiomas con los que es compatible la extensión. Cada idioma contendrá archivos que especifican texto usado por el paquete pero para un idioma concreto. Ilustración 8: Directorio locale.

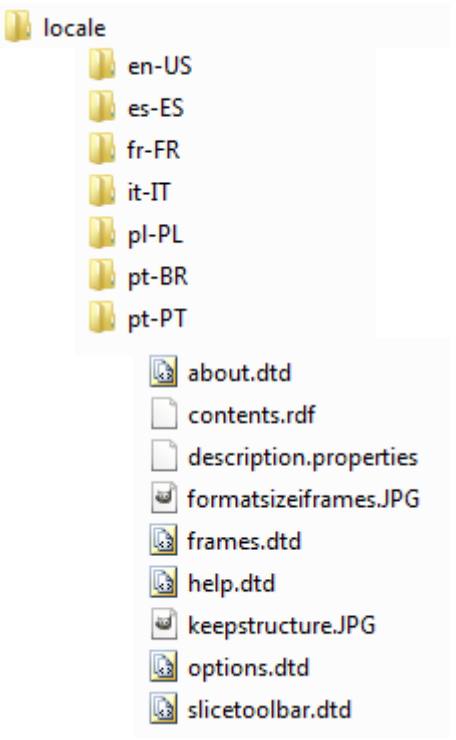


Ilustración 8: Directorio locale.

El texto de idioma es almacenado en dos tipos de archivos: archivos DTD y archivos de propiedades. Los archivos DTD tienen una extensión .dtd y contienen declaraciones de identidad, una para cada cadena de texto usada en una ventana. Por ejemplo, el archivo slicetoolbar.dtd contiene declaraciones de identidad para cada comando del menú. Además, los atajos de teclado para cada comando también están definidos, ya que pueden ser diferentes en cada idioma. Los archivos DTD son usados por los archivos XUL por eso, en general, tendrás uno por cada archivo XUL. La parte de configuración regional también contiene archivos de propiedades, que son similares, pero son usados por archivos script.

Esta estructura nos permite traducir nuestra extensión a un idioma distinto sólo añadiendo una nueva configuración regional para ese idioma. No tienes que cambiar la parte de XUL. Además, otra persona puede proporcionar un paquete separado que aplica una piel (skin) o configuración regional a tu parte de contenido, proporcionando soporte de esta manera a un nuevo tema o idioma sin tener que modificar el paquete original.

2.1.3.4 Paquetes

Un paquete es un conjunto de ficheros XUL y scripts que definen la funcionalidad de una interfaz de usuario. Los paquetes se pueden instalar dentro de Mozilla y referenciar con una URL del chrome. Un paquete puede contener cualquier tipo de ficheros que pueden estar separados en distintos subdirectorios dentro del paquete. Un paquete puede estar contenido como un directorio o como un fichero JAR.

2.1.3.5 Archivos Manifest

Un archivo manifest describe un paquete y mapea su localización en disco a una URL. El archivo manifest en el directorio chrome se examinará al iniciarse la aplicación Mozilla para comprobar los paquetes instalados. Esto significa que todo lo que necesitamos hacer para instalar un nuevo paquete es añadir un nuevo archivo manifest en el directorio chrome de la aplicación o en el directorio chrome específico del usuario. Este último es el utilizado preferentemente, dado que es posible que no tengamos suficientes permisos para escribir en el directorio de la aplicación.

El primer campo 'content' indica un paquete contenido. Para los temas utilizamos 'skin', mientras que para la configuración regional utilizamos 'locale'.

El campo final es la ruta donde se localiza el archivo. Esta puede ser una ruta de archivo local utilizando una URL de archivo, o un archivo JAR utilizando una URL jar, la cual describiremos a continuación. Puede especificar múltiples paquetes incluyendo otras líneas en el fichero manifest.

El fichero chrome.manifest usado por nuestra extensión es el que se puede ver en la Ilustración 9.

```
content     slicetoolbar                chrome/content/
skin        slicetoolbar    classic/1.0 chrome/skin/
locale      slicetoolbar    en-US       chrome/locale/en-US/
locale      slicetoolbar    es-ES       chrome/locale/es-ES/
locale      slicetoolbar    fr-FR       chrome/locale/fr-FR/
locale      slicetoolbar    it-IT       chrome/locale/it-IT/
locale      slicetoolbar    pl-PL       chrome/locale/pl-PL/
locale      slicetoolbar    pt-BR       chrome/locale/pt-BR/
locale      slicetoolbar    pt-PT       chrome/locale/pt-PT/


overlay   chrome://browser/content/browser.xul   chrome://slicetoolbar/content/slicetoolbar.xul
overlay   chrome://browser/content/browser.xul   chrome://content/options/options.xul
```

Ilustración 9: Fichero chrome.manifest

En el fichero se definen las referencias a las distintas partes del paquete slicetoolbar.jar del que hemos hablado con anterioridad. Al directorio content y al directorio skin que contienen el contenido y el layout que tendrá la aplicación así como a los distintos idiomas que son soportados por nuestra extensión.

A las líneas anteriores se les ha añadido un campo extra para indicar que tanto el tema, como la configuración regional se aplican al navegador. El nombre del tema es 'classic/1.0'. En este caso se usa un número de versión como parte del nombre del tema pero sería opcional en caso de que se necesitara crear un nuevo tema.

Además se están especificando dos recubrimientos (overlays), lo cual permite combinar contenidos de distintos paquetes. Las extensiones harán mayor uso de los recubrimientos (overlays), dado que ellas fusionan su UI con la del navegador.

2.1.3.6 Instalación de un paquete

Para instalar una aplicación, se necesita crear un instalador para ella o incluirlo como parte de otra aplicación.

El método usado depende del tipo de desarrollo que se lleve a cabo. Para las extensiones, se necesita incluir en el paquete de instalación un fichero llamado install.rdf, que contiene información sobre la extensión desarrollada. La información contenida en este fichero se refiere al autor y colaboradores en el desarrollo de la extensión, aunque también incluye las versiones del navegador mínimas y máximas con las que es compatible la extensión, así como información referente a la aplicación.

También se necesita una estructura de directorios específica ya que las extensiones están limitadas a donde los ficheros deben de ser instalados. Una extensión es empaquetada en un fichero XPI. XPI es la abreviatura de **XPInstall** y es usada por Mozilla para instalar componentes. Al igual que los ficheros JAR, un fichero XPI sólo es un fichero ZIP al que se le ha cambiado la extensión por lo que puedes crear y ver los ficheros contenidos en un paquete XPI con cualquier herramienta ZIP.

El administrador de extensiones de Mozilla Firefox maneja automáticamente las extensiones instaladas empaquetadas en ficheros XPI.

A continuación se muestra una plantilla para este tipo de ficheros:

```
<?xml version="1.0"?>

<RDF:RDF xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:chrome="http://www.mozilla.org/rdf/chrome#">

  <RDF:Seq about="urn:mozilla:package:root">
    <RDF:li resource="urn:mozilla:package:mi_aplicacion"/>
  </RDF:Seq>

  <RDF:Description about="urn:mozilla:package:mi_aplicacion"
         chrome:displayName="titulo"
         chrome:author="autor"
         chrome:name="mi_aplicacion"
         chrome:extension="true"/>

</RDF:RDF>
```

Ilustración 10: Fichero Install.rdf

Y ahora se muestra el fichero install.rdf que se utilizará en el software desarrollado:

```
<?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">
    <em:id>filteringtoolbar@foo.net</em:id>
    <em:version>1.5</em:version>
    <em:type>2</em:type>

    <!-- Target Application this extension can install into,
         with minimum and maximum supported versions. -->
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>1.5</em:minVersion>
        <em:maxVersion>5.0.*</em:maxVersion>
      </Description>
    </em:targetApplication>

    <!-- Front End MetaData -->
    <em:name>Web Filtering Toolbar</em:name>
    <em:description>A Toolbar to Filter Webpages.</em:description>
    <em:creator>Josep Silva</em:creator>
    <em:contributor>Mercedes Garcia</em:contributor>
    <em:contributor>Tatiana Tomas</em:contributor>
    <em:contributor>Sergio Lopez</em:contributor>
    <em:contributor>Carlos J. Castillo</em:contributor>
    <em:contributor>Hector Valero</em:contributor>
    <em:contributor>Domingo Morello</em:contributor>
    <em:homepageURL>http://www.dsic.upv.es/~jsilva/webfiltering</em:homepageURL>
    <em:optionsURL>chrome://slicetoolbar/content/options/options.xul</em:optionsURL>
    <em:iconURL>chrome://slicetoolbar/content/images/scissors32.png</em:iconURL>
  </Description>
</RDF>
```

Ilustración 11: Fichero Instal.rdf de la extensión

Una vez tenemos toda la estructura mencionada en este apartado, la empaquetaremos en un fichero .xpi. Para ello basta con comprimir los directorios chrome, defaults y los ficheros chrome.manifest e install.rdf en un mismo fichero. En el caso que nos ocupa se ha utilizado el compresor 7zip.

1. Se seleccionan los ficheros y se comprimen utilizando la herramienta nombrada anteriormente.

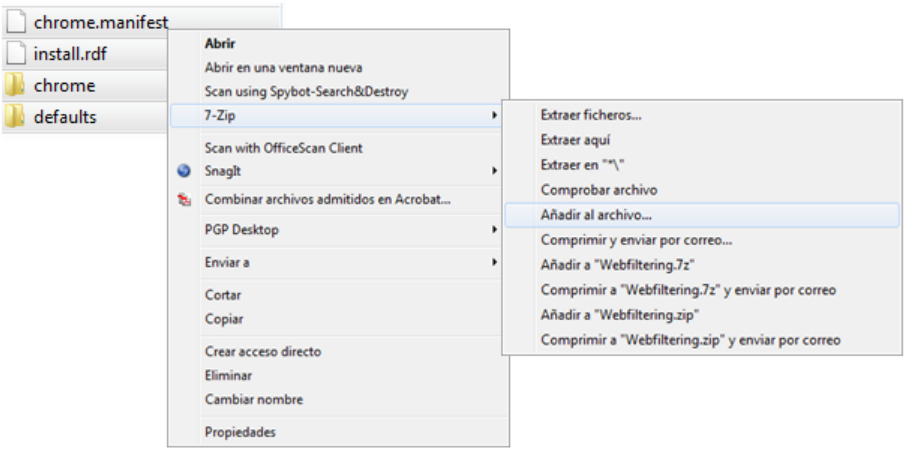


Ilustración 12: Directorios incluidos en el paquete .xpi

2. Seleccionamos el nombre que queremos para el paquete de instalación. En nuestro caso filteringtoolbar@foo.net.xpi:

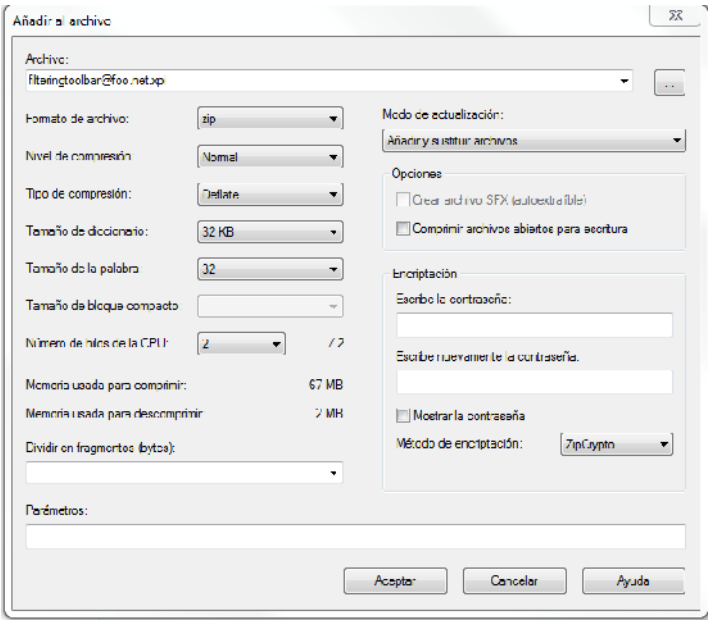


Ilustración 13: Creación paquete filteringtoolbar.xpi

Una vez realizado los pasos anteriores se arrastrará el fichero generado sobre el navegador firefox en el que comenzará el proceso de instalación.

2.2 Document Object Model (DOM)

En castellano podríamos traducirlo por Modelo de Objeto de Documento, los objetos del DOM modelan tanto la ventana del navegador como el historial, el documento o página web, y todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos, etc.

A través del DOM se puede acceder, utilizando las funciones adecuadas de Javascript, a cualquiera de estos elementos, es decir a sus objetos para alterar sus propiedades o invocar a sus métodos. Cualquier elemento de la página queda disponible, para ser modificado, suprimido, crear nuevos elementos y colocarlos en la página, etc.

El DOM está definido y administrado por el W3C, por lo que los distintos navegadores simplemente aplican las especificaciones del World Wide Web Consortium (W3.ORG, 2004), para dar soporte al DOM en sus aplicaciones. El DOM no sólo permite modificar páginas web en HTML, sino también documentos XML.

A lo largo de la historia de los navegadores, se han ido aplicando en mayor o menor manera las características del DOM. A medida que se sucedían versiones de los navegadores también se iba dando un mayor soporte a las especificaciones del DOM, en lo que se han llamado los niveles del DOM. El primero que empezó a dejar disponibles por medio de objetos los componentes de la página fue Netscape 2.0, que incorporaba lo que se llama el DOM nivel 0. Actualmente, la última especificación publicada es DOM nivel 4.

Es importante destacar ahora que, dado que los niveles del DOM cambian de versión a versión del navegador y que las especificaciones se han entendido de manera distinta por las distintas organizaciones creadoras de los navegadores, se ha producido un marco donde trabajar con los objetos de la página difiere de un navegador a otro.

En el caso concreto de nuestro proyecto el hecho de que DOM cambie entre navegadores no nos afecta, puesto que nuestra extensión será usada en Mozilla Firefox, por lo que nos ceñiremos a utilizar los métodos que sean válidos en este navegador.

2.2.1 Métodos de Acceso a DOM

A continuación mostraremos algo de documentación sobre los métodos de DOM (W3.ORG, 2004) que hemos utilizado en la parte del proyecto que hemos desarrollado.

Los elementos que podemos encontrar en un documento HTML se encuentran vinculados a un tipo de objeto en DOM que implementan distintas interfaces entre las que se encuentran las que enumeraremos a continuación:

- **Interfaz Node:** La interfaz Node es el tipo de dato primario para la integridad del Modelo de Objetos del Documento. Representa un nodo individual del árbol del documento. Mientras que todos los objetos que implementan la interfaz Node exponen métodos para tratar con hijos, no todos los objetos que implementan la interfaz Node pueden tener hijos. Por ejemplo, los nodos Text no pueden tener hijos, y al añadir hijos a tales nodos provoca una excepción DomException.

  Los atributos nodeName, nodeValue y attributes se han incluido como un mecanismo para obtener información del nodo sin destruir la interfaz específica derivada. En los casos donde no hay un mapa obvio de estos atributos para un nodeType específico (Por Ejemplo, nodeValue para un Element o attributes para un Comment), esto devuelve null. Observe que las interfaces especializadas pueden contener mecanismos adicionales y convenientes para obtener y establecer la información relevante.

- **Interfaz Element:** La interfaz Element representa un elemento en un documento HTML o XML. Los elementos pueden tener atributos asociados a ellos; como la interfaz Element se hereda de Node, puede utilizarse el atributo de la interfaz genérica Node attributes para obtener el conjunto de todos los atributos de un elemento. Existen métodos en la interfaz Element para obtener o bien un objeto Attr por su nombre o bien un valor de atributo por su nombre. En XML, en el cual un valor de atributo puede contener referencias a entidades, debería obtenerse un objeto Attr para examinar el posiblemente complejo sub-árbol que representa el valor del atributo. Por otra parte, en HTML, en el cual todos los atributos tienen valores de cadenas simples, pueden emplearse con seguridad métodos más convenientes para acceder directamente al valor de un atributo.
- **Interfaz document:** Representa el documento HTML o XML completo. Conceptualmente, es la raíz del árbol del documento, y proporciona el acceso primario a los datos del documento. Como los elementos, nodos de texto, comentarios, instrucciones de procesamiento, etc., no pueden existir fuera del contexto de un Document, la interfaz Document también contiene los métodos constructores necesarios para crear estos objetos. Los objetos Node creados tienen un atributo ownerDocument que los asocia con el Document dentro de cuyo contexto fueron creados.

2.2.1.1 Algunos Metodos de document

- ***getElementById(Identificador):***
  Devuelve el nodo Elemento cuyo id es el Identificador que se le pasa como parámetro.
  - ***Ej:*** *document.getElementById("cuadroblanco")*
- ***getElementsByTagName(Nombre):***
  Devuelve una lista ordenada con todos los nodos Element que tengan como nombre de etiqueta el Nombre que se pasa como parámetro.
  - ***Ej:*** *var listaLI=DOMPars.getElementsByTagName('li');*
- ***createElement(Tipo):***
  Crea un nodo tipo Element del tipo especificado en Tipo.
  - ***Ej:*** *webfiltering_button=document.createElement("button");*
- ***createTextNode(cadena):***
  Crea un nodo tipo Text con el valor Cadena
  - ***Ej:*** *webfiltering_button_txt=document.createTextNode("Aceptar");*

2.2.1.2 Algunos Metodos de Node

- ***appendChild(nuevoNodo):***
  Añade el nodo que se le pasa como parámetro al final del conjunto de hijos del nodo al que se aplica.
  - ***Ej:*** *webfiltering_button.appendChild(webfiltering_button_txt);*
- ***removeChild(Nodo):***
  Retira el nodo que se le pasa como parámetro del conjunto de hijos del nodo al que se aplica.
  - ***Ej:*** *div.parentNode.removeChild(div);*

2.2.1.3 Metodo de Element

- ***setAttribute(Nombre,Valor):***
  Añade al elemento un nuevo atributo Nombre, estableciendo Valor.
  - ***Ej:*** *webfiltering_button.setAttribute("type","button");*

2.2.1.4 Metodo de HTMLInputElement

- ***value:***
  Propiedad que almacena el contenido del control de formulario correspondiente.
  - ***Ej:*** *document.getElementById("cuadroblanco").value+=… ("cuadroblanco").value, velocidad);*

2.3 JAVASCRIPT

Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, es un lenguaje que utiliza Windows y sistemas X-Windows, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

2.3.1 ¿Cómo nace Javascript?

Javascript fue concebido para ofrecer a los desarrolladores de webs de la década de los 90, la posibilidad de intercambiar datos con los visitantes de la web. Fue el primer paso hacia las páginas webs dinámicas tal como las conocemos hoy en día.

"El lenguaje creado entre las paredes de la compañía Netscape ha sido denominado de múltiples formas a lo largo de su historia. En un primer momento su creador lo denominó "Mocha", para posteriormente cambiar de nombre sobre el año 1995 coincidiendo con el momento en que Netscape agrega soporte para la tecnología Java a su navegador web Netscape Navigator en su versión 2.0B3. Este cambio de nombre se dice que fue fruto de una estrategia para obtener mayor prestigio por parte de la compañía". (Movses-bot, 2011)

"En diciembre de 1995, Netscape y Sun Microsystems se unen con el objetivo de colaborar en el desarrollo de este lenguaje, renombrando el lenguaje como Javascript. En respuesta a la popularidad de Javascript, Microsoft lanzo su propio lenguaje de programación a base de script, VBScript (una pequeña versión de Visual Basic). En el año de 1996 Microsoft se interesa por competir con Javascript por lo que lanza su lenguaje llamado Jscript, introducido en los navegadores de Internet Explorer. A pesar de las diferentes críticas que se le hacen al lenguaje Javascript, este es uno de los lenguajes de programación más populares para la web. Desde que los navegadores incluyen el Javascript, no necesitamos el Java Runtime Environment (JRE), para que se ejecute". (Valdés, 2007)

El lenguaje que estamos describiendo se ha convertido en imprescindible a la hora de realizar cualquier desarrollo web, y en la actualidad es considerado como el primer escollo hacia la nueva generación de aplicaciones web dinámicas. Esto ha sido causado a raíz de su estandarización por parte de la "ECMA"(European Computer Manufacturers Association) (ecma-international, 2011)y W3C DOM (W3.ORG, 2004).

En la época que vive la web, cada vez está más extendido el uso de este lenguaje en aplicaciones web, haciendo cada vez más raro encontrar una web que no incluya líneas de código en este lenguaje. Podemos encontrar **Javascript** (W3SCHOOLS, 2011) en distintos elementos de una web, como pueden ser menús desplegables, contadores de visitas, calendarios, etc.

2.3.2 ¿Cómo identificar código Javascript?

Generalmente el código **Javascript** lo podemos encontrarlo en la estructura de un documento HTML, y normalmente se inserta entre etiquetas <script></script>, normalmente ubicados en la cabecera del documento etiquetada como <head></head>.

Las funciones escritas en **Javascript** se pueden encontrar definidas en el mismo documento HTML o pueden ser referenciadas en el documento y estar ubicadas en otro fichero con extensión ".js". Para referenciar este tipo de ficheros desde un documento **HTML** se utilizará la siguiente estructura de la etiqueta <script>

```
<script type="text/javascript" src="micodigo.js"></script>
```

Su sintaxis es similar a la usada en Java y C, al ser un lenguaje que se ejecuta en el navegador del usuario, es el propio navegador el que debe dar soporte para ejecutar las funciones definidas en la web.

Podemos conocer más características del lenguaje consultando su material de referencia. (W3SCHOOLS, 2011)

2.3.3 ¿Es compatible con navegadores?

La mayoría de navegadores existentes dan soporte al lenguaje Javascript, Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

Con el surgimiento de lenguajes como PHP del lado del servidor y Javascript del lado del cliente, surgió Ajax en acrónimo de (Asynchronous Javascript And XML). El mismo es una técnica para crear aplicaciones web interactivas. Este lenguaje combina varias tecnologías:

- HTML y Hojas de Estilos CSS para generar estilos.
- Implementaciones ECMAScript, uno de ellos es el lenguaje Javascript.
- XMLHttpRequest es una de las funciones más importantes que incluye, que permite intercambiar datos asincrónicamente con el servidor web, utilizando lenguajes del lado del servidor.

Debemos tener en cuenta que aunque Javascript sea soportado en gran cantidad de navegadores, los usuarios que acceden a la web, pueden elegir la opción de Activar/Desactivar el Javascript en los mismos, por lo que los desarrolladores deberían proporcionar un método de acceso a la visualización de la web alternativo independientemente de si está activada o desactivada esta funcionalidad en el navegador de los visitantes de la web. (Valdés, 2007)

2.4 XMLHttpRequest

Este objeto (SUBGURIM.NET, 2006) es utilizado para cambiar información con el servidor, de su uso se desprenden múltiples posibilidades entre las que se encuentran:

- Actualizar la página sin necesidad de recargar toda la página.
- Realizar peticiones al servidor después de haber cargado la página.
- Recibir información del servidor después de recargar la página.
- Enviar información al servidor en segundo plano.

2.4.1 Como utilizar el objeto XMLHttpRequest

Para explicar cómo debemos utilizar este tipo de objeto dividiremos el proceso en 4 pasos:

2.4.1.1 Crear el objeto XMLHttpRequest:

Como casi siempre en javascript, el XMLHttpRequest se obtiene de manera distinta para los distintos navegadores, pero en el caso que nos ocupa el desarrollo está destinado a ser usado en el navegador Firefox, por lo que la forma de obtener el objeto será la siguiente:

```
var xmlHttp;
// 1.- Creamos el objeto xmlHttpRequest
if(!xmlHttp && typeof XMLHttpRequest != "undefined")
{
   xmlHttp = new XMLHttpRequest();
}
```

Ilustración 14: Objeto XMLHttpRequest

2.4.1.2 *Enviar la Información*

En este paso lo que haremos es enviar la información del cliente al servidor:

```
// 2.- Definimos la llamada para hacer un simple GET.
var ajaxRequest = 'http://wordnetweb.princeton.edu/perl/webwn'

// 3.- Marcar qué función manejará la respuesta
//xmlHttp.onreadystatechange = recogeInfo;

// 4.- Enviar
xmlHttp.open("GET", ajaxRequest, false);
xmlHttp.send("");
```

Ilustración 15: Objeto XMLHttpRequest

Como vemos en el código anterior, lo que estamos haciendo es:

- Asignar la URL donde mandaremos la petición y guardar en su QueryString la información que vamos a mandar.
- Enviamos la información, con el comando **xmlHttp.open("GET", ajaxRequest, false)** lo que indica que utilizaremos el método GET, enviaremos la información a la URL definida y elegimos que se comporte de modo síncrono. Para que se comporte de forma asíncrona hay que marcar true en el parámetro correspondiente.

2.4.1.2.1 Tratar la información en el servidor

La información que enviamos a la URL será tratada por el servicio que proporciona Wordnet por el que enviamos una palabra y nos devolverá conceptos relacionados semánticamente con ella.

2.4.1.2.2 Recoger la información en cliente:

Una vez que el servidor concluye el envío de la respuesta a nuestra petición, continuará la ejecución de nuestro método, que continúa con el tratamiento de la información recibida, para ello utilizamos la propiedad **responseText** del objeto XMLHttpRequest y comenzará el tratamiento que queramos dar a la información recibida.

```
var DOMPars = HTMLParser(xmlHttp.responseText);
```

Ilustración 16: Objeto XMLHttpRequest

2.5 WORDNET

Wordnet (Princeton University, 2011) es un sistema online de referencia léxica cuyo diseño está inspirado por las teorías psicolingüísticas actuales acerca de la memoria léxica humana.

Sustantivos, verbos, adjetivos y adverbios están organizados en conjuntos de sinónimos, cada uno de los cuales representa un concepto léxico subyacente. Dichos conjuntos están unidos por diferentes relaciones.

WordNet fue desarrollada por el Laboratorio de Ciencias Cognitivas de la Universidad de Princeton bajo la dirección del Profesor George A. Miller (Investigador Jefe). A lo largo de los años mucha gente ha contribuido al éxito de WordNet. Se empezó a desarrollar en 1985. A lo largo de dos años, el proyecto recibió alrededor de 3 millones de dólares en donaciones, principalmente de agencia gubernamentales interesadas en la traducción automática. Se trata de un diccionario semántico para la lengua inglesa. Agrupa palabras inglesas en grupos de sinónimos llamados synsets, provee definiciones cortas, y almacena las distintas relaciones semánticas entre estos grupos de sinónimos. El propósito es doble, producir una combinación de diccionario y de tesauro que se pueda utilizar de una forma más intuitiva, y con el objetivo de soportar el análisis automático de textos de las aplicaciones de inteligencia artificial. La base de datos y las herramientas software son públicas bajo una licencia de tipo BSD y pueden ser descargadas y distribuidas libremente. La base de datos también puede ser consultada en línea. En el año 2005, la base de datos contenía 150000 palabras organizadas en 115000 synsets para un total de 203000 pares con significado; comprimidos ocupan alrededor de 12 megabytes de tamaño.

WordNet distingue entre nombres, verbos, adjetivos y adverbios asumiendo que estos son alojados en el cerebro humano de una forma distinta. Cada synset contiene un grupo de palabras sinónimas o colocaciones (una colocación es una secuencia de palabras que van juntas para formar un significado especifico, tal como “préstamo de coche”); las palabras, típicamente, participan en varios synsets. El significado de los synsets se encuentra posteriormente clarificado con una serie de frases que lo definen.

Un synset típico de ejemplo con estas frases es (Troyano, 2002):

> *bueno, correcto, oportuno -- (lo más adecuado o correcto para un propósito particular; "un buen momento para plantar tomates"; "el momento correcto para actuar"; "el momento es oportuno para grandes cambios sociales")*

Cada synset está conectado a otros synsets a través de varias relaciones. Estas relaciones pueden variar dependiendo del tipo de palabra:

- **Sustantivos**:
  - **Sinónimos**: synsets con significados similares
  - **Hiperónimos**: Y es un hiperónimo de X si cada X es un (tipo de) Y o hipónimos: Y es un hipónimo de X si cada Y es un (tipo de) X
  - **Términos coordinados**: Y es un termino coordinado de X si X e Y comparten un hiperónimo
  - **Holónimo**: Y es un holónimo de X si X es parte de Y
  - **Merónimo**: Y es un merónimo de X si Y es parte de X
- **Verbos**:
  - **Sinónimos**
  - **Hiperónimo**: el sustantivo Y es un hiperónimo del verbo X si la actividad X es un (tipo de) Y.
  - **Términos coordinados**: aquellos verbos compartiendo un hiperónimo.
- **Adjetivos**:
  - Sinónimos y sustantivos relacionados.
  - Antónimos: adjetivos de significado opuesto.
- **Adverbios**:
  - **Sinónimos** y adjetivos raíz.
  - **Antónimos.**

WordNet también proporciona el contador polisémico de una palabra: el número de synsets que contienen esa palabra. Si una palabra está presente en varios synsets (por ejemplo, tiene varios significados), entonces típicamente algunos significados son más comunes que otros. **WordNet** contabiliza esto a través de la puntuación de frecuencia: en algunos textos de ejemplo todas las palabras fueron semánticamente anotadas en el synset correspondiente, además de contabilizar con qué frecuencia una palabra aparecía con un significado definido.

La interfaz de la base de datos es capaz de deducir la forma raíz de una palabra a partir de la entrada del usuario; solamente la forma principal se almacena en la base de datos. (Troyano, 2002).

3 DESARROLLO DEL PROYECTO

3.1 Definición de Requisitos

3.1.1 Requisitos Externos

Número de requisito	RE0		
Nombre de requisito	Formulario de Opciones de Diccionario.		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Formulario que contendrá los elementos que seleccionarán las funcionalidades del software.		

Número de requisito	RE1		
Nombre de requisito	Agrupación de Estado.		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Agrupación de las opciones de Activación/Desactivación del diccionario, cuya etiqueta será “Estado”. Aparecerá contenida en el formulario especificado en la RE0.		

Número de requisito	RE2		
Nombre de requisito	Etiqueta de Activación del Diccionario.		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE1. En esta etiqueta aparecerá el texto “Activo”.		

Número de requisito	RE3
Nombre de requisito	Botón de Selección Activación del Diccionario.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE2.

Número de requisito	RE4
Nombre de requisito	Etiqueta de desactivación del Diccionario.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE1. En esta etiqueta aparecerá el texto “Desactivar”.

Número de requisito	RE5
Nombre de requisito	Botón de Selección desactivación del Diccionario.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE4.

Número de requisito	RE6
Nombre de requisito	Agrupación de Modo de Uso.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Agrupación de las opciones de Modo de uso Local/Online del diccionario, cuya etiqueta será “Modo de Uso”. Aparecerá contenida en el formulario especificado en la RE0.

Número de requisito	RE7		
Nombre de requisito	Etiqueta Modo Online		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Etiqueta que identificará la opción de utilización de Diccionario en el modo Online. Esta opción se incluirá en la agrupación especificada en el requisito RE6.		

Número de requisito	RE8		
Nombre de requisito	Botón de Selección Modo Online		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE7.		

Número de requisito	RE9		
Nombre de requisito	Etiqueta Modo Local		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE6. En esta etiqueta aparecerá el texto “Local”.		

Número de requisito	RE10		
Nombre de requisito	Botón de Selección Modo Local		
Tipo	☒ Requisito	☐ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE9.		

Número de requisito	RE11
Nombre de requisito	Agrupación de Velocidad de búsqueda.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Agrupación de las opciones de velocidad de búsqueda del diccionario, rápida, media o lenta, cuya etiqueta será "Velocidad de respuesta". Aparecerá contenida en el formulario especificado en la RE0.

Número de requisito	RE13
Nombre de requisito	Etiqueta Búsqueda Rápida
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE11. En esta etiqueta aparecerá el texto "Local".

Número de requisito	RE14
Nombre de requisito	Botón Búsqueda Rápida
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE13.

Número de requisito	RE15
Nombre de requisito	Etiqueta Búsqueda Media.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE11. En esta etiqueta aparecerá el texto "Local".

Número de requisito	RE16
Nombre de requisito	Botón Búsqueda Rápida.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE15.

Número de requisito	RE17
Nombre de requisito	Etiqueta Búsqueda Lenta.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Etiqueta que aparecerá contenida la agrupación de opciones especificada en el requisito RE11. En esta etiqueta aparecerá el texto “Local”.

Número de requisito	RE18
Nombre de requisito	Botón Búsqueda Lenta.
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Aparecerá un botón de tipo Radio a la izquierda de la etiqueta especificada en el RE17.

Número de requisito	RE19
Nombre de requisito	Botón (x)
Tipo	☒ Requisito ☐ Restricción
Fuente del requisito	
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Eliminaremos el botón de la barra de herramientas que permitía ocultar la barra, dado que se trata de una de las opciones que proporciona el navegador Mozilla Firefox.

Número de requisito	RE20		
Nombre de requisito	Botón Opciones de Diccionario.		
Tipo	☐ Requisito	☒ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	En el menú “Opciones” de la Barra de herramientas debe aparecer una nueva opción que al seleccionarla mostrará el formulario de “Opciones de Diccionario”		

Número de requisito	RE21		
Nombre de requisito	Cambiar alerts por notificaciones del navegador.		
Tipo	☐ Requisito	☒ Restricción	
Fuente del requisito			
Prioridad del requisito	☐ Alta/Esencial	☒ Media/Deseado	☐ Baja/ Opcional
Descripción	Mozilla Firefox nos pide que cambiemos la forma de comunicarnos con los usuarios, utilizaremos notificaciones del navegador para motrar las notificaciones.		

3.1.2 Requisitos Funcionales

Número de requisito	RF0
Nombre de requisito	Almacenar Estado
Tipo	☒ Requisito ☐ Restricción
Entradas	entero
Salidas	vacio
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Cuando se seleccione una opción en la agrupación definida por el requisito RE1 (Agrupación de Estado), se enviará el valor seleccionado a esta función que se encargará de guardar el valor en una variable del navegador. En caso de que se seleccione la opción “Inactivo” se impedirá poder modificar la configuración del Modo de Uso, así como la velocidad de Búsqueda. Si se volviera a seleccionar la opción “Activo” se volverían a activar el resto de opciones.

Número de requisito	RF1
Nombre de requisito	Almacenar Modo de Uso
Tipo	☒ Requisito ☐ Restricción
Entradas	Entero
Salidas	vacio
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Cuando se seleccione una opción en la agrupación definida por el requisito RE6 (Agrupación de Modo de Uso), se enviará el valor seleccionado a esta función que se encargará de guardar el valor en una variable del navegador. En caso de seleccionar el Modo de uso Local, se desactivará el poder seleccionar velocidad de búsqueda. En caso de volver a seleccionar “Online” se volverá a permitir al usuario el modificar las opciones de velocidad.

Número de requisito	RF2
Nombre de requisito	Almacenar Velocidad de Búsqueda
Tipo	☒ Requisito ☐ Restricción
Entradas	Entero
Salidas	vacio
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Cuando se seleccione una opción en la agrupación definida por el requisito RE11 (Agrupación de Modo de Uso), se enviará el valor seleccionado a esta función que se encargará de guardar el valor en una variable del navegador.

Número de requisito	RF3
Nombre de requisito	Cargar Preferencias de Diccionario.
Tipo	☒ Requisito ☐ Restricción
Entradas	vacio
Salidas	vacio
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional
Descripción	Cuando se muestra el formulario de Opciones de Diccionario del requisito RE0, se cargarán las preferencias que anteriormente se han guardado utilizando la función del requisito RF2, RF1 y RF0.
Número de requisito	RF4

Nombre de requisito	Búsqueda en Diccionario Local
Tipo	☒ Requisito ☐ Restricción
Entradas	criterio
Salidas	Filtro que se aplicará
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción	En el criterio de búsqueda el usuario podrá utilizar el elemento “OR” para especificar búsquedas múltiples. Se creará un vector y se le pasará a la función de búsqueda de índices. RF5.

Número de requisito	RF5
Nombre de requisito	Búsqueda de Índices
Tipo	☒ Requisito ☐ Restricción
Entradas	Vector de criterios
Salidas	cadena
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción	Se encarga de realizar la búsqueda en el fichero de la línea que coincide con cada elemento del vector de criterios de búsqueda. Devolverá el resultado de las búsquedas separado por “OR”.

Número de requisito	RF6
Nombre de requisito	Búsqueda en Diccionario Online
Tipo	☒ Requisito ☐ Restricción
Entradas	criterio, velocidad
Salidas	Filtro que se aplicará
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción	En el criterio de búsqueda el usuario podrá utilizar el elemento "OR" para especificar búsquedas múltiples. Se realizará una consulta a Wordnet (RF7) para cada elemento encontrado en el criterio de búsqueda

Número de requisito	RF7
Nombre de requisito	Consultar WordNet
Tipo	☒ Requisito ☐ Restricción
Entradas	Criterio, velocidad
Salidas	cadena
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Descripción	Se realizará una consulta al servicio proporcionado por Wordnet, al que se le pasará el campo "criterio". El campo "velocidad" indica el nº de elementos que devolveremos en la salida separados por "OR".

Número de requisito	RF8		
Nombre de requisito	Descargar Diccionario Local		
Tipo	☒ Requisito	☐ Restricción	
Entradas	vacio		
Salidas	vacio		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional
Descripción	Esta función descargará el fichero de índices de diccionario ubicado en uno de los servidores de la Universidad Politécnica de Valencia. Tras realizar con éxito la descarga mostrará un pop-up informando de su correcta finalización RF7.		

Número de requisito	RF9		
Nombre de requisito	Mostrar pop-up.		
Tipo	☒ Requisito	☐ Restricción	
Entradas	cadena		
Salidas	vacio		
Prioridad del requisito	☐ Alta/Esencial	☐ Media/Deseado	☒ Baja/ Opcional
Descripción	Mostrará un bocadillo que contendrá la cadena que se le pasa como parámetro.		

Número de requisito	RF10		
Nombre de requisito	Función inicial.		
Tipo	☒ Requisito	☐ Restricción	
Entradas			
Salidas			
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional
Descripción	Es la función que realiza el filtrado de la web, ya existente en las anteriores versiones del software. Añadimos la funcionalidad del diccionario, para ello comprobamos las preferencias de configuración y realizamos la búsqueda en uno de los diccionarios online u local según las preferencias.		

3.2 Definición del proyecto

3.2.1 Propósitos

En la actualidad y con la continua aparición de webs, blogs y redes sociales, donde los usuarios publican información de todo tipo, la world wide web se ha convertido en una inagotable fuente de conocimiento. Al mismo tiempo los diferentes sitios albergados en esta red de redes ofrecen al visitante información alternativa a la que les llevó a visitar la página, desviando así la atención del usuario hacia otra información de menor relevancia, anuncios y ofertas que no son lo que estamos buscando.

Es por esto que se produce la aparición de herramientas que permitan al usuario organizar a la información obtenida, como es el caso de *"Webfiltering toolbar"*, que permite al usuario realizar una búsqueda selectiva, por concepto, en la web que está visitando y mostrar toda la información que tenga que ver con los términos introducidos en el campo de búsqueda.

El propósito principal de este proyecto es proporcionar una nueva funcionalidad a una herramienta que ya existe, en concreto proporcionará al usuario la posibilidad de utilizar conceptos relacionados semánticamente con los términos que pudieran introducir en el campo de búsqueda en un primer momento.

3.2.2 Objetivos

Los objetivos del proyecto son:

- Entender la filosofía de la programación de extensiones para Mozilla Firefox.
- Encontrar y realizar una comparativa de algunos servicios de internet que proporcionan acceso a un Lexicón.
- Integración de los servicios seleccionados con la herramienta.
- Optimización.
- Redacción de la memoria del Proyecto.

3.3 Planificación

3.3.1 Fases del Proyecto:

El proyecto en un primer momento se planificó para una duración aproximada de 5 meses, en el periodo comprendido entre 12 de Febrero de 2011 al 12 de Junio de 2011. Estas cuatro fases pasamos a enumerarlas a continuación.

3.3.1.1 Fase 1: Estudio previo

- Búsqueda de Información sobre el tipo de software a desarrollar
- Aplicaciones necesarias para el desarrollo del software.
- Comprensión del software ***"Webfiltering toolbar"*** (inicial).

3.3.1.2 Fase 2: Análisis y diseño

- Especificación de Requisitos
- Definir el Sistema de Diccionario Online
- Definir el Sistema de Diccionario Local
    - Descarga de Índices
    - Utilización de Índices

3.3.1.3 Fase 3: Implementación

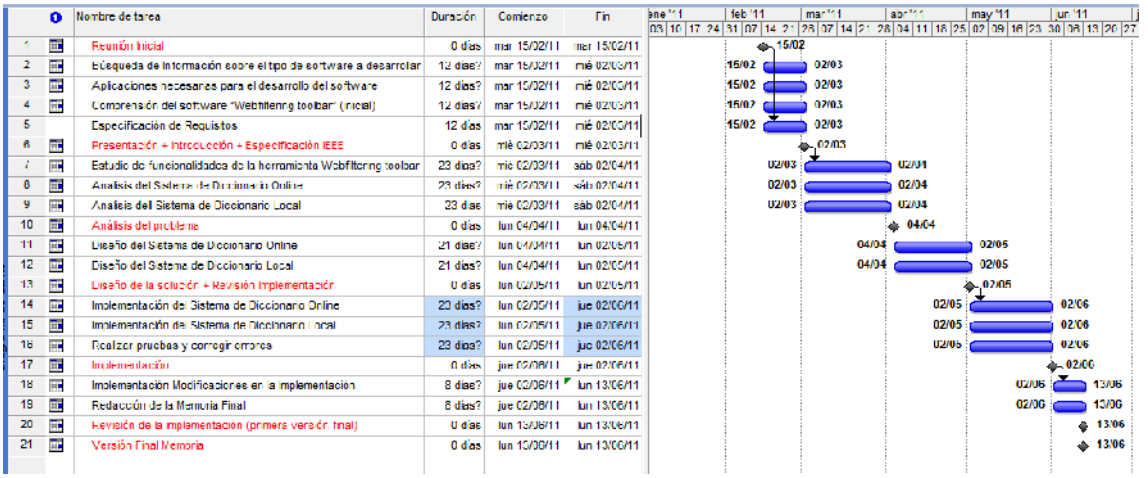
- Implementación del Sistema de Diccionario Online
- Implementación del Sistema de Diccionario Local
- Realizar pruebas y corregir errores.

3.3.1.4 Fase 4: Memoria

- Confeccionar la memoria del Proyecto.

3.3.2 Diagrama de Gantt inicial

En la Ilustración 17 podemos observar el diagrama de Gantt de la planificación inicial del proyecto. Aparecen marcados en rojo los diferentes puntos de control que tendrán como objetivo realizar el seguimiento del proyecto durante las fases de Analisis, Diseño e Implementación, y la posterior redacción de la memoria.



	Nombre de tarea	Duración	Comienzo	Fin
1	Reunión Inicial	0 días	mar 15/02/11	mar 15/02/11
2	Búsqueda de información sobre el tipo de software a desarrollar	12 días?	mar 15/02/11	mié 02/03/11
3	Aplicaciones necesarias para el desarrollo del software	12 días?	mar 15/02/11	mié 02/03/11
4	Comprensión del software "Webfiltering toolbar" (inicial)	12 días?	mar 15/02/11	mié 02/03/11
5	Especificación de Requisitos	12 días	mar 15/02/11	mié 02/03/11
6	Presentación + Introducción + Especificación IEEE	0 días	mié 02/03/11	mié 02/03/11
7	Estudio de funcionalidades de la herramienta Webfiltering toolbar	23 días?	mié 02/03/11	sáb 02/04/11
8	Análisis del Sistema de Diccionario Online	23 días?	mié 02/03/11	sáb 02/04/11
9	Análisis del Sistema de Diccionario Local	23 días	mié 02/03/11	sáb 02/04/11
10	Análisis del problema	0 días	lun 04/04/11	lun 04/04/11
11	Diseño del Sistema de Diccionario Online	21 días?	lun 04/04/11	lun 02/05/11
12	Diseño del Sistema de Diccionario Local	21 días?	lun 04/04/11	lun 02/05/11
13	Diseño de la solución + Revisión implementación	0 días	lun 02/05/11	lun 02/05/11
14	Implementación del Sistema de Diccionario Online	23 días?	lun 02/05/11	jue 02/06/11
15	Implementación del Sistema de Diccionario Local	23 días?	lun 02/05/11	jue 02/06/11
16	Realizar pruebas y corregir errores	23 días?	lun 02/05/11	jue 02/06/11
17	Implementación	0 días	jue 02/06/11	jue 02/06/11
18	Implementación Modificaciones en la implementación	8 días?	jue 02/06/11	lun 13/06/11
19	Redacción de la Memoria Final	8 días?	jue 02/06/11	lun 13/06/11
20	Revisión de la implementación (primera versión final)	0 días	lun 13/06/11	lun 13/06/11
21	Versión Final Memoria	0 días	lun 13/06/11	lun 13/06/11

Ilustración 17: Diagrama de Gantt inicial.

3.3.3 Diagrama de Gantt final

A continuación se puede observar en la Ilustración 18 las diferencias en el tiempo empleado en cada fase del Proyecto con respecto al Diagrama de Gantt inicial de la Ilustración 17. Estas desviaciones aparecen por la necesidad de realizar un nuevo análisis y diseño de algunas funcionalidades que comprende el proyecto.

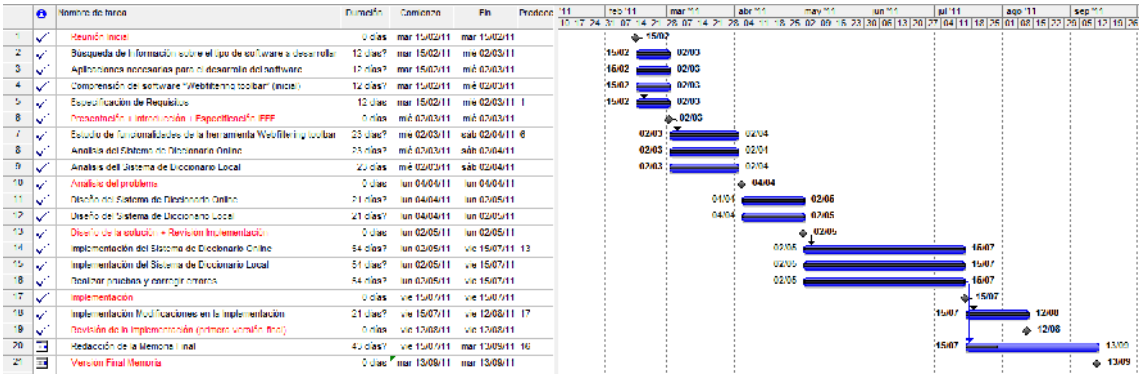


Ilustración 18: Diagrama de Gantt final.

3.4 Desarrollo y Seguimiento del Proyecto

3.4.1 Información Inicial

Como ya hemos comentado anteriormente, el Proyecto se trata de una ampliación de funcionalidad para una extensión Firefox ya existente, desde el comienzo somos informados de manera adecuada sobre las tecnologías utilizadas en el proyecto (Javascript, XPCOM, Ajax) y posibles entornos de programación que podremos utilizar durante el desarrollo de la solución final.

3.4.2 Análisis del Problema

El siguiente paso de nuestro proyecto consistía en realizar un análisis del requisito principal de nuestro proyecto, junto con un análisis de la secuencia de ejecución de la aplicación de partida, tras lo cual, aparecen identificadas las tareas iniciales a realizar en la implementación.

Dado que la función principal consiste en realizar el filtrado de una web en función de unos conceptos relacionados semánticamente con el introducido en un campo de búsqueda, llegamos a la conclusión de que necesitaremos desarrollar una función a la que se le pase como parámetro el concepto introducido por el usuario de la aplicación y que devuelva como resultado sus conceptos relacionados semánticamente.

Comenzamos nuestro análisis de la secuencia de llamadas que se produce desde que se introduce, en el campo de búsqueda, el concepto elegido para realizar el filtrado, hasta que realiza el filtrado, ya que será en este momento cuando deberemos realizar la llamada a la función que devolverá el resultado con los conceptos relacionados semánticamente con el concepto inicial introducido por el usuario de la aplicación.

3.4.3 Líneas de Investigación Iniciales

En este paso del proyecto se explicarán algunas de las líneas de investigación que se siguieron al inicio de nuestra actividad.

En la red de internet, encontramos distintos servicios que ofrecen la posibilidad de realizar búsquedas por conceptos y devuelven el resultado en una página web, por lo tanto una de las líneas de investigación consistía en realizar una petición mediante el objeto XMLHttpRequest, realizando un tratamiento específico para cada servicio en función del resultado esperado. Este tipo de servicios utilizan como base para proporcionar nuevos resultados el servicio proporcionado por la universidad de Princeton que conocemos como Wordnet, y además devolvían el resultado de la búsqueda en XML, lo que facilitaría el proceso, pero aún así sería necesario realizar tratamiento mediante JavaScript en la máquina del cliente. El servicio en concreto se trata del proporcionado por la Universidad de Toulouse y que podemos encontrar en la siguiente página. (Bernard Bou [Universidad de Touluse], 2008).

Otra posibilidad que se tuvo en cuenta, tras comentarlo con el director del proyecto, fue la de incluir el fichero de índices de diccionario en el paquete de instalación de nuestra extensión. En este caso lo que se pretendía era ubicar la base de datos de Wordnet en el equipo del cliente final, de esta forma el tiempo de respuesta sería mínimo. El problema de esta solución como era de esperar sería que el tamaño del paquete de instalación aumentaría por encima de los 7 u 8 MB, por lo que no cumpliríamos con los requisitos de Mozilla Firefox.

La última opción sería realizar la petición directamente al servicio de Wordnet, aunque el resultado obtenido no se trata de un XML, sí que se puede conseguir un resultado que podríamos tratar utilizando JavaScript y XPCOM para convertir el resultado obtenido del servicio Wordnet en un objeto DOM.

Tras este primer análisis se decidió implementar únicamente la característica que conectaba con el servicio de Wordnet para recuperar los conceptos relacionados con el criterio de búsqueda, debido a la limitación en el tamaño del paquete de instalación y por que de esta manera no dependeremos en el futuro de servicios de terceros.

Más tarde en una de las reuniones se nos ocurrió que se podría realizar la descarga del fichero de índices tras haber instalado la extensión en el navegador, al iniciar el análisis de esta nueva funcionalidad no estaba muy convencido de que realmente se pudiera realizar, por las muchas limitaciones que tiene JavaScript a la hora de manejar ficheros en la parte del cliente. Sin embargo, gracias a las características de nuestra aplicación podremos hacer uso de XPCOM de Mozilla, que proporciona una serie de componentes que permiten realizar tratamiento de ficheros en la parte del cliente, tal y como se ha explicado anteriormente.

3.4.4 Detalles de la Interfaz

La interfaz de usuario, no sufre mayores modificaciones con respecto a su anterior versión. Únicamente desaparece el botón de ocultar la barra, debido a que esa funcionalidad ya la proporciona el propio navegador Mozilla.

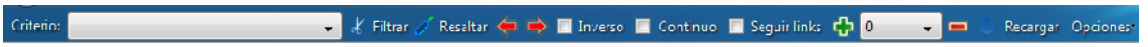


Ilustración 19: Interfaz de usuario de la extensión.

Otro pequeño cambio es la aparición en el menú Opciones de un submenú llamado "Diccionario" con el que se proporciona acceso a realizar los cambios en la configuración del diccionario.

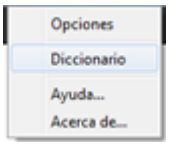


Ilustración 20: Menú desplegable de Opciones.

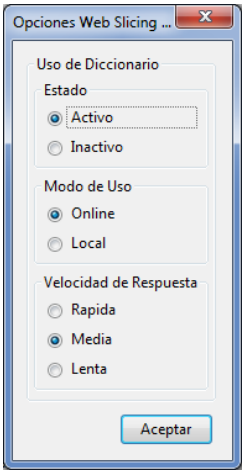


Ilustración 21: Opciones de Diccionario

Las opciones configurables por el usuario son el Estado del Diccionario (Activo o Inactivo), el Modo de Uso (Online o Local), y la velocidad de respuesta (Rápida, Media o Lenta). En función de la selección del usuario se ejecuta una función que guarda en una variable del navegador la configuración seleccionada por el usuario. De manera que cuando se realiza la búsqueda introduciendo el criterio de búsqueda, se comprueba inicialmente la configuración seleccionada por el usuario y se actúa en consecuencia. Realizando el filtrado Online, Local o sin el uso del Diccionario.

3.4.5 Detalles de la Implementación

En este apartado se proporcionan detalles concretos acerca de la implementación, entrando en detalle en las funcionalidades más importantes del proyecto.

Las funciones a las que debe su funcionalidad nuestra aplicación son las que se encargan de realizar la búsqueda en los distintos diccionarios utilizados (Servicio Wordnet online, Wordnet local).

Para realizar la llamada a estas funciones, hemos incrustado código en alguna de las funciones existentes inicialmente en el Proyecto Webfiltering. En concreto se ha insertado en el fichero **slicetoolbar_funciones_auxiliares.js**, en la función **webfiltering_inicializar_opciones.**

En esta parte del código se comprueban los diferentes ajustes que puede realizar el usuario referentes a si el diccionario se encuentra activado o no y el modo de utilización, en caso de ser online se llamará a la búsqueda online y en el caso de que el modo sea Local, se llamaría a la función apropiada. A continuación se muestra el código resultante.

```
var diccionario = webfiltering_getIntegerPreference("slice.synsetMode",true);

if(diccionario==2){
   //DESACTIVADO
}
else{//ACTIVADO
   var velocidad = webfiltering_getIntegerPreference("slice.synsetBusqueda",true);
   var modo = webfiltering_getIntegerPreference("slice.synsetModeLocal",true);
   if(modo==2){//LOCAL
       //ESTE ES EL FICHERO QUE UTILIZAMOS PARA LAS BUSQUEDAS EN LOCAL
       var file = Components.classes["@mozilla.org/file/directory_service;1"]
       .getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
       file.append("index_M2.sense");
       if ( file.exists() == true ){//SI EXISTE SE REALIZA BÚSQUEDA LOCAL
           document.getElementById("cuadroblanco").value=
           buscaEnDiccionarioLocal(document.getElementById("cuadroblanco").value.toLowerCase()
           .replace(" && "," and ").replace(" || "," or "));

       }
       else{ //SI NO EXISTE EL FICHERO COMENZARÁ LA DESCARGA.
           popup('Información WebFilteringToolbar','Ha seleccionado el uso del Diccionario local. A continuación se
           document.getElementById("cuadroblanco").value=
           diccionarioOnline(document.getElementById("cuadroblanco").value.toLowerCase()
           .replace(" && "," and ").replace(" || "," or "), velocidad);
           getFile(file);

           }
   }
   else{//ONLINE

       document.getElementById("cuadroblanco").value=diccionarioOnline(document.getElementById("cuadroblanco")
       .value.toLowerCase().replace(" && "," and ").replace(" || "," or "), velocidad);

       }
   }
```

Ilustración 22: Código de Filtrado con Diccionario

El código de estas funciones se encuentra en el fichero "***webfiltering_diccionario.js***" contenido en el paquete de instalación.

La primera de ellas se encarga de recopilar la información que devuelve como resultado el servicio online al que se realiza la petición. Para cada una de las palabras introducidas en el criterio de búsqueda, separadas por los operadores "OR", "||". Para que el lector se haga una idea de lo que queremos decir, a continuación se muestra un ejemplo del resultado obtenido al realizar una búsqueda en el servicio proporcionado por la universidad de Princeton.

En la figura de la izquierda hemos obtenido la respuesta de la búsqueda que usaremos para especificar cuál será el tratamiento que se realizará en el lado del cliente, mediante el uso de **JavaScript** y **DOM**, para extraer todas las palabras relacionadas. Para facilitarnos el trabajo he utilizado una extensión llamada **Firebug** (FirebugWorkingGroup, 2011), que nos muestra el código HTML de la página y nos facilita la visualización del DOM de la página. En la figura de la derecha se muestra el código HTML correspondiente a las primeras etiquetas "LI" que encontramos en el documento, que a su vez contendrán etiquetas "A".

A continuación explicaré el código de la implementación correspondiente a una búsqueda online. Esta función recibe dos parámetros de entrada, el criterio de búsqueda y la velocidad, de los cuales el primero se corresponde a la palabra o palabras introducidas por el usuario y el segundo parámetro determina el número máximo de conceptos que devolverá la función.

Con la intención de aumentar la funcionalidad e incluir búsquedas múltiples, se añade una función que se encarga de comprobar que el tipo de operador utilizado en el criterio de búsqueda se trata del tipo "OR" u "||", por ejemplo en el caso que el usuario utilizara el operador "AND" para separar diferentes conceptos de búsqueda, se mostraría al usuario una notificación indicando que no es posible realizar búsquedas con ese tipo de operador. Después de realizar esta comprobación, para cada criterio encontrado se realiza la búsqueda online.

```
function diccionarioOnline(txt,velocidad){
    busqueda = [];
    comprobarAND = txt.split(" and ");
    comprobarOR= txt.split(" or ");
    if(comprobarAND.length>1){
    popup('Información WebFilteringToolbar','No está permitido el uso de operadores mientras está
    return txt;
    }
    do{
                busqueda.push(getWordnet(comprobarOR.splice(comprobarOR.length-1,1), velocidad));
    }while(comprobarOR.length>0)
    return busqueda.join(" OR ");
}
```

Ilustración 23: Código de Búsqueda online

El primer paso en la función que realiza la búsqueda online es la creación del objeto **XMLHttpRequest** y la definición de la URL a la que se realizará la petición de búsqueda.

```
var xmlHttp;
// 1.- Creamos el objeto xmlHttpRequest
if(!xmlHttp && typeof XMLHttpRequest != "undefined")
{
   xmlHttp = new XMLHttpRequest();
}
// 2.- Definimos la URL a la que se realizará la petición.
var ajaxRequest = 'http://wordnetweb.princeton.edu/perl/webwn?s='+txt
```

Ilustración 24: Código de Búsqueda Online

El siguiente paso en nuestra implementación consiste en enviar la petición al servicio Wordnet, para ello utilizaremos el método open del objeto **XMLHttpRequest** que hemos creado con anterioridad, y especificaremos que se realizará una conexión síncrona, por lo que esperaremos a que se reciba la información, seguidamente estableceremos un timeout en el que si no se ha recibido una respuesta se notificará al usuario que el servicio no está disponible, esto lo conseguimos utilizando la función setTimeout, indicándole como parámetros la función que se ejecutará y el tiempo límite en milisegundos. A continuación realizamos el envío de la petición, mediante el método send() del objeto xmlhttprequest. Una vez termina la ejecución del método send, eliminaremos el timeout establecido anteriormente. De esta manera no se mostrará la notificación al usuario.

```
// 3.- Enviamos la petición
xmlHttp.open("GET", ajaxRequest, false);
id=setTimeout('popup(\'Información WebFilteringToolbar\'
xmlHttp.send("");
clearTimeout(id);
```

Ilustración 25: Código de Búsqueda Online

Crearemos una variable de tipo array en la que iremos añadiendo los elementos encontrados, a la que añadiremos el criterio de búsqueda como primer elemento.

```
var busqueda = [];
busqueda.push(txt);
```

Ilustración 26: Código de Búsqueda Online

Llegado a este punto de la función y una vez se ha recibido la respuesta del servidor en texto plano, pasaremos a convertirla en un documento HTML para poder acceder con facilidad a la estructura del documento. Esto lo conseguimos mediante la utilización de la función HTMLParser, a la que le pasamos como parámetro el texto plano y se encarga de devolverlo en forma de documento HTML.

```
//Parseamos la respuesta recibida.
var DOMPars = HTMLParser(xmlHttp.responseText);
```

Ilustración 27: Código de Búsqueda Online

Tras obtener el documento HTML, pasaremos a buscar en el documento los nodos etiquetados como <LI>, mediante el uso de la función **getElementsByTagName**, que devuelve un array que contendrá todos los elementos LI del documento.

```
//COGEMOS LOS ELEMENTOS LI del documento.
var listaLI=DOMPars.getElementsByTagName('li');
```

Ilustración 28: Código de Búsqueda Online

Para cada elemento <LI> debemos recuperar los nodos etiquetados como <A>, una vez los hemos recuperado, en función de la velocidad (0:Lenta,1:Media,2:Rápida), se recuperarán un determinado número de palabras.

Una vez determinado el número de palabras máximo que se recuperaran, para cada una de ellas se comprobará que no existen en el array que hemos creado anteriormente, en caso de que no exista, se realizará la comprobación de si se trata de un concepto compuesto, en este caso se encerrará el concepto entre comillas simples. Este detalle de la implementación viene especificado tras darnos cuenta de que si no se realizaba esta comprobación el filtrado de nuestra extensión devolvía resultados incoherentes con los resultados de la búsqueda obtenidos, al realizar combinaciones de tipo 'AND' con todos los elementos de la búsqueda.

Después de comprobar esto se añade al array de búsqueda.

```
for(var i =0;i<listaLI.length;i++)
{
    var listaA = listaLI[i].getElementsByTagName('a');
    var num;
    //Comprobamos la velocidad seleccionada
    switch(velocidad){
    case 2: num=3;
            if(num>listaA.length)
                num=listaA.length;
        break;
    case 1: num=5;
            if(num>listaA.length)
                num=listaA.length;
        break;
    case 0: num=listaA.length;
        break;
    }
    for(var j=2; j<num;j++)
    {
        if(listaA[j]!="(n)" && listaA[j]!="(v)" && busqueda.indexOf(listaA[j].text)==-1){
            if(listaA[j].text.split(" ").length >1){
                busqueda.push("'"+ listaA[j].text + "'");
            }else{
                busqueda.push(listaA[j].text);
            }
        }
    }
}
```

Ilustración 29: Código de Búsqueda Online

Por último devolvemos todos los elementos del array de búsqueda, separados por la operación 'OR'.

```
,
return busqueda.join(" OR ");
```

Ilustración 30: Código de Búsqueda Online

Tras realizar la implementación de la búsqueda online y realizar su primer testeo, se nos ocurrió que si en algún momento el servicio proporcionado por Wordnet dejara de funcionar, nos afectaría directamente. La posibilidad de que este hecho se produjera hacia que comenzáramos a plantearnos la idea de proporcionar un método alternativo para poder utilizar el diccionario. (Durante el desarrollo del proyecto se sucedieron algunas interrupciones del servicio con un tiempo máximo de caída de 48 horas). La idea que veníamos persiguiendo era conseguir que se descargaran los ficheros al equipo del usuario final, sin necesidad de que estuviesen incluidos en el paquete de instalación. Es decir, que el usuario final debería decidir cuándo descargarlos.

Una vez que el usuario ha activado el uso del diccionario local, nuestra extensión realizará una petición al servidor Einstein, ubicado en la Universidad Politécnica de Valencia y se descargará el fichero al directorio de configuración de Mozilla, para acto seguido proceder a su utilización. Para proceder a descargar el fichero del diccionario local, anteriormente el usuario debe haber seleccionado en los Ajustes del diccionario el Modo Local, y en el siguiente filtrado que se realiza, se realizará la descarga del fichero. La función que realiza esta acción comienza creando el objeto XMLHttpRequest y definiendo la URL a la que se realizará la petición.

```
// 1.- Creamos el objeto xmlHttpRequest

if(!xmlHttpFile && typeof XMLHttpRequest != "undefined")
 {
    xmlHttpFile = new XMLHttpRequest();
 }

 // 2.- Definimos la llamada para hacer un simple GET.
 //var ajaxRequest = 'http://localhost:8080/synset/index.sense';
 var ajaxRequest = 'http://einstein.dsic.upv.es/webfiltering/index_M2.sense';
```

Ilustración 31: Código Descarga de Diccionario

En esta ocasión es posible realizar la acción de descargar el fichero de forma asíncrona. Lo que significa que no se interferirá la ejecución de la aplicación mientras se esperamos la respuesta del servidor, por lo que habrá que definir qué debe hacer el programa cuando acabe la recepción del fichero.

```
xmlHttpFile.onreadystatechange = function(){
    if (xmlHttpFile.readyState==4 && xmlHttpFile.status==200){
    //Escribimos en el fichero.
        var content = xmlHttpFile.responseText;
        file.create( Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 420 );
        var outputStream = Components.classes["@mozilla.org/network/file-output-stream;1"].createInsta
        outputStream.init( file, 0x04 | 0x10, 420, 0 );
        var result = outputStream.write( content, content.length );
        outputStream.close();

        popup('Información WebFilteringToolbar','La descarga del Diccionario local ha finalizado.');

    };
```

Ilustración 32: Código de Descarga de Diccionario

Por último se realiza el envío de la petición de manera similar a como la realizábamos en la búsqueda online, con la diferencia que en esta ocasión pasaremos el valor “true” en el tercer parámetro de la función open(), que indica que la conexión se realizará de manera asíncrona tal y como hemos explicado anteriormente.

```
xmlHttpFile.open("GET", ajaxRequest, true);
xmlHttpFile.send("");
```

Ilustración 33: Código Descarga de Diccionario

A continuación se muestra un ejemplo del contenido del fichero que se descargará al equipo del usuario final, en el que podemos observar que existe un separador que divide en dos partes cada línea del fichero. La primera parte corresponde a las palabras contenidas en el fichero sobre las que se realizará la búsqueda según los criterios introducidos por el usuario y en la segunda parte aparecen los conceptos relacionados semánticamente, que serán devueltos por nuestra implementación como resultado de la búsqueda.

```
programing%'computer programing' OR 'computer programming' OR programming OR programming OR scheduling
programing language%'programming language'
programme%broadcast OR 'computer program' OR 'computer programme' OR 'course of study' OR curriculum OR plan OR
programme music%'program music'
```

Tras la breve explicación del contenido del fichero explicaremos brevemente el algoritmo de búsqueda en el fichero.

Al igual que ocurría en las funciones correspondientes a la búsqueda online, utilizamos una primera función en la que comprobamos los operadores utilizados y se realiza la llamada a la función que realiza la búsqueda en el diccionario local, para cada palabra.

```
function buscaEnDiccionarioLocal(criterio){
    criterios = [];
    criterios = criterio.split(" or ");
    comprobarAND = criterio.split(" and ");
    if(comprobarAND.length>1){
    popup('Información WebFilteringToolbar','No está permitido el uso del operador AND en búsquedas
    return criterio;
    }
    return findIndex(criterios);
}
```

Ilustración 34: Código de búsquedaEnDiccionarioLocal

La función findIndex, recibe como parámetro un array con los criterios y consigue en una única pasada al diccionario, recuperar todos los conceptos relacionados en él, con las palabras contenidas en el criterio de búsqueda.

Para entender mejor la funcionalidad de esta función, explicaremos su funcionamiento dividiéndola en tres partes.

En la primera se especifica el fichero que se recorrerá, usando la funcionalidad que proporciona **XPCOM** para manejo de ficheros locales. Después de especificar que se utilizará un objeto de tipo **nsiFile**, se determina el nombre del fichero que utilizará de los ubicados en el directorio de configuración de Mozilla, mediante el método append(), al que se le pasa la cadena con el nombre del fichero. Podemos observar que se crea un array donde se almacenarán los resultados de la búsqueda y en el que añadimos los criterios iniciales introducidos por el usuario. Por último declararemos una variable istream que utilizaremos a como buffer, de la cual iremos leyendo cada línea del fichero.

```
function findIndex(criterios){
var file = Components.classes["@mozilla.org/file/directory_service;1"]
.getService(Components.interfaces.nsIProperties).get("ProfD", Components.interfaces.nsIFile);
file.append("index_M2.sense");
var busqueda = [];
busqueda.push(criterios.join(" OR "));
// open an input stream from file
var istream = Components.classes["@mozilla.org/network/file-input-stream;1"].
              createInstance(Components.interfaces.nsIFileInputStream);
istream.init(file, 0x01, 0444, 0);
istream.QueryInterface(Components.interfaces.nsILineInputStream);
```

Ilustración 35: Código de findIndex.

En la siguiente parte de la función, se continuarán leyendo líneas de fichero mientras que existan elementos en el array de criterios. Cada vez que se lee una línea se realiza una división utilizando el divisor de cadena que hemos comentado con anterioridad en el punto de la estructura del fichero local, tras esta acción se comprueba si el primer elemento de la línea existe en el array de criterios y si fuera así se añade la segunda parte de la línea al array de búsqueda y se elimina del array de criterios el elemento que hemos encontrado, así hasta que no quedan más criterios.

En el peor de los casos se leerán todas las líneas del diccionario, una única vez, independientemente del número de criterios introducidos por el usuario.

```
// read lines into array
var line = {}, lines = [], hasmore;
var num=3;
do {
  hasmore = istream.readLine(line);
  var v_text = line.value.split("%");
  var indice = criterios.indexOf(v_text[0]);
  if(indice!=-1){
     busqueda.push(v_text[1]);
     criterios.splice(indice,1);
  }
} while(criterios.length>0 && hasmore && num>0);
istream.close();
```

Ilustración 36: Código de findIndex.

Después de finalizar el bucle por no existir más elementos en el criterio de búsqueda, se devuelve el resultado.

```
return busqueda.join(" OR ");
}
```

Ilustración 37: Código de findIndex.

Después de la ejecución tanto si se realiza una búsqueda online, como local, se muestran los resultados en el criterio de búsqueda.

3.4.6 Tiempo de respuesta del producto final

Las pruebas que hemos venido realizado sobre las distintas versiones de nuestra implementación tenían como objetivo comprobar el tiempo de respuesta medio de las distintas búsquedas.

El equipo utilizado para la realización de las pruebas se trata de un Portátil de reducidas dimensiones con un procesador Intel Core 2 duo a 1.30 GHz y con 4 GB de RAM, además el sistema operativo utilizado para la realización de las pruebas ha sido un Windows 7 Home Premium, funcionando en una red doméstica conectado por wifi a una conexión de 6 MB.

Al realizar las pruebas sobre una de las primeras implementaciones realizadas se obtuvieron unos tiempos de respuesta medios demasiado altos, que no se asemejaban a los tiempos de respuesta ideales. Teóricamente la búsqueda mediante el uso del diccionario local, debía resultar mucho más rápida que la búsqueda online, y tal y como se puede observar en la tabla, eso no se ajustaba a la realidad. Esto era debido a que el fichero de índices de diccionario estaba estructurado de manera distinta, teniendo que realizar varias pasadas para encontrar todas las ocurrencias del concepto buscado. Tras comprobar esto se decidió modificar el fichero de índices para que optimizar el tiempo de búsqueda.

Respecto a la búsqueda Online, teóricamente debía disminuir el tiempo de búsqueda en función de la velocidad seleccionada por el usuario, pero esto no siempre era cierto pues el tiempo de respuesta dependía también del lo saturado que esté el servicio de Wordnet en ese momento. En este caso se podría mejorar el algoritmo utilizado en el cliente, pero sin embargo no podemos ganar tiempo en la parte de la comunicación con Wordnet.

Tiempos de respuesta primeras versiones del producto

	LOCAL			ONLINE		
	Lenta	Media	Rápida	Lenta	Media	Rápida
car	5341	4732	2671	2511	1221	1895
game	6825	6488	4843	2540	1380	1113
dog	5668	3886	1413	2626	1834	1306

Ilustración 38: Tabla de tiempos de respuesta primeras versiones.

Para esta prueba se escogieron las palabras “car”,”game” y “dog” que aparecen indicadas tanto en la tabla como en la gráfica, por que el tiempo de respuesta no dependía de donde estaba situada la palabra en el fichero de índices, si no que el tiempo de respuesta era mucho mayor debido a las pasadas que se realizaban sobre el fichero.

Para la confección del nuevo fichero de índices se utilizó un algoritmo que realizaba el recorrido del fichero para todas las palabras y buscaba todos los conceptos relacionados. En el primer momento de comenzar a trabajar con el fichero de índices, este contenía una línea por cada relación semántica que tenía el concepto, podemos observar que la estructura del fichero era la siguiente:

```
dog%1:05:00:: 02084071 1 42
dog%1:06:00:: 03901548 6 0
dog%1:06:01:: 02710044 7 0
dog%1:13:01:: 07676602 5 0
dog%1:18:00:: 10023039 3 0
dog%1:18:01:: 10114209 2 0
dog%1:18:02:: 09886220 4 0
dog%2:38:00:: 02001858 1 2
```

Ilustración 39: Estructura de fichero de índices inicial.

Basándonos en las líneas que se muestran en la figura anterior, podríamos vaticinar que se realizarán mínimo dos pasadas al fichero. La primera en la que se recolectan todos los índices correspondientes a la palabra “dog” y la segunda pasada que recorrerían todas las líneas comparando el índice de cada concepto con los índices encontrados en la primera pasada.

En la siguiente gráfica se pueden observar los tiempos de respuesta de la extensión en las pruebas realizadas sobre las primeras versiones de la aplicación.

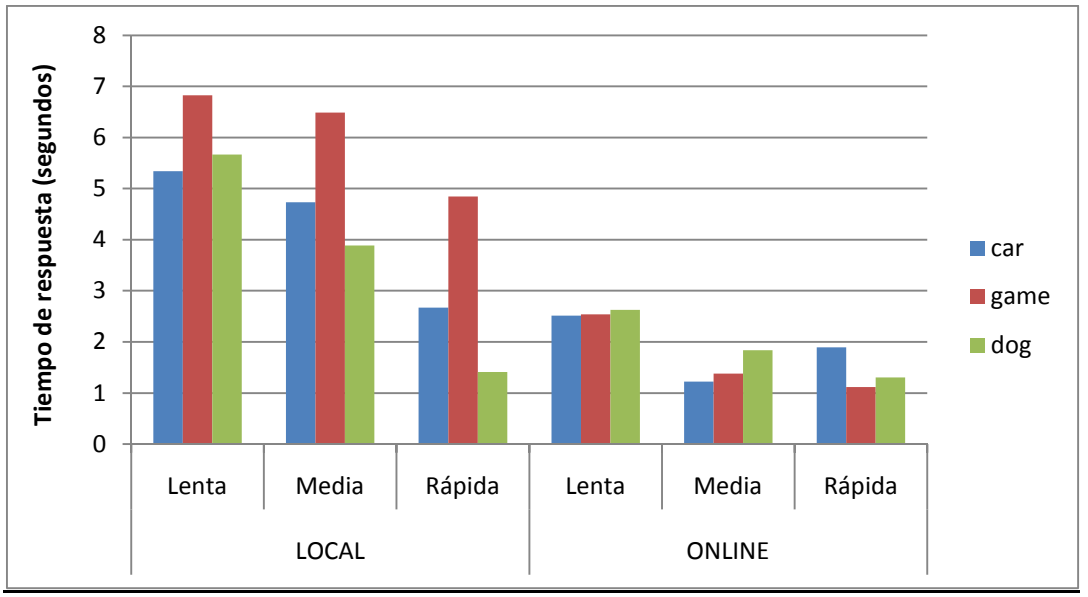


Ilustración 40: Gráfica Tiempos de respuesta primeras versiones.

A continuación se muestra cómo varía el tiempo de respuesta tras realizar las modificaciones pertinentes en la implementación, ocasionadas por la optimización del fichero de índices de diccionario.

Observamos que en la búsqueda local se consigue disminuir considerablemente el tiempo de respuesta con respecto a anteriores versiones de la extensión, al conseguir una velocidad que en el peor de los casos ronda un tiempo medio de respuesta de 1000 ms, se decidió eliminar la opción de establecer la velocidad de búsqueda en el diccionario local.

En esta nueva versión de la extensión se modifica la estructura del fichero de índices, conteniendo una única ocurrencia por concepto, reduciendo de esta manera el número de líneas total que encontramos en el fichero y consiguiendo el resultado de búsqueda después de realizar tan solo una pasada a todo el fichero, en el peor de los casos.

```
dog%andiron OR blackguard OR bounder OR cad OR 'canis familiaris'
dog's-tooth check%'dogs-tooth check' OR 'dogstooth check' OR 'houn
dog's-tooth violet%dogtooth OR 'dogtooth violet'
dog's breakfast%'dog's dinner'
dog's dinner%'dog's breakfast'
dog's mercury%'dog mercury' OR 'mercurialis perennis'
dog-day cicada%'harvest fly'
dog-eared%eared
dog-iron%andiron OR dog OR firedog
dog-tired%exhausted OR fagged OR fatigued OR 'played out' OR spent
```

Ilustración 41: Estructura del fichero de índices final.

Con respecto a la búsqueda online, los resultados pueden variar en función del número de peticiones que esté recibiendo el servicio **Wordnet** en ese momento, por lo que se realizaron modificaciones en el algoritmo del cliente pero no se ha conseguido una mejoría significativa con respecto a otras versiones de la extensión.

Tiempo de respuesta del producto final. (milisegundos).

	LOCAL		ONLINE					
Bloque de Búsqueda	Simple	Multiple	Lenta	Media	Rápida	Multiple Lenta	Multiple Media	Multiple Rápida
A - I	200	207	1251	1209	817	1983	1776	1581
J - R	617	837	2442	1239	1076	2371	1876	1760
S - Z	978	1040	2175	1664	1081	2494	1803	1750

Ilustración 42: Tabla tiempos de respuesta versión final.

A continuación se obtiene la gráfica con los resultados

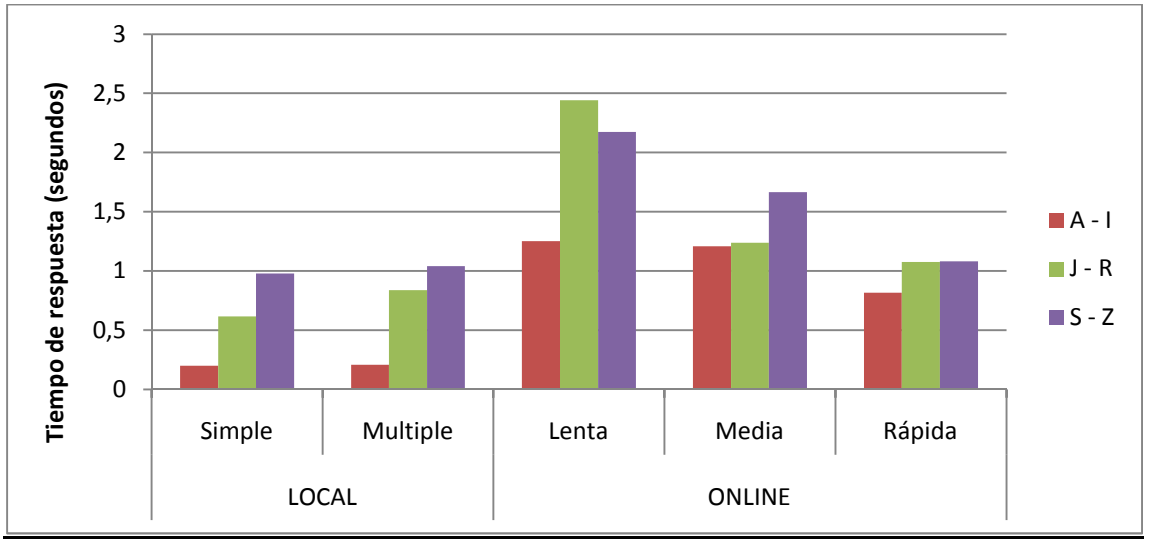


Ilustración 43: Gráfico Tiempos de respuesta versión final.

3.4.7 Mejoras y requisitos transmitidos por Mozilla Firefox

Desde Mozilla Firefox, tras realizar un análisis de nuestra extensión, nos indicaron que sería bueno para mejorar la experiencia del usuario la no utilización de la función alert proporcionada por JavaScript y nos propuso la utilización de divs.

Tras la recepción de esta propuesta nos pusimos manos a la obra a desarrollar esta nueva funcionalidad, y en poco tiempo conseguimos mostrar un div en el que se mostrarían los distintos mensajes producidos por la aplicación al usuario final. Pero después de conseguir dotar de esta funcionalidad a nuestra extensión se concluyó que lo deseable sería que los mensajes de notificación al usuario se desplegaran de la propia barra del navegador.

3.4.8 Solución con Divs

La primera solución implementada con javascript y DOM, mostraba un DIV con la notificación para el usuario:

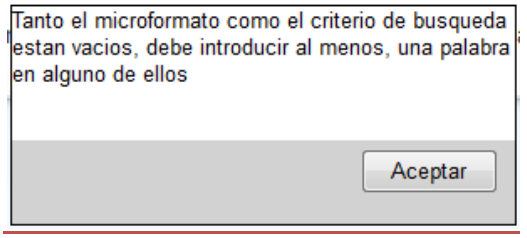


Ilustración 44: DIV flotante con el mensaje.

La codificación html de una web cuando se muestra el elemento div podría ser la que se muestra a continuación.

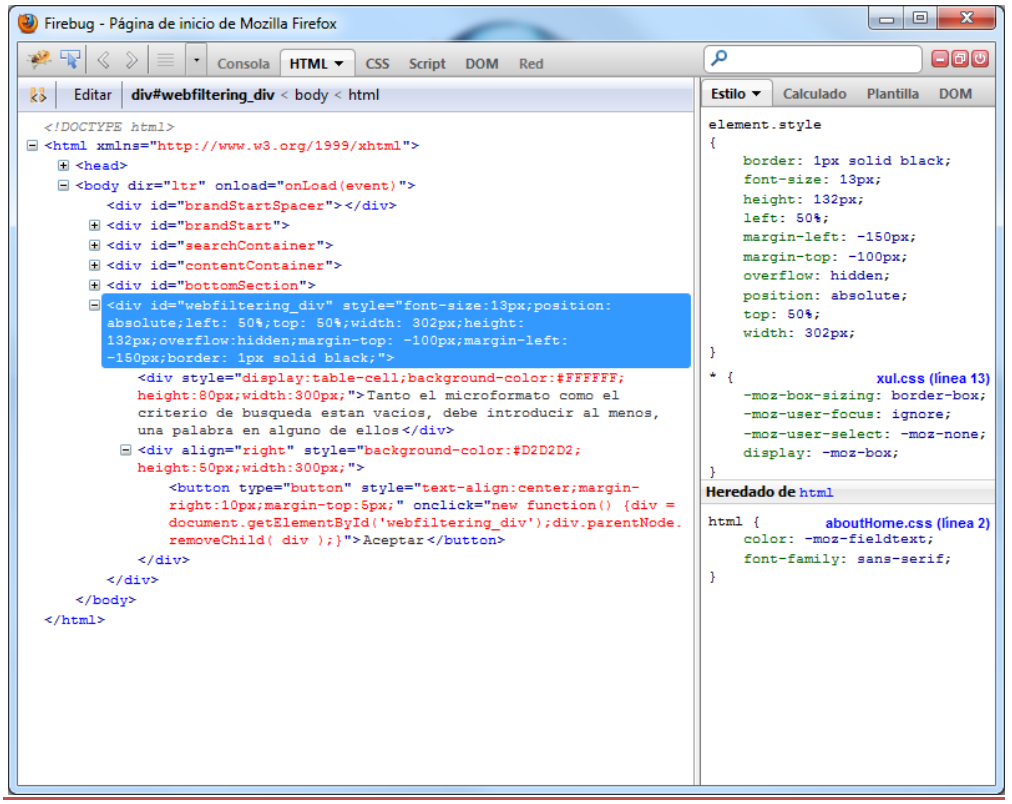


Ilustración 45: Código DIV flotante generado en página visitada.

La función que permite mostrar el mensaje en la web que se visualiza en el navegador recibía como parámetro el mensaje que se deseaba poner en conocimiento del usuario.

```
function notificar(mensaje){
```

Ilustración 46: Creación elemento div flotante.

Para comenzar se almacena el documento html en la variable webfiltering_contentDocument, y posteriormente se le irán añadiendo los distintos nodos necesarios para mostrar el div en la web que se está visualizando actualmente.

```
this.webfiltering_contentDocument = document;
```

Ilustración 47: Creación elemento div flotante.

El primer nodo que crearemos se trata del que contiene el mensaje que se le ha pasado como parámetro. Esto lo hacemos mediante la utilización de la función createTextNode(), pasándole como parámetro la cadena con el mensaje de notificación.

```
webfiltering_mensaje=document.createTextNode(mensaje);
```

Ilustración 48: Creación elemento div flotante.

Seguidamente creamos el Nodo <P> utilizando el método createElement y añadimos como hijo mediante la función appendChild el nodo de texto que hemos creado con el mensaje.

```
webfiltering_p=document.createElement("p");
webfiltering_p.appendChild(webfiltering_mensaje);
```

Ilustración 49 Creación elemento div flotante.

Después de esto creamos el nodo correspondiente al botón que utilizaremos para Aceptar la notificación y ocultar el div que hemos mostrado al usuario, mediante la función createElement creamos un elemento <input> y posteriormente establecemos la propiedad type y el valor correspondientes. Además especificamos que cuando se haga clic en el botón, se ocultará el div.

```
webfiltering_button=document.createElement("input");
webfiltering_button.type="button";
webfiltering_button.value="Aceptar";
webfiltering_button.onclick = function() {
            webfiltering_div.style.visibility = "hidden";
        }
```

Ilustración 50: Creación elemento div flotante.

A continuación se crearan dos elementos div. Uno de ellos contendrá como hijo el elemento que contiene el mensaje, llamado webfiltering_p. El otro contendrá el botón que permitirá al usuario aceptar el mensaje y devolver la web a su estado normal.

```
webfiltering_div_texto=document.createElement("div");
webfiltering_div_texto.setAttribute("style","height:100px;width:300px");
webfiltering_div_texto.appendChild(webfiltering_p);
```

Ilustración 51: Creación elemento div flotante.

```
webfiltering_div_button=document.createElement("div");
webfiltering_div_button.setAttribute("style","height:100px;width:300px");
webfiltering_div_button.appendChild(webfiltering_button);
```

Ilustración 52: Creación elemento div flotante.

Ahora se crea el div que engloba a los elementos que contienen el mensaje y el botón creados anteriormente.

```
webfiltering_div=document.createElement("div");
webfiltering_div.id="webfiltering_div";
webfiltering_div.setAttribute("style","background-color:transparent;opacity:0.5; position: absolute;left:
webfiltering_div.appendChild(webfiltering_div_texto);
webfiltering_div.appendChild(webfiltering_div_button);
```

Ilustración 53: Creación elemento div flotante.

Por último se añade el elemento webfiltering_div mediante el uso de la función appendChild al final del nodo body de la web que se está visitando.

```
    this.webfiltering_contentDocument.body.appendChild(webfiltering_div);
}
```

Ilustración 54: Creación elemento div flotante.

3.4.9 Solución con elementos de notificación de XPCOM:

Tras realizar la implementación decidimos cambiar los divs por elementos proporcionados por la tecnología XPCOM de Mozilla. Estos elementos son las barras de notificación del navegador y los pop-ups que se despliegan desde la barra de estado.

3.4.9.1.1 Barra de Notificación

La barra de notificación, permanecerá visible hasta que el usuario la cierre, y se mostrará con el siguiente aspecto al usuario.

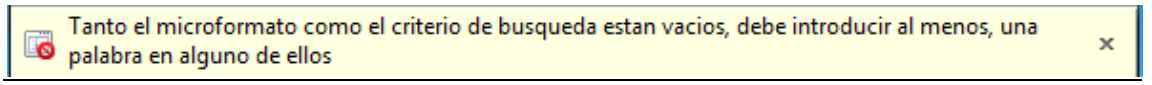


Ilustración 55: Barra de notificación.

La función **notificar** es la encargada de mostrar la barra de notificación, la cual recibe el mensaje que se ha de mostrar como parámetro. Las barras de notificación del navegador tienen un identificador, que en nuestro caso hemos denominado "info". Antes de pasar a crear el elemento por primera vez, se comprueba que no exista ya en el navegador mediante el uso de las líneas:

"var nb = gBrowser.getNotificationBox();" y **"var n = nb.getNotificationWithValue('info');"**

Y la comprobación de si en la variable **"n"** se ha obtenido algú objeto después de la ejecución de las líneas comentadas anteriormente.

En caso de que no exista ningún elemento con el identificador **"info"** se procede a añadirlo al navegador mediante el uso de la función appendNotification(), que recibe como parámetros:

- El mensaje
- El identificador
- La imagen
- Prioridad
- El vector de botones especificando para cada botón el comportamiento individual. En nuestro caso no se utilizan botones por no ser necesarios puesto que se trata de información que se le dá al usuario y no es necesario tomar ninguna decisión sobre la información que se muestra.

```
function notificar(mensaje){
    var message = mensaje
    var nb = gBrowser.getNotificationBox();
    var n = nb.getNotificationWithValue('info');
    if(n) {
        n.label = message;
    } else {

    var buttons=[];

    const priority = nb.PRIORITY_WARNING_MEDIUM;
    nb.appendNotification(message, 'info',
                          'chrome://browser/skin/Info.png',
                           pricrity, buttons);
}


}
```

Ilustración 56: Función que muestra la Barra de notificación.

3.4.9.1.2 Pop-up

Los pop-up que se mostrarán se desplegarán desde la barra de estado del navegador y tras unos segundos desaparecerá de la vista del usuario. En principio estos mensajes mostrarán información relacionada con el uso del diccionario.

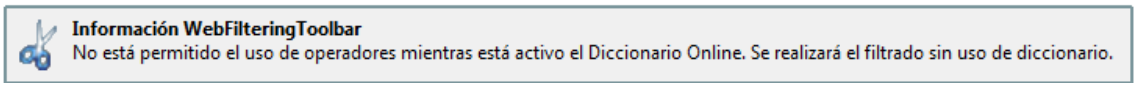


Ilustración 57: Ejemplo de Pop-up del navegador.

La función que implementa la aparición de este elemento en el navegador es la llamada "popup" y recibirá como parámetros el título y el texto que contendrá el propio pop-up.

```
//FUNCIÓN QUE MUESTRA EL POP-UP DEL NAVEGADOR
function popup(title, text) {
  try {
    Components.classes['@mozilla.org/alerts-service;1'].
               getService(Components.interfaces.nsIAlertsService).
               showAlertNotification("chrome://slicetoolbar/skin/scissors32.png", title, text, false, '', null);
  } catch(e) {
    // prevents runtime error on platforms that don't implement nsIAlertsService
  }
}
```

Ilustración 58: Función que muestra el Pop-up.

Después de la ejecución de este código se visualizará el mensaje durante unos segundos y posteriormente desaparecerá de la pantalla.

4 Conclusiones

A la hora de seleccionar este tipo de proyecto, fue determinante el hecho de que se tratase de una oportunidad de conocer una nueva tecnología relacionada con el desarrollo de extensiones para el navegador Mozilla Firefox. Aunque a la vez supuso un reto y fue necesario un trabajo previo de investigación para conseguir dominar los distintos aspectos de esta tecnología.

Una vez superado el objetivo de conocer la tecnología, se debía comparar los distintos servicios que existen en internet. Tras lo que se decidiría finalmente el elegido y el que formaría parte del proyecto. Tal como se ha explicado a lo largo de la redacción del presente documento, este servicio fue Wordnet por proporcionar múltiples vías para consultar los conceptos contenidos en su base de datos, un servicio web y un diccionario local.

La parte más costosa del proyecto fue la definición de los algoritmos de consulta que conseguirían a la postre integrar los servicios online y local de Wordnet y la extensión "Webfiltering toolbar", debido en parte a que no existen herramientas suficientemente extendidas y utilizadas para el desarrollo de este tipo de aplicaciones.

Tras conseguir la implementación del algoritmo inicial, se realizan tomas de tiempos sobre distintos conceptos utilizando ambos modos de utilización. Después de realizar un análisis de los datos obtenidos se toma la decisión de optimizar los algoritmos utilizados. En este caso, tras la reestructuración del Diccionario Local, y la modificación del algoritmo teniendo en cuenta la nueva estructura, se consigue rebajar el tiempo de respuesta de la barra, tal y como se explica en el punto 3.4.6.

Por último, la realización de la memoria ha permitido asentar los conocimientos sobre las herramientas y lenguajes utilizados durante las distintas etapas del proyecto así como adquirir nuevos conocimientos sobre la redacción de documentos técnicos.

Finalmente, tras concluir la realización de la memoria del Proyecto Final de Carrera, cabe resaltar el hecho de haber contribuido al crecimiento funcional de la herramienta "Webfiltering toolbar", finalizando con éxito el objetivo principal planteado al inicio del proyecto, que consistía en añadir la funcionalidad de Filtro con Diccionario.

5 Líneas futuras de trabajo

En la parte inicial del desarrollo del proyecto, se realizó un estudio sobre los distintos servicios que se encuentran en Internet y que ofrecen la posibilidad de realizar búsquedas por conceptos sobre relaciones semánticas.

Los servicios encontrados, aunque ofrecen soporte para múltiples idiomas, no aportaban la riqueza léxica que sí que encontramos en Wordnet (Servicio proporcionado por la Universidad de Princeton), con más de 200.000 conceptos en su base de datos, además de la posibilidad de realizar filtrados en modo local, lo que finalmente fue determinante para convertir este servicio en el seleccionado para formar parte del desarrollo del proyecto.

En la parte final del proyecto se vuelve a realizar un estudio sobre los distintos servicios que se podrían encontrar, fue en esta ocasión cuando se descubre que se está llevando a cabo un desarrollo de un servicio Wordnet en castellano. Al encontrarse en una fase temprana de desarrollo, se decide no incluirlo para esta versión del producto final.

La idea de proporcionar soporte para el idioma Castellano, a la funcionalidad de filtrado semántico e incluso la posibilidad de incluir otros idiomas, debería incluirse como línea de investigación para futuras versiones de la barra **"Webfiltering toolbar".**

Como alternativa a la anterior idea, podemos encontrar en internet multitud de servicios de traducción que pueden ser utilizados para proveer a la herramienta de una funcionalidad similar a la descrita anteriormente; la cual supondría la integración del filtrado semántico descrito en el proyecto, con uno de los servicios de traducción seleccionado. Por las características de la extensión, será determinante a la hora de seleccionar uno de los servicios de traducción el tiempo de respuesta resultante.

En el futuro, se debe tener en cuenta que la actual versión de la extensión está disponible únicamente para Mozilla Firefox, eso quiere decir que existen distintos navegadores que los usuarios pueden elegir para navegar por internet y en otros no tendrán la barra disponible. Los navegadores más utilizados en la actualidad, son principalmente tres. El navegador de Microsoft es el que obtiene mayor cuota de mercado, seguido de Mozilla Firefox y Google Chrome. Sería una buena estrategia de expansión para la extensión, el desarrollo de versiones adaptadas para otros navegadores como puedan ser Internet Explorer 9 o Google Chrome.

En ocasiones, los usuarios de internet tienden a consultar recurrentemente unas determinadas URL's, en busca de actualizaciones en la información contenida en ellas. Con la intención de optimizar al máximo el tiempo que se dedica a la búsqueda de este tipo de información sería posible la implementación de un sistema de avisos que se integraría en la barra y ofrecería al usuario información sobre las distintas actualizaciones que se producen en las páginas incluidas en los marcadores del usuario.

Es posible que con el paso del tiempo se necesite actualizar el fichero de índices descargado en los equipos de los usuarios de la extensión, por esta razón se debería incluir como mejora futura la implementación de un sistema de actualización de los ficheros de índices. Este sistema comprobaría la existencia de nuevas versiones del Diccionario en el servidor "Einstein" y realizaría la descarga del fichero en caso de existir una nueva versión en el servidor. Esta ampliación permitiría gestionar de manera independiente la actualización de versiones de la extensión y la actualización de versiones del fichero de índices del Diccionario.

6 Bibliografía

(Akilaa [Mozilla Firefox]. (2011). *MDN Docs - Gecko FAQ*. Retrieved 2011 йил 8-Mayo from Gecko FAQ: https://developer.mozilla.org/en/Gecko_FAQ

Bernard Bou [Universidad de Touluse]. (6 de Julio de 2008). Obtenido de http://jws-champo.ac-toulouse.fr:8080/wordnet-xml/

ecma-international. (2011). *ECMA.* Retrieved 2011 йил 6-Septiembre from http://www.ecma-international.org/

FirebugWorkingGroup. (2011). *Firebug.* From Complementos para Firefox: https://addons.mozilla.org/es-es/firefox/addon/firebug/

hack.augusto [Mozilla Firefox]. (2011 йил 7-Mayo). *MDN Docs - XPCOM*. Retrieved 2011 йил 7-Mayo from https://developer.mozilla.org/en/XPCOM

Madarche [Mozilla Firefox]. (2011 йил 14-Julio). *MDN Docs - nsiFile*. Retrieved 2011 йил 10-Agosto from nsiFile: https://developer.mozilla.org/en/nsIFile

Monzonís, F. J. (2008 йил 2-Junio). *Aplicación Práctica sobre el Proyecto.* Retrieved 2011 йил 7-Mayo from Universitat Jaume I: http://forja.uji.es/frs/download.php/133/memoria.pdf

Movses-bot. (2011 йил 11-Agosto). *Wikipedia.* Retrieved 2011 йил 25-Agosto from http://es.wikipedia.org/wiki/JavaScript

Mozilla Firefox. (2011). *Navegador web Firefox*. Retrieved 2011 йил 10-Septiembre from http://www.mozilla.org/es-ES/firefox/

Philikon [Mozilla Firefox]. (2011 йил 12-Abril). *MDN Docs - nsIFileInputStream*. From nsIFileInputStream: https://developer.mozilla.org/en/nsIFileInputStream

Princeton University. (2011 йил 21-Junio). *WordNet*. From http://wordnet.princeton.edu/wordnet/

Sheppy [Mozilla Firefox]. (2011 йил 21-Abril). *MDN Docs - NsIFileOutputStream*. From https://developer.mozilla.org/En/NsIFileOutputStream

SUBGURIM.NET. (2006 йил 07-Mayo). *Subgurim.net.* Retrieved 2011 йил 03-Abril from http://www.subgurim.net/Articulos/ajax-y-javascript/54/ajax-a-pelo-xmlhttprequest.aspx

Tonymec [Mozilla Firefox]. (2011 йил 27-Junio). *MDN Docs - XUL*. Retrieved 2011 йил 6-Agosto from https://developer.mozilla.org/En/XUL

Trevorh [Mozilla Firefox]. (2010 йил 20-10). *MDN Docs - nsILocalFile*. From nsILocalFile: https://developer.mozilla.org/en/XPCOM_Interface_Reference/nsILocalFile

Trevorh [Mozilla Firefox]. (2010 йил 22-10). *MDN Docs - nsIScriptableUnescapeHTML*. Retrieved 2011 йил 1-Abril from nsIScriptableUnescapeHTML: https://developer.mozilla.org/en/nsIScriptableUnescapeHTML

Troyano, J. A. (2002 йил 16-Mayo). *WordNet.* Retrieved 2011 йил 19-Junio from http://www.lsi.us.es/~troyano/documentos/wordnet.pdf

Valdés, D. P. (2007 йил 3-Julio). *Maestros del web*. Retrieved 2011 йил 17-Julio from http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/

W3.ORG. (2004 йил 4-Abril). Retrieved 2011 йил 12-Julio from http://www.w3.org/2005/03/DOM3Core-es/nucleo.html

W3SCHOOLS. (2011). *w3schools - HTML.* From http://www.w3schools.com/tags/default.asp

W3SCHOOLS. (2011). *W3Schools - Javascript*. Retrieved 2011 йил 10-Julio from http://www.w3schools.com/js/default.asp

7 Anexo 1: Manual de usuario

7.1 Descripción

El presente documento, pretende poner en manos del usuario final una herramienta con la que poder iniciarse en el uso de la extensión "**Webfiltering toolbar**".

7.2 Requisitos

Para completar con éxito la instalación de la extensión es necesario disponer en el equipo una versión actualizada del navegador Mozilla Firefox. Si fuera necesario se puede realizar la descarga del navegador desde la web oficial de Mozilla firefox. (Mozilla Firefox, 2011): Mozilla Firefox. (2011). *Navegador web Firefox*. Recuperado el 10 de Septiembre de 2011, de http://www.mozilla.org/es-ES/firefox/

7.3 Instalación de la Extensión

Es posible realizar la instalación de varias maneras distintas, la que se explica en el actual manual será descargando el paquete de instalación desde la web del autor.

7.3.1 Descarga e instalación desde la web del autor

En primer lugar debemos acceder a la sección de descargas de la web del autor donde descargaremos la última versión disponible. Será posible acceder dirigiéndonos a la siguiente dirección URL (http://users.dsic.upv.es/~jsilva/webfiltering/index.htm#downloads).

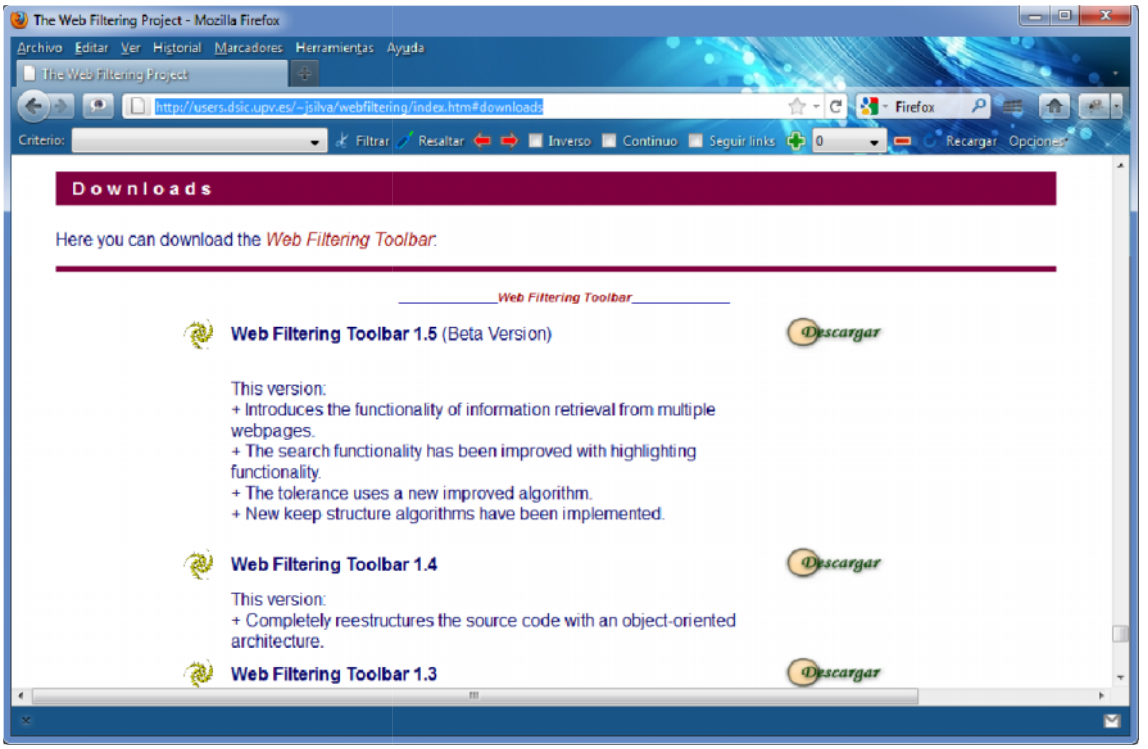


Ilustración 59: Versiones disponibles para descargar

Una vez visualizamos la web, descargaremos el paquete de instalación haciendo clic en el botón “Descargar” que aparecerá a la derecha de la versión correspondiente de la herramienta.

En este momento se mostrará al usuario el formulario típico de descarga de ficheros, mostrado en la Ilustración 60.

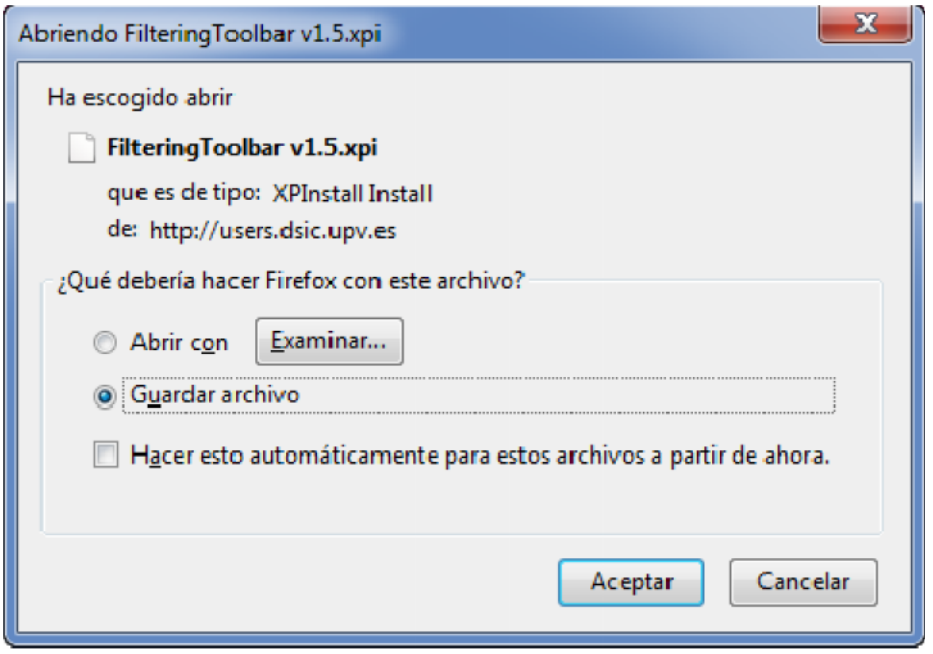


Ilustración 60: Descarga de Paquete de instalación

Donde seleccionaremos la opción “Abrir con”, donde debemos seleccionar la aplicación “Mozilla Firefox” si estuviera disponible entre las que se muestran en la Ilustración 61.

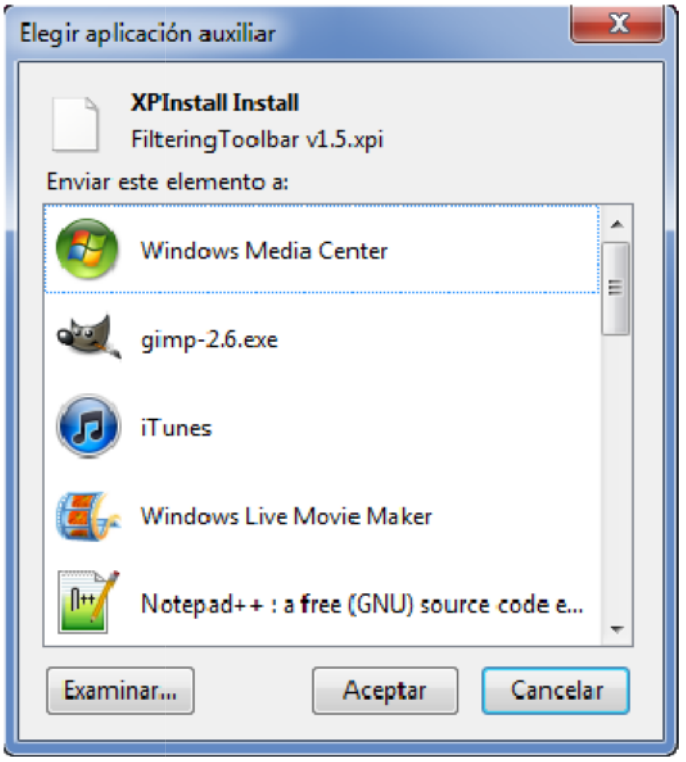


Ilustración 61: Abrir con Aplicación seleccionada

Si no se encuentra entre las mostradas anteriormente se debe hacer clic con el ratón en el botón “Examinar” y navegaremos por los directorios hasta el directorio de instalación de Mozilla Firefox, seleccionando su ejecutable, tal y como se muestra en la Ilustración 62.

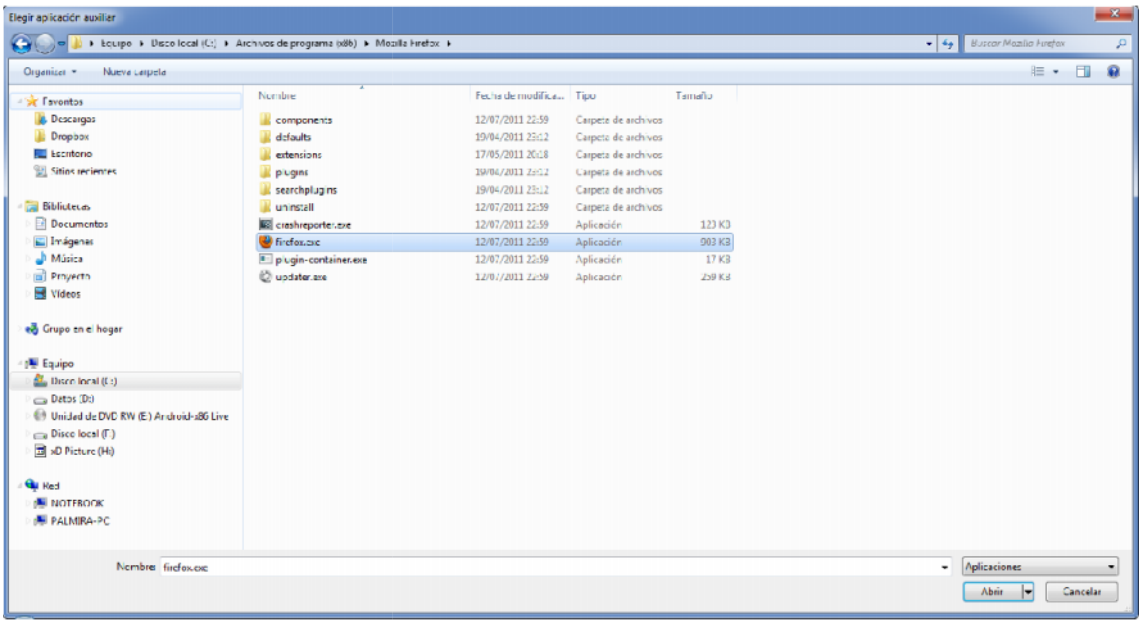


Ilustración 62:Directorio Mozilla Firefox

Una vez seleccionada, seleccionaremos la opción “Abrir”, con lo que nos devolverá al formulario de descarga, donde aparecerá seleccionada la aplicación que se ha elegido anteriormente.

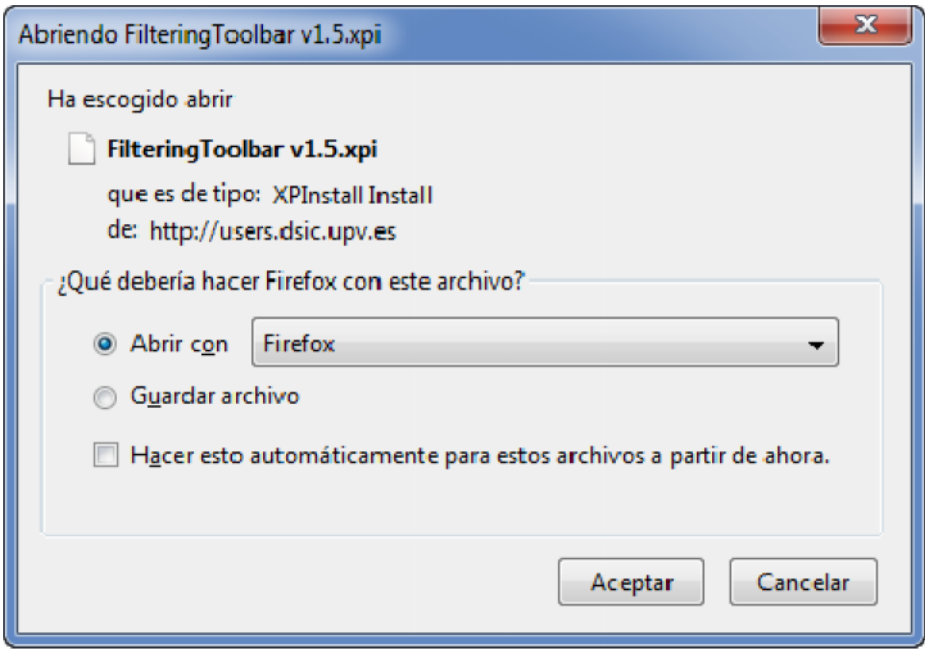


Ilustración 63: Abrir fichero con Firefox

En este momento haremos clic en el botón “Aceptar” y a partir de ese momento es el navegador Mozilla Firefox el encargado de continuar con el resto del proceso de instalación.

7.4 Elementos de la Extensión

7.4.1 Campo criterio

El campo criterio es donde es posible introducir el criterio de búsqueda sobre el que se realizará el filtrado de la página que se visualiza en el navegador **Mozilla Firefox**.

Ilustración 64: Criterio de búsqueda

Se podrán seleccionar criterios utilizados con anterioridad haciendo clic en el desplegable tal como se observa en la siguiente figura.

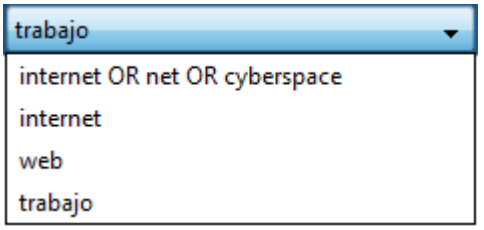


Ilustración 65: Histórico de búsqueda

7.4.2 Botón Filtrar

Este botón ejecuta el filtro sobre la página visitada en cada momento por el usuario, utilizando el criterio introducido.

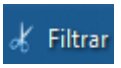


Ilustración 66: Botón Filtrar

7.4.3 Botón Resaltar

Tal como indica su propio nombre, resalta en amarillo las palabras coincidentes con el criterio de búsqueda.

Ilustración 67: Botón Resaltar

En la siguiente figura se muestra un ejemplo de utilización en la web, en el que se ha utilizado como criterio de búsqueda el concepto “web”. Se puede observar que aparecen resaltados tanto palabras coincidentes, como elementos que contienen entre sus etiquetas texto coincidente con el criterio de búsqueda.

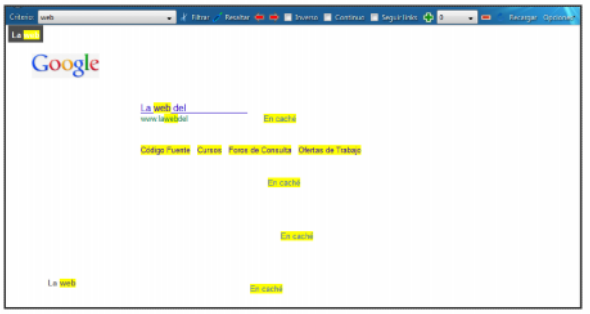


Ilustración 68: Ejemplo Resaltar

7.4.4 Botones de desplazamiento

El control al que se refiere este apartado del manual permite al usuario, desplazarse por el texto de la página que coincide con el criterio introducido por el usuario, por orden de aparición.

Ilustración 69: Botones de desplazamiento

En este caso se resaltará únicamente una palabra coincidente, saltando cada vez que se presione una de las flechas a la siguiente en orden de aparición.

7.4.5 Casilla de verificación de Filtro Inverso

Inicialmente este elemento de la barra no se encuentra seleccionado. Si en algún momento se encontrase marcado, se activa el filtrado inverso, que consiste en ocultar los elementos que contengan texto coincidente con el criterio de búsqueda.

Ilustración 70: Botón inverso

Podemos ver el ejemplo realizado en una conocida web de dispositivos, en la que se realiza un filtrado inverso sobre la palabra “ipod”.

Podemos observar las diferencias entre la Ilustración 71 y la Ilustración 72, en la que desaparecen todos los elementos que coinciden con el criterio de búsqueda nombrado.

Ilustración 71: Página Tienda Apple.

Ilustración 72: Página Tienda Apple después de aplicar el filtro

7.4.6 Casilla de verificación de Filtro Continuo

La activación de este control activa en la barra “Webfiltering toolbar”, la funcionalidad de filtrado continuo. Conforme se acceda a nuevas páginas, se realizará el filtrado con el criterio de búsqueda introducido, sin necesidad de hacer clic en el botón “Filtrar”. Ver Botón Filtrar.

Ilustración 73: Casilla de verificación de FiltroContinuo

7.4.7 Casilla de verificación de Filtro Seguir links

El control actual permite activar la opción de seguir links que, como su propio nombre indica, realiza una búsqueda a lo largo de todas las páginas que se encuentran enlazadas desde la página visitada y devuelve como resultado las secciones de estas páginas que contienen información coincidente con el criterio de búsqueda seleccionado.

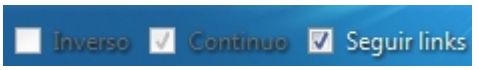


Ilustración 74: Control Seguir links

Al seleccionar este control, automáticamente se desactivan las búsquedas “Inverso” y “Continuo”, que no son compatibles con este tipo de filtrado.

Debido a que el número de enlaces que existe en una página puede ser muy alto, puede llevar mucho tiempo, por ello se puede especificar un tiempo de respuesta máximo en el que se devolverá el resultado. Por defecto son 10 segundos, pero existe la posibilidad de personalizar este valor.

Además es posible personalizar también la estructura con la que se mostrará el resultado de la búsqueda. Ver

Opciones de Seguir links.

7.4.8 Tolerancia

Este control se refiere a la tolerancia en la búsqueda para incluir en el resultado secciones de la página colindantes a los elementos que contienen información que coincide con el criterio de búsqueda.

Ilustración 75: Control Tolerancia

La tolerancia que se aplicará al realizar el filtro sobre la página visualizada se puede aumentar o disminuir utilizando los botones  y  , o seleccionando en el menú desplegable el valor deseado. Además se puede configurar la tolerancia por defecto en las opciones generales de la aplicación. Ver Opciones de configuración.

7.4.9 Recargar

Al hacer clic en el botón “Recargar”, se cargará de nuevo la página visitada con su aspecto original.

Ilustración 76: Botón Recargar

7.4.10 Opciones

Este control es el que da acceso a las Opciones generales de la aplicación, así como a las Opciones del Diccionario. Ver Opciones de configuración.

Ilustración 77: Botón Opciones

7.5 Opciones de configuración

7.5.1 Acceso a las Opciones de Configuración

A continuación se realizará una explicación de las opciones de configuración aplicables a la hora de utilizar la funcionalidad de Filtrado. Para acceder a la ventana de opciones, es necesario acceder al menú "Opciones" que se encuentra en la barra **"Webfiltering toolbar"**. Tal y como se muestra en la siguiente figura.

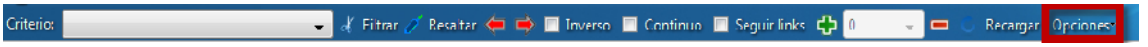


Ilustración 78: Extensión Webfiltering toolbar

Será necesario hacer clic con el botón izquierdo del ratón sobre el elemento "Opciones" para visualizar los distintos accesos a configuraciones, "Opciones", y "Diccionario", así como los elementos que proporcionan acceso a información sobre la extensión, "Ayuda…" y "Acerca de…".

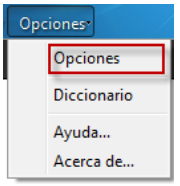


Ilustración 79: Menú Opciones

Si hacemos clic en la opción denominada "Opciones", se obtendrá acceso a la configuración de las opciones relacionadas con la estructura del filtrado. Ver

Sin embargo si hacemos clic en el menú “Diccionario”, aparecerá en el navegador el formulario de configuración del Diccionario. Ver Opciones de Diccionario.

Ilustración 81: Opciones de Diccionario

Opciones de Filtro. Además a través del mismo formulario se podrán modificar las preferencias referentes al modo de uso seguir links. Ver

Opciones de Seguir links.

Una vez hacemos clic en Opciones, aparecerá el siguiente formulario.

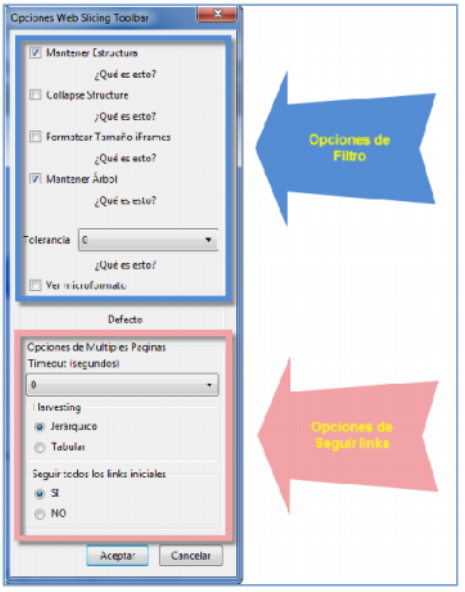


Ilustración 80: Formulario de Opciones

Sin embargo si hacemos clic en el menú “Diccionario”, aparecerá en el navegador el formulario de configuración del Diccionario. Ver Opciones de Diccionario.

Ilustración 81: Opciones de Diccionario

7.5.2 Opciones de Filtro

Tal y como se ha comentado en el punto anterior, para acceder a las Opciones de Filtro, se debe hacer clic en la primera opción que aparece en el desplegable, que tiene por nombre “Opciones”, tras la realización de esta acción, se mostrará al usuario un formulario en el navegador Mozilla Firefox, en el que se visualizan las opciones que se explican en este apartado del manual.

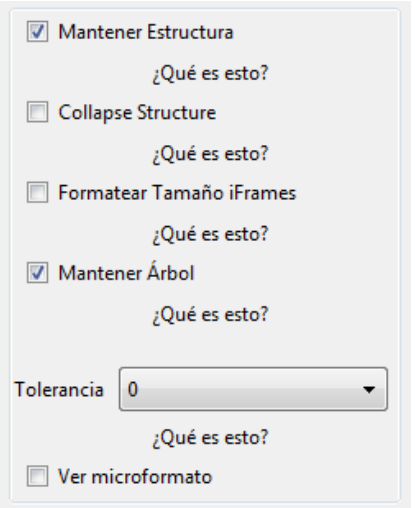


Ilustración 82: Opciones de Filtro

Las opciones que a continuación se detallan permiten al usuario especificar la estructura de los resultados que serán obtenidos tras realizar el filtrado sobre la página en la que se utiliza la

herramienta, que serán aplicadas al mismo tiempo que cualquier configuración que tenga que ver con el modo de uso.

- **Tolerancia:** permite al usuario especificar el nivel de detalle con el que desea realizar el filtrado. Cuanto mayor sea el valor seleccionado, mayor información aparecerá en el resultado, mientras que si se selecciona el valor "0" se visualizarán solo los elementos que coincidan con el criterio especificado en el campo de búsqueda.
- **Mantener estructura:** si aparece seleccionada indica que al obtener el resultado se respetará la estructura de la página original, mostrando tan solo los elementos que coinciden con el elemento de búsqueda.
- **Comprimir estructura:** si aparece marcada esta opción en lugar de "Mantener estructura", se comprimirá la información encontrada en la web sobre la que se realiza el filtrado al inicio de la página, eliminando todos las secciones que no coincidan con el criterio de búsqueda.
- **Formatear tamaño de frames:** si aparece seleccionada indica que se desea mantener el tamaño original de los frames que tenía la página. En caso contrario indica que el tamaño de los frames aparecerá formateados de acuerdo a su contenido.
- **Mantener el árbol:** permite visualizar los elementos que contengan el criterio de búsqueda elegido por el usuario
- **Ver microformato: A activar esta opción, la herramienta busca y recupera automáticamente microformatos en la página que coincidan con el texto utilizado en la búsqueda.**

En este momento se puede seleccionar entre las distintas opciones, la que mejor se adapte a las necesidades del usuario, pudiendo utilizarla junto a los distintos modos de uso que tiene la aplicación, que se detallan en el siguiente punto.

7.5.3 Opciones de Seguir links

El acceso a la configuración del modo de funcionamiento de Seguir links o Múltiples páginas, se realiza de la manera explicada en el apartado Acceso a las Opciones de Configuración.

De esta forma tendremos acceso al formulario que se muestra en la Ilustración 83. Donde podemos observar los distintos elementos de configuración que tiene este modo de funcionamiento. Ver Casilla de verificación de Filtro Seguir links.

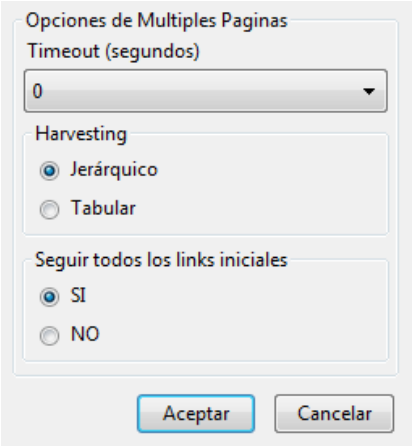


Ilustración 83: Opciones de Múltiples Páginas

En la Ilustración 83, se observan las distintas opciones de configuración que tiene la barra "Webfiltering toolbar" con respecto a su funcionalidad de recuperación de secciones en múltiples páginas.

Timeout: Este campo delimita el tiempo total que la herramienta "Webfiltering toolbar" dedicará a la búsqueda de información en la maraña de links disponibles en la página.

Harvesting: Este elemento de selección se utiliza para especificar la forma en la que aparecerán los resultados de la búsqueda en múltiples páginas. El modo **Jerárquico** (Ver Ilustración 84Ilustración 84: Búsqueda Múltiples Páginas Jerárquica) devolverá el resultado de la búsqueda como si de una estructura de directorios se tratara. Mientras que el modo **Tabular** (Ver Ilustración 85) mostraría el enlace y a continuación la página que contiene el enlace, para cada uno de los enlaces que se encuentre en la página, si existen varios enlaces en la página se mostrarán después de la página que los contiene.

Ambos modos se encuentran limitados por el **Timeout** especificado por el usuario con anterioridad.

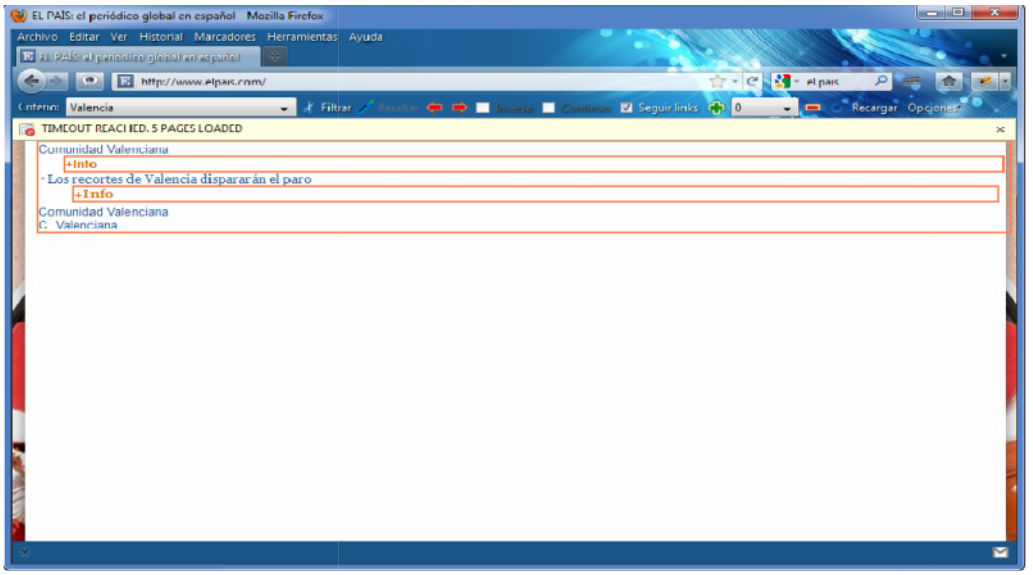


Ilustración 84: Búsqueda Múltiples Páginas Jerárquica

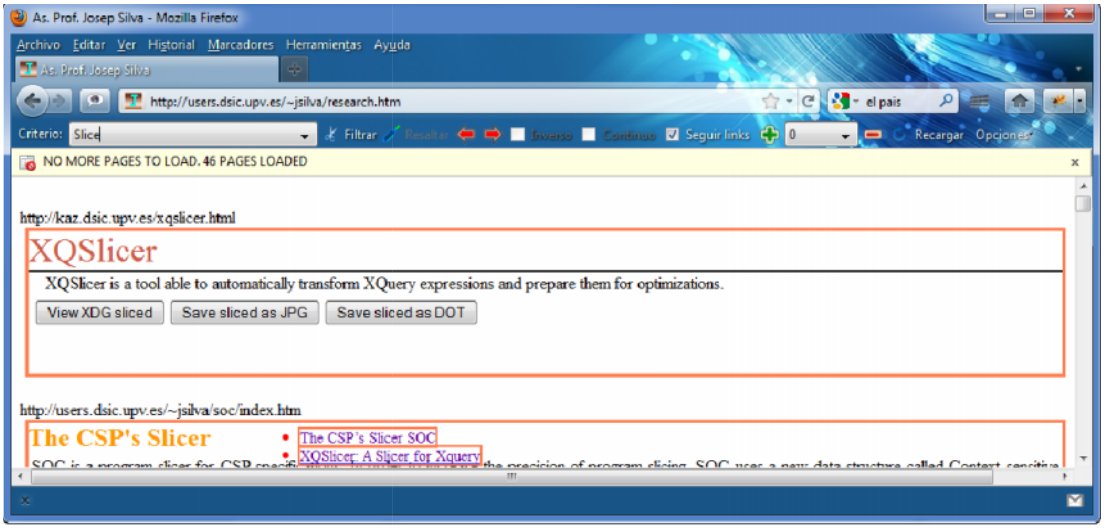


Ilustración 85: Búsqueda Múltiples Páginas Tabular

Seguir links iniciales: Al marcar esta opción, forzamos a la herramienta a explorar todos los enlaces de la página donde iniciamos la búsqueda. De esta manera se garantiza un espacio de búsqueda de varias páginas aunque en la página inicial no aparezca la palabra buscada. En todas las páginas accedidas a continuación, solamente se seguirán analizando aquellos enlaces que sean relevantes de acuerdo al criterio de búsqueda.

7.5.4 Opciones de Diccionario

Una vez accedemos al formulario de configuración del diccionario se observa que existen tres opciones de configuración, las correspondientes al Estado (Activo/Inactivo), el Modo de Uso (Online/Local) y la Velocidad (Rápida, Media, Lenta).

7.5.4.1 Configuración del estado

El estado del diccionario es el que determina si antes de realizar el filtrado sobre la página (utilizando el criterio de búsqueda) se realizará una búsqueda previa del concepto seleccionado en el diccionario, o no.

De manera que, si el estado del diccionario es "Inactivo" (ver Ilustración 86: Estado del diccionario), no se utilizará el diccionario durante la utilización de la herramienta. En este caso las demás opciones de configuración aparecerán deshabilitadas, puesto que no será posible su utilización.

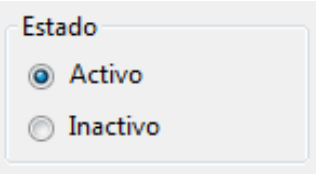


Ilustración 86: Estado del diccionario

En el caso de que el estado fuera Activo, se utilizaría el servicio proporcionado por la Universidad de Princeton, que utilizaría el criterio introducido por el usuario para recuperar conceptos relacionados con el mismo y utilizarlos en el momento de realizar el filtro. Ver WORDNET.

7.5.4.2 Configuración del modo de uso

El modo de uso del Diccionario es configurable tan solo si el Estado del diccionario es Activo, y sirve para determinar dónde se producirá la búsqueda de los conceptos relacionados con el concepto introducido por el usuario.

Si el modo seleccionado fuese "**Online**", tal como se ha comentado anteriormente, la extensión recuperaría la información proporcionada directamente del servicio web de Wordnet, mientras que si se trata del modo "**Local**", la siguiente búsqueda se realizaría la búsqueda de manera Local, y si esta fuera la primera, se procedería a la descarga del Diccionario a carpeta de configuración de Mozilla Firefox de manera automatizada.

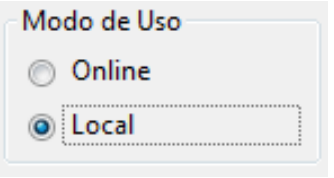


Ilustración 87: Modos de Uso

7.5.4.3 Velocidad de respuesta

La velocidad de respuesta aparece desactivada si se encuentra seleccionado el Modo de uso Local. Solo está disponible la posibilidad de determinar la velocidad de respuesta, si está habilitado el modo de uso "Online".

La extensión mostrará más resultados en función de la velocidad que se haya seleccionado, de manera que a mayor velocidad, se devolverán menor número de resultados.

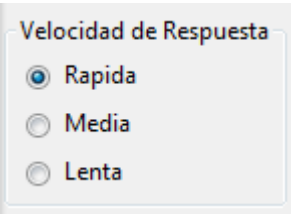


Ilustración 88: Velocidad de Respuesta

7.6 Modos de utilización

Desde el momento en que se publicó la primera versión de esta herramienta se han añadido nuevos elementos y funcionalidades a la extensión, de manera que a continuación pasaremos a detallar la utilización de la extensión enumerando cada uno de los modos de utilización de ***"Webfiltering toolbar"***.

7.6.1 Filtrado Normal

Para realizar el filtrado utilizando este modo de uso será necesario observar que aparecen sin marcar los elementos de la barra que activan los modos de uso "**Inverso**", "**Continuo**", o "**Seguir links**" (marcados con el rectángulo rojo, en la Ilustración 89).

Ilustración 89Ilustración 89 Extensión Webfiltering toolbar

Una vez se ha comprobado que la barra tiene la configuración adecuada se podrá realizar el primer filtrado en el modo que se ha denominado "Normal". [1]Para ello se introducirá en el campo "Criterio" una o varias palabras, entre las que se podrá utilizar una serie de operadores para especificar el criterio de búsqueda que será aplicado sobre la página.

Para la realización de los ejemplos que se muestran a continuación ha sido utilizada la web del proyecto "**Webfiltering toolbar**" que se muestra cargada en la siguiente figura.

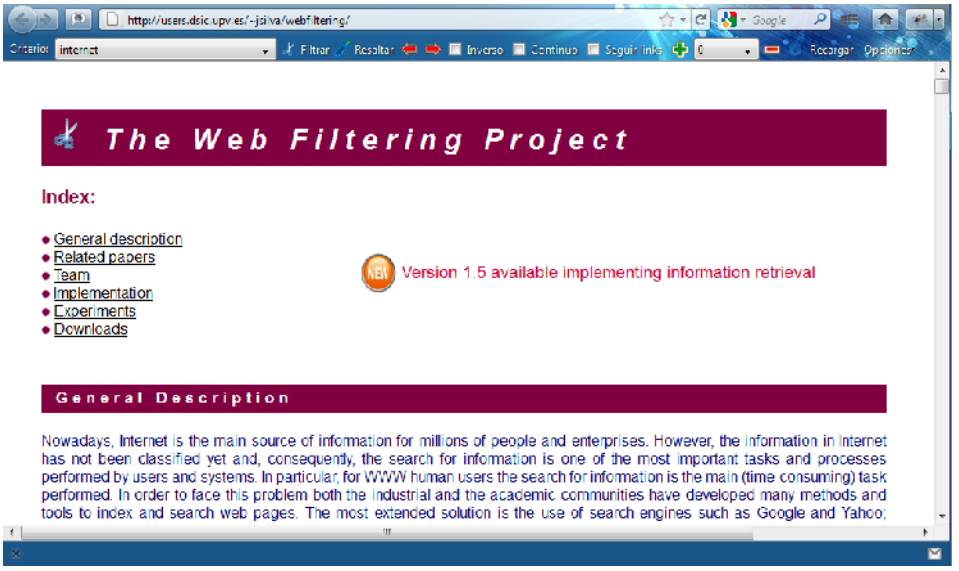


Ilustración 90: Web de la extensión Webfiltering toolbar

A continuación se realizaran distintos ajustes en las Opciones de la extensión, en función del resultado que se desea obtener. Se puede obtener más información 76en la página 76 de este manual.

[1] Los operadores que actualmente acepta la extensión son los siguientes: **&&, and, AND, (espacio), ||, or, OR**.

7.6.1.1 Mantener estructura

En el ejemplo que se muestra a continuación se ha utilizado la web mostrada en la Ilustración 90, y se han configurado las opciones por defecto, mostradas en la Ilustración 91.

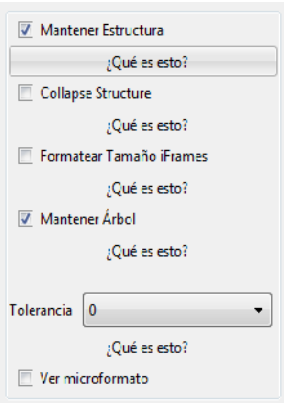


Ilustración 91: Opciones por Defecto

Después de realizar los ajustes apropiados en las "Opciones", se ha introducido la palabra "internet" en el criterio de búsqueda, y después se ha presionado el botón "**Filtrar**", con lo que se ha obtenido el resultado que se muestra en la

Ilustración 92.

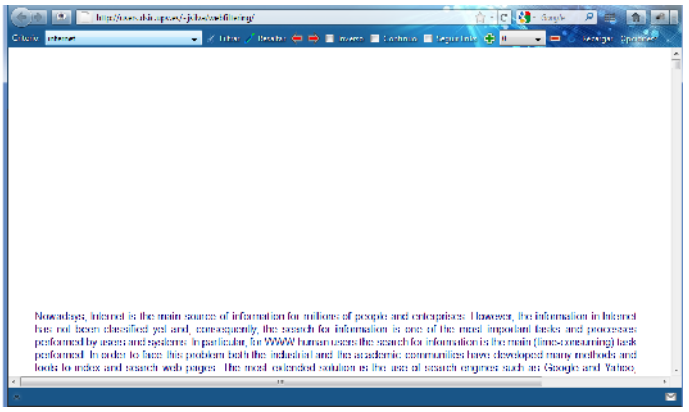


Ilustración 92: Filtrado Mantener Estructura

Se puede observar que lo que se ha logrado con esta configuración es que se visualicen tan solo las secciones de la página que tienen coincidencias con el criterio de búsqueda, manteniendo la estructura de la página.

7.6.1.2 Comprimir Estructura

Del mismo modo que en el ejemplo anterior se realiza la configuración de las opciones tal como aparecen en la Ilustración 93.

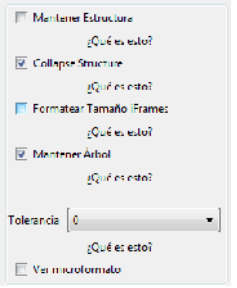


Ilustración 93: Opciones de configuración

En este caso tal como se ha especificado en la configuración, la estructura obtenida tenía que ser comprimida al inicio de la página. Como criterio de búsqueda se ha utilizado el concepto “internet”. Ver Ilustración 94.

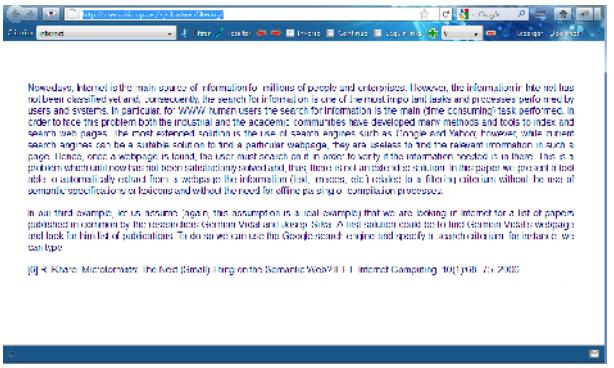


Ilustración 94: Estructura comprimida al Inicio

7.6.2 Filtrado con Diccionario

Para utilizar el Diccionario, se debe acceder a las Opciones de diccionario, tal y como se explica en el punto **Opciones de Diccionario**, y verificar que el Estado esté “Activo”, también deberemos seleccionar un Modo de uso y en caso de que se trate del modo de uso “Online”, determinar la velocidad de respuesta esperada.

Es necesario conocer que aunque es posible introducir palabras en cualquier idioma, el diccionario únicamente reconocerá las introducidas en el idioma Inglés y que además se encuentren en la Base de datos de **Wordnet**.

A modo de ejemplo podemos configurar el Diccionario para utilizarlo en modo local. Y realizaremos una búsqueda sobre la página del proyecto, nuevamente. Ver Ilustración 90.

Los ajustes necesarios son los que se muestran a continuación. Ilustración 95

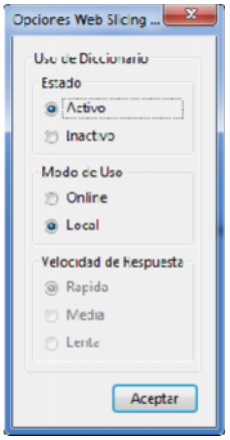


Ilustración 95: Utilización de Diccionario en Modo Local

Una vez establecidos, introduciremos en el criterio de búsqueda la cadena “information” y hacemos clic en el botón “Filtrar”. Al tratarse de la primera vez que realizamos la búsqueda local, se nos informará debidamente de que se está realizando la descarga del Diccionario.

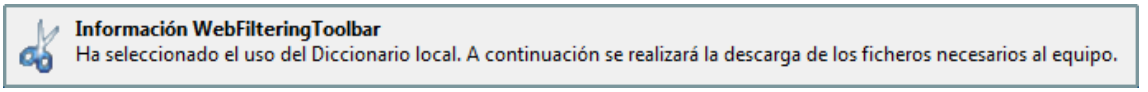


Ilustración 96: Pop-up Descarga de Diccionario

Después de unos instantes la extensión volverá a mostrarnos un pop-up indicándonos que ha terminado la descarga del fichero.

Ilustración 97: Información Descarga finalizada

Una vez realizada la descarga se realizará la búsqueda en el fichero y se añadirán al criterio de búsqueda los conceptos relacionados semánticamente con el concepto seleccionado previamente. Mostrando la página habiendo realizado el filtrado. En nuestro caso el concepto seleccionado previamente se trataba de "Information" y podemos observar que tras realizar la búsqueda aparecen junto a él otros conceptos relacionados con él. Ver Ilustración 98.

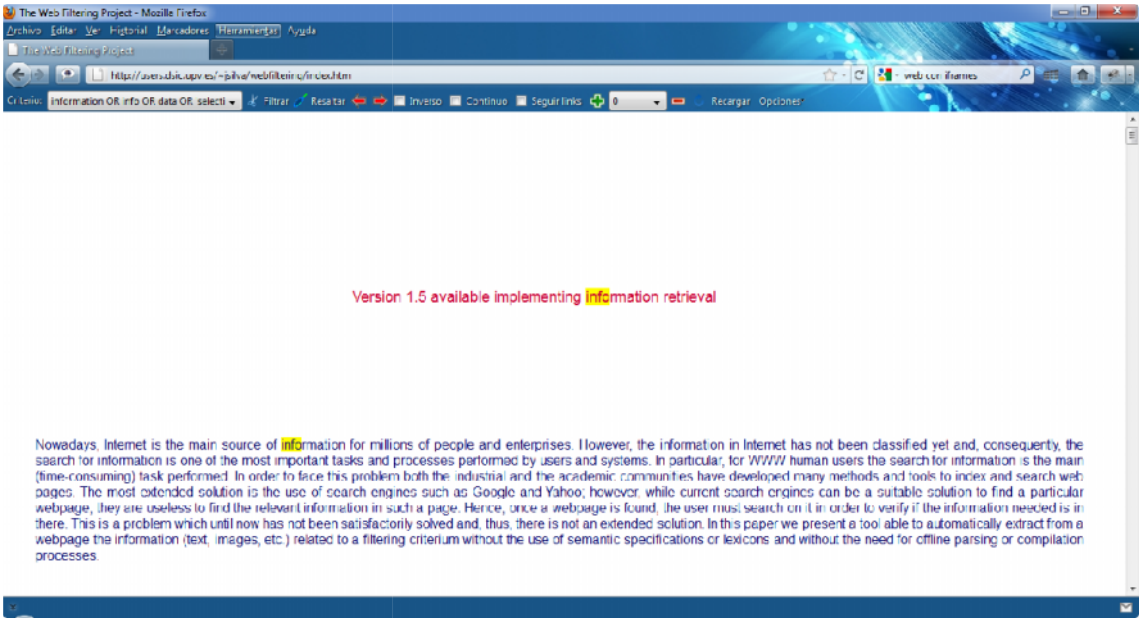


Ilustración 98: Filtrado con Diccionario