

BADWORK (control de acceso)

**2016**

Carlos Cañas Agenjo

IES Virgen de la Paz

08/06/2016



Índice

[1. Justificación del proyecto y objetivos. 3](#_Toc452877872)

[2. Introducción 4](#_Toc452877873)

[3. Metodología y desarrollo del proyecto 5](#_Toc452877874)

[3.1. Metodología 5](#_Toc452877875)

[3.2. Desarrollo del proyecto 6](#_Toc452877876)

[3.2.1. Ciclo 0. 6](#_Toc452877877)

[3.2.1.1. Comunicación con el cliente / Objetivo. 6](#_Toc452877878)

[3.2.1.2. Planificación. 6](#_Toc452877879)

[3.2.1.2.1. Diagrama gráfico de Gantt 7](#_Toc452877880)

[3.2.2. Ciclo 1. Liberación de BadWork 0.0.1 8](#_Toc452877881)

[3.2.2.1. Comunicación con el cliente / Objetivo. 8](#_Toc452877882)

[3.2.2.2. Planificación. 8](#_Toc452877883)

[3.2.2.2.1. Diagrama gráfico de Gantt 8](#_Toc452877884)

[3.2.2.3. Análisis de riesgo 9](#_Toc452877885)

[3.2.2.3. Ingeniería 9](#_Toc452877886)

[3.2.2.4. Evaluación del cliente / Casos de uso y pruebas 15](#_Toc452877887)

[3.2.2.5. Construcción y entrega 16](#_Toc452877888)

[3.2.3. Ciclo 2. Liberación de BadWork 1.0.1. 17](#_Toc452877889)

[3.2.3.1. Comunicación con el cliente / Objetivo. 18](#_Toc452877890)

[3.2.3.2. Planificación. 18](#_Toc452877891)

[3.2.3.2.1. Diagrama gráfico de Gantt 18](#_Toc452877892)

[3.2.3.3. Análisis de riesgo 19](#_Toc452877893)

[3.2.3.4. Ingeniería 19](#_Toc452877894)

[3.2.3.4. Evaluación / Pruebas 26](#_Toc452877895)

[3.2.3.5. Construcción y entrega 27](#_Toc452877896)

[4. Resultado y discusión 34](#_Toc452877897)

[5. Conclusiones 35](#_Toc452877898)

[6. Bibliografía y referencias 36](#_Toc452877899)

[7. Anexos/Otros 37](#_Toc452877900)

[7.1. Anexo I 37](#_Toc452877901)

[7.2. Anexo II 37](#_Toc452877902)

[7.3. Anexo III 41](#_Toc452877903)

# Justificación del proyecto y objetivos.

Los navegadores populares Chrome y Firefox nos dan la posibilidad de añadir mejoras y personalizarlos al uso cotidiano de cada usuario, ésta característica es lo que me ha llevado a la elección de este proyecto.

Estos programas que instalamos en nuestros navegadores, hacen de ellos una herramienta muy potente debido a que no sólo realizan su uso funcional, sino que la optimizan y pueden tener diferentes usos y características.

El desarrollo de esta aplicación la realizaré para el navegador Mozilla Firefox ya que comparto su filosofía de que el conocimiento debe ser libre, gratuito y debe ser difundido para toda la comunidad del mundo.

El proyecto que voy a realizar será un control de acceso, centrado en la actividad laboral de una empresa, cuya función es que el navegador bloquee las páginas no deseadas por el administrador.

Mi elección ha sido debida a que en los últimos años se ha visto un aumento de bajo rendimiento laboral debido a la libertad que nos da internet, lo que conlleva que los empleados hagan un mal uso de las nuevas tecnologías y en sus horas de actividad profesional no sólo accedan a contenido relacionado con el desarrollo de su profesión, sino también a contenido personal y de ocio.

Los objetivos que me planteo para dicho proyecto son los siguientes:

* Principal:

1. Bloquear las páginas no deseadas por el administrador.
2. Gestionar las páginas bloqueadas

* Secundarios:

1. Prevenir el mal uso de las nuevas tecnologías.
2. Aumentar el rendimiento de los empleados, por no tener acceso libre a internet.

# 2. Introducción

El control de acceso es una herramienta que se puede habilitar por parte del administrador de forma rápida y sencilla, además de personalizarlo según las necesidades de cada usuario.

Su funcionalidad es filtrar el contenido web y la actividad de registro.

La habilitación y deshabilitación de dicho control de acceso normalmente está bloqueada por una contraseña, que únicamente conoce el administrador.

Las principales características del control de acceso son:

* Bloqueo de páginas web: restringe el acceso a contenido inapropiado por medio del bloqueo de ciertos sitios web.
* Restricción de acceso por categorías: aparecen en conjuntos, se muestra un listado donde se selecciona la categoría deseada a bloquear.
* Registro de actividad: permite observar un registro detallado de la actividad dentro del control de acceso.
* Selección rápida: se crean usuarios y según cada rol, se le permite el acceso a determinado contenido o no. Ejemplo: niño, adolescente, padre.
* Control de tiempo: limita el tiempo del ordenador conectado a internet.

El control de acceso es aplicable a diversos campos:

* Control en el hogar familiar, dónde los padres serían los administradores.
* Control escolar, en los centros educativos, bloqueando ciertos contenidos inapropiados e innecesarios durante las horas lectivas.
* Control laboral, en las empresas, para que así los empleados no hagan un mal uso de las nuevas tecnologías ni abusen del libre acceso por internet.

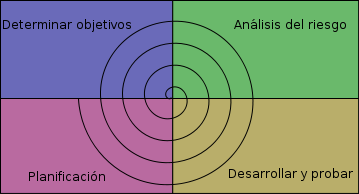
Mi proyecto de control de acceso se va a centrar en un control laboral, dentro de los ordenadores de una empresa, realizando un bloqueo de páginas web, para que los empleados puedan acceder sólo a los contenidos necesarios que se relacionen con su desarrollo laboral y así conseguir un aumento de su rendimiento. Se llamará “BADWORK” (mala palabra).

# 3. Metodología y desarrollo del proyecto

## 3.1. Metodología

La metodología que voy a seguir para desarrollar este proyecto será un modelo en espiral.

Cada bucle representa un conjunto de actividades, esto formará una caracola, que tiene dos dimensiones, la angular (indica el avance del proyecto) y la radial (indica el aumento de costes, porque por cada nueva interacción quiere decir que se pasa más tiempo desarrollando).

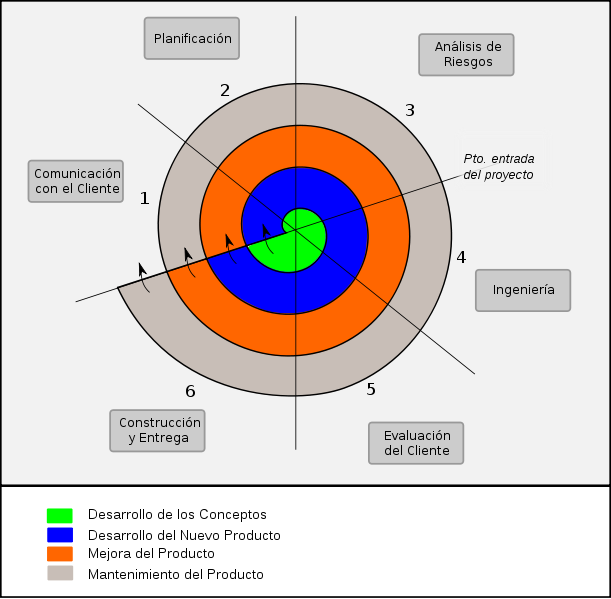
[](https://www.google.es/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwjig5HqlJbMAhVElxoKHaXhAH8QjRwIBw&url=https://es.wikipedia.org/wiki/Desarrollo_en_espiral&psig=AFQjCNHyma1VyqUC40jiTRmyewAZ2VANDQ&ust=1460999311743809)

Cada ciclo tendrá:

* Objetivos.
* Análisis de riesgos y alternativas.
* Desarrollo y prueba.
* Planificación, donde se probará finalmente y se verá si en necesario comenzar un nuevo ciclo.

El modelo en espiral de divide en regiones de tareas:

* Comunicación con el cliente.
* Planificación.
* Análisis de riesgos.
* Ingeniería: tareas para construir representaciones de la aplicación.
* Evaluación del cliente: casos de usos y pruebas.
* Construcción y entrega.



## 3.2. Desarrollo del proyecto

### 3.2.1. Ciclo 0.

En este ciclo se abordará el estudio de la tecnología XUL. Este ciclo que tomará como una fase de estudio previo del desarrollo del proyecto.

### 3.2.1.1. Comunicación con el cliente / Objetivo.

* El cliente solicita que realice un control de acceso para su empresa, su intención es que los empleados no puedan acceder a determinadas páginas web.

Objetivo: bloquear y desbloquear la página actual, y que esto quede guardado en la sesión.

### 3.2.1.2. Planificación.

Este apartado estará dividido en dos subcategorías:

* Definición de la tecnología.
* Diagrama gráfico sobre la planificación de las tareas a realizar.

#### XUL (XML Lenguaje de Interfaz de Usuario)

Esta tecnología fue creada para facilitar el desarrollo del navegador Mozilla Firefox. Es un lenguaje XML (eXtensible Markup Language), por lo que todas las características de XML también están incluidas en XUL.

La mayoría de aplicaciones necesitan ser desarrolladas usando una plataforma específica, convirtiendo el desarrollo de la misma en algo costoso. Sin embargo XUL permite ser implementada y modificada de forma fácil y rápida.

XUL tiene las ventajas de otros lenguajes XML como XHTML (eXtensible HyperText Markup Language) y también aplicar hojas de estilos para modificar la apariencia de la interfaz del usuario.

Estos son algunos elementos que pueden ser creados:

* Controles de entrada tales como cuadros de texto y cajas de chequeo.
* Barra de herramientas con botones u otros contenidos.
* Menús en barras de menú o menú emergente.
* Pestañas de diálogo.
* Árbol de información jerárquica o tabulada.
* Teclas de accesos directo.

Para desarrollar en XUL el usuario debe tener conocimientos de HTML (HyperText Markup Language), XML y CSS (Cascading Style Sheets).

Reglas que debe tener en cuenta:

* Los elementos en XUL y sus atributos deben introducirse en minúsculas, ya que XML distingue entre mayúsculas/minúsculas (a diferencia de HTML).
* Los valores de los atributos en XUL, deben colocarse entre comillas, aunque sean números.
* Los archivos XUL por lo general se dividen en cuatro ficheros, uno para la disposición de los elementos, otro para la definición del estilo, otro para declarar las entidades (usadas en las localizaciones) y otro para los "script". Además se pueden tener archivos para las imágenes o para datos específicos de una plataforma.

Para crear fácilmente el desarrollo de aplicaciones con esta tecnología es necesario instalar:

* IDE (integrated developement environment) “entorno de desarrollo interactivo”: XUL Explorer, facilita la creación de ficheros con extensión “.xul”.
* Remote xul manager (extensión de Firefox para desarrollar y probar ficheros XUL).

#### Justificación de cambio de tecnología

Comienzo a desarrollar el proyecto con la tecnología XUL, y me encuentro con varias dificultades que lo hacen incompatible, por lo que cambio a una tecnología más actual llamada ADD-ONS SDK (Software Development Kit). Este cambio es comentado y aceptado por el cliente ya que conlleva un retraso en la planificación del proyecto.

### 3.2.1.2.1. Diagrama gráfico de Gantt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tareas | Fecha de inicio Programada | Fecha de inicio | Días | Días Extra | Fecha fin |
| Inicio del Proyecto | 05/04/2016 | 05/04/2016 | 14 | 0 | 19/04/2016 |
| Documentación | 06/04/2016 | 06/04/2016 | 13 | 0 | 19/04/2016 |
| Planificación | 09/04/2016 | 09/04/2016 | 1 | 0 | 09/04/2016 |
| Ingeniería | 10/04/2016 | 10/04/2016 | 4 | 6 | 19/04/2016 |
| Implementación | 11/04/2016 | 14/04/2016 |  | 5 | 19/04/2016 |

### 3.2.2. Ciclo 1. Liberación de BadWork 0.0.1

En este ciclo se hará una aplicación para bloquear y desbloquear páginas y se realizará la entrega con una aplicación firmada y preparada para instalar en nuestro navegador Firefox.

### 3.2.2.1. Comunicación con el cliente / Objetivo.

* El cliente solicita que realice un control de acceso para su empresa, su intención es que los empleados no puedan acceder a determinadas páginas web.

Objetivo: bloquear y desbloquear la página actual, y que esto quede guardado en la sesión.

### 3.2.2.2. Planificación.

#### ADD-ONS SDK

Usando Add-on SDK puedes crear complementos Firefox utilizando las tecnologías Web estándar: JavaScript, HTML y CSS. SDK incluye una API de JavaScript que se puede utilizar para crear complementos y herramientas de creación, funcionamiento, pruebas y empaquetado llamadas JPM que será descrita en el apartado de ingeniería.

### 3.2.2.2.1. Diagrama gráfico de Gantt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tareas | Fecha de inicio Programada | Fecha de inicio | Días | Días Extra | Fecha fin |
| Inicio del Proyecto | 20/04/2016 | 20/04/2016 | 16 | 0 | 05/05/2016 |
| Documentación | 21/04/2016 | 21/04/2016 | 15 | 0 | 05/05/2016 |
| Planificación | 24/04/2016 | 24/04/2016 | 2 | 0 | 25/04/2016 |
| Ingeniería | 26/04/2016 | 26/04/2016 | 8 | 0 | 03/05/2016 |
| Implementación | 04/05/2016 | 04/05/2016 | 1 | 0 | 05/05/2016 |

### 3.2.2.3. Análisis de riesgo

Llevo a cabo el estudio de las posibles amenazas y eventos no deseados, evaluando y proponiendo diferentes alternativas:

* Al buscar información sobre los controles de acceso y el tipo de tecnología usada habitualmente, una de las dificultades que se me presentan es que la mayoría de la documentación viene en lengua extrajera. Por lo que veo el idioma como un posible riesgo. Solucionable al poder utilizar software para traducir idiomas.
* Otro de los riesgos que me planteo es encontrarme con versiones no actualizadas de la tecnología a utilizar para desarrollar el proyecto, como alternativa a esto me propongo intentar realizar el proyecto con la tecnología más óptima.
* Retraso en los plazos de entrega, por la necesidad de solucionar diferentes riesgos. Decido añadir algún día de margen al plazo de entrega.

### 3.2.2.3. Ingeniería

Para desarrollar el proyecto usaré las siguientes herramientas:

* Editor de texto Sublime Text
* Navegador Firefox con las siguientes extensiones instaladas: firebug, colorzilla.

Para crear complementos para Firefox utilizando el SDK, tenemos una serie de requisitos:

* Navegador Firefox este en la versión 38 o superior.
* Tener instalado Node.js.
* Tener instalado el paquete JPM.

JPM es distribuido por un gestor de paquetes nodo NPM que podemos conseguir a través de Node.js.

Una vez instalado Node.js ejecutaremos los siguientes comandos en nuestra terminal:

* **git clone https://github.com/mozilla-jetpack/jpm.git**
* **cd jpm**
* **npm install**
* **npm link**

La herramienta JPM es el reemplazo para [CFX](https://developer.mozilla.org/en-US/Add-ons/SDK/Tools/cfx) . Se le permite probar, ejecutar, y crear paquetes de complementos a través de la línea de comandos.

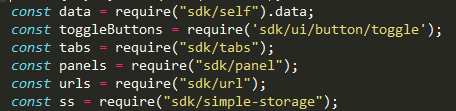
Estos son algunos de los comandos que puede utilizar:

|  |  |
| --- | --- |
| jpm init | Crear una estructura del add-ons a desarrollar |
| jpm run | Lanza una instancia a Firefox con el complemento instalado |
| jpm test | Ejecuta las pruebas de la unidad del complemento |
| jpm xpi | Empaqueta todos los archivos del complemento en XPI, que es el formato de archivos para instalar complementos Firefox |
| jpm post | Empaquete los archivos en XPI y les asigna una ruta |
| [jpm watchpost](https://developer.mozilla.org/en-US/Add-ons/SDK/Tools/jpm#jpm_watchpost) | Empaqueta los archivos en XPI cada vez que hay cambios en el archivo y lo publica en cierta URL |
| jpm sign | Empaqueta los archivos en XPI y válida para posteriormente firma el complemento por Mozilla. |

Add-ons SDK hay dos grupos de APIS/módulos que sirven para desarrollar:

* APIs de alto nivel: son aquellas que sirven para la construcción de complementos como: creación de interfaz del usuario, interactuando con la web, e interactuando con el navegador. Están son las APIs de alto nivel que se utilizaran para liberar esta versión:
  + simple-storage
  + self
  + tabs
  + panel
* APIs de bajo nivel: estos módulos todavía están en desarrollo pero se espera no ser utilizado en futuras versiones. Se dividen en módulos para poder crear colecciones, módulos para crear eventos y trabajadores, y por ultimo módulos que sirven como complemento de APIS de alto nivel. De este grupo se van a usar los siguientes módulos:
  + ui/button/toggle

Para utilizar las diferentes APIs primero hay que declarar una variable igualándola a la librería requerida:



Explicación de los módulos utilizados:

**Simple-storage**: permite generar un almacén de datos con persistencia, quedando guardado en la memoria del perfil del navegador Firefox. Funciona de forma similar al almacenamiento del DOM en la Web (local.storage), con la excepción de que solo está disponible para estos complementos.

Datos que puede almacenar:

* strings
* boolean
* números
* arrays
* objetos
* nulos

**Self**: Este modulo permite acceder a los datos que hay dentro del propio add-on. No se debe confundir con el objeto global **self** que se utiliza en los scripts de contenido para comunicarse con el código añadido.

Métodos:

Self.data.url (nombre): apunta a un fichero de datos incrustado, muy útil para visualizar el contenido de un fichero HTML que puede ir incrustado, por ejemplo, en el constructor de un panel.

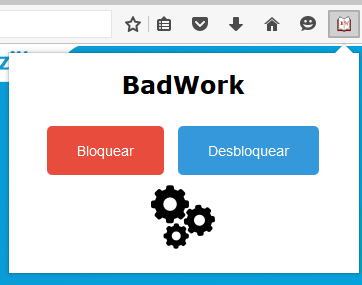
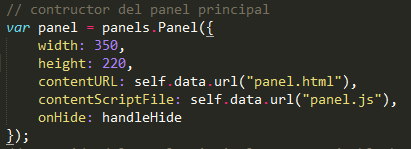
**Tabs:** abrir, manipular, recibir mensajes y eventos de las pestañas. En esta versión se usará esta API por sus diferentes propiedades:

* **activeTab:** llama al objeto pestaña activado.
* **url:** devuelve o carga una url de la pestaña activa.

**Panels:** crea diálogos transitorios que forman parte de la interfaz del complemento.

Constructor:

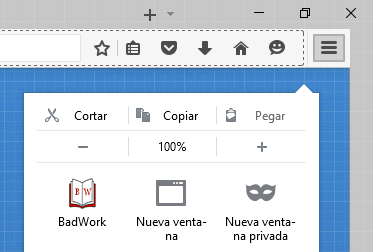
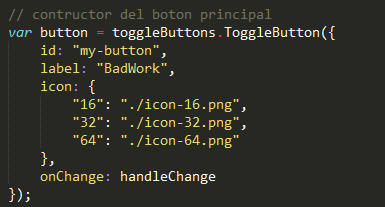
|  |  |
| --- | --- |
| Opciones | Descripción |
| With | Ancho del panel. Valor numérico |
| Heigth | Alto del panel. Valor numérico |
| contentURL | Con el modulo **self** podremos inyectar en el contenido del panel un fichero HTML ya guardado en la extensión |
| onHide | Incluye una función para escuchar eventos ocultos del panel |



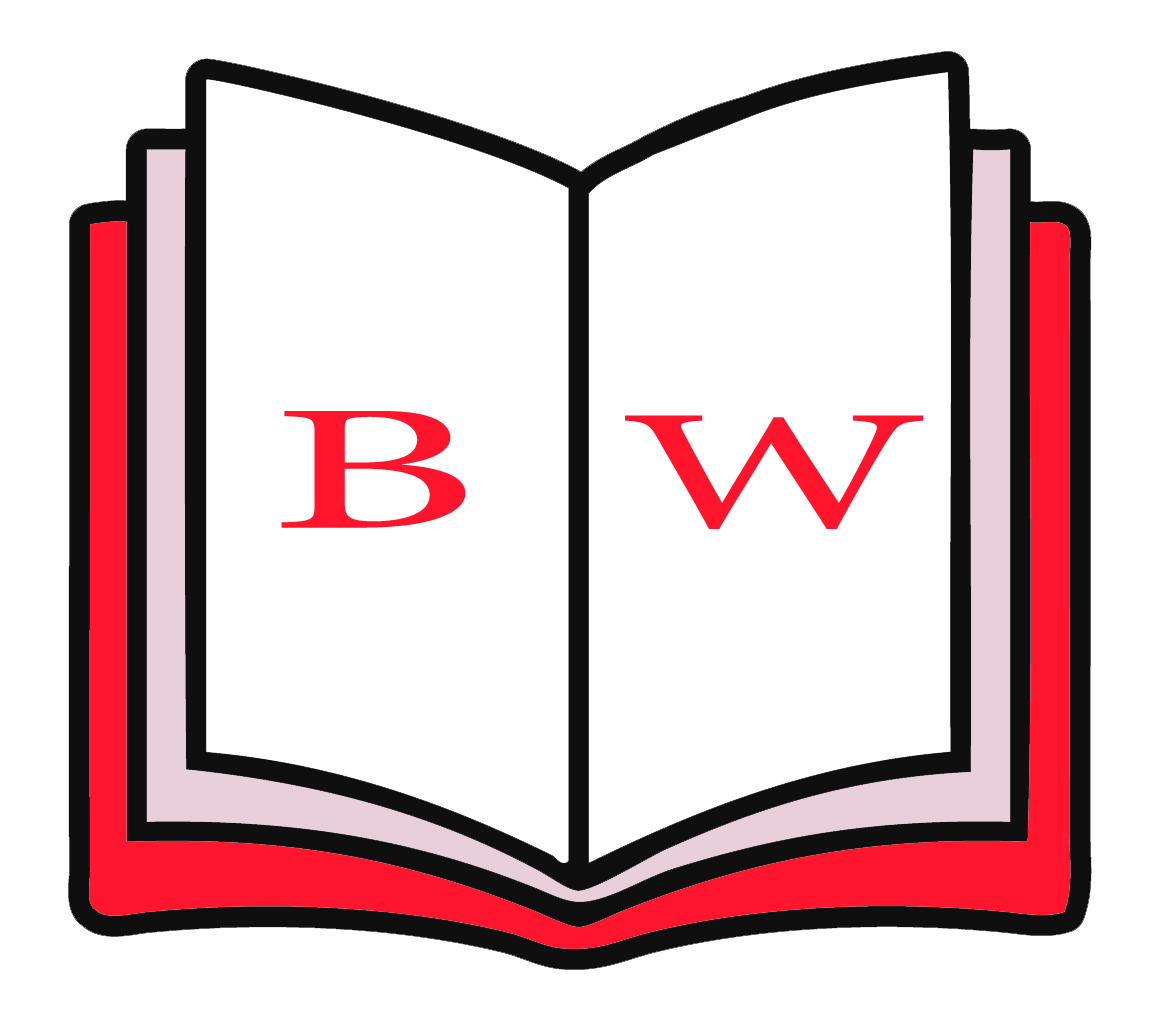
**Ui/button/toggle:** Agregar un botón de activación a la interfaz del usuario de Firefox. Con este módulo puedes crear botones que funcionan como una casilla de verificación, como si representará un interruptor.

En el constructor de este modulo podremos determinar los diferentes tamaños de los iconos que representarán nuestra extensión y asignarle un evento que al pulsarlo se active. Por ejemplo que se muestre el panel construido anteriormente.

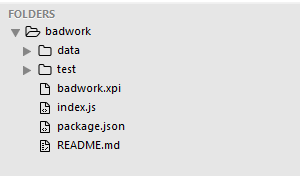
Screenshot_3.png

****

Para empezar el desarrollo lo primero que hago es pensar un nombre para el programa, ya que el objetivo es evitar distracciones en el empleo creo que una buena posibilidad es llamarlo “BADWORK”, el símbolo elegido para representarlo es el siguiente:



Con el comando **JPM INIT** hemos creado una estructura base a la que le vamos a añadir diferentes cambios y explicar los diferentes directorios y archivos:



**Badwork:** carpeta raíz.

**Data:** carpeta donde se van almacenar las imágenes, JS, HTML, CSS.

**Test:** carpeta que va a almacenar ficheros JS para hacer pruebas.

**Badwork.xpi:** será el fichero compilado preparado para instalar.

**Index.js:** será el fichero que llevara la lógica de la API.

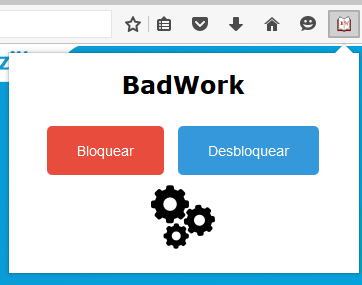
**Package.json:** archivo que describe características de la API, autor, colaborador, versión soportada, descripción, nombre, etc.

**README.md:** pequeña descripción del funcionamiento de la API.

Almaceno en la carpeta data tres imágenes con distinto tamaño:

* Icon-16.png
* Icon-32.png
* Icon-64.png

En adelante se crean los diferentes archivos HTML y CSS para dar estilo al panel. Se codifica la lógica en el index.js y se construyen los diferentes objetos, de los que he hablado con anterioridad, el panel y el botón de activación, quedando como el resultado siguiente:

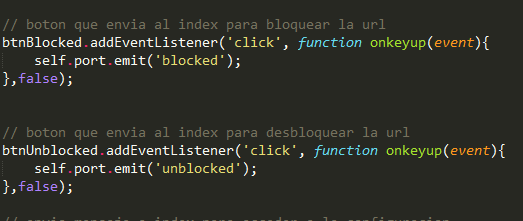
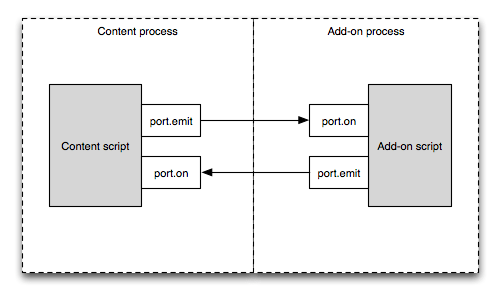


Para habilitar la comunicación entre sí de las secuencias de comandos de add-ons y las secuencias del contenido de un modulo add-on sdk, tienen que crear un puerto objeto:

* Para enviar mensajes se usa port.emit()
* Para recibir el mensaje del otro lado se usa port.on()

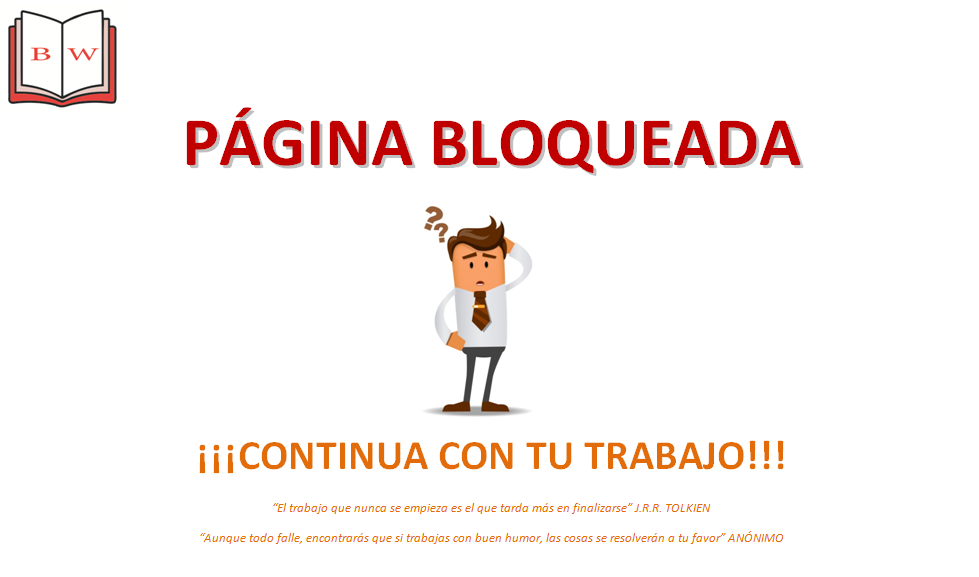
Estos mensajes son asíncronos, es decir, no se espera una respuesta del receptor, una vez enviado el mensaje el script sigue su procesamiento.

Para habilitar secuencias de comandos y secuencias de comandos de contenido del add-ons para comunicarse entre sí, cada extremo de la conversación tiene acceso a un puerto de objeto. A continuación se muestra un esquema y un ejemplo:



En este ejemplo vemos como se generan dos mensajes emit y se enviará uno de ellos dependiendo del botón pulsado del panel.

Cuando el index.js recoja el mensaje de bloqueo, almacenará la ruta y redireccionará la página al fichero ‘blocked.html’ quedando de la siguiente manera.



### 3.2.2.4. Evaluación del cliente / Casos de uso y pruebas

Usuario

|  |  |
| --- | --- |
| Nombre | UC-01 bloquear página |
| Secuencia principal | 1. El usuario solicita bloquear www.mozilla.org 2. El sistema recoge la información de la url 3. El sistema almacena el dominio en un array 4. El sistema redirecciona a la página de bloqueo |
| Error / Secuencia alternativa | 1.1.i. El usuario solicita www.mozilla.org en una nueva pestaña.  1.1.p La página es bloqueada  1.2.i. El usuario abre varias pestañas con la url www.mozilla.org es y solicita en una pestaña el bloqueo de la página.  1.2.p. ERROR. Solo se bloquea la página activa, las demás pestañas con la misma url no se han bloqueado.  1.2.i. El usuario solicita desbloquear www.mozilla.org. Caso de uso finaliza y comienza UC-02 desde la secuencia 2. |
| Post Codificación | El usuario ve que la pagina está bloqueada |

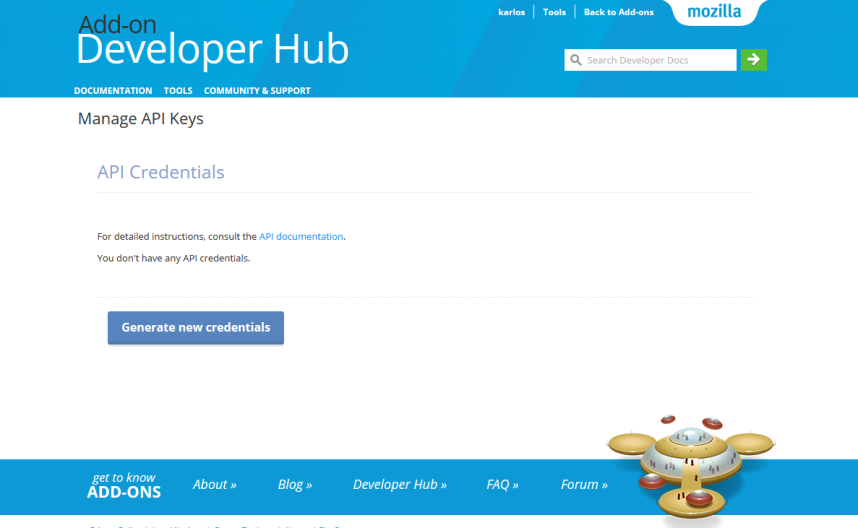
|  |  |
| --- | --- |
| Nombre | UC-02 desbloquear página |
| Secuencia principal | 1. El usuario solicita desbloquear www.mozilla.org 2. El sistema encuentra la página para desbloquear 3. El sistema borra el dominio del array 4. El sistema redirecciona a la página www.mozilla.org |
| Error / Secuencia alternativa | 1.1.i. El usuario solicita www.mozilla.org en una nueva pestaña.  1.1.p. La página no aparece bloqueada.  1.2.i. El usuario abre varias pestañas con la url www.mozilla.org, pagina bloqueada, y solicita desbloquear en una de ellas.  1.2.p. ERROR. Solo se desbloquea la pestaña activa. Las demás pestañas siguen bloqueadas |
| Post Codificación | El usuario vuelve a ver la página |

### 3.2.2.5. Construcción y entrega

Para conseguir construir nuestra add-on es muy sencillo, solo basta con usar el comando **JPM XPI** y se comprimirá toda la aplicación. Con este archivo podremos instalarlo en el navegador solamente arrastrándolo sobre él. Pero esta técnica conlleva un problema, el navegador sabe que esa aplicación no está firmada por los agentes de Mozilla y te advierte que no es seguro.

En esta versión voy a mostrar como tener la firma de los agentes de Mozila de forma sencilla.

* En esta dirección nos proporcionará un usuario y una clave privada: <https://addons.mozilla.org/en-US/developers/addon/api/key/>



* En nuestra línea de comandos ejecutaremos lo siguiente:

**jpm sign --api-key ${AMO\_API\_KEY} --api-secret ${AMO\_API\_SECRET}**

* Pasa una serie de validaciones que pueden dar los siguientes mensajes:
  + El proceso de validación fue correcto, Mozilla firmó el proyecto y generó un archivo XPI firmado.
  + El proceso de validación fue fallido y no se firmo el complemento.
  + El número de versión de su complemento ya existe. Incrementa el número de la versión en el package.json y vuelve a ejecutar el comando.

Si todo ha funcionado ya poseemos nuestro complemento firmado y ya lo podremos instalar en nuestro navegador. Esta forma que tiene JPM para qué firme Mozilla tiene una peculiaridad. El complemento es cierto que está en los repositorios de Mozilla, pero está en una zona no listada, por lo que para distribuir la aplicación deberemos hacerlo nosotros mismos.

### 3.2.3. Ciclo 2. Liberación de BadWork 1.0.1.

Este ciclo será el último para la entrega del proyecto. Se realizarán muchos cambios, tanto a nivel de diseño como de codificación, por lo pasará por varias versiones hasta llegar a la actual. Este ciclo termina pudiendo descargar la aplicación desde los complementos de Firefox e instalarlo en nuestro navegador

### 3.2.3.1. Comunicación con el cliente / Objetivo.

* El cliente quiere añadir varias funcionalidades más:

1. Una contraseña de acceso para que sólo el administrador pueda añadir o quitar páginas bloqueadas.
2. Una plataforma de configuración dónde se listan las páginas bloqueadas y la posibilidad de cambiar la contraseña desde esta zona.

Objetivo: crear una mayor seguridad en la aplicación para que sólo el administrador pueda añadir o borrar las páginas que desea bloquear.

### 3.2.3.2. Planificación.

### 3.2.3.2.1. Diagrama gráfico de Gantt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tareas | Fecha de inicio Programada | Fecha de inicio | Días | Días Extra | Fecha fin |
| Inicio del Proyecto | 05/05/2016 | 05/05/2016 | 15 | 0 | 19/05/2016 |
| Documentación | 06/05/2016 | 06/05/2016 | 14 | 0 | 07/05/2016 |
| Planificación | 07/05/2016 | 07/05/2016 | 2 | 0 | 08/05/2016 |
| Ingeniería | 09/05/2016 | 09/05/2016 | 6 | 2 | 16/05/2016 |
| Implementación | 16/05/2016 | 17/05/2016 | 2 | 1 | 19/05/2016 |

### 3.2.3.3. Análisis de riesgo

Llevo a cabo el estudio de las posibles amenazas y eventos no deseados, evaluando y proponiendo diferentes alternativas:

* La seguridad al utilizar contraseñas sin encriptación, ya que cabe la posibilidad de que una tercera persona quiera hacer un mal uso de la aplicación. Como solución todas las contraseñas serán encriptados con el algoritmo SHA256.
* Incomunicación entre los diferentes ficheros javascript al pasar variables entre ellos. Para solventarlo utilizaré funciones de llamadas de mensajes provistas por el propio lenguaje.
* Perdida de datos del proyecto desarrollado por daños en el sistema o en los ficheros. Como solución, crear un repositorio GitHub para mantener un control de versiones y asegurar el funcionamiento correcto de cada versión.

### 3.2.3.4. Ingeniería

En esta versión cada cambio en el código de las diferentes aéreas han hecho que cree un repositorio de control de versiones en GitHub, así se resuelve el problema si en un punto de lo programado no funciona y poder volver al punto donde si lo hacía.

En esta tabla se muestran algunos comandos que se han utilizado:

|  |  |
| --- | --- |
| Git clone | Clonar el repositorio |
| Git status | Saber el estado de los ficheros locales frente a los del repositorio |
| Git diff | Ver la diferencia de los cambios entre un fichero del repositorio y un fichero local |
| git checkout <nombre fichero> | Volver un fichero local al estado del repositorio del mismo |
| Git commit –a | Hacer un commit a todos los archivos modificados |
| Git commit –m | Hacer un commit añadiendo un comentario descriptivo |
| Git add <nombre fichero> | Añadir un fichero al repositorio |
| Git rm <nombre fichero> | Borrar un fichero del repositorio |
| Git pull | Actualiza el repositorio local con el repositorio remoto |
| Git push | Inyecta los cambios realizados en el repositorio local al repositorio remoto |

A continuación vamos a hablar de la tecnología Add-ons SDK.

Como comentamos en el ciclo 1, Add-ons SDK posee dos módulos para desarrollar aplicaciones. En este ciclo se hablará en detalle del API de alto nivel **page-mod.**

**Page-mod:** Ejecutar secuencias de comandos en el contexto de las páginas web cuya URL coincide con un patrón determinado.

Para utilizar la página-mod, se especifica:

* una o más secuencias de comandos. El SDK llama a estos scripts "guiones de contenido".
* un patrón URL de una página que debe coincidir.

Constructor:

|  |  |
| --- | --- |
| Opciones | Descripción |
| Include | Introduciendo un String o un Array de direcciones para que page-mod aplique las definiciones del contenido |
| Exclude | Parecido a include. Estas rutas son descartadas a la hora de aplicar las definiciones de contenido. |
| contentFileScript | Introduciendo un String o un Array. Se incluirán en esta página los diferentes archivos javascript al contenido de la pagina a modificar |
| contentScriptWhen | Modifica la manera en la que se carga el contenido javascript.  Tres estados:   * start, cargar los contenidos script antes de que el DOM haya sido cargado * ready, carga los script una vez que el DOM ha sido cargado. * end, carga los script una vez que todos los archivos (DOM, JS, CSS, imágenes) han sido cargadas. |
| attachTo | Adjunta opciones que añadirán la funcionalidad de page-mod a las pestañas. Tres opciones que se pueden combinar:   * exist, page-mod se aplicara automáticamente en las pestañas abiertas * top, page-mod se aplicará a la pestaña de más alto nivel. * Iframe, page-mod se aplicará al marco flotante dentro de la pestaña. |

**Eventos**:

**OnAttach**: Este evento se emite cuando el contenido de scripts de la page-mod está asociado a un documento cuya URL coincide con el patrón incluido en page-mod.

La función de escucha se pasa un objeto trabajador (**worker**) que se puede utilizar para [comunicarse con los guiones de contenido](https://developer.mozilla.org/en-US/Add-ons/SDK/High-Level_APIs/page-mod#Communicating_With_Content_Scripts) de su página-mod ha cargado.

El evento se dispara cada vez que los scripts de los contenidos de la página-mod son llamados por el usuario. Cada vez que se habrá una página que coincida con el patrón URL de page-mod se realizará una instancia nueva del objeto trabajador.

El **Trabajador** puede ser utilizado para la comunicación de script de contenido como se explico en el ciclo 1 en el API de alto nivel Panel.

**APIs de terceros**

A parte de estas APIs Add-ons SDK se pueden usar módulos de terceros que se incluyen dentro de Firefox, pero también modulos propios y ponerlos a disposición de cualquier persona. Antes de la herramienta JPM estuviera disponible se hacía muy difícil encontrar módulos desarrollados por otros programadores, tanto instalarlos como actualizarlos.

Con JPM, utilizamos el gestor de paquetes nodo NGP que hacen envíos a SDK dentro de Firefox. Los desarrolladores pueden publicar sus módulos en NGP para que posteriormente podamos incorporarlos a través de NPM. Ejemplo de instalación de un menú:

* **NPM instalación menuitem –save**

No todo el desarrollo de nuestra aplicación debe estar en nuestro fichero index.js también podemos separar el desarrollo en módulos propios y exportar los diferentes resultados para que nuestro fichero principal los maneje.

Se pueden crear modulos muy potentes de bajo nivel pero deben llevar privilegios de “chrome”, lo que nos da acceso a Components object, con lo que no nos proporciona ninguna restricciones al sistema anfitrión.

Estos componentes objetos son los denominados XPCOM. Son modelos multiplataforma, que permiten ser utilizados por gran cantidad de lenguajes como javaScript, java, Pyton, además de C++.

XPConnect es la herramienta que hace de puente entre JavaScript y XPCOM. Con XPConnect se pueden usar componentes de código JavaScript e interactuar con objetos JavaScript dentro de los componentes XPCOM. XPConnect es parte de Firefox y se utiliza activamente en aplicaciones XUL.

Para poder crear estos objetos, como hemos reflejado anteriormente, hay que darle cierta autoridad al sistema anfitrión. Para ello debemos declarar una variable incluyendo los diferentes componentes, pudiendo solamente escribir sus alias:

* Cc (Components.classes)
* Ci (Components.interfaces)
* Cu (Components.utils)
* Cr (Components.results)
* Cm (Components.manager)

Ejemplo:



Hay tres pasos a seguir a la hora de llamar a un componente XPCOM.

* Obtener el componente
* Obtener la parte del componente que implementa la interfaz que queremos utilizar.
* Llamar a la función que necesitamos

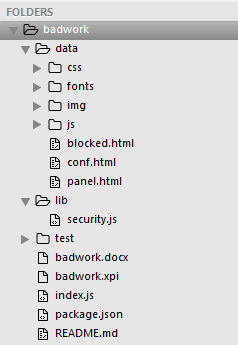
En este proyecto se incluye una librería para llevar a cabo la funcionalidad de control de acceso de la aplicación a través de una contraseña. Esta librería incluida se llama **security.js**, para ello se utilizaran los siguientes XPCOM:

* nsIPromptService: para mostrar una pop-up tipo prompts con diferentes usos.
* nsICryptoHash: para encryptar un string o un fichero con diferentes algoritmos de encriptación.
* nsIScriptableUnicodeConverter: para convertir el dato binario en un dato hexadecimal.



[*\* Para más información revisar anexo I.*](#_Anexos/Otros)

En esta versión en cuanto a la estructura se han realizado varios cambios para que todos los archivos estén distribuidos de una forma más ordenada. En la imagen siguiente podemos ver su estructura:



A diferencia de la versión anterior tenemos que destacar las varias carpetas y ficheros:

**Data:** se han creado subcarpetas para separar los tipos de ficheros dejando en su raíz los HTML:

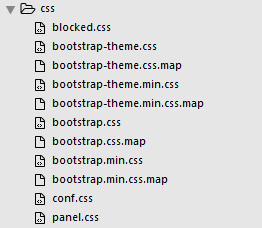
* **Css**: se almacén las hojas de estilos.
* **Fonts**: las fuentes que se pueden utilizar.
* **Img**: las imágenes de la aplicación.
* **Js**: todos los archivos javascript

**Lib:** esta carpeta servirá para almacenar modulos externos, donde creó un fichero security.js que manejara la seguridad de las contraseñas.

En esta versión he incluido librerías de BootStrap y Jquery para facilitar la programación tanto del diseño como los ficheros javascript.

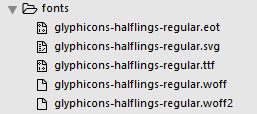
Para incluir la librería solamente he tenido que descargar las versiones más actuales de estas librerías e incluirlas en las diferentes subcarpetas de Data:

La carpeta Css:



Gran parte de estos archivos css son incluidos por la librería bootstrap, los más importantes son bootstrap.css que contiene todas las clases y etiquetas con su estilo característico y bootstrap.min.css que es la hoja de estilos minimizada.

La carpeta Fonts:



Estos son las fuentes que representan los iconos de bootstrap.

La carpeta JS:



En esta carpeta están incluidos el fichero bootstrap minimizado y la librería jQuery que es esencial para que funcione bootstrap.

Gracias a bootstrap y jquery se logró el siguiente efecto en las siguientes páginas

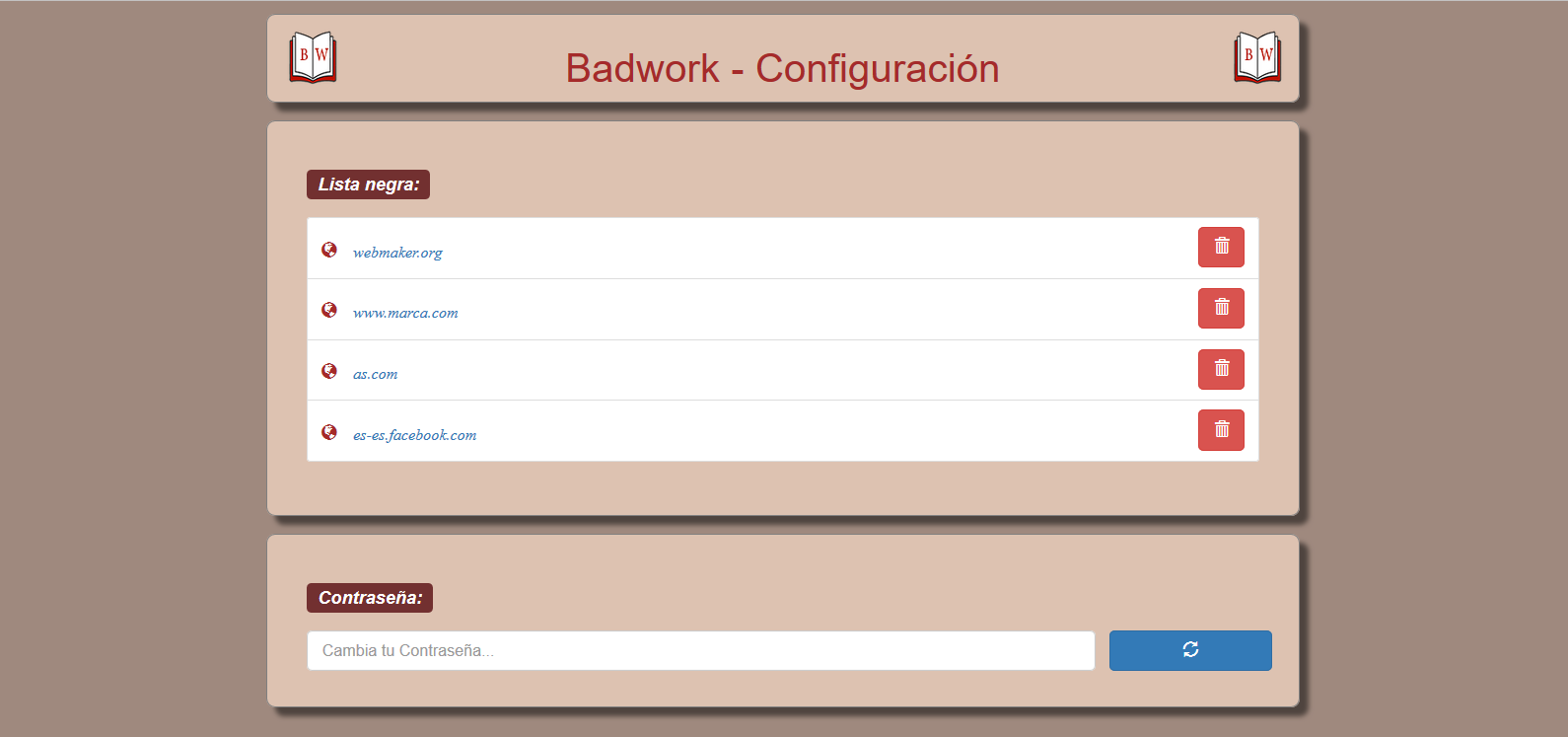
**Blocked.html:**



Observaciones:

* Se ha creado un header con los iconos de la aplicación acompañado de un texto explicativo sobre que dominio a bloqueado
* Se ha rediseñado la imagen y se le ha dado una animación CSS
* Todas las cajas poseen un estilo de sombra, con bordes redondeados
* En el footer se ha incluido varias citas de ánimo para el buen trabajador, que van cambiando aleatoriamente en cada página bloqueada.

**Conf.html:**



Observaciones:

* El header es exactamente igual al de BLOCKED.HTML salvo que cambia el texto.
* Se ha creado un contenedor con una lista NEGRA, de las paginas que se han ido bloqueando, y va actualizando según la interacción de la aplicación.
* Se incluyen botones a cada ítem para tener la posibilidad de eliminarlos desde la página de configuración. Se incluye el icono de papelera, icono proporcionado por la librería bootstrap.
* La sección de abajo es la zona para cambiar la contraseña. Es un input tipo TEXT con estilo propio al pulsar en él.

A continuación se muestra una tabla con la descripción de las funciones, constructores y mensajes de los archivo index.js y security.js, que son los ficheros que llevan la lógica de la aplicación:

|  |  |
| --- | --- |
| **Index.js** | |
| elementos | Descripción |
| Función getCurrentUrl | Devuelve la última url conocida dependiendo de la pestaña que este activa en ese mismo momento. |
| Función storagePageBloqued | Se almacena la pagina que se deseo bloquear en la aplicación. |
| Función deletePage | Borra la página almacenada en la aplicación. |
| Función checkPages | verifica si una página está guardada en almacenamiento o no. |
| Función reloadPageDelete | Recorrer todas las pestañas de la ventana del navegador para desbloquear las páginas que se han desbloqueado y actualizar la **lista negra** de la página de configuración. |
| Constructores page-mod | * Page-mod para cualquier página. * Page-mod para la página de bloqueo. * Page-mod para la página de configuración. |
| Mensajes | * “storages”: envió de la variable de las paginas almacenadas. * “delete”: indica qué elemento debe ser borrado. * “changePassword”: recibe el password que ha sido cambiado y se cifra en el algoritmo SHA256. * “blocked”: determina la url que debe ser bloqueada. * “unblocked”: determina la url que debe ser desbloqueada. * “config”: determina que se quiere acceder a la configuración de la aplicación. |

[*\* Para más información revisar anexo II.*](#_Anexos/Otros)

|  |  |
| --- | --- |
| **Security.js** | |
| Elementos | Descripción |
| Función hasPassword | Determina si existe una contraseña almacenada. |
| Función passwordHash | Codifica la contraseña con un algoritmo de cifrado SHA256. |
| Función statusPassword | Determina el estado de la contraseña. Recién creado o si está activo |
| Función toHexString | devuelve dos dígitos hexadecimales para cada byte |
| Función checkPassword | Verifica si la contraseña introducida es verdadera. |

[*\* Para más información revisar anexo III.*](#_Anexos/Otros)

### 3.2.3.4. Evaluación / Casos de uso y pruebas

Administrador

### 

Usuario

|  |  |
| --- | --- |
| Nombre | UC-01 Crear contraseña (administrador) |
| Secuencia principal | 1. El usuario administrador pulsa para iniciar la aplicación. 2. El sistema solicita crear una contraseña. 3. El usuario introduce una contraseña. 4. El sistema recoge y cifra la contraseña en un algoritmo SHA256 y la guarda. 5. El sistema solicita verificar contraseña. 6. El usuario introduce una contraseña. 7. El sistema compara la contraseña guardada con la introducida. |
| Error / Secuencia alternativa | 2.1.i. El sistema tiene ya una contraseña guardada.  2.1.p. Caso de uso finaliza comenzando el UC-02 verificar contraseña. |
| Post Codificación | El sistema abre el panel de control de acceso. |

|  |  |
| --- | --- |
| Nombre | UC-02 Verificar contraseña (usuario) |
| Secuencia principal | 1. El sistema solicita verificar la contraseña. 2. El usuario introduce una contraseña. 3. El sistema recoge la contraseña y la compara con la guardada |
| Error / Secuencia alternativa | 3.1.i. El usuario ingresa una contraseña correcta.  3.1.p. El sistema abre el panel de control de acceso. |
| Post Codificación | El sistema muestra un mensaje de alerta de contraseña incorrecta. |

|  |  |
| --- | --- |
| Nombre | UC-03 Bloquear página |
| Secuencia principal | 1. El administrador solicita bloquear www.mozilla.org 2. El sistema recoge la información de la url 3. El sistema almacena el dominio en un array 4. El sistema redirecciona a la página de bloqueo |
| Error / Secuencia alternativa | 1.1.i. El usuario solicita www.mozilla.org en una nueva pestaña.  1.1.p La página es bloqueada  1.2.i. El usuario abre varias pestañas con la url www.mozilla.org y solicita en una pestaña el bloqueo de la página.  1.2.p. Las pestañas que coinciden con esa url son redireccionadas a la página de bloqueo.  1.2.i. El usuario solicita desbloquear www.marca.es. Caso de uso finaliza y comienza UC-04 desde la secuencia 2. |
| Post Codificación | El usuario ve que la pagina está bloqueada |

|  |  |
| --- | --- |
| Nombre | UC-04 Desbloquear página |
| Secuencia principal | 1. El usuario solicita desbloquear www.mozilla.org 2. El sistema encuentra la página para desbloquear 3. El sistema borra el dominio del array 4. El sistema redirecciona a la página www.mozilla.org |
| Error / Secuencia alternativa | 1.1.i. El usuario solicita www.mozilla.org en una nueva pestaña.  1.1.p. La página no aparece bloqueada.  1.2.i. El usuario abre varias pestañas con la url www.mozilla.org, que está bloqueada, y solicita desbloquear en una de ellas.  1.2.p. Todas las pestañas dónde coincida la url serán desbloqueadas. |
| Post Codificación | El usuario vuelve a ver la página |

|  |  |
| --- | --- |
| Nombre | UC-05 Cambiar contraseña. |
| Secuencia principal | 1. El administrador introduce una nueva contraseña. 2. El administrador pulsa el botón cambiar contraseña. 3. El sistema sustituye la contraseña antigua por la nueva 4. El sistema alerta que la contraseña ha sido cambiada. 5. El sistema reinicia la solicitud de verificación. 6. El sistema cierra la pestaña de configuración. |
| Error / Secuencia alternativa | 1.2.i. El usuario introduce una contraseña poco segura.  1.2.p. ERROR. El sistema no avisa de que la contraseña es segura y realiza el cambio contraseña sin verificación.  2.1.i. El usuario pulsa botón sin introducir contraseña.  2.1.p. No se realiza el cambio de contraseña. ERROR no se avisa al usuario de que debe ingresar una contraseña. |
| Post Codificación | La contraseña de acceso al panel de control cambió. |

### 3.2.3.5. Construcción y entrega

En el ciclo 1 para que Mozilla firmara nuestra aplicación utilizamos JPM, con la particularidad de que la aplicación no estaría en la lista de complemento de Mozilla, si no que nos tocaría a los desarrolladores distribuirla por cuenta propia.

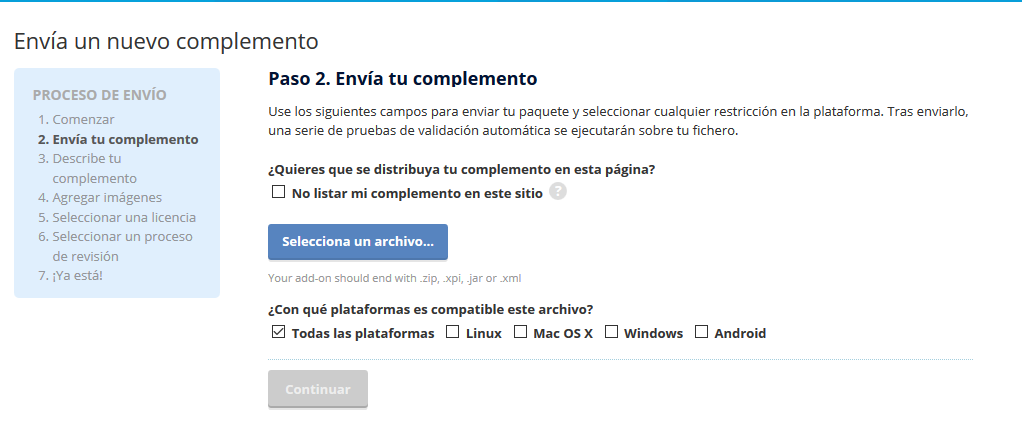
En este ciclo vamos a firmar el complemento a través de la página de add-ons de Mozilla para que esta vez aparezca en la lista de extensiones y toda la comunidad pueda descargársela e instalarla.

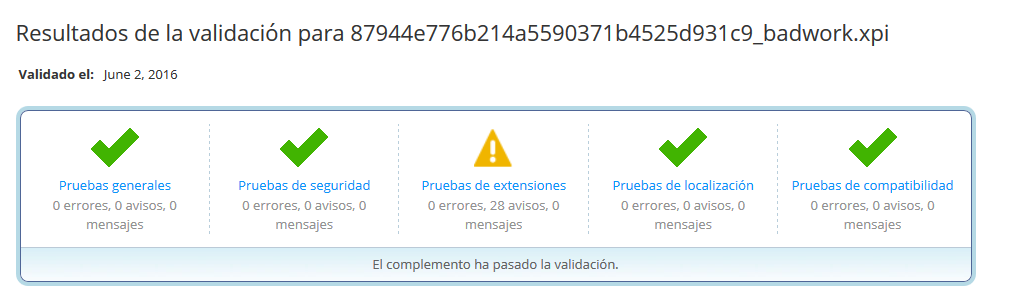
Requisitos:

* generar un archivo XPI de la aplicación (**JPM XPI**).
* Ingresar en <https://addons.mozilla.org>
* Subir el complemento

Pasos a seguir:

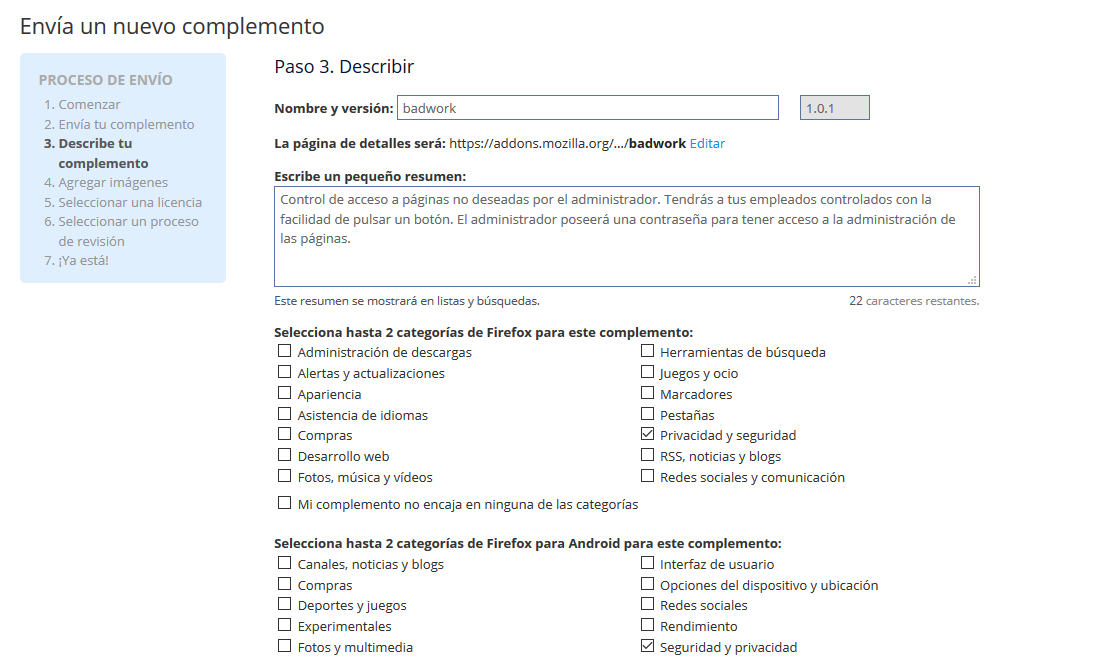
**Envía tu complemento**. En este proceso subiremos nuestro archivo XPI, una vez subido se irán haciendo una serie de validaciones del complemento.





Si el proceso se realizó con existo podemos proseguir.

**Describe tu complemento.** En este apartado se debe hacer un resumen para saber en qué consiste tu aplicación y elegir una categoría para clasificarla a la hora de buscar un complemento.

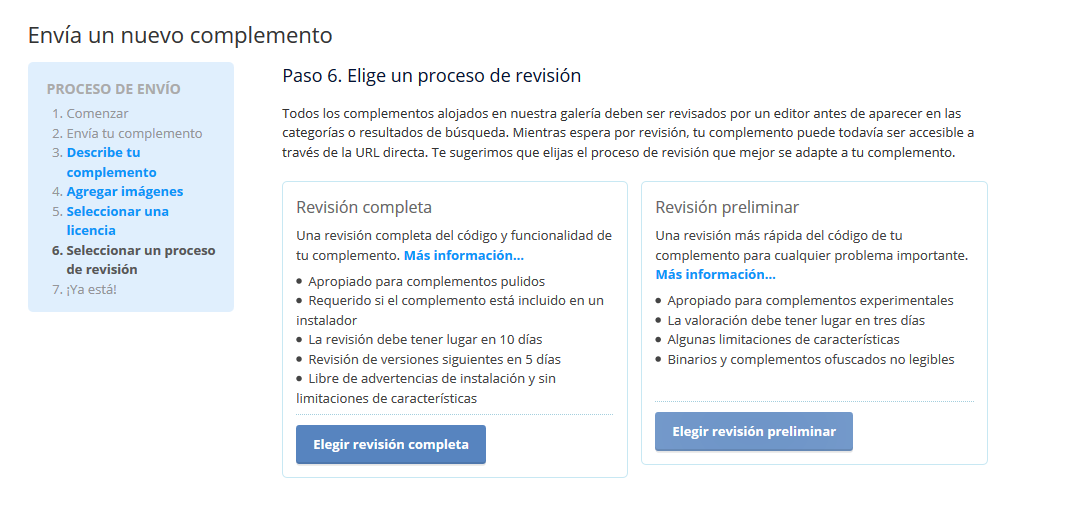


**Agregar imágenes.** Agregamos las imágenes que van a representar los iconos de la aplicación y el funcionamiento de la misma.



**Elegir una licencia:** La licencia MIT pone muy pocas restricciones en la reutilización de software. Los derechos otorgados al [Usuario final](https://es.wikipedia.org/wiki/Usuario_final) en la licencia MIT, incluyen: el derecho a usar, copiar, modificar, fusionar, publicar, distribuir, cambiar la licencia y/o vender el software

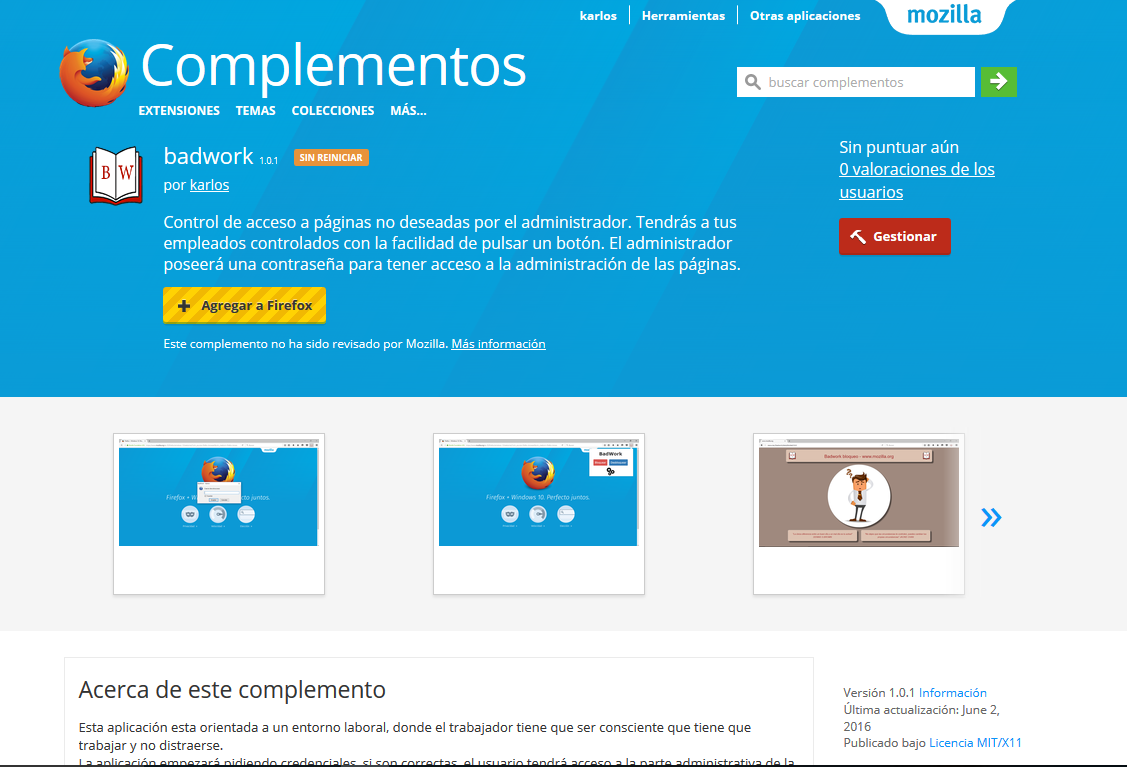


**Seleccionar un proceso de revisión**: en este apartado debemos elegir el método de revisión. Esto puede tardar varios días dependiendo del método que elijamos.

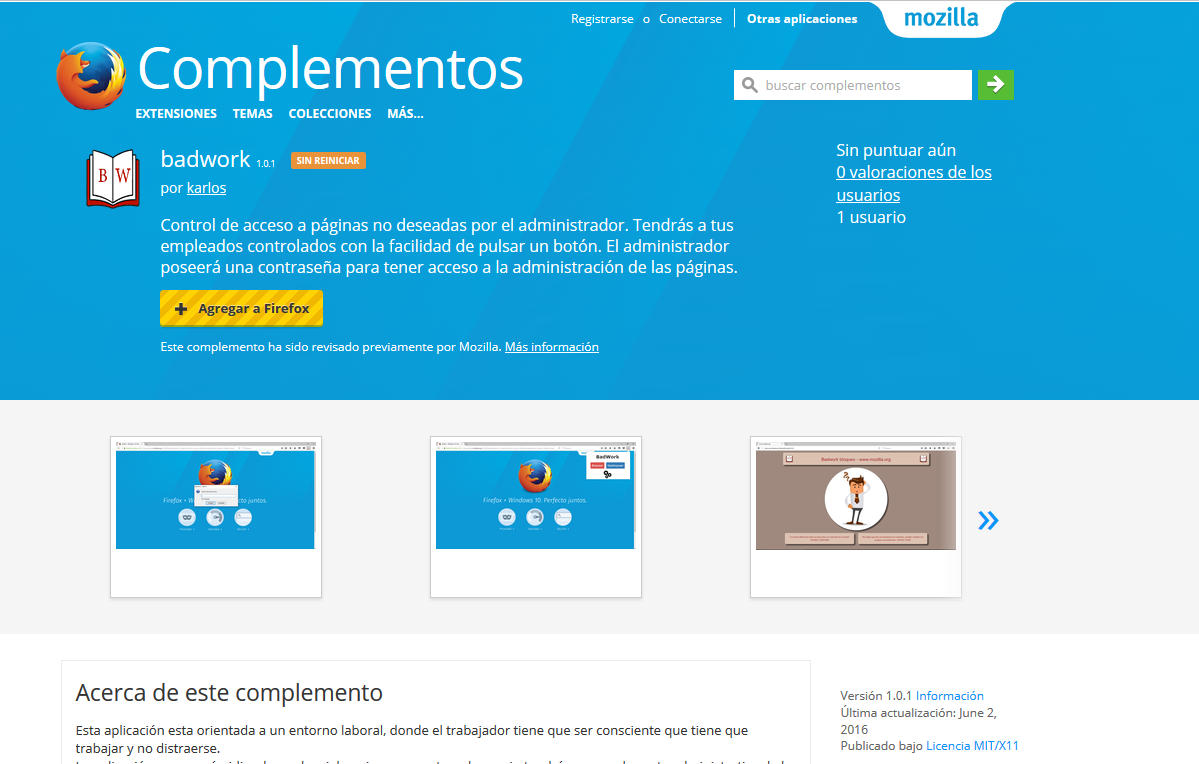
**¡Ya está!** El proceso se completo correctamente. Ya esta nuestra versión liberada, pero no revisada, esto quiere decir que hasta que el desarrollador no reciba un mail con los resultados de la revisión no aparecerá en la búsqueda de galería de complementos de Mozilla, pero si a través de la url proporcionada.

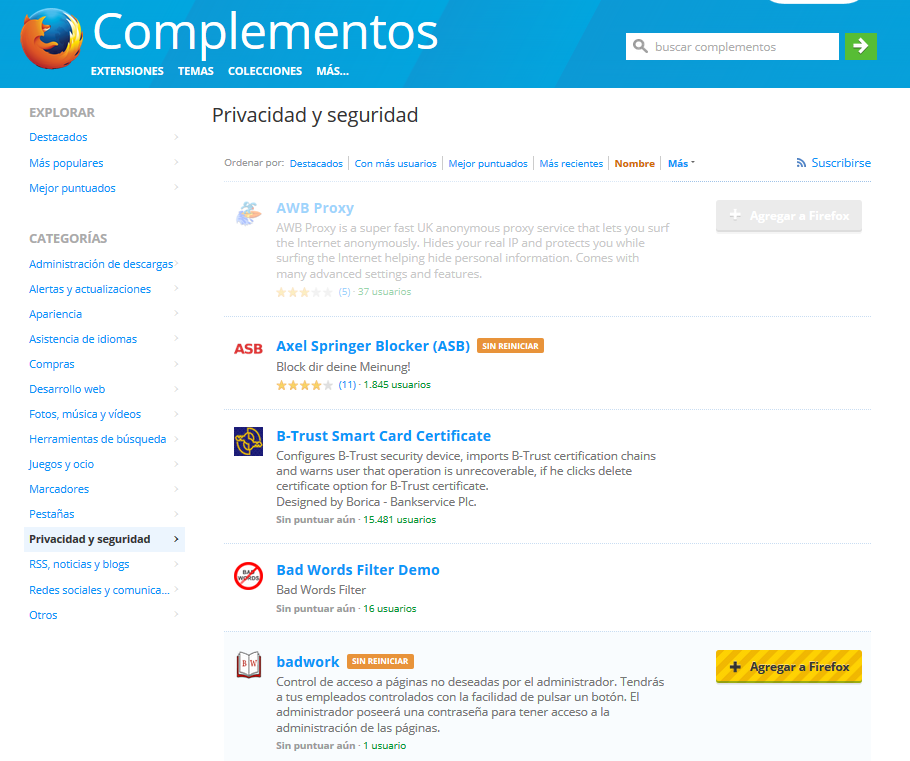


**Complemento no revisado**:



**Complemento revisado:**





# 4. Resultado y discusión

Tras documentarme decido utilizar para este proyecto la tecnología “XUL”. Al empezar a programar con esta tecnología comienzo a encontrarme diversas dificultades:

* Cuando trabajo localmente con los archivos con extensión “.xul” hayo incompatibilidades con las versiones más actuales de Firefox.
* A la hora de instalar el API (Application Programming Interface) necesita la firma de verificación del grupo de desarrollo de Mozilla.

Por estos inconvenientes me decido finalmente realizar el proyecto en una tecnología que no esté tan desfasada e implementar el proyecto utilizando Add-on SDK (Kit de Desarrollo Software) que te permite desarrollar APIs de Firefox siguiendo las tecnologías Web estándar: JavaScript, Css y Html.

Add-ons SDK tiene numerosas ventajas respecto a XUL:

* Posee una herramienta por línea de comando denominada **JPM**, que tiene diferentes funciones:
  1. Inicializar un proyecto desde cero (creando una estructura base de directorios y archivos). **Jpm init**
  2. Ejecutar el proyecto en local. **Jpm run**
  3. Probar
  4. Empaquetar proyecto para posteriormente instalarlo. **Jpm xpi**
* JPM está basada en Node.js y sustituye a la herramienta CFX.
* JPM se puede utilizar en versiones de Firefox 38 en adelante.
* Posee módulos ya definidos que te ayudan a la hora de crear el API e interactuar con el usuario.
* Puedes añadir módulos de terceros o incluso poner tus creaciones.

# 5. Conclusiones

Estos casi tres meses desarrollando este proyecto me han aportado grandes cosas. Una de ellas ha sido el saber mis habilidades y limitaciones a la hora de utilizar y programar una tecnología desde cero. Ha sido un trabajo de muchas horas de documentación y tutoriales antes de ponerme a programar la codificación. Gracias a este proyecto he mejorado mis técnicas y comprensión en el leguaje JavaScript, que pienso que es el futuro, debido a la gran potencia de los navegadores quitando el peso de trabajo a los servidores.

En mi opinión Add-ons SDK es una tecnología muy sencilla si quieres introducirte en la programación de complementos para Mozilla. Posee una documentación, en inglés, muy fácil de entender y con muchos ejemplos. Por otra parte, la propia comunidad de Mozilla te puede ayudar en foros y eso supone una gran satisfacción y te anima a que tú también contribuyas con otros programadores.

Aunque está aplicación es sencilla, creo que tiene grandes posibilidades y seguiré desarrollando para darle una versión más estable y profesional.

Por falta de tiempo hubo versiones que no se pudieron realizar como por ejemplo:

* Páginas almacenadas por defecto para su bloqueo por categorías: Redes, Noticia Deportiva, Juegos. Pudiendo activarse y desactivarse.
* Botón para introducir un fichero con un formato determinado dónde se listarán las páginas que quiere bloquear el usuario.
* Modificación de roles de la aplicación, dependiendo para quién fuera orientado, dando a un botón cambiaría todo el estilo y las imágenes. Por ejemplo para un entorno escolar, para niños o temática variada.

Como he mencionado anteriormente seguiré desarrollando esta aplicación para llevarla a un nivel más alto, pero si tuviera que desarrollar otra aplicación distinta para Mozilla creo que utilizaría Web extensión, que es una tecnología más actual, ya que se está intentando alejarse de Addon SDK, aunque lentamente.

# 6. Bibliografía y referencias

<http://soporte.eset-la.com/kb2798/?locale=es_ES>

<https://es.wikipedia.org/wiki/Desarrollo_en_espiral>

<http://www.sites.upiicsa.ipn.mx/polilibros/portal/polilibros/P_externos/Administracion_informatica_de_las_organizaciones_Ramon_E_Enriquez_Gonzalez/AIO2_Mod_ESPIRAL.html>

<http://sofware1nathalygrijalva.blogspot.com.es/2012/10/modelo-espiral.html>

<http://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL/Tutorial>

<https://developer.mozilla.org/en-US/Add-ons/SDK>

<https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software>

[https://developer.mozilla.org/en-US/Add-ons/SDK/Tools/jpm#Installation](https://developer.mozilla.org/en-US/Add-ons/SDK/Tools/jpm%23Installation)

<https://nodejs.org/en/>

<https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface>

<https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIScriptableUnicodeConverter>

<https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsICryptoHash>

<https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIPromptService>

**Imágenes:**

**Trabajador con dudas fue facilitada por Vector desde:**

<http://www.freepik.es/fotos-vectores-gratis/negocios>

**Iconos (libro y engranajes) facilitada por Freepik desde:**

<http://www.flaticon.com>

# 7. Anexos/Otros

## 7.1. Anexo I

**NsIPromptService:** esta interfaz sirve para mostrar diálogos simples.

|  |  |
| --- | --- |
| Métodos | Descripción |
| Alert | Muestra un mensaje de alerta con un botón. Trabaja de la misma manera que Windows.alert, con la diferencia que acepta poner un titulo a la ventana modal. |
| promptPassword | Muestra un dialogo con campo editable tipo password y un chexbox opcional. |

**NsICryptoHash:** esta interfaz es usada para cifrar ficheros o datos con varios algoritmos de cifrado (MD2, MD5, SHA256, SHA384 y SHA512).

|  |  |
| --- | --- |
| Métodos | Descripción |
| Init | Inicia el objeto de cifrado. |
| Update | Añade una matriz de datos que deben ser cifrados por el objeto inicializado. |
| Finish | Finaliza el objeto de cifrado iniciado completando con los datos cifrados actuales. |

**NsIScriptableUnicodeConverter:** esta interfaz convierte cadenas utilizadas en un script en datos binarios para utilizarlos a la hora de cifrar.

|  |  |
| --- | --- |
| Métodos | Descripción |
| convertToByteArray | Convierte un cadena en un array de bytes |

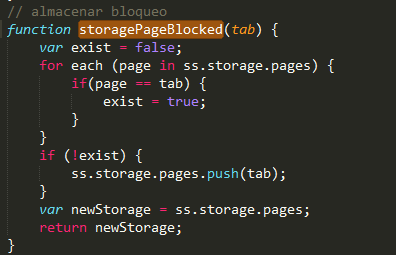
## 7.2. Anexo II

**Funciones**:

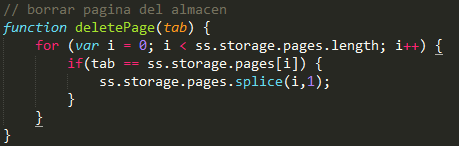
GetCurrentUrl:



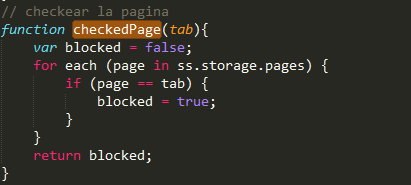
StoragePagesBlocked:



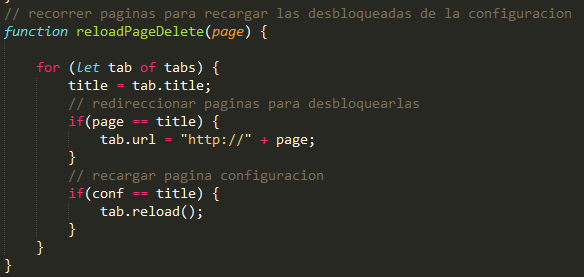
DeletePage:



CheckedPage:

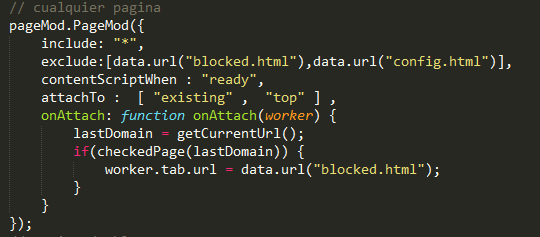


ReloadPageDelete:



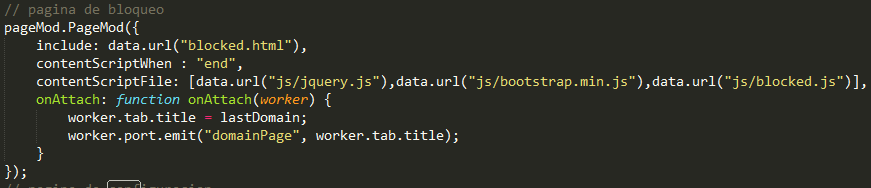
**Constructores de page-mod para diferentes paginas:**

Cualquier página:



Este constructor va atacar a todas las paginas una sola vez, excluyendo claramente a las paginas cuya URL sea BLOCKED.HTML y CONFIG.HTML. Este constructor se le asigna una función que ataca a la página comprobando si debe de estar bloqueada, si fuera cierto, hace un redireccionamiento a BLOCKED.HTML.

Para la página de bloqueo:



Este constructor solo atacara a las paginas cuya URL sea Blocked.HTML, se incluirán varios archivos JavaScript (BootStrap, Jquery y blocked.js).

Se asignará una función que ataque a la pagina activa, modificando el titulo de la misma y enviado un mensaje, que será recibido por el archivo blocked.js para poder modificar el contenido de la pagina.

* Archivo Blocked.js incrustado en la página:



En este fichero js ocultamos las frases de ánimo y calculamos un número aleatorio para que solo salgan dos cada vez. Se recibe la variable enviada por index.js para modificar el header de la página.

Para la página de configuración:

****

Constructor muy parecido a la página de bloqueo, salvo por los diferentes mensajes que son enviados y son recibidos.

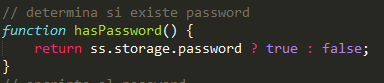
* Mensajes enviados de index.js a conf.js:
  + “storages”: envió de la variable de las paginas almacenadas
* Mensajes recibidos de index.js desde conf.js:
  + “delete”: recibe un índice (numérico), esto indica el elemento que debe ser borrado del array almacenado.
  + “changePassword”: recibe el password que ha sido cambiado y se cifra en el algoritmo SHA256, Se reinicia la petición de la contraseña al realizar el cambio.
* **Archivo Conf.js incrustado en la página:**
  + Recibe el array con las páginas bloqueadas, según el número de páginas se irán creando diferentes etiquetas HTML en el DOM.
  + Las etiquetas <LI> serán padres de la etiqueta <A> y un <BUTTON>
  + La etiqueta <A> contendrá el texto de cada ítem del array
  + <BUTTON> tendrá como identificador el índex representativo del ítem del array
  + Cada <BUTTON> se le asignara un evento “onclick” que enviará al index.js información sobre que página hay que borrar del bloqueo.
  + Por último en el apartado de password, se asignara también un evento “onclick” recogiendo el nuevo password introducido en el input para posteriormente enviarlo al fichero index.js



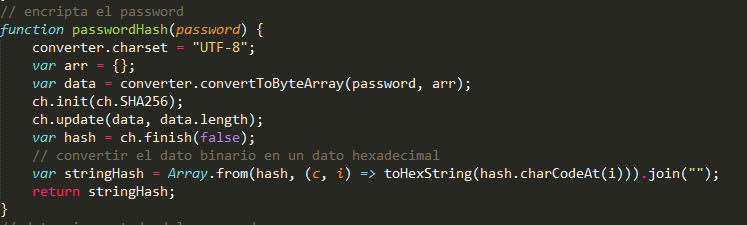
## 7.3. Anexo III

**Funciones**:

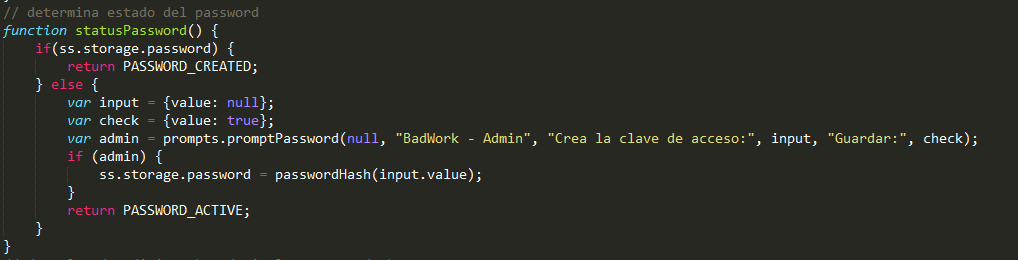
HasPassword:



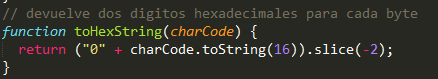
PasswordHash:



StatusPassword:



ToHexString:



CheckPassword:

