

BADWORK (control de acceso)

**2016**

Carlos Cañas Agenjo

IES Virgen de la Paz

01/01/2016



Índice

[1. Justificación del proyecto y objetivos. 3](#_Toc452315444)

[2. Introducción 4](#_Toc452315445)

[3. Metodología y desarrollo del proyecto 5](#_Toc452315446)

[3.1 Metodología 5](#_Toc452315447)

[3.2. Desarrollo del proyecto 6](#_Toc452315448)

[3.2.1. Comunicación con el cliente / Objetivo. 6](#_Toc452315449)

[3.2.2. Planificación. 6](#_Toc452315450)

[3.2.2.1. Documentación 6](#_Toc452315451)

[3.2.2.2. Diagrama Gantt 8](#_Toc452315452)

[3.2.3. Análisis de riesgo 8](#_Toc452315453)

[3.2.4. Ingeniería 9](#_Toc452315454)

[3.2.5. Evaluación / Pruebas 9](#_Toc452315455)

[3.2.6. Construcción y entrega 14](#_Toc452315456)

[4. Resultado y discusión 14](#_Toc452315457)

[5. Conclusiones 16](#_Toc452315458)

[6. Bibliografía y referencias 17](#_Toc452315459)

[7. Anexos/Otros 18](#_Toc452315460)

# 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.

# 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.

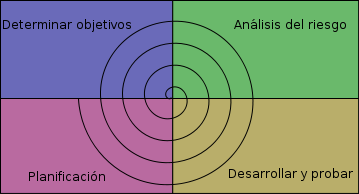
Mi proyecto de control de acceso se va a centrar en un control laboral, dentro de los ordenadores de una empresa, realizando un bloqueo de páginas web, para que los empleados puedan acceder sólo a los contenidos necesarios que se relacionen con su desarrollo laboral y así conseguir un aumento de su rendimiento.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

# 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).

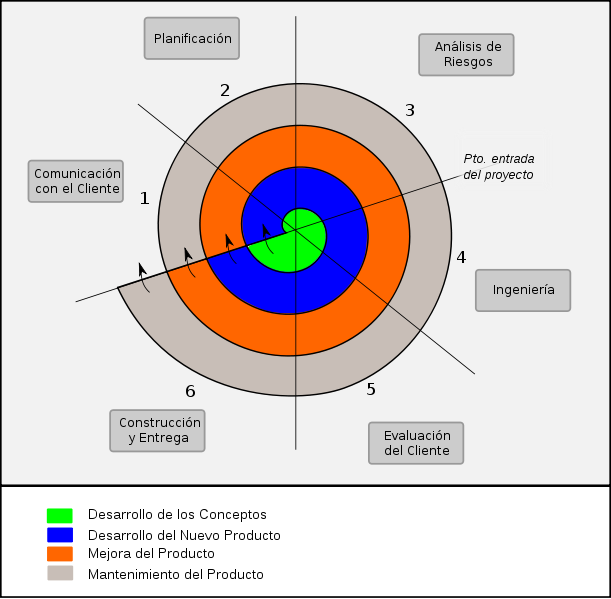
[](https://www.google.es/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwjig5HqlJbMAhVElxoKHaXhAH8QjRwIBw&url=https://es.wikipedia.org/wiki/Desarrollo_en_espiral&psig=AFQjCNHyma1VyqUC40jiTRmyewAZ2VANDQ&ust=1460999311743809)

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. 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.

* 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.

### 3.2.2. Planificación.

Este apartado estará dividido en dos subapartados:

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

### 3.2.2.1. Definición de la tecnología

#### 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

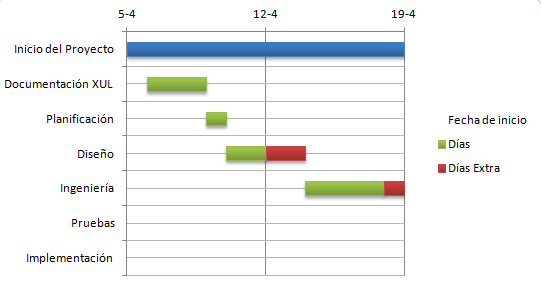
Comienzo a desarrollar el proyecto con la tecnología XUL, y me encuentro con varias dificultades que lo hacen incompatible, por lo que cambio a una tecnología más actual llamada ADD-ONS SDK (Software Development Kit). Este cambio es comentado y aceptado por el cliente ya que conlleva un retraso en la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

#### ADD-ONS SDK

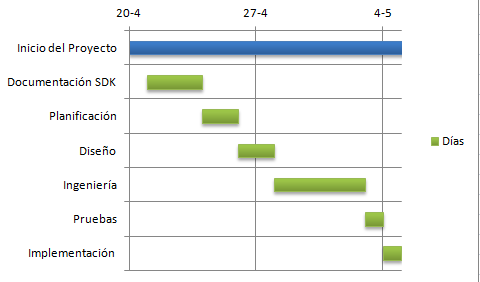
Usando Add-on SDK puedes crear complementos Firefox utilizando las tecnologías Web estándar: JavaScript, HTML y CSS. SDK incluye una API de JavaScript que se puede utilizar para crear complementos y herramientas de creación, funcionamento, pruebas y empaquetado llamadas JPM que será descrita en el apartado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

### 3.2.2.2. 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 XUL | 06/04/2016 | 06/04/2016 | 3 | 0 | 08/04/2016 |
| Planificación | 09/04/2016 | 09/04/2016 | 1 | 0 | 09/04/2016 |
| Diseño | 10/04/2016 | 10/04/2016 | 2 | 2 | 13/04/2016 |
| Ingeniería | 11/04/2016 | 14/04/2016 | 4 | 2 | 19/04/2016 |
| Pruebas | 15/04/2016 |  | 2 | 0 |  |
| Implementación | 17/04/2016 |  | 2 | 0 |  |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 SDK | 21/04/2016 | 21/04/2016 | 3 | 0 | 23/04/2016 |
| Planificación | 24/04/2016 | 24/04/2016 | 2 | 0 | 25/04/2016 |
| Diseño | 26/04/2016 | 26/04/2016 | 2 | 0 | 27/04/2016 |
| Ingeniería | 28/04/2016 | 28/04/2016 | 5 | 0 | 02/05/2016 |
| Pruebas | 03/05/2016 | 03/05/2016 | 1 | 0 | 03/05/2016 |
| Implementación | 04/05/2016 | 04/05/2016 | 1 | 0 | 04/05/2016 |



### 3.2.3. 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.
* 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 encriptadas 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.

### 3.2.4. Ingeniería

Para desarrollar el proyecto usaré las siguientes herramientas:

* Editor de texto Sublime Text
* Navegador Firefox con las siguientes extensiones instaladas: firebug, colorzilla.
* Creacion de un repositorio GitHub para mantener un control de versiones del proyecto.

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/modulos 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. De este grupo se van a usar los siguientes modulos:
  + simple-storage
  + self
  + tabs
  + page-mod
  + ui
  + url
  + panel
* APIs de bajo nivel: estos modulos todavía están en desarrollo pero se espera no ser utilizado en futuras versiones. Se dividen en modulos para poder crear colecciones, modulos para crear eventos y trabajadores, y por ultimo modulos que sirven como complemento de APIS de alto nivel. De este grupo se van a usar los siguientes modulos:
  + ui/button/toggle

A parte de estos módulos Add-ons SDK se pueden usar modulos 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, por lo que estos modulos 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 su alias:

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

Ejemplo:

* var {Cc, Ci} = require("chrome");

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 utilizaran los siguientes XPCOM que se explicarán en el desarrollo de las versiones de la aplicación:

* 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.

Modulos de alto nivel:

**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 del añadido.

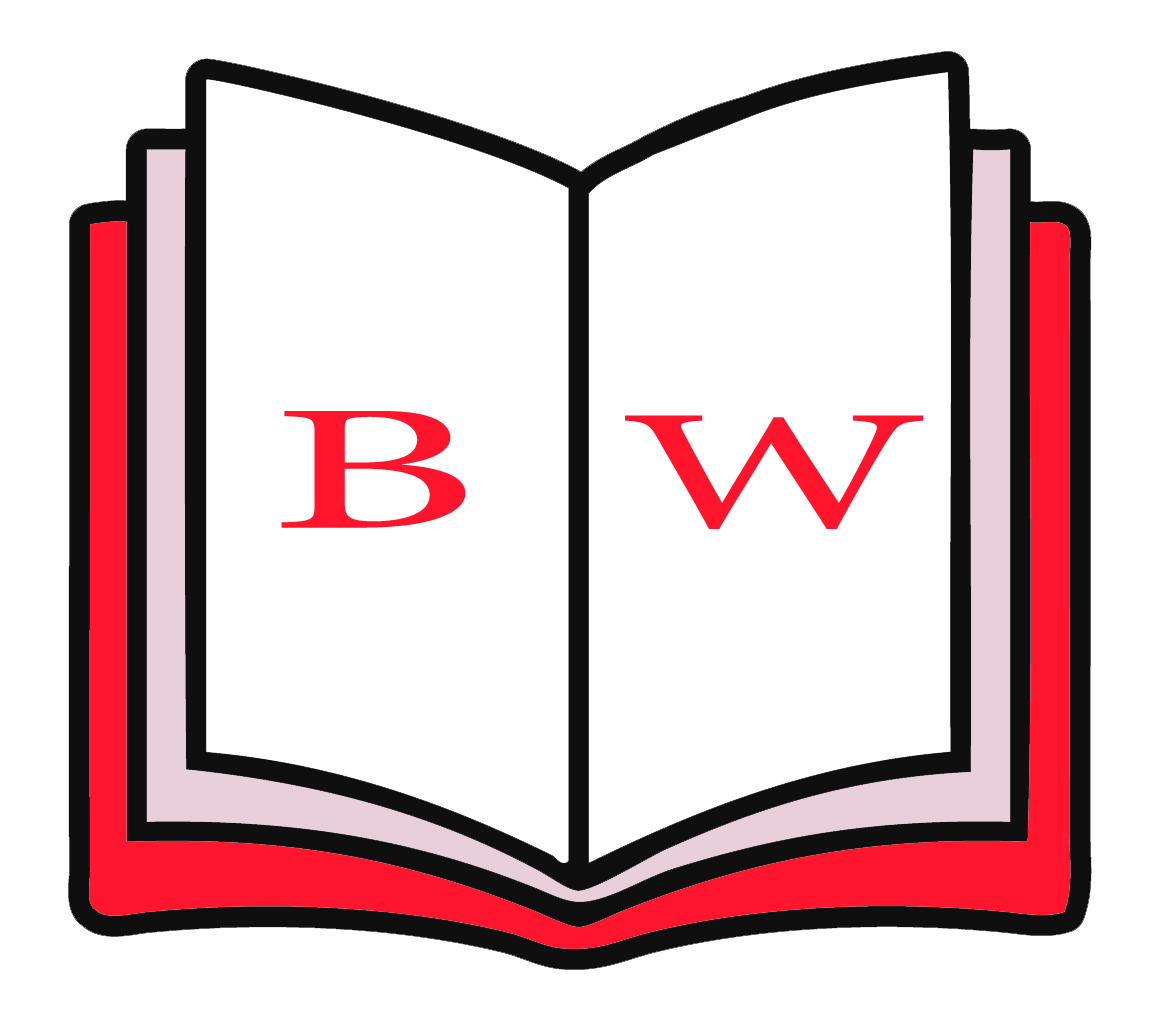
Métodos usados en este proyecto:

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.

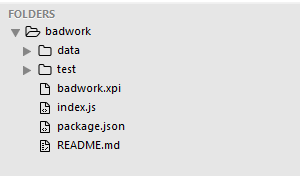
**Tabs:**

#### 3.2.4.1. BadWork 0.0.1.

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 y 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.

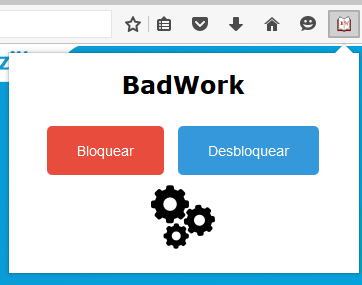
**Package.json:** archivo que describe características de la API, autor, colaborador, versión soportada, descripción, nombre, etc.

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

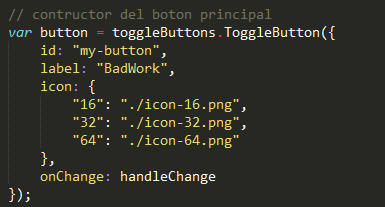
Almaceno en la carpeta data tres imágenes con distinto tamaño:

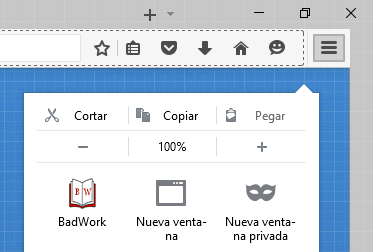
* Icon-16.png
* Icon-32.png
* Icon-64.png

Se crea un fichero panel.html y una hoja de estilos style.css en la carpeta data para dar el siguiente aspecto:



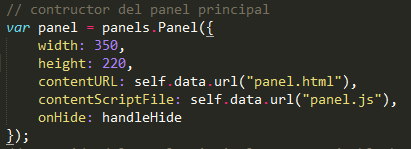
Se crean los diferentes constructores que dependen de los módulos (botón, panel).

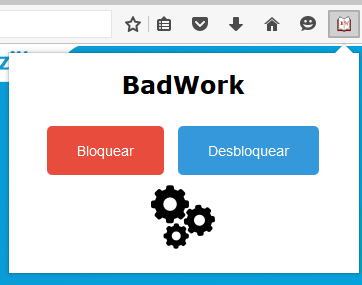


Se construye el botón y se le asigna un id, un label, las imágenes que están en la carpeta data y una función que al pulsarlo haga que se muestre el panel.

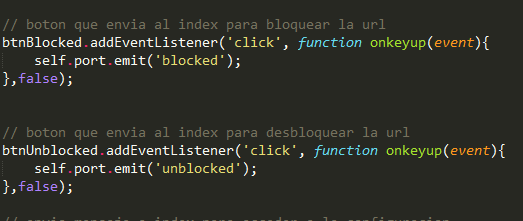
Screenshot_3.png

Se construye el panel donde especificamos el ancho y alto, el contenido del panel (que será el archivo panel.html), también irá asociado un archivo javaScript y una función que hace que sólo se active el panel cuando se pincha el icono, sino quedaría oculto.





Con el archivo panel.js añadimos eventos a los botones que al pulsarlos van a mandar un mensaje que será recogido por el archivo Index.js. Esto se utiliza para recoger las acciones que se produzcan en el panel e interactuar con el usuario. Los mensajes pueden ser codificados a través de un JSON, string, array y objects. En este caso mandaremos un string.

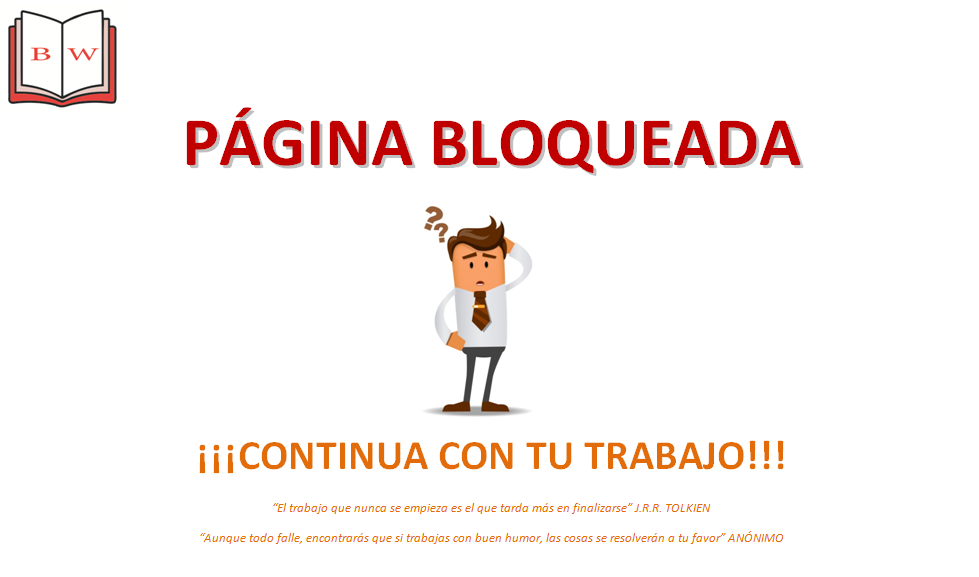


Se construye un array donde se almacenarán todas las páginas que se bloqueen.

En index.js al recibir los mensajes ‘blocked’ se almacenará la página actual en el array y se bloqueará la página mostrando blocked.html.

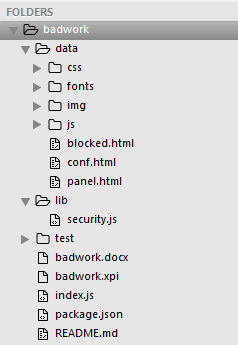
Si por el contrario el mensaje es ‘unblocked’ se borrará la página del array y mostrará la página que fue bloqueada.

Para el diseño, me he centrado en algo llamativo y actual, para así sorprender al trabajador y que su reacción sea positiva, para animarle a que continúe con su actividad laboral y no se distraiga con el acceso libre que permita internet.



#### 3.2.4.1. BadWork 0.0.2.

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:

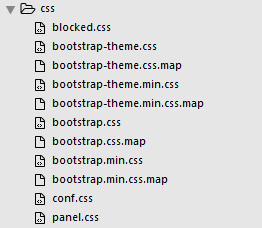
* **Css**: se almacén las hojas de estilos.
* **Fonts**: las fuentes que se pueden utilizar.
* **Img**: las imágenes de la aplicación.
* **Js**: todos los archivos javascript

**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.

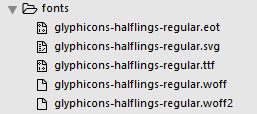
Para incluir la librería solamente he tenido que descargar las versiones más actuales de estas librerías e incluirlas en las diferentes subcarpetas de Data:

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

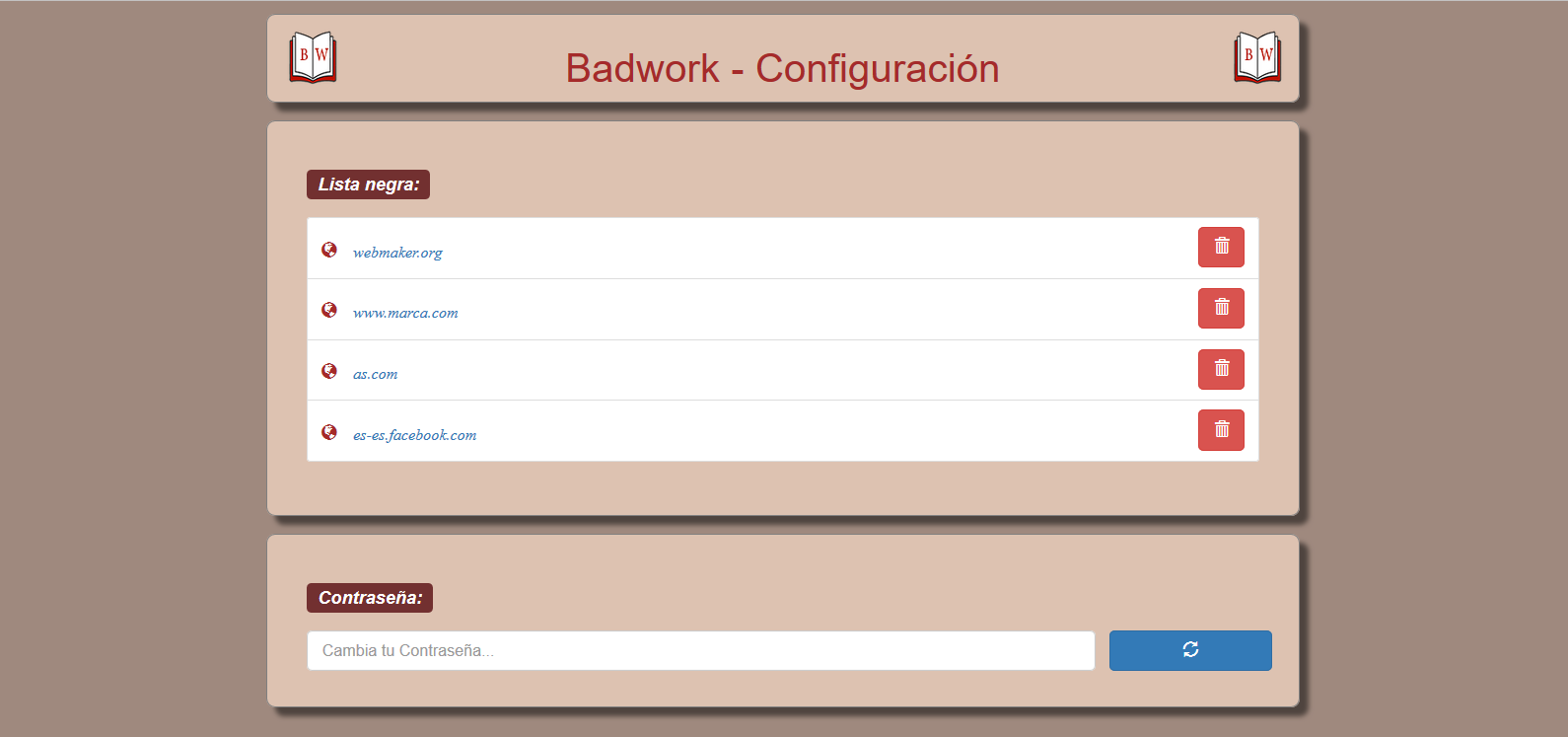
**Blocked.html:**



Como podemos observar:

* 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:**



Como podemos observar:

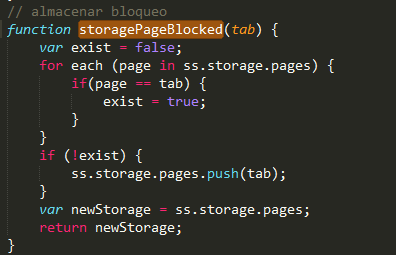
* El header es exactamente igual al de BLOCKED.HTML salvo que cambia el texto.
* Se ha creado un contenedor con una lista NEGRA, de las paginas que se han ido bloqueando, y va actualizando según la interacción de la aplicación.
* Se incluyen botones a cada ítem para tener la posibilidad de eliminarlos 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.

Se añaden diferentes funciones para optimizar el código:

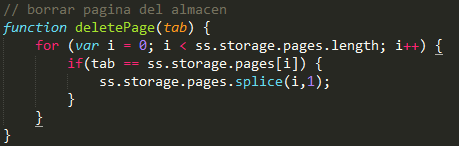
**GetCurrentUrl:** devuelve la última url conocida dependiendo de pestaña que este activa en ese mismo momento.



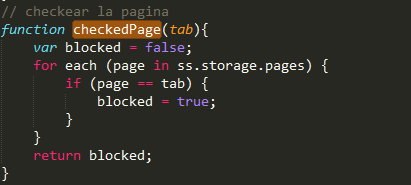
**StoragePageBlocked:** se almacena la pagina que se deseo bloquear en la aplicación.



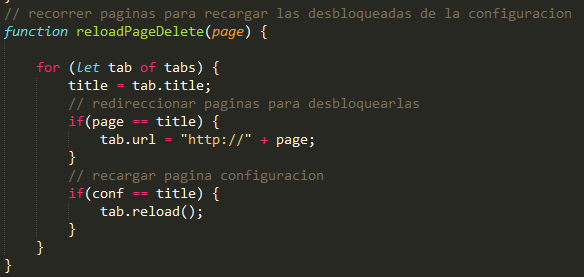
**DeletePage:** borra la página almacenada en la aplicación.



**CheckPages:** verifica si una página está guardada en almacenamiento o no.

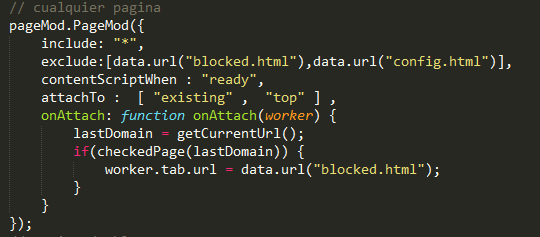
****

**ReloadPageDelete:** recorrer todas las pestañas de la ventana del navegador para desbloquear las páginas que se han desbloqueado en el navegador y recargar la página de configuración para actualizar la LISTA NEGRA.



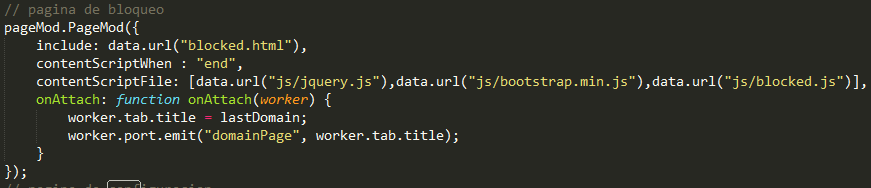
Se añaden tres constructores para modificar las páginas según su URL:

**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.

**Blocked.html:**



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.

* **Blocked.js:**



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.

**Conf.html:**

****

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 encripta en el algoritmo SHA256, Se reinicia la petición de la contraseña al realizar el cambio.
* **Conf.js:**
  + 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



### 3.2.5. Evaluación / Pruebas

Casos de uso

1. Desarrollo y prueba

La estructura a seguir:

- Crear un archivo “.manifest” e “install.rdf” para poder generar el programa y probarlo.

- Comprimir todo el proyecto en formato “.zip” y luego cambiarlo a formato “.xpi”.

- Comprimir el contenido del proyecto excepto “.manifest” e “install.rdf” en formato”.zip” y cambiarlo a formato “.jar”.

### 3.2.6. Construcción y entrega

Documentación sobre la firma de Mozilla y subida a la lista de Complementos

# 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.

# Conclusiones

# 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>

# Anexos/Otros