Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas.

Extensión de Mozilla Firefox para su personalización

Autor:

José Ernesto Cortes Mendez

Tutores:

Ing. Inst. Ivan Romay Aragón

Ing. Luis Daniel Sierra Corredera

Ing. Inst. Yanet Cabeza Chávez

La Habana, Julio 2018

Agradecimientos

A mi familia por su apoyo incondicional, específicamente a mis padres por confiar siempre en mí.

A mis tutores por la paciencia, y el apoyo que me han ofrecido.

A mis amigos por estar siempre ahí, como esa segunda familia, por soportarme y alentarme en todo momento.

A todas aquellas personas que de una forma u otra me aconsejaron, sugirieron o se preocuparon por mí en todo este proceso.

Dedicatoria

Este trabajo de diploma está dedicado a toda mi familia especialmente a mi hermano

Victor Manuel Cortes Mendez.

Declaración de autoría

Declaro por este medio que yo José Ernesto Cortes Mendez, con carné de identidad 94042310944 soy el autor principal del trabajo titulado “Extensión de Firefox para su personalización” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los ___días del mes de junio del año 2018.

_____

Firma del Autor

_____ _____

Ing. Yanet Cabeza Chávez — Tutora

Ing. Luis Daniel Sierra Corredera — Tutor

_____

Ing. Iván Romay Aragón

Tutor

Resumen

En el Centro de *Software* Libre de la Universidad de las Ciencias Informáticas, se desarrolla Nova Elecciones, el cual es un ambiente con restricciones de mínimo acceso desde el navegador web *Mozilla Firefox* y concebido para ser la base tecnológica que se empleará en los próximos procesos electorales del país. El actual sistema no permite crear personalizaciones en el navegador de manera fácil, pues no cuenta con una interfaz sencilla para dicho propósito, además, las extensiones web utilizadas con anterioridad no son compatibles con las nuevas versiones de Mozilla Firefox. La presente investigación propone la realización de una extensión web con el objetivo de facilitar la creación de personalizaciones en *Mozilla Firefox*. Para ello se realiza un estudio de las principales herramientas que permiten la realización de personalizaciones de acuerdo a las características de la propuesta de solución. Se define como metodología de desarrollo AUP versión UCI, la cual guiará el proceso de desarrollo de *software.* Para la implementación de la solución propuesta se define como Entorno de Desarrollo Integrado (IDE) *WebStorm*, como herramienta de modelado *Visual Paradigm* y como lenguaje de programación JavaScript. Finalmente se obtuvo como resultado práctico de la investigación, la extensión web Nova Kiosko la cual posibilita la creación de personalizaciones en el Navegador web *Mozilla Firefox*.

Palabras clave: extensión web, navegador web, nova elecciones, personalización.

Índice de Contenido

Introducción 1

Capítulo 1. Fundamentación teórica sobre extensiones web para la creación de personalizaciones1

1.1 Definiciones asociadas al tema investigativo 1

1.1.1 Navegador WEB Mozilla Firefox 1

1.2 Navegador de Mozilla Firefox Quantum o Firefox 57 3

1.2.1 Cambios para programadores web 3

1.3 Desarrollo de extensiones web para la personalización de navegadores 6

1.3.1 Anatomía de una extensión web en la actualidad 6

1.3.2 WebExtensions API 9

1.3.3 Desarrollo de extensiones web antes de WebExtensions 10

1.4 Análisis de extensiones de Firefox que permiten crear personalizaciones 11

1.5 Tecnologías de implementación 13

1.5.1 Metodología de desarrollo de software 13

1.5.2 Lenguaje de modelado: UML 14

1.5.3 Herramienta CASE para la modelación del sistema 14

1.5.4 Lenguajes de programación 14

1.5.5 Lenguajes de etiquetado 15

1.5.6 Entorno de Desarrollo Integrado (IDE). 16

1.6 Conclusiones del capítulo 17

Capítulo 2. Análisis y diseño de la extensión web para la creación de personalizaciones en Mozilla Firefox. 18

2.1 Descripción de la propuesta de solución 18

2.2 Requisitos 19

2.2.1 Requisitos funcionales 19

2.2.2 Requisitos no funcionales 21

2.2.3 Historias de usuario (HU) ... 22
2.3 Análisis y diseño ... 26
2.3.1 Arquitectura ... 26
2.3.2 Diagrama de paquetes ... 27
2.3.3 Patrones de diseño ... 28
2.3.4 Patrón Fachada (Facade) ... 28
2.3.5 Patrón Observador (Observer) ... 29
2.3.6 Patrón de Devolución de llamada (Callback) ... 30
2.4 Conclusiones del capítulo ... 31
Capítulo 3. Implementación y pruebas de la extensión web para la creación de personalizaciones en Mozilla Firefox 32
3.1 Diagrama de componentes ... 32
3.2 API WebExtensions para desarrollo de extensiones web ... 32
3.3 Código Fuente ... 36
3.3.1 Estándar de codificación ... 36
3.4 Pruebas de Software ... 37
3.4.1 Niveles de Prueba ... 38
3.4.2 Métodos de Prueba ... 38
3.4.3 Diseño de los casos de prueba ... 42
3.5 Resultados obtenidos de los casos de prueba ... 45
3.6 Pruebas de Aceptación ... 46
3.7 Conclusiones del capítulo ... 47
Conclusiones ... 48
Recomendaciones ... 49
Referencias ... 50
Anexos ... 54

Índice de Ilustraciones

Ilustración 1: Anatomía de una extensión web (Mozilla Firefox) 7
Ilustración 2: Propuesta de solución (Elaboración propia). 19
Ilustración 3: Diagrama de paquetes (Elaboración propia). 28
Ilustración 4: Evidencia del patrón Fachada (Elaboración propia). 29
Ilustración 5: Evidencia del patrón Observador (Elaboración propia). 30
Ilustración 6: Evidencia del patrón Devolución de llamada (Elaboración propia). 30
Ilustración 7: Diagrama de componentes (Elaboración propia). 32
Ilustración 8: Descripción de la propuesta de solución para requisitos que no se desarrollan con API de WebExtensions 34
Ilustración 9: Código fuente del archivo userChrome.css 35
Ilustración 10: Ejemplo del navegador personalizado por mediación de la extensión web y el archivo userChrome.css 35
Ilustración 11: Ejemplo de Declaraciones de métodos y variables.(Elaboración propia) 37
Ilustración 12: Ejemplo del Estándar de Codificación Saltos de líneas, después de un punto y coma o abrir una llave.(Elaboración propia) 37
Ilustración 13: Esquema del Método de Caja Negra. 39
Ilustración 14: Aplicación de las pruebas unitarias mediante el método de caja blanca con la técnica camino básico (Elaboración propia). 40
Ilustración 15: Gráfica del programa mediante nodos y aristas (Elaboración propia). 41
Ilustración 16: Resultado de las pruebas funcionales (Elaboración propia). 46
Ilustración 17: Comparación de los navegadores web más utilizados actualmente en el mundo según www.w3counter.com 68
Ilustración 18: Comparación de los navegadores web más utilizados actualmente en el mundo según https://gs.statcounter.com 68
Ilustración 19: Comparación de los navegadores web más utilizados actualmente en el mundo según www.w3schools.com 69
Ilustración 20: Comparación de los navegadores web más utilizados actualmente en el mundo según www.w3counter.com 69

Índice de tablas

Tabla 1 Requisitos Funcionales (Elaboración propia) 20
Tabla 2: Requisitos no funcionales (Elaboración propia) 22
Tabla 3: Historia de usuario No 4 22
Tabla 4: Historia de usuario No 6 23
Tabla 5: Historia de usuario No 14 24
Tabla 6 Historia de usuario No 15 25
Tabla 7: Requisitos funcionales desarrollados mediante el archivo userChrome.css 33
Tabla 8: Listado de caminos independientes.(Elaboración propia) 41
Tabla 9: Caso de prueba de unidad, camino1(Elaboración propia) 42
Tabla 10: Caso de prueba de unidad, camino 2.(Elaboración propia) 42
Tabla 11 :Caso de Prueba de Validación para la Historia de Usuario Gestionar opciones de guardado.(Elaboración propia) 43
Tabla 12: Historia de usuario No.1 54
Tabla 13: Historia de usuario No.3 54
Tabla 14: Historia de usuario No.7 55
Tabla 15: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar botones del ratón.(Elaboración propia) 57
Tabla 16: Caso de Prueba de Validación para la Historia de Usuario Personalizar Menú Contextual.(Elaboración propia) 58
Tabla 17: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar combinaciones de teclas.(Elaboración propia) 59
Tabla 18: Caso de Prueba de Validación para la Historia de Usuario Personalizar apertura de vínculos.(Elaboración propia) 61
Tabla 19: Caso de Prueba de Validación para la Historia de Usuario Personalizar apertura de documentos descargados.(Elaboración propia) 62
Tabla 20: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar gestión de las pestañas.(Elaboración propia) 64

Introducción

La historia de Internet se remonta al temprano desarrollo de las redes de comunicación en la segunda mitad del pasado siglo, específicamente, en el año 1969 (RAMOS, 2011). Su surgimiento potenció el crecimiento vertiginoso de muchas ramas de la ciencia y la tecnología, posibilitó un acceso más fácil a la información, fomentando la creación de foros virtuales para compartir conocimientos y generar debates científicos, sociales y culturales.

El auge de este fenómeno como factor principal de la globalización de la información es la causa fundamental por la que herramientas que propician su utilización avanzan, debido a que esta evolución de los medios ha acaparado gran atención académica y profesional. Estas herramientas informáticas llamadas Navegadores Web, Buscadores web o *Browsers* son las que permiten acceder a toda la información contenida en Internet mediante enlaces de hipertexto (Diccionario de la Real Academia Española, 2018).

A lo largo de los años se han desarrollado disímiles navegadores que han evolucionado con el avance de las tecnologías, algunos de estos son: *Mosaic, Netscape Navigator, Opera, Safari, Google Chrome, Internet Explorer (Microsoft Edge para versiones de Microsoft Windows modernas)* y *Mozilla Firefox* entre otros. A finales del 2004 aparece en el mercado Firefox, una rama de desarrollo de Mozilla. Este proyecto pretendía eliminar todas las funciones ajenas a un navegador propiamente dicho y mejorar su código e interfaz, haciéndolo más rápido, seguro, y completamente personalizable por el usuario (Huelga, 2010).

Para acceder a la web los navegadores más utilizados actualmente son *Google Chrome*, *Safari*, *Mozilla Firefox* e *Internet Explorer* según los sitios *W3Counter*[1], *StatCounter* [2] y *W3Schools* [3]. (Ver Anexo 4)

[1] *W3Counter*: Sitio web fundado en el año 2004 que tiene como principal objetivo brindar una alternativa para el análisis de trafico web.

[2] *StatCounter*: Herramienta para el análisis del tráfico web creado en 1999. Según su sitio oficial, más de dos millones de sitios web usan *StatCounter*.

[3] *W3Schools*: Es un sitio web para aprender tecnologías web en línea, contiene tutoriales de *HTML*, *CSS*, *JavaScript* entre otras. Fue lanzado en 1999 y desde 2002 almacena estadísticas del uso de los navegadores web, mensualmente recibe más cincuenta millones de visitas.

Estos navegadores encabezan desde hace años la competencia por su rendimiento, calidad, interfaz, rapidez, seguridad y extensibilidad. El navegador que más se utiliza en ambientes de desarrollo del Proyecto GNU Linux es *Mozilla Firefox,* con una filosofía de Software Libre que lo hace casi completamente personalizable por el usuario mediante complementos, denominados también extensiones web*.* Las extensiones son populares en *Firefox*, pues la intención de los desarrolladores de Mozilla es reducir los errores de *software*, manteniendo un alto grado de extensibilidad para que los usuarios individuales puedan agregar las características que ellos prefieren (Huelga, 2010). Actualmente el sitio web oficial donde los usuarios pueden subir e instalar extensiones web es *AMO* (del acrónimo *Addons.Mozilla.Org*).

Existen varias herramientas para desarrollar extensiones de *Firefox*, como *CFX* y *JPM*; otras como los complementos de superposición, complementos *bootstrapped* y *Add-on SDK*[4], quedaron obsoletas desde finales de noviembre de 2017, cuando se decide que las extensiones deben construirse utilizando las *API*[5] de *WebExtensions*. Las extensiones desarrolladas usando las *API* de *WebExtensions* para *Firefox* están diseñadas para ser compatibles entre navegadores. En la mayoría de los casos, se ejecutarán en *Chrome*, *Edge* y *Opera* con pocos o ningún cambio. También son totalmente compatibles con multiprocesos *Firefox* (developer.mozilla, 2017).

La Universidad de Ciencias Informáticas (UCI) surge en el año 2002, concebida como un centro de nuevo tipo, de alcance nacional, atípica y con tareas concretas en el proceso de informatización de la sociedad cubana, con énfasis en la producción de software. Siguiendo este objetivo, durante estos 16 años de funcionamiento y perfeccionamiento de la universidad, se han creado varios centros de desarrollo, entre los que se encuentra el Centro de Software Libre (CESOL) de la Facultad 1. Este centro presenta como uno de sus productos más importantes la Distribución Cubana de GNU/Linux Nova.

[4] SDK: *SDK* es el acrónimo de "*Software Development Kit*" (Kit de desarrollo de software). El *SDK* reúne un grupo de herramientas que le permite al programador o desarrollador de software crear una aplicación informática para un sistema concreto.

[5] *API*: Una *API* es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Las siglas *API* vienen del inglés *Application Programming Interface*. En español sería Interfaz de Programación de Aplicaciones

Se desarrollan tres variantes de Nova para contribuir al desarrollo de la informatización de la sociedad: Nova Escritorio, Nova Ligero y Nova Servidores, de acuerdo a las necesidades de los OACE [6]. A pesar de ser estas las principales líneas de productos en desarrollo, también se realizan personalizaciones a la medida por encargo y necesidad de algunas instituciones nacionales. Una de estas personalizaciones es Nova Elecciones, el cual es un sistema a la medida basado en Nova y un grupo mínimo de aplicaciones de escritorio, entre ellas Firefox. Este presenta una interfaz personalizada para cumplir con un grupo bien definido de restricciones que garantizan la seguridad de la información que se maneja en el sistema electoral cubano.

Para personalizar Firefox hasta la versión 52 se utilizan un conjunto de extensiones, lo que permitió cumplir con los requerimientos del software, entre ellos: *R-Kiosk*, *Tab Mix Plus* y *Max Tabs*.

El Centro de Software Libre decide hacer una extensión de Firefox debido a que:

- Se hace necesario unificar todas las funcionalidades de las anteriores extensiones (*R-Kiosk*, *Tab Mix Plus* y *Max Tabs*) que presentan similares características a la que se propone desarrollar, además de crear nuevas funcionalidades, todas estas en una sola extensión para dar la posibilidad al usuario de personalizar el navegador.
- A partir de noviembre del 2017 Mozilla cambió las tecnologías para el desarrollo de extensiones web, actualmente se utilizan las *WebExtensions*. A partir del Firefox en su versión 57 (Quantum) las extensiones desarrolladas con tecnologías como XPCOM, add-ons SDK, bootstraped u overlays son obsoletas, inutilizables por el navegador. Debido a toda esta situación se ratifica la necesidad e importancia de desarrollar una extensión web para Mozilla Firefox con esta nueva tecnología, posibilitando usar las nuevas versiones del navegador de Mozilla en las próximas jornadas electorales.

A partir de la situación anteriormente expuesta se identifica el siguiente **problema a resolver**:

¿Cómo facilitar la creación de personalizaciones para *Mozilla Firefox* desde una extensión web?

[6] OACE: Organismos de Administración Central del Estado

Como **objeto de estudio** se propone: El proceso de desarrollo de extensiones web para la creación de personalizaciones. El **campo de acción** se enmarca en el desarrollo de extensiones web para la creación de personalizaciones desde el navegador web *Mozilla Firefox* a partir de la versión 57.

Para dar solución al problema existente se propone como **objetivo general** desarrollar una extensión web de *Mozilla Firefox* que facilite la creación de sus personalizaciones.

Objetivos específicos

1. Elaborar el marco teórico referente al proceso de desarrollo de extensiones web de *Mozilla Firefox* para la creación de sus personalizaciones.
2. Diseñar una extensión web de *Mozilla Firefox* para la creación de sus personalizaciones.
3. Implementar una extensión web de *Mozilla Firefox* para la creación de sus personalizaciones.
4. Evaluar la extensión web de *Mozilla Firefox* para la creación de sus personalizaciones.

Preguntas científicas

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de desarrollo de extensiones web de Mozilla Firefox para la creación de sus personalizaciones?
2. ¿Qué aspectos se deben tener en cuenta para diseñar una extensión web de *Mozilla Firefox* para la creación de sus personalizaciones?
3. ¿Cuáles son las tecnologías más adecuadas para implementar una extensión web de *Mozilla Firefox* para la creación de sus personalizaciones?
4. ¿Cuáles pruebas de *software* aplicar en la evaluación de la extensión web de *Mozilla Firefox* para la creación de personalizaciones?

Tareas de investigación:

1. Elaboración del marco teórico metodológico referente al proceso de desarrollo de extensiones web para la creación de personalizaciones.
2. Análisis de los requerimientos de la propuesta de solución.
3. Diseño de la propuesta de solución.
4. Evaluación de las salidas del complemento para la puesta en acción del mismo, seguridad, requerimientos necesarios y calidad.
5. Evaluación de los resultados obtenidos a partir de los métodos de prueba definidos en la investigación.

Los **Métodos de investigación científica** utilizados para darle solución al objetivo propuesto son:

Métodos teóricos:

Analítico – Sintético: Este método permite analizar las teorías y los documentos referentes al objetivo de la investigación, facilitando de esta forma la extracción de los elementos más importantes relacionados con el objeto de estudio. Además, posibilita construir el camino a seguir a partir del análisis detallado de cada uno de los documentos previamente mencionados.

Histórico – Lógico: Es utilizado en el análisis de los sistemas homólogos, de manera que permite identificar elementos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada.

La estructura del documento se caracteriza por la presencia de: introducción, tres capítulos, conclusiones, bibliografía y los anexos. El contenido de cada capítulo se describe a continuación:

Capítulo 1: Fundamentación teórica sobre de extensiones web para la creación de personalizaciones: En este capítulo se describen los elementos fundamentales relacionados a las extensiones web en Firefox, además de sus características ventajas y desventajas. Se detallan las principales características de la metodología que guiará el proceso de desarrollo de la extensión. Se definen las herramientas y tecnologías que son utilizadas en la implementación.

Capítulo 2: Análisis y diseño de la extensión web para la creación de personalizaciones en *Mozilla Firefox*: En este capítulo se hará referencia a la implementación del sistema donde los elementos del diseño se convierten en elementos de implementación en términos de componentes. Se definen la arquitectura y los patrones de diseño a utilizar además de los requisitos funcionales y no funcionales con los que debe cumplir la propuesta de solución.

Capítulo 3: Implementación y pruebas de la extensión web para la creación de personalizaciones en *Mozilla Firefox*: En este capítulo se documentan los artefactos asociados a la implementación de la propuesta de solución y se realizan los casos de prueba para lograr el desarrollo de un software con la calidad requerida. Además, se establecen los estándares de codificación que se tuvieron en cuenta para el desarrollo de la extensión.

Capítulo 1. Fundamentación teórica sobre extensiones web para la creación de personalizaciones

En el presente capítulo se definen los principales conceptos asociados al tema de investigación. Se realiza un estudio del navegador web *Mozilla Firefox* y se describe el desarrollo de extensiones web para la personalización de navegadores. También se aborda todo lo relacionado a las principales herramientas a utilizar además de un estudio de las tecnologías necesarias para la implementación de la propuesta de solución.

1.1 Definiciones asociadas al tema investigativo

Concepto de *WebExtensions*: *WebExtensions* es un sistema para desarrollar extensiones de navegador en *Firefox*. Este sistema está definido bajo el estándar de la W3C[7] y proporciona *API*, que en gran medida son compatibles con diferentes navegadores como *Mozilla Firefox, Google Chrome, Opera Browser* y *Microsoft Edge* (developer.mozilla, 2017).

Concepto de Extensión web: Las extensiones son un conjunto de archivos que tienen la capacidad de extender y modificar la funcionalidad de un navegador web. Están escritos utilizando los estándares de tecnología Web (regidos por la W3C), *JavaScript*, *HTML*, y *CSS* - más algunas *API* dedicadas *JavaScript*. Una extensión puede agregar nuevas características al navegador o cambiar la apariencia o contenido de un sitio web en particular (developer.mozilla, 2017)

1.1.1 Navegador WEB Mozilla Firefox

Mozilla surge en 1998 como un proyecto de la comunidad *Open Source*, a partir de Netscape[8]. Se originó gracias a una comunidad de personas que querían construir algo a su gusto, muy diferente a los deseos

[7] *W3C*: El *World Wide Web Consortium* (W3C) es la principal organización de estándares internacionales para la *World Wide Web* (abreviado *WWW* o *W3*).

[8] *Netscape Navigator* es un navegador web desarrollado por la compañía *Netscape Communications*, creada por Marc Andreessen. Netscape fue el primer navegador comercial.

de las compañías que solo desean lanzar sus productos. Son cerca de 200 personas trabajando en Mozilla, pero también tienen miles de aliados por todo el mundo desarrollando código y decenas de miles de personas que se dedican a probar sus productos (Rodriguez, 2010).

Mozilla Firefox es un navegador independiente, originado a partir del navegador incorporado en la *suite*[9], que ofrece control sobre las páginas que se están viendo, bloqueando ventanas emergentes molestas y que a su vez incluye soporte para accesibilidad DHTML[10]. Posee un corrector ortográfico integrado para evitar cometer errores, permite agregar funcionalidades mediante extensiones, y es compatible con los estándares de W3C[11], así como una alta seguridad.

Firefox comenzó como una rama experimental del proyecto Mozilla a cargo de Dave Hyatt, Joe Hewitt y Blake Ross. En su opinión, las exigencias comerciales del patrocinio de *Netscape* y el gran número de características de *Mozilla Application Suite* comprometían la utilidad de este. Para combatir lo que ellos denominaban inflada *Mozilla Application Suite*, crearon un navegador independiente con la intención de reemplazarla.

El 3 de abril de 2003, la Organización *Mozilla* anuncia que centraría sus esfuerzos en *Firefox* y *Thunderbird*. El 9 de febrero de 2004 se bautiza finalmente como ***Mozilla Firefox***, a menudo referido simplemente como "*Firefox*" y abreviado como "Fx" o "fx", o más común como "FF". Este nombre se eligió por su semejanza con *Firebird* y por ser único en la industria informática. (Rodriguez, 2010).

[9] *Suite*: En la informática, una suite ofimática **o** de oficina es un conjunto de programas (*software*) que permite realizar distintas tareas típicas del mundo laboral. Redactor documentos, realizar balances contables e imprimir información forman parte de las funciones habituales de una suite.

[10] *DHTML*: *HTML* Dinámico o *DHTML* (del inglés *Dynamic HTML*) designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de algún lenguaje de marcado estático (como *HTML*), un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de hojas de estilo en cascada (*CSS*) y la jerarquía de objetos de un *Document Object Model* (*DOM*).

1.2 Navegador de Mozilla Firefox Quantum o Firefox 57

Se define a Firefox 57 como Quantum por el proyecto de ingeniería Firefox Quantum, cuyo objetivo ha sido el de reconstruir Firefox desde cero para darle un rendimiento y una estabilidad excelentes, así como mejorar su apariencia visual. Esta es la primera versión de Firefox en incluir algunos de estos cambios (developer.mozilla, 2017).

El nuevo procesador de CSS en paralelo de Firefox, también denominado ***Quantum CSS*** o ***Stylo*** está activado de manera predeterminada en Firefox 57 para escritorio; las versiones aun no cuentan con esta característica. Los programadores no deben notar ninguna diferencia importante, aparte de la amplia gama de mejoras de rendimiento. Sin embargo, existen algunas diferencias menores de funcionalidad en *Stylo*, las cuales se han implementado para corregir comportamientos no estándares de Gecko que habrían de desaparecer (developer.mozilla, 2017).

1.2.1 Cambios para programadores web

Herramientas de desarrollo

No hay ningún cambio.

HTML

Los tipos «date» y «time» de <input> ya están activados en todas las compilaciones.

CSS

- Ahora son admitidos los valores minimal-ui y standalone de la consulta display-mode.
- Se corrige un problema que causaba que la notación abreviada de la cuadrícula quedara restablecida al utilizar grid-row-gap y grid-column-gap.
- Se ha eliminado la preferencia layout.css.clip-path-shapes.enabled, que permitía activar o desactivar la compatibilidad de <basic-shape> con clip-path.

Notas sobre Firefox Quantum

- Los valores de degradados radiales tales como radial-gradient (circle gold, red) funcionan en el sistema de estilos de Gecko anterior, pero no debería por la coma faltante entre circle y gold. Quantum corrige este defecto.
- Cuando se da animación a un elemento ubicado fuera de la pantalla y se establece un tiempo de retardo, Gecko no actualiza la visualización correctamente en algunas plataformas, como Windows. Esto ha quedado corregido en Quantum.
- En Gecko, no es posible desplegar de manera predeterminada los elementos mediante el atributo open si tienen una propiedad animation activa. Quantum corrige este problema.
- En Gecko, los efectos transitions no funcionan si existe una transición desde un text-shadow con un color especificado a un text-shadow sin ningún color. Este comportamiento se ha corregido en Quantum.
- En Gecko, cancelar una animación de relleno (p. ej., con animation-fill-mode: forwards definido) puede desencadenar un conjunto de transiciones en el mismo elemento, aunque únicamente una vez. Por lo general, las animaciones declaratorias no deberían desencadenar animaciones. Esto se ha corregido en Quantum.
- En Gecko, las animaciones que utilizan ems como unidad no reciben las modificaciones realizadas en font-size en el elemento padre del elemento animado, aunque deberían. Quantum CSS corrige esto.
- Gecko además maneja la herencia de font-size de manera diferente a la de Quantum CSS, por lo que, para algunas configuraciones de idioma, los tamaños de letra heredados quedaban más pequeños de lo esperado. Quantum corrige esto.
- Gecko reutiliza el mismo mecanismo empleado al analizar una ficha de URL durante el análisis de las funciones domain() o url-prefix() para la regla @-moz-document. Quantum CSS no utiliza el mismo mecanismo y no considera las fichas como no válidas cuando contienen paréntesis o comillas.
- En Gecko, cuando se define un tipo de letra del sistema como el valor del tipo de letra de algún contexto del lienzo 2D (p. ej., menu), la obtención del valor del tipo de letra no devuelve el resultado correcto (no devuelve nada). Este problema quedó solucionado en Quantum.
- En Gecko, cuando se crea un subárbol desligado (p. ej., un creado a través de createElement() que todavía no se inserta en el DOM), el elemento raíz del subárbol se define

como un elemento de nivel bloque. En Quantum CSS, el elemento se define como alineado, tal como se define en la especificación.

- Gecko rechaza las expresiones calc —lo que invalida el valor— cuando se utilizan como el componente radial de una función radial-gradient. Quantum CSS lo resuelve.
- En Gecko, calc (1*2*3) no es analizado debidamente; Quantum CSS soluciona el problema.
- En Quantum CSS, calc() se admite en todos los sitios que la especificación indica. En Gecko, este no es el caso.
- Gecko contiene un defecto que provoca que los pseudoelementos ::before y ::after se generen aun si el valor de la propiedad content se define como normal o none. Este comportamiento es contrario a la especificación y se ha corregido en Quantum.
- Otro defecto de Gecko provoca que la propiedad background-position no pueda transicionarse entre dos valores que contengan números diferentes de valores <position>, por ejemplo, background-position: 10px 10px; y background-position: 20px 20px, 30px 30px. Quantum resuelve el error (developer.mozilla, 2017).

JavaScript

- El bucle no estándar for each...in (E4X) fue eliminado. En su lugar, debe utilizarse for...of.
- Se marcan como obsoletos los métodos Object.prototype.watch() y unwatch(). Se emitirá una alerta si se utilizan y serán eliminados en el futuro próximo.
- Se eliminaron los objetos Iterator y StopIteration y el protocolo de iteración heredado (developer.mozilla, 2017).

DOM

- Se ha implementado la propiedad Selection.type de la API Selection.
- Se admite Document.createEvent('FocusEvent') ahora.
- La propiedad files de la interfaz HTMLInputElement puede definirse ahora.
- El método HTMLDocument.getSelection() se ha trasladado a la interfaz Document para que esté disponible para los documentos XML.
- Se ha implementado el suceso messageerror, el cual puede ejecutar código tras su desencadenamiento por parte de manejadores de sucesos utilizados en destinos de mensajes.
- Cuando se emplea la iteración en valores Headers, estos quedarán organizados en orden lexicográfico y se combinarán los valores de los nombres de cabeceras duplicados (developer.mozilla, 2017).

Seguridad

- Los URL resource:// ya no filtran información.
- Los URI de datos ahora se manipulan como orígenes opacos únicos, en vez de heredar el origen del objeto de configuración responsable de la navegación (developer.mozilla, 2017).

1.3 Desarrollo de extensiones web para la personalización de navegadores.

1.3.1 Anatomía de una extensión web en la actualidad

Una extensión se compone de una colección de archivos, empaquetados para su distribución e instalación.

Toda extensión debe contener un archivo llamado "*manifest*.*json*". Este puede tener enlaces a otros tipos de archivos:

- Páginas en segundo plano (*Background page*): implementan la lógica de larga ejecución.
- *Scripts* de contenido (*Content scripts*): interactúan con las páginas web (no es lo mismo que un elemento <*script*> dentro de una página).
- Archivos de acción del navegador (*Browser action*): implementan los botones de la barra de herramientas.
- Archivos de acción de la página (*Action page*): Implementan botones en la barra de direcciones.
- Páginas de opciones (*Options page*): Definen una IU[12] para que los usuarios vean y cambien las configuraciones de la extensión.
- Recursos Web accesibles (*Web accesible resources*): Hace posible que el contenido empaquetado acceda a páginas web y scripts (developer.mozilla, 2017).

[12] IU: Interfaz de usuario, es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

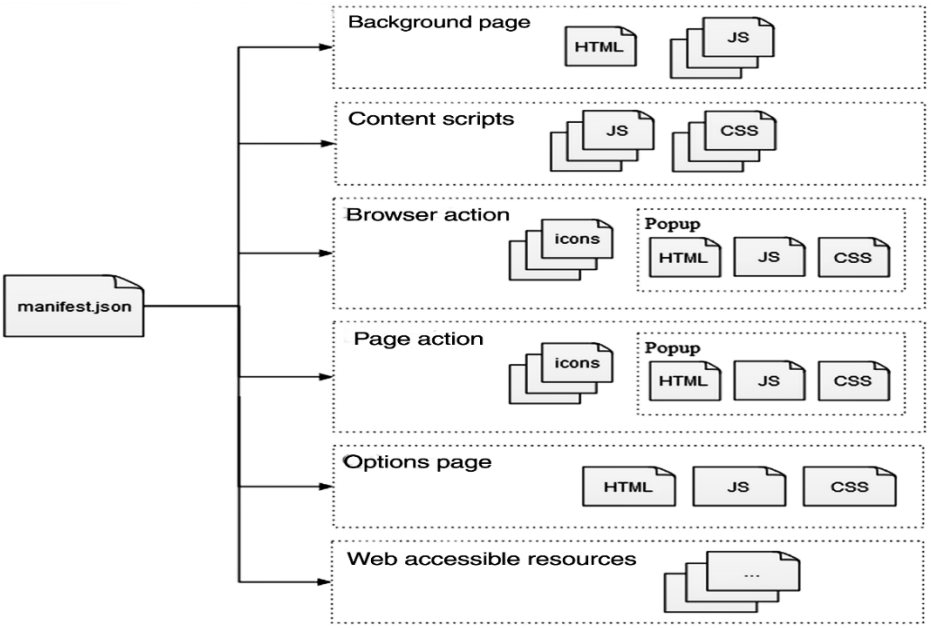


Ilustración 1: Anatomía de una extensión web (Mozilla Firefox)

Manifest.json

El *manifest.json* es un archivo en formato *JSON*, y es el único archivo que toda *WebExtension* debe contener necesariamente.

Usando el *manifest.json*, se puede especificar los metadatos básicos de la extensión como nombre y versión, también se puede especificar aspectos funcionales de la extensión, como *scripts* en segundo plano, y las acciones del navegador (developer.mozilla, 2017).

Scripts en segundo plano

Las extensiones a menudo necesitan mantener estados de larga duración, o ejecutar operaciones a largo plazo, independientemente del tiempo de vida de una página web en particular o las ventanas del navegador.

Los scripts en segundo plano se inician cuando la extensión es cargada y se mantienen en ejecución hasta que la extensión es deshabilitada o desinstalada. Se puede usar cualquier *API* de WebExtensions en el script, siempre y cuando se haya solicitado el permiso necesario. Los scripts en segundo plano o *background* se ejecutan en el contexto de páginas especiales llamadas páginas en segundo plano. Esto

le da un variable global *window*, junto con todas las *API* estándar del *DOM* [13]que proporciona (developer.mozilla, 2017).

Para incluir un script en segundo plano se usa la propiedad (llave) *background* en el *manifest.json*:

```
"background": {
  "scripts": ["background-script.js"]
}
```

Scripts de contenido

Los scripts de contenido se usan para acceder y manipular páginas web, son cargados dentro de las páginas web y ejecutados en el contexto particular de esa página.

Los scripts de contenido son scripts provistos por la extensión, los cuales se ejecutan en el contexto de la página web; estos difieren de los scripts que son cargados por la página misma, incluye aquellos que son proporcionados en los elementos <*script*> dentro de la página.

Los scripts de contenido pueden ver y manipular el *DOM* de las páginas, igual que los scripts cargados normalmente por la página.

A diferencia de los scripts normales, ellos pueden:

- Usar un pequeño subconjunto de las *API* de *WebExtension*
- Intercambiar mensajes con sus scripts en segundo plano, y por esta vía, tener acceso indirecto a todas las *API* de *WebExtension*.

Los scripts de contenido no pueden acceder directamente a los scripts normales de una página web, pero pueden intercambiar mensajes con ellos usando la *API* estándar *window.postMessage()* (developer.mozilla, 2018).

[13] DOM: *Document Object Model* o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos *HTML*, *XHTML* y XML.

Para incluir un script de contenido se usa la propiedad (llave) *content_scripts* en el *manifest*.*json*:

```
"content_scripts":
 {
   "js": ["content-script.js"]
 }
```

Recursos web accesibles

Los recursos web accesibles son recursos como imágenes, *HTML*, *CSS* y JavaScript que pueden ser incluidos en la extensión y pueden hacerse accesible a los scripts en segundo plano y los scripts de las páginas. Los recursos que son accesibles desde la web pueden ser referenciados desde scripts de páginas web y scripts de contenido mediante un esquema especial de URI[14].

Para incluir un recurso web accesible se usa la propiedad (llave) *web_accesible_resources* en el *manifest*.*json* (developer.mozilla, 2017).

```
"web_accessible_resources": ["images/my-image.png"]
```

1.3.2 WebExtensions API

Para el desarrollo de extensiones actualmente Mozilla cuenta con un total de cuarenta y tres *API* disponibles, de las cuales serán listadas a continuación las más significativas a tener en cuenta para el desarrollo de la propuesta de solución con una breve explicación en cada caso (developer.mozilla, 2018).

1. Menus: Agrega elementos al sistema de menú del navegador,
2. Tabs: Interactúa con el sistema de pestañas del navegador.
3. BrowsingSettings: Permite que una extensión modifique ciertas configuraciones globales del navegador. Cada propiedad de esta API es un objeto BrowserSettings que brinda la posibilidad de modificar una configuración en particular.

[14] URI: Un identificador de recursos uniforme o URI —del inglés *uniform resource identifier*— es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

4. Storage: permite almacenar y recuperar datos, y escuchar cambios a los elementos almacenados.
5. BrowsingData: Permite borrar los datos que se acumulan mientras el usuario está navegando.
6. Notifications: Muestra notificaciones al usuario, utilizando el mecanismo de notificación del sistema operativo subyacente. Debido a que esta API utiliza el mecanismo de notificación del sistema operativo, los detalles de cómo aparecen y se comportan las notificaciones pueden diferir según el sistema operativo y la configuración del usuario.
7. Commands: Escucha al usuario que ejecuta los comandos que ha registrado utilizando la llave del *manifest.json* commands.

Para utilizar cada una de las API de WebExtensions es necesario declarar el permiso en el archivo manifest.json, que generalmente lleva el mismo nombre de la API mediante la llave “permissions” (developer.mozilla, 2018).

Ejemplo:

"permissions": [**"menus"**, **"tabs"**, **"browserSettings"**, **"storage"**, **"browsingData"**, **"notifications"**,]

1.3.3 Desarrollo de extensiones web antes de *WebExtensions*

Anteriormente existían diferentes maneras de crear una extensión para *Firefox*, y las más significativas están cubiertas en las secciones que siguen. Antes de *WebExtensions*, se podía desarrollar complementos de *Firefox* utilizando uno de los tres diferentes sistemas: *XUL*/*XPCOM overlays*, extensiones *bootstrapped*, o el complemento *SDK*. Todas estas arquitecturas de desarrollo quedaron obsoletas por la nueva arquitectura propuesta por *Mozilla*: *WebExtensions,* a partir de noviembre del 2017 (developer.mozila, 2017).

Extensiones con el *Add-ons SDK*

El *Add-ons SDK* es un grupo de API simples que pueden utilizarse para crear rápidamente buenas extensiones para *Firefox*. El *SDK* abstrae la mayoría de la infraestructura *XUL* / *XPCOM* en *Firefox* y brinda un ambiente *HTML* y *JS* más familiar para trabajar.

El enfoque para crear una extensión web mediante el *SDK* consiste en descargar e instalar el mismo, programar localmente, y luego empaquetar la extensión (archivo *XPI*) utilizando la herramienta *JPM*

incluida con el *SDK* (nota: *JPM* reemplaza a la anterior herramienta llamada *CFX*) (developer.mozilla, 2017).

Extensiones *Bootstrapped*

Al igual que las extensiones hechas con el *SDK*, las extensiones *bootstrapped* no necesitan de un reinicio para instalarlas, pero tampoco tienen acceso directo a los *API* del *SDK* o aislamiento de seguridad. Básicamente se hace todo manualmente, como monitorear las ventanas para agregar o quitar la interfaz de usuario. Sin embargo, existen varias herramientas disponibles mediante módulos *JavaScript*, como *CustomizableUI.jsm* para las barras de herramientas y *Services.jsm* para componentes de *Firefox* usados frecuentemente.

Comparado con la antigua forma de crear extensiones, las únicas pérdidas destacadas son las *overlays*. Las extensiones *Bootstraped* tienen, en cambio, el archivo bootstrap.js, que sirve como punto de partida para el código. Aparte de eso, todo lo demás debería ser familiar: *install.rdf*, *manifest* y archivos de código, todos comprimidos en un archivo ZIP con la extensión *.xpi* (developer.mozilla, 2017).

1.4 Análisis de extensiones de Firefox que permiten crear personalizaciones

Se analizan las siguientes extensiones web que permiten crear personalizaciones con el fin de tener en cuenta o no sus funcionalidades para el desarrollo de la propuesta de solución:

Características de *R-Kiosk*: *Real Kiosk* es una extensión de *Firefox* que se predetermina a pantalla completa, desactiva todos los menús, barras de herramientas, comandos de teclado y menús de botón derecho.
Se puede habilitar la barra de herramientas de navegación agregando lo siguiente a user.js: *user_pref("rkiosk.navbar",true);*
Es posible eliminar el cuadro de diálogo de impresión agregando las siguientes líneas a *user.js:*
user_pref("print.always_print_silent",true);
user_pref("print.show_print_progress",false);

El usuario aún puede cerrar *Firefox* con Alt-F4 y obtener acceso a la computadora. La extensión *R-kiosk* solo puede eliminarse en el modo seguro de *Firefox*.

Esta extensión web no permite al usuario la posibilidad de personalizar, ya sea habilitando o deshabilitando características específicas del *Firefox*. Una vez instalado esta extensión todo el navegador se convierte en un kiosko, un navegador en blanco, sin la posibilidad de cambiar nada, la única opción de regresar a la configuración anterior del navegador es desactivándolo o eliminándolo de

Firefox. Esta extensión web presenta características y funcionalidades similares a las descritas en *nova-kiosko* por lo que puede ser útil para la implementación de la misma. Esta extensión resulta obsoleta para navegadores modernos debido al cambio de arquitectura de desarrollo de extensiones de Mozilla.

Características de *Tab- Mix- Plus*: *Tab Mix Plus* mejora las capacidades de navegación de pestañas de *Firefox*. Incluye funciones tales como la duplicación de pestañas, el control del enfoque de pestañas, las opciones de hacer clic en la pestaña, deshacer pestañas y ventanas cerradas, y mucho más. También incluye un administrador de sesión con recuperación de fallos que puede guardar y restaurar combinaciones de pestañas y ventanas abiertas.

Tab Mix Plus presenta una ventana de opciones que incluye todas sus preferencias y agrega una interfaz de usuario a las preferencias ocultas importantes de *Firefox*. La ventana de opciones está disponible desde el Administrador de complementos o desde el menú Herramientas (a excepción que se desactive esta opción).

Todas las preferencias de *Tab Mix Plus* están almacenadas en el archivo prefs.js en el directorio de perfil de usuario. A estas preferencias también se pueden acceder y cambiar al ingresar "*about*: *config*" en la barra de URL y "*extensions*. *tabmix*" en el cuadro de filtro para restringir los resultados de la búsqueda.

Además de presentar características muy similares a las de *nova-kiosko*, con una interfaz para las opciones de configuración, prescinde de otras funcionalidades importantes para el desarrollo de la extensión de *Firefox*. Esta extensión presenta un panel de opciones que permite al usuario personalizar varios componentes del navegador, muy importante y a tener en cuenta a la hora de desarrollar la extensión web pero carece de funcionalidades que se consideran imprescindibles en la propuesta de solución.

Características de *Max- Tabs*: Se caracteriza por:

- Establecer el número máximo de pestañas abiertas en una ventana.
- Controlar y restringir la apertura de demasiadas pestañas.
- Evitar que las pestañas se sobrecarguen y desborde la barra de pestañas.
- De forma predeterminada, está configurado para un máximo de 10 pestañas por ventana, configurable a través del botón de opciones.

Esta extensión, aunque sea simple será de ayuda para desarrollar funcionalidades descritas en *nova-kiosko* como por ejemplo toda la gestión de las pestañas de la página. También carece de todo un cúmulo

de características y funcionalidades que se tiene en cuenta en la extensión de *Firefox* para su personalización.

Características de *Modern Kiosk:*

Extensión tipo kiosko para buscadores modernos (Firefox >=57). Este add-on está en desarrollo activo y no está terminado como un kiosko completo. Esta extensión implementa la funcionalidad del kiosko para ser utilizado por el público, para evitar el acceso al navegador en ejecución. Permite definir una dirección URL que será la única con la que se tendrá acceso a navegar y a través del menú contextual se puede entrar en modo kiosko (*FullScreen*).

Esta extensión carece de otras funcionalidades importantes para el desarrollo de la propuesta de solución, aunque se tendrá en cuenta sus características como una extensión en modo kiosko para navegadores modernos.

1.5 Tecnologías de implementación

En este epígrafe se aborda todo lo referente a herramientas y lenguajes de modelado, los lenguajes de programación, entorno de desarrollo integrado (IDE). Esto permitirá la implementación eficiente de la extensión de Firefox.

1.5.1 Metodología de desarrollo de software

La metodología de desarrollo de software empleada para la obtención de la propuesta de solución es Variación de AUP (*Agile Unified Process*, Proceso Unificado Ágil) para la UCI. Se elige esta metodología debido a que describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software, se adapta al ciclo de vida de la actividad productiva que se realiza en la universidad y permite aumentar la calidad del software.

La metodología AUP-UCI posee tres fases:

1. **Inicio**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
2. **Ejecución (Elaboración, Construcción, Transición)**: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto

considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

3. **Cierre**: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

En la fase de Ejecución es donde se centra el desarrollo de la propuesta de solución ya que en la misma se desarrollan las disciplinas: Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de aceptación.

En el desarrollo de la investigación se utiliza el Escenario No 4 ya que: se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos para poder implementarlos, probarlos y validarlos además de que el proyecto no sea muy extenso.

1.5.2 Lenguaje de modelado: UML

El Lenguaje Unificado de Modelado (*UML*, *Unified Modeling Language*) es un lenguaje de modelado visual que permite especificar, construir y documentar artefactos de un software. Se puede usar en las diferentes etapas del ciclo de vida de un proyecto. Incluye conceptos semánticos, notación y principios generales de un sistema. Contiene además construcciones organizativas para agrupar los modelos en paquetes, lo que permite dividir grandes sistemas en piezas de trabajo más simples (Hernández, 2015).

1.5.3 Herramienta CASE para la modelación del sistema

Visual Paradigm es una herramienta *UML* (*Unified Modeling Language*) profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño Orientados a Objetos, construcción, pruebas y despliegue. El *software* de modelado *UML* ayuda a una más rápida construcción de aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación (Quintana, y otros, 2011).Se selecciona esta herramienta debido a que todos los diagramas del proceso de desarrollo de la extensión de Firefox Nova Kiosko en su versión actual están realizados en la misma.

1.5.4 Lenguajes de programación

Los lenguajes de programación permiten crear programas mediante un grupo de instrucciones, operadores y reglas de sintaxis, que son puestos en manos del programador para que este pueda comunicarse con los dispositivos de *hardware* y con el *software* existente en el ordenador.

JavaScript es un lenguaje de programación que se define como orientado a objetos. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas (Huelga, 2010) .

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, *Java* y *JavaScript* no están relacionados y tienen semánticas y propósitos diferentes (Huelga, 2010).

Ventajas:

- No requiere tiempo de compilación.
- Los scripts pueden desarrollarse en un período de tiempo relativamente corto.
- Posee características de interfaz, que son gestionados por el navegador y por el código *HTML*.
- Los programas *JavaScript* tienden a ser pequeños y compactos, no requieren mucha memoria ni tiempo adicional de transmisión.
- Es independiente de la plataforma de hardware o sistema operativo, siempre y cuando exista un navegador con soporte *JavaScript*.

1.5.5 Lenguajes de etiquetado

Un lenguaje de etiquetado, de marcado o lenguaje de marcas se utiliza en documentos que incorporan junto al texto etiquetas o marcas para de alguna forma codificar dicho documento. El más utilizado en el mundo es *HTML*, el cual es el fundamento de la Red de Redes.

Se trata de un conjunto de símbolos y reglas que se usan para especificar el formato y delimitar el contenido de un documento dado, que después será interpretado por un programa específico. También se le denomina lenguaje de marcación, de etiquetas (*tags*) o de anotaciones (Rodriguez, 2010).

1.5.5.1 Lenguaje de Marcado de Hipertexto

El Lenguaje de Marcado de Hipertexto (del inglés **H**yper**T**ext **M**arkup **L**anguage) o *HTML* nació públicamente en un documento llamado *HTML Tags* (Etiquetas *HTML*), publicado por primera vez en Internet por *Tim Berners-Lee* en 1991. De esta manera, se puede afirmar que han existido distintas versiones de *HTML* a lo largo de la historia de internet. Se trabaja con el *HTML* estándar actual, que es el utilizado por los navegadores y páginas web de hoy en día.

HTML es el lenguaje que se emplea para el desarrollo de páginas de internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. *HTML* dispone de etiquetas

para imágenes, hipervínculos que permiten dirigirse a otras páginas, saltos de línea, listas, tablas, etc. Podríamos decir que *HTML* sirve para crear páginas web, darles estructura y contenido (Gutiérrez, 2009).Se utiliza en la creación de la interfaz de usuario para permitir las personalizaciones en el navegador mediante la extensión web.

1.5.5.2 Lenguaje de marcado o de Diseño gráfico *CSS*

Las Hojas de Estilo en Cascada (del inglés ***C****ascading* ***S****tyle* ***S****heets*) o *CSS* es el lenguaje utilizado para describir la presentación de documentos *HTML* o *XML*, esto incluye varios lenguajes basados en *XML* como son *XHTML* o *SVG*. *CSS* describe como debe ser renderizado el elemento estructurado en pantalla o en otros medios (developer.mozilla, 2018). Se utiliza para dar un estilo agradable a la interfaz de usuario desarrollada mediante *HTML.*

1.5.6 Entorno de Desarrollo Integrado (*IDE*).

Un entorno de desarrollo integrado (*IDE*) es un programa compuesto por un conjunto de herramientas para que el programador las utilice. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. El entorno de desarrollo es imprescindible en la producción de un software.

El IDE utilizado es el *WebStorm*, una herramienta de desarrollo que ayuda a la implementación de una aplicación en *JavaScript.* Con este entorno de desarrollo se implementará todo el código *JavaScript, HTML y CSS* necesario para crear la extensión de Firefox. El autor decide usar este *IDE* debido a la alta calidad y gama que presenta la *suite* de *JetBrains*, las opciones de depuración son precisas, es un IDE enfocado a JavaScript principalmente lo que lo hace idóneo para el desarrollo de la extensión web además que el autor se siente cómodo y familiarizado con este entorno de desarrollo (González, 2016).

1.6 Conclusiones del capítulo

A lo largo de este capítulo han sido expuestos los principales puntos de interés relacionados con los conceptos fundamentales sobre los cuales gira la presente investigación y los antecedentes que dieron origen a la misma, lo que ha permitido una mejor comprensión del objeto de estudio. El análisis de extensiones con similar propósito permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución. Se detallan las tecnologías utilizadas, las cuales llevaron una investigación previa para seleccionar las más indicadas, lo que posibilitó obtener la base tecnológica para el desarrollo de la extensión web. Como resultado de esta investigación previa se decidió utilizar: como metodología de desarrollo de *software:* AUP variación UCI, como lenguajes de programación y lenguaje de etiquetado: *JavaScript* y *HTML* respectivamente, como entorno de desarrollo: *WebStorm* y como herramienta *Case*: *Visual Paradigm*.

Capítulo 2. Análisis y diseño de la extensión web para la creación de personalizaciones en Mozilla Firefox.

En el presente capítulo se establecen las principales características y funcionalidades con que contará la solución, ya sean requisitos funcionales o no funcionales. Se generan los diagramas correspondientes a las etapas de planificación y diseño que propone la metodología de desarrollo AUP-UCI. Además, se describen los patrones de diseño y arquitectura que se utilizarán, así como el modelo de dominio del sistema y el diagrama de clases del diseño.

2.1 Descripción de la propuesta de solución

Para la propuesta de solución se tiene en cuenta los resultados definidos hasta este punto de la investigación y para resolver la problemática planteada se propone el desarrollo de la extensión Nova-Kiosko, que permitirá personalizar el navegador web *Mozilla Firefox*, haciendo uso de las diferentes funcionalidades que presenta. La solución propuesta está regida por la arquitectura de desarrollo basada en componentes que permitirá al usuario modificar dicho navegador tanto funcionalmente, así como también su interfaz.

La extensión despliega una serie de funcionalidades mediante las opciones de configuración que le dan la posibilidad al usuario de habilitar o deshabilitar características específicas del navegador web Mozilla Firefox como, por ejemplo: limitar cantidad de pestañas abiertas en una ventana, el menú contextual, las combinaciones de teclas y opciones de guardado de información en general. El usuario mediante una interfaz sencilla y un navegador personalizable por la extensión web se convierte en su principal administrador.

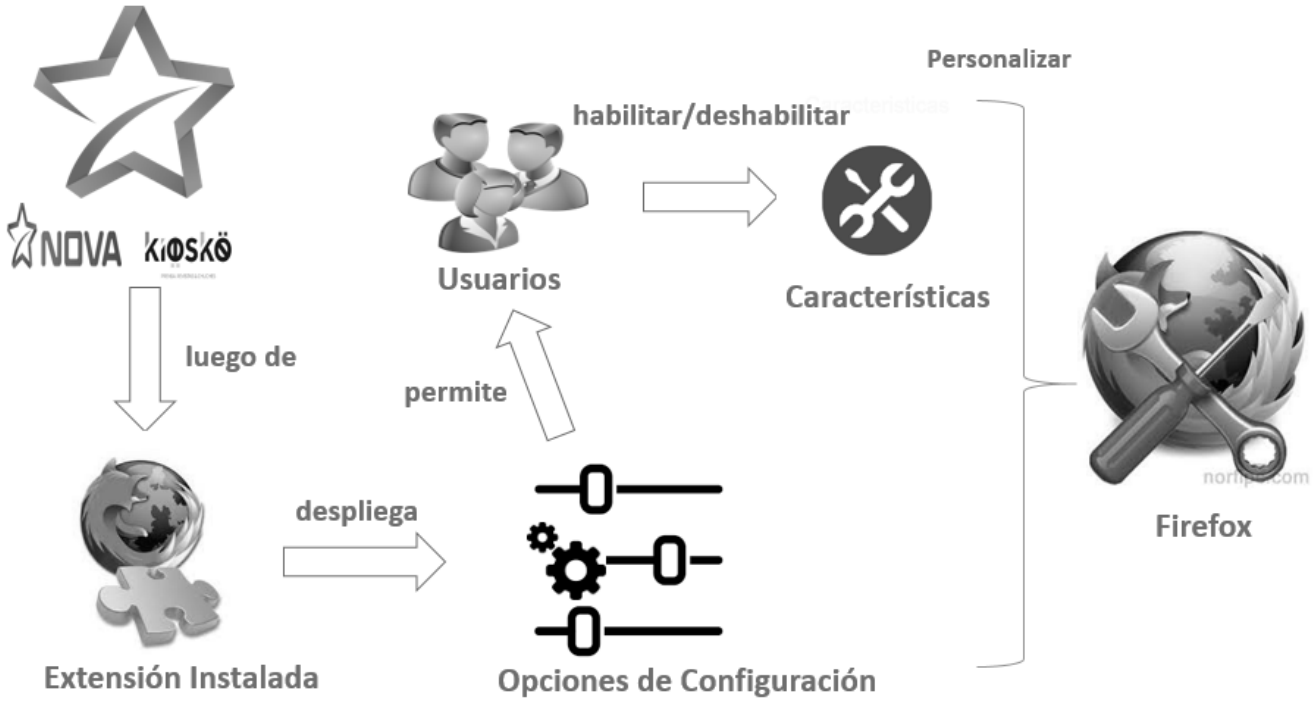


Ilustración 2: Propuesta de solución (Elaboración propia).

2.2 Requisitos

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento, estos aportan valor y utilidad al cliente y a los usuarios finales. Se clasifican en requisitos funcionales(RF) y requisitos no funcionales(RNF) (DECSAI, 2010).

Fuentes de obtención de requisitos

- Especialistas de CESOL

Técnicas para identificar requisitos

- Análisis documental
- Tormenta de ideas

2.2.1 Requisitos funcionales

Un requisito funcional define una función del software o sus componentes; pueden ser cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir (Sommerville, 2006).

Tabla 1 Requisitos Funcionales (Elaboración propia)

ID	Nombre del requisito	Descripción del requisito	Prioridad	Complejidad
RF1	Deshabilitar/ habilitar botones del ratón	El sistema debe permitir al usuario deshabilitar/habilitar todas las acciones que se realizan con todos los botones del ratón que no sean el izquierdo (dejando que la rueda haga scroll).	Media	Media
RF2	Deshabilitar opciones de desarrollador	El sistema debe permitir deshabilitar las opciones de desarrollador.	Media	Alta
RF3	Personalizar menú contextual	El sistema debe permitir personalizar a los usuarios el menú contextual.	Media	Baja
RF4	Deshabilitar/ habilitar combinaciones de teclas	El sistema debe permitir al usuario deshabilitar/habilitar todas las combinaciones de teclas excepto F5, Tab, Enter, y las flechas de direcciones.	Alta	Alta
RF5	Deshabilitar/ habilitar opciones de guardado	El sistema debe permitir al usuario deshabilitar/habilitar todas las opciones de guardado de información de una página dígase: historial de formularios, descargas, navegación, cookies, sesiones.	Media	Alta
RF6	Personalizar apertura de vínculos	El sistema debe abrir todos los vínculos por defecto en una misma pestaña, evitar que se solicite abrir una nueva pestaña, una nueva ventana o una ventana emergente.	Alta	Alta

RF7	Deshabilitar/ habilitar apertura de documentos	El sistema debe permitir al usuario deshabilitar/habilitar que al abrir documentos descargados se abran en una nueva pestaña	Media	Media
RF8	Deshabilitar campo *URL*	El sistema debe permitir deshabilitar el campo de direcciones URL	Media	Alta
RF9	Deshabilitar controles de navegación	El sistema debe permitir deshabilitar los controles de navegación (botones de Atrás y Adelante)	Media	Baja
RF10	Deshabilitar la barra de menú	El sistema debe permitir deshabilitar la barra de menú	Media	Media
RF11	Deshabilitar la barra de marcadores	El sistema debe permitir deshabilitar la barra de marcadores	Media	Media
RF12	Deshabilitar/habilitar gestión de las pestañas	El sistema debe permitir al usuario deshabilitar/ habilitar la gestión de las pestañas (fijar pestañas, fijar las URL que deben abrir estos).	Alta	Alta
RF13	Gestionar opciones de guardado	El sistema debe permitir al usuario gestionar el historial, decidir si se guarda el historial o hay que limpiarlo con cada reinicio del navegador.	Alta	Alta

2.2.2 Requisitos no funcionales

Un requisito no funcional especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Sirven de apoyo a los requisitos funcionales (Barbosa Guerrero, 2006).

Tabla 2: Requisitos no funcionales (Elaboración propia)

ID	Categoría	Descripción
RNF1	Usabilidad	La extensión debe permitir al usuario el acceso fácil a todas las opciones de configuración permisibles
RNF2	Usabilidad	La extensión debe ser capaz de validar las opciones seleccionadas y mostrar mensajes de error en caso que estas lo permitan
RNF3	Soporte	La herramienta deberá permitir posteriores actualizaciones y se debe garantizar el mantenimiento de la misma
RNF4	Apariencia	La extensión debe ser implementada con HTML 5.0 y CSS 3.0
RNF5	Software	La extensión como se establece por el grupo de desarrollo de Mozilla Firefox debe ser instalable en Firefox >= 57.0
RNF6	Software	La extensión debe ser implementada usando JavaScript 1.8
RNF7	Software	La extensión como se establece por el grupo de desarrollo de Mozilla Firefox debe ser un fichero nova-kiosko.xpi

2.2.3 Historias de usuario (HU)

En la metodología AUP-UCI una de las formas que se utiliza para describir los requisitos de *software* son las Historias de Usuario (HU). Se describe brevemente y en un lenguaje natural, las características que el sistema debe poseer. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en un corto período de tiempo ((XP), 2017).

Tabla 3: Historia de usuario No 4

Historia de Usuario	
Número: HU_4	**Nombre del requisito:** Deshabilitar/ habilitar combinaciones de teclas
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1 semana

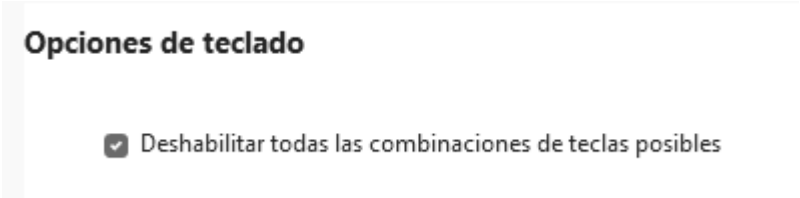
Descripción: La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar” debajo de la etiqueta “Opciones de teclado”, luego de terminar la acción que indica que las combinaciones de teclas están deshabilitadas o habilitadas presiona el botón de “Guardar Preferencias” dependiendo de la opción marcada y no permitirá que el usuario realice ninguna combinación de teclas exceptuando las teclas F5, Tab, Enter, flechas de direcciones y las combinaciones de teclas que el navegador no permite remover como (Alt+ F4, Ctrl+W, Ctrl+ T, Ctrl+Shift+W) o simplemente el usuario puede ejecutar cualquier combinación de teclas si está habilitada la opción.
Prototipo de la interfaz gráfica del usuario


Tabla 4: Historia de usuario No 6

Historia de Usuario	
Número: HU_6	**Nombre del requisito:** Personalizar apertura de vínculos
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3 días
Descripción: La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar”, debajo de la etiqueta ”Opciones de Pestañas”, la cual contiene “Abrir vínculos en la misma pestaña”, luego de terminar la acción el sistema abrirá todos los vínculos por defecto en una misma pestaña y evitará que se solicite abrir una nueva pestaña.	
Prototipo de la interfaz gráfica del usuario	

Tabla 5: Historia de usuario No 14

Historia de Usuario	
Número: HU_12	**Nombre del requisito:** Deshabilitar/ habilitar gestión de las pestañas
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1 día
Descripción: La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar” debajo de la etiqueta “Opciones de “Pestañas” en la parte superior de las opciones de configuración de la extensión y pueda limitar el número de pestañas abiertas en el navegador, fijar las URL que abrirán por defecto al iniciar la extensión y así gestionar las pestañas mediante una barra con un rango determinado y un cuadro de texto respectivamente.	

Prototipo de la interfaz gráfica del usuario

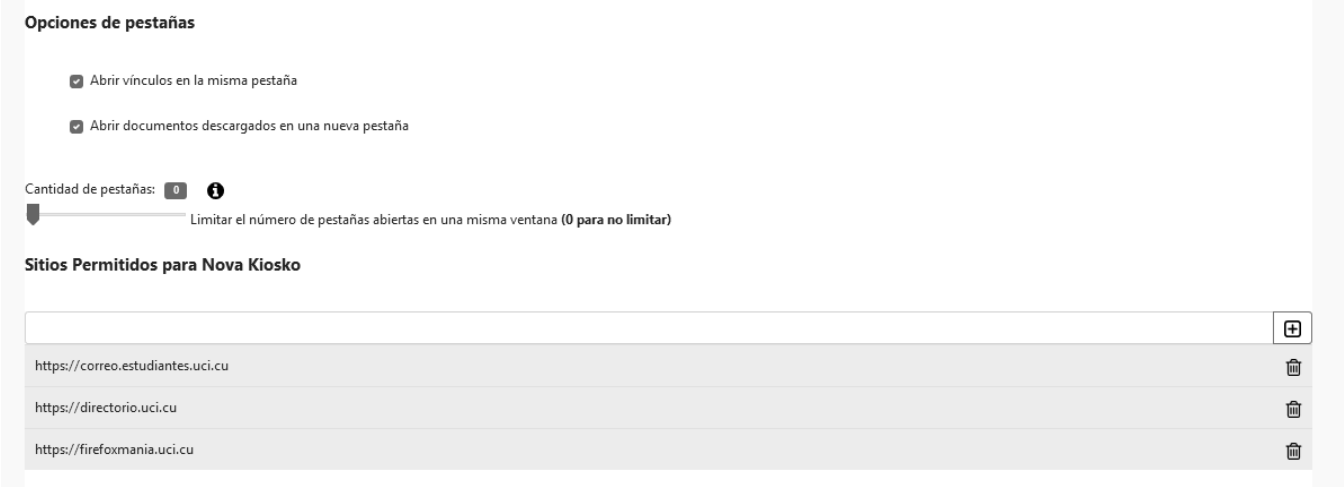
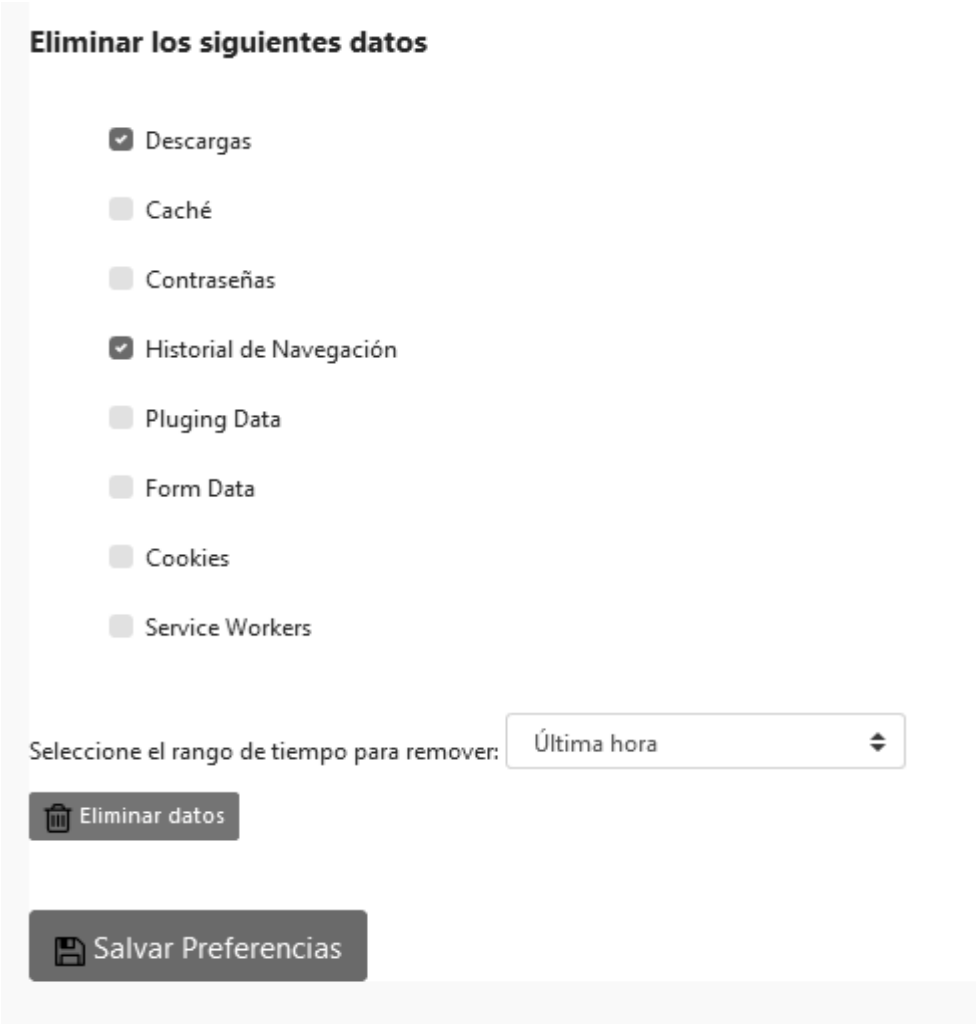


Tabla 6 Historia de usuario No 15

Historia de Usuario	
Número: HU_13	**Nombre del requisito:** Gestionar opciones de guardado
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 semana
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1 semana
Descripción La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar” debajo de la etiqueta “Eliminar los siguientes datos”, esta contiene las siguientes opciones “Historial de Navegación”, ”Descargas”, ”Cookies”, “Cache”, “Contraseñas”, “Pluging Data”, “Form Data” y “Service Workers”, puede marcar todas las opciones en cada checkbox, estas opciones implican que el navegador elimine o no la información que se describe en las opciones.	

Prototipo de la interfaz gráfica del usuario



Las restantes historias de usuario podrán ser consultadas en el Anexo 1.

2.3 Análisis y diseño

2.3.1 Arquitectura

En esencia, un componente es una pieza de código preelaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar (Terreros, 2017).

Para el desarrollo de la solución se utilizó una arquitectura basada en componentes, la misma consiste en una rama de la Ingeniería de *software* en la cual se trata con énfasis la descomposición del *software* en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en

piezas más grandes de *software*. Este proceso de construcción de una pieza de *software* con componentes ya existentes, da origen al principio de reutilización del *software*, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro (Huelga, 2010).

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como **Desarrollo de Software Basado en Componentes**. El uso de este paradigma posee algunas ventajas:

1. **Reutilización del software.** Permite a alcanzar un mayor nivel de reutilización de software.
2. **Simplifica las pruebas.** Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
3. **Simplifica el mantenimiento del sistema.** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
4. **Mayor calidad.** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

2.3.2 Diagrama de paquetes

Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema (Group, 2001).

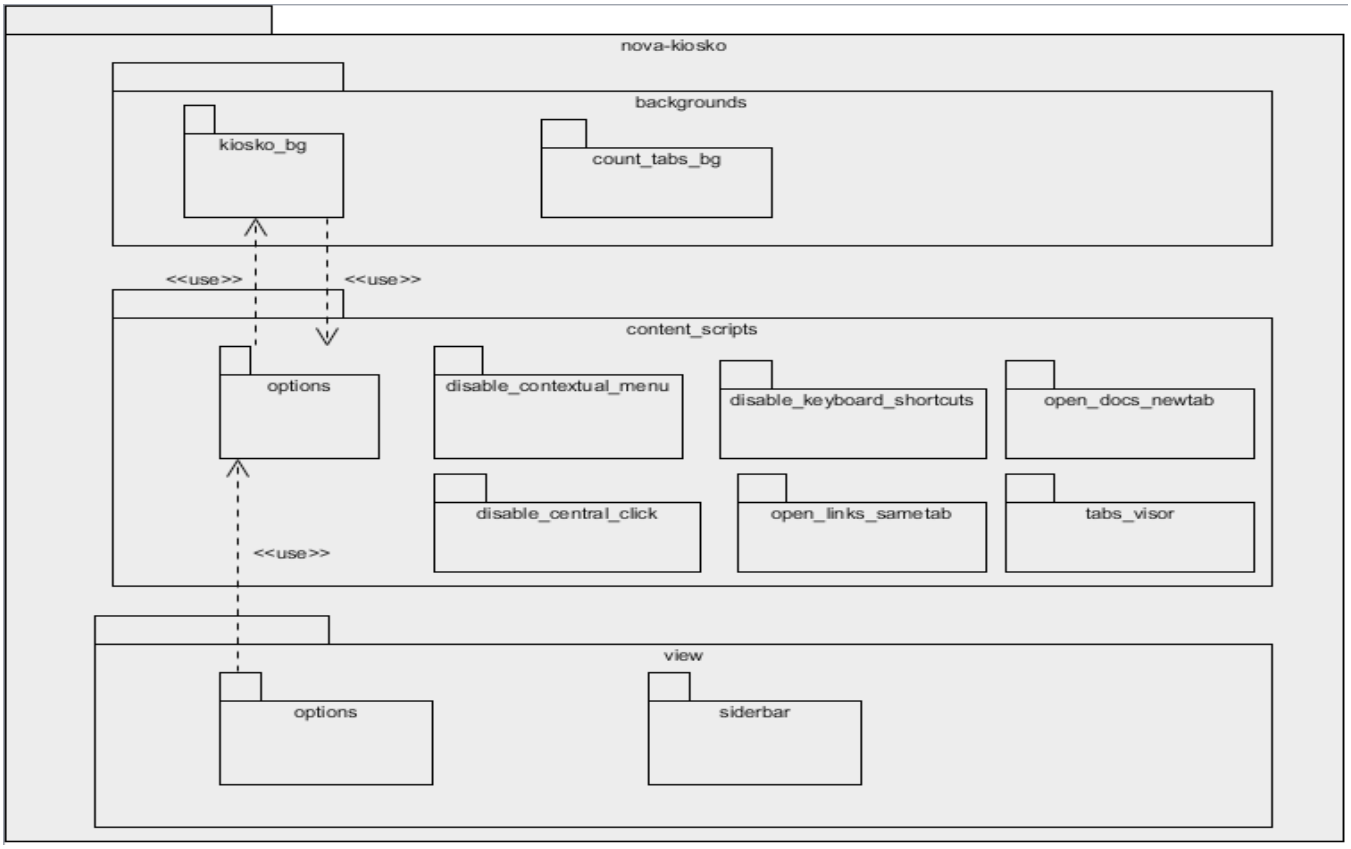


Ilustración 3: Diagrama de paquetes (Elaboración propia).

2.3.3 Patrones de diseño

Un patrón de diseño es una buena práctica documentada de la solución de un problema que ha sido aplicado satisfactoriamente en múltiples entornos. Es una solución recurrente a un problema común observado o descubierto durante el estudio o construcción de numerosas aplicaciones. Su principal objetivo es incrementar la calidad del *software* en términos de reusabilidad, mantenimiento y extensibilidad (Craig LARMAN, 2003).

2.3.4 Patrón Fachada (*Facade*)

La fachada es un patrón simple; proporciona solo una interfaz alternativa a un objeto. Es una buena práctica de diseño mantener sus métodos cortos y no hacer que trabajen demasiado. Siguiendo esta práctica, se terminará con una mayor cantidad de métodos que si se tiene muchos métodos con muchos parámetros. A veces, dos o más métodos comúnmente se pueden llamar juntos. En tales casos, tiene sentido crear otro método que envuelva las llamadas al método repetitivo. (Stefanov, 2010)

Su utilización se evidencia en el script *kiosko_bg*.js en la función *checkTabsSettings*();

Ejemplo de su utilización:

```
/**
 * Check all tabs settings of NovaKiosko
 * @param {*} settings
 * @param {*} tabs
 */
function checkTabsSettings(settings, tabs) {
  // Open links same tab
  if (settings.kiosko_open_links_same_tab) {
    for (let tab of tabs) {
      executeScript(settings, tab, 'open_links_same_tab')
    }
  }
  // Open docs new tab
  if (settings.kiosko_open_docs_new_tab) {
    for (let tab of tabs) {
      executeScript(settings, tab, 'open_docs_newtab')
    }
  }
  // Limit number of open tabs
  if (settings.kiosko_max_open_tabs !== "0" && tabs.length !== Number(settings.kiosko_max_open_tabs) ) {
    limitOpenTabs(settings)
  }
  // Only allow browsing over kiosko url and internals pages
  if (settings.kiosko_url) {
    browser.tabs.query({}).then( openTabs => {
      let whiteList = settings.kiosko_internals_url
      whiteList.push(settings.kiosko_url.split("/")[2])
      for (let pestana of openTabs) {
        if ( !isAllowed(whiteList, pestana.url)) {
          closeTab(pestana)
        }
      }
    }, onError)
  }
}
```

Ilustración 4: Evidencia del patrón Fachada (Elaboración propia).

2.3.5 Patrón Observador (*Observer*)

El patrón de observador se usa ampliamente en la programación de JavaScript del lado del cliente. Todos los eventos del navegador (*mousedown*, *keydown*, etc.) son ejemplos del patrón (Stefanov, 2010). Su utilización se evidencia en el script disable_keyboard_shortcuts.js.

Ejemplo de su utilización:

```
window.onkeydown = function(e) {
  // teclas del F1...F12
  if (e.keyCode ===112 || e.keyCode === 113 || e.keyCode === 114 || e.keyCode === 115 ||
    e.keyCode === 117 || e.keyCode === 118 || e.keyCode === 119 || e.keyCode === 120 ||
    e.keyCode === 121 || e.keyCode === 122 ||e.keyCode ===123) {
    e.preventDefault();
  }
  // combinacion de teclas de shift + [special keys]
  if(e.shiftKey && special_keys[e.keyCode] || e.ctrlKey && special_keys[e.keyCode]){
    e.preventDefault();
  }
  // combinaciones letras + Alt
  if( e.altKey && keys[e.keyCode]){
    e.preventDefault();
  }
  // combinaciones letras + Ctrl y signos de mas y menos+-
  if (e.ctrlKey && (keys[e.keyCode] || e.keyCode ===44 || e.keyCode === 45)) {
    e.preventDefault();
    //combinaciones de teclas para Ctrl+At+letras
    if(e.ctrlKey && e.altKey && keys[e.keyCode]){
      e.preventDefault();
    }
    //combinaciones de teclas para Ctrl+At+letras
  } else if (e.ctrlKey && e.shiftKey && keys[e.keyCode]) {
    e.preventDefault();
  }

}
```

Ilustración 5: Evidencia del patrón Observador (Elaboración propia).

2.3.6 Patrón de Devolución de llamada (*Callback*)

Las funciones son objetos, lo que significa que se pueden pasar como argumentos a otras funciones. La devolución de llamada puede ser una función existente, o puede ser una función anónima, que se crea cuando llama a la función principal (Stefanov, 2010).

Ejemplo de su utilización:

```
function handleInstalled(details) {
  if (details.reason === "install") {
    // set defalut options
    browser.storage.local.set(kiosko_default_options).then( () => {
      checkSettings(kiosko_default_options)
    }, onError)
  } else if (details.reason === "update") {
    // get saved settings and check all
    browser.storage.local.get().then( (settings) => {
      for (let [key, value] of Object.entries(kiosko_default_options) ) {
        if ( settings[key] === undefined ) {
          settings[key] = value
          browser.storage.local.set(settings)
        } else {
          console.log("Kiosko option: " + key + " : " + settings[key])
        }
      }
    }, onError)
  } else {
    console.log("Nova Kiosko will be: " + details.reason)
  }
  openAllowedURLs(kiosko_default_options.white_urls)
}
```

Ilustración 6: Evidencia del patrón Devolución de llamada (Elaboración propia).

2.4 Conclusiones del capítulo

En este capítulo se han abordado los principales elementos del análisis y diseño de la extensión de Firefox para su personalización, en el cual se puede arribar a las siguientes conclusiones:

- La arquitectura de software Desarrollo Basado en Componentes, así como los patrones de diseño definidos, permitieron una mayor organización en la implementación, fomentándose buenas prácticas en el desarrollo de la propuesta de solución.
- La propuesta de solución permitió conocer las principales características y funcionalidades que debe cumplir la aplicación para lograr la aceptación por el cliente, en este proceso se identificaron 13 requisitos funcionales y 8 no funcionales.
- Las descripciones de las historias de usuario y la elaboración del diagrama de paquetes posibilitaron una mejor comprensión del funcionamiento de la propuesta de solución.

Capítulo 3. Implementación y pruebas de la extensión web para la creación de personalizaciones en Mozilla Firefox

El objetivo fundamental de este capítulo es explicar el diseño de la solución; así como los principales componentes y las relaciones que existen entre ellos. A través de diferentes diagramas se ofrece una panorámica del funcionamiento del componente dentro de la plataforma y cómo ocurren los principales flujos de procesos.

3.1 Diagrama de componentes

Un componente representa una parte de un sistema modular, desplegable y reemplazable, que encapsula la implementación y expone un conjunto de interfaces (Group, 2001). Podría ser, por ejemplo, código fuente, binario o ejecutable.

En el siguiente diagrama se muestra la relación que posee el componente *kiosko_bg.js* (*background*) con el resto de los componentes en los content_scripts y el componente *manifest.json*.

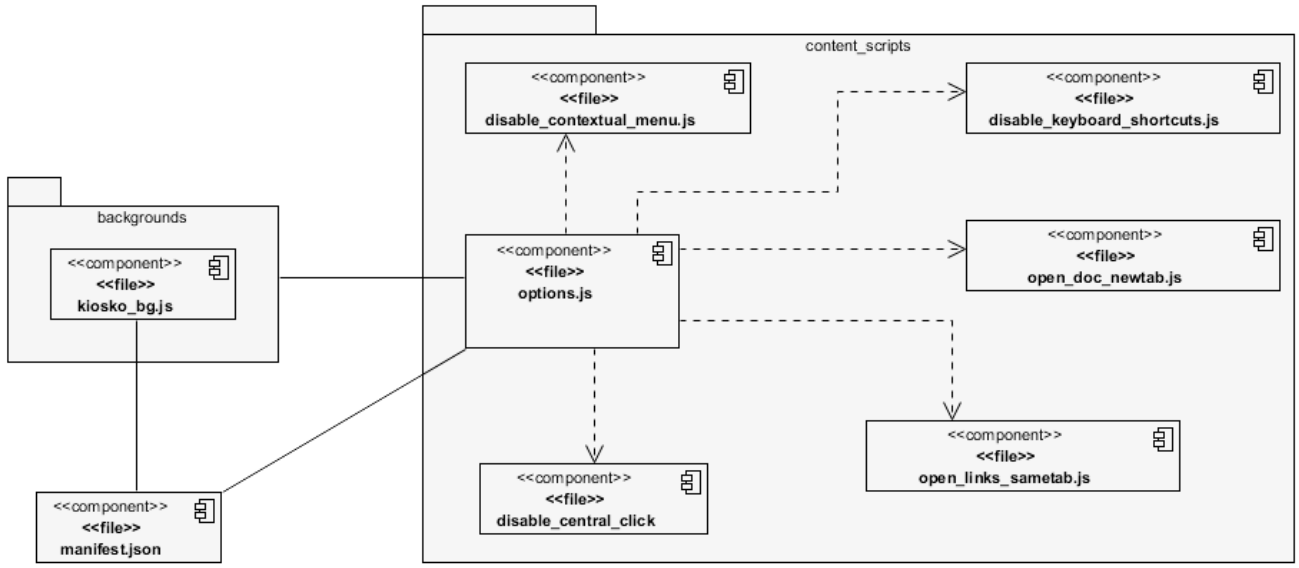


Ilustración 7: Diagrama de componentes (Elaboración propia).

3.2 API WebExtensions para desarrollo de extensiones web

Hasta este punto de la investigación y desarrollo de la extensión web Nova Kiosko, luego de indagar en todas las fuentes de información de *Mozilla* se concluye que este no cuenta con las *API* necesarias para el desarrollo de los siguientes requisitos funcionales, debido a que estos son preferencias por defecto del navegador y no pueden ser modificadas mediante las *API* de *WebExtensions* (developer.mozilla, 2018).

Tabla 7: Requisitos funcionales desarrollados mediante el archivo userChrome.css

ID	Nombre del requisito	Descripción del requisito	Prioridad	Complejidad
RF2	Deshabilitar opciones de desarrollador	El sistema debe permitir deshabilitar las opciones de desarrollador.	Media	Alta
RF8	Deshabilitar campo URL	El sistema debe permitir deshabilitar el campo de direcciones URL	Media	Alta
RF9	Deshabilitar controles de navegación	El sistema debe permitir deshabilitar los controles de navegación (botones de Atrás y Adelante)	Media	Baja
RF10	Deshabilitar la barra de menú	El sistema debe permitir deshabilitar la barra de menú	Media	Media
RF11	Deshabilitar la barra de marcadores	El sistema debe permitir deshabilitar la barra de marcadores	Media	Media

Antes de *WebExtensions* era posible cumplir con el desarrollo de los requisitos mencionados anteriormente pues el *Add-on SDK* incluía la *API preferences/service* que tenía la capacidad de modificar las preferencias del navegador web *Mozilla Firefox.* Actualmente, luego que *Mozilla* cambió su arquitectura de desarrollo de extensiones web, no existe un equivalente a esta *API* en *WebExtensions* (developer.mozilla, 2017).

Descripción de la solución definida para estas funcionalidades

Debido a las razones antes expuestas, por las que estas funcionalidades no pueden ser desarrolladas empleando las actuales *API* de *WebExtensions,* se procede a generar un archivo llamado ***userChrome.css*** que permite editar el diseño por defecto del navegador Mozilla Firefox para que el usuario no tenga acceso a las siguientes interfaces:

- Menú de Opciones de Desarrollador
- Campo de Direcciones URL
- Controles de Navegación
- Barra de Menú
- Barra de Marcadores

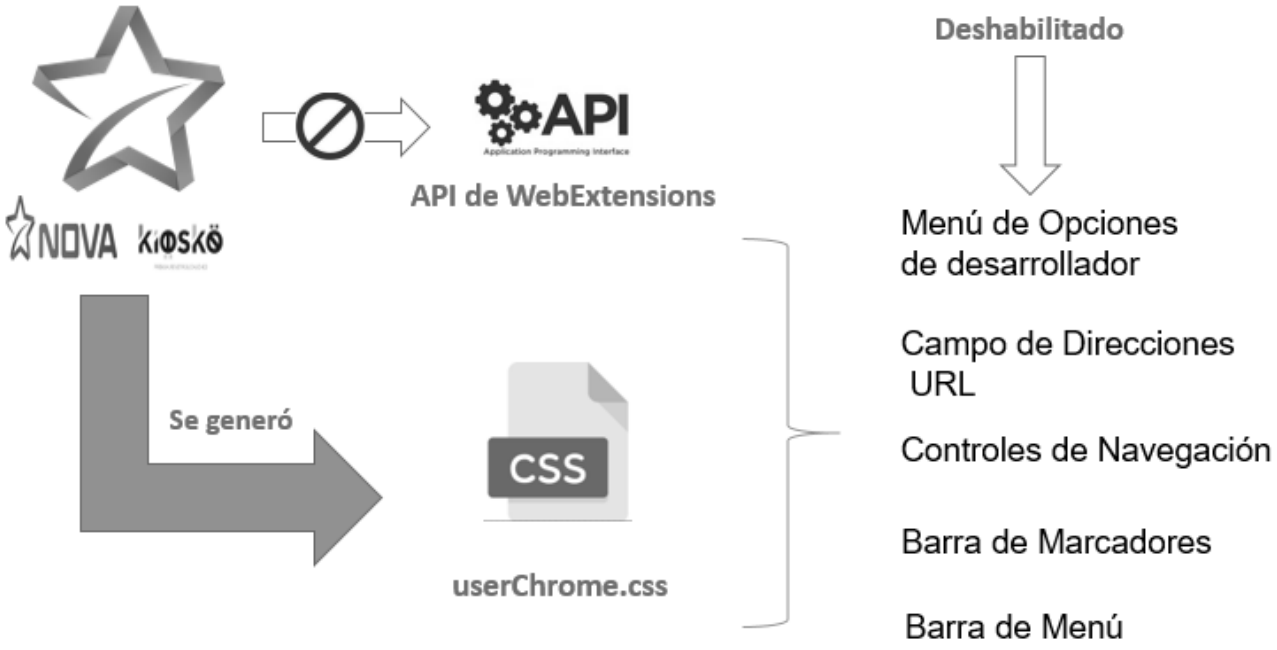


Ilustración 8: Descripción de la propuesta de solución para requisitos que no se desarrollan con API de WebExtensions

Pasos para una correcta puesta en acción del archivo *userChrome.css*

1. El archivo debe ser colocado para su correcto funcionamiento en el siguiente directorio: C:\Users***NombredeUsuario***\AppData\Roaming\Mozilla\Firefox\Profiles***CarpetaDePerfilDelUsuario.***
2. Una vez situado en la carpeta del usuario, es necesario crear un nuevo directorio con el nombre Chrome y dentro de la misma añadir el archivo, el cual no debe variar su nombre y consta de las siguientes líneas de código:

```
/*
 * Do not remove the @namespace line -- it's required for correct functioning
 */
@namespace url("http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"); /* set default namespace to XUL */

/*
 * Hide tab bar, navigation bar and scrollbars
 * !important may be added to force override, but not necessary
 */
#TabsToolbar {visibility: collapse;}
#navigator-toolbox {visibility: collapse;}
#content browser {margin-right: -14px; margin-bottom: -14px;}

#toolbar-context-menu #toggle_PersonalToolbar { visibility: collapse; }

#navigator-urlbar {visibility: collapse;}
```

Ilustración 9: Código fuente del archivo userChrome.css

Luego de colocar el archivo userChrome.css manualmente en el directorio especificado el navegador queda personalizado de manera tal que el usuario solo tenga la posibilidad de navegar en los sitios permitidos bajo las restricciones concebidas con anterioridad en las opciones de configuración de la extensión Nova Kiosko, habilitando una barra lateral como visor de pestañas para su navegación, la cual presenta una combinación especial de teclas que permite su visualización o no (Ctrl + Shift + L) para no estorbar con la navegación en el sitio web.

Ejemplo:

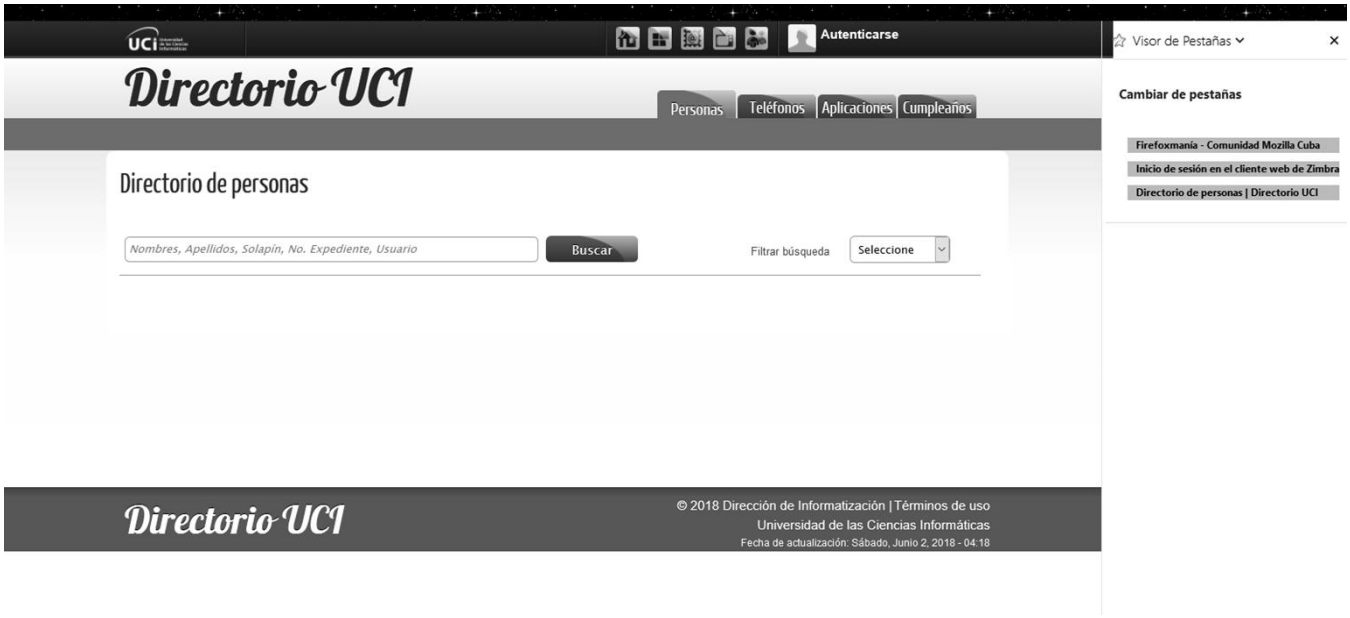


Ilustración 10: Ejemplo del navegador personalizado por mediación de la extensión web y el archivo userChrome.css

3.3 Código Fuente

El código fuente de un sistema informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para así poder ejecutar las funcionalidades requeridas por el *software*. El código fuente varía según el lenguaje de programación y debe ser traducido al lenguaje máquina para que sea ejecutado.

3.3.1 Estándar de codificación

Para garantizar la uniformidad del código en el desarrollo de la extensión, se definió un estándar de codificación, lo cual posibilita cometer menos errores durante la implementación y que la lectura y comprensión del mismo sea mucho más fácil para otros desarrolladores. Cuando el proyecto de software incorpora código fuente previo, o cuando realiza el mantenimiento de un sistema de software el estándar de codificación debería establecer cómo operar con la base de código existente. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

A continuación, se muestran varios ejemplos del estándar utilizado:

Longitud de la línea:

Las líneas tienen siempre no más de 80 caracteres.

Rompiendo líneas:

Cuando una expresión no entra en una línea, se rompe de acuerdo con estos principios:

- Romper después de una coma.
- Romper después de un operador

Declaraciones

- Los métodos y variables se declaran con minúscula, si se hace necesario un nombre del método o variable compuesto se declara con mayúscula en cada inicio de palabra lógica del método o variable.

Ejemplo:

```
function getTypes() {

  let dataTypes = [];
  for (let item of document.querySelectorAll("#kiosko_remove_browsingData [type=checkbox]:checked")) {

    dataTypes.push(item.id)
  }
  return dataTypes;
}
```

Ilustración 11: Ejemplo de Declaraciones de métodos y variables.(Elaboración propia).

Saltos de línea.

- Añadir un salto de línea después de abrir una llave.

Ejemplo:

```
function transformLinksWithDocuments() {

  let links = document.getElementsByTagName('a');
  for (let link of links) {

    let lastCharecters = link.href.substr(-4, 4);
    for (document of documents) {

      if (document === lastCharecters) {

        link.target = '_blank';
      }
    }
  }
}
```

Ilustración 12: Ejemplo del Estándar de Codificación Saltos de líneas, después de un punto y coma o abrir una llave.(Elaboración propia).

3.4 Pruebas de Software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un software, involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (Pressman, 2002).

3.4.1 Niveles de Prueba

- **Pruebas Funcionales:**

Se escoge este nivel de prueba debido a que se desea verificar las funcionalidades del sistema a través de las interfaces que presenta la extensión, específicamente la de opciones de configuración.

Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica (Pressman, 2002).

- **Unitarias**

La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente (Pressman, 2009).

3.4.2 Métodos de Prueba

Caja Blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que (Pressman, 2009).

1. Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
2. Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
3. Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
4. Ejerciten estructuras de datos internos para asegurar su validez.

Caja Negra

Se escoge el método de caja negra ya que los niveles de pruebas escogidos pretenden probar el funcionamiento de la interfaz y de las funcionalidades del sistema, para esto es necesario este método ya que permite comprobar el comportamiento del *software* a través de los requisitos funcionales (48), obviando el funcionamiento interno y la estructura del programa.

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y de terminación.

Ilustración 13: Esquema del Método de Caja Negra.

Técnicas de Prueba

Técnica de prueba Partición equivalente

Características

- Divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba.
- Descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general.
- Reducir al máximo el total de casos de prueba que deben desarrollarse.

El diseño consiste:

- Identificar clases de equivalencia.
- Crear los casos de prueba.

Clase de equivalencia

- Conjunto de estados válidos o inválidos para condiciones de entrada.

Las clases de equivalencia se definen de acuerdo a las siguientes directrices:

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de

equivalencia válida y otra no válida.

4. Si una condición de entrada es booleana, se define una clase de equivalencia válida y otra no válida.

Camino básico

La prueba de camino básico es una técnica de prueba de caja blanca. Este método permite que el diseñador de casos de pruebas obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico deben garantizar que se ejecutan cada instrucción del programa por lo menos una vez durante la prueba (Pressman, 2009).

3.3.4. Aplicación de las pruebas

Pruebas unitarias

Las pruebas unitarias fueron realizadas mediante el método de caja blanca a través de la técnica camino básico. A continuación, se realiza la técnica del camino básico, al método *"handleStartup* ()"


```
4  function handleStartup() {
5    browser.storage.local.get().then( (settings) => {
6      if ( !settings.hasOwnProperty("kiosko_url") ) {
7        browser.storage.local.set({
8          kiosko_url: "http://firefoxmania.uci.cu",
9          kiosko_disable_keyboard_shortcuts: true,
10         kiosko_disable_contextual_menu: false,
11         kiosko_open_links_same_tab: true,
12         kiosko_open_docs_new_tab: true,
13         kiosko_max_open_tabs: "0",
14         kiosko_password: "secret",
15         kiosko_remove_browsingData: {
16           dataTypes: ["history", "downloads"],
17           since: 2
18         },
19         kiosko_disable_pocket: true,
20         kiosko_disable_screemshots: true,
21         kiosko_start_fullscreen_mode: false,
22         kiosko_disable_slow_startup_notification: false,
23         kiosko_internals_url: ["about:", "chrome:", "moz-extension:", "opera:", "file:", "chrome-extension:"]
24       }).then( () => {
25         startOnModeKiosko(settings)
26         checkSettings(settings)
27       }, onError)
28     } else {
29       startOnModeKiosko(settings)
30       checkSettings(settings)
31     }
32   }, onError)
33 }
```

Ilustración 14: Aplicación de las pruebas unitarias mediante el método de caja blanca con la técnica camino básico (Elaboración propia).

Luego de numerar las líneas de código, se diseña la gráfica del programa que describe el flujo de control lógico empleando nodos y aristas.

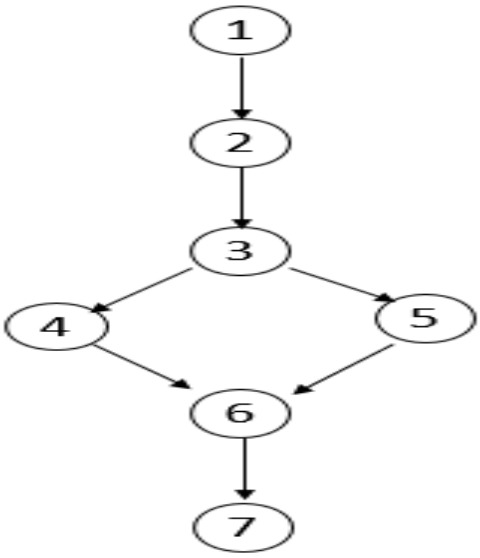


Ilustración 15: Gráfica del programa mediante nodos y aristas (Elaboración propia).

A partir del grafo obtenido con 7 nodos y 7 aristas, se calcula la complejidad ciclomática *V(G),* la cual constituye una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa.

V(G) = cantidad_aristas – cantidad_nodos + 2

V(G) = 7 –7+ 2 = 2

Un camino independiente es cualquier camino del programa que introduce al menos un nuevo conjunto de sentencias de proceso o una nueva condición (Pressman, 2009). La cantidad de caminos independientes se establece por la complejidad ciclomática, por tanto, se identifican 2 caminos como se muestra en la siguiente tabla.

Tabla 8: Listado de caminos independientes.(Elaboración propia)

No.	Camino
1	1-2-3-4-6-7
2	1-2-3-5-6-7

El valor de la complejidad ciclomática ofrece además un límite superior para la cantidad de pruebas que se deben diseñar y ejecutar para garantizar que se cumplen todas las sentencias del programa (Pressman, 2009). A continuación, se muestran los casos de pruebas para cada camino independiente.

Tabla 9: Caso de prueba de unidad, camino1(Elaboración propia)

Caso de prueba de unidad	
No. Camino: 1	**Camino:** 1-2-3-4-6-7
Nombre de la persona que realiza la prueba: José Ernesto Cortes Mendez	
Descripción de la prueba: *handleStartup ().*	
Acción: Actualizar	
Resultado esperado: Se actualizan las preferencias por defecto	
Evaluación de la prueba: Satisfactoria. Preferencias por defecto actualizadas.	

Tabla 10: Caso de prueba de unidad, camino 2.(Elaboración propia)

Caso de prueba de unidad	
No. Camino: 2	**Camino:** 1-2-3-5-6-7
Nombre de la persona que realiza la prueba: José Ernesto Cortes Mendez	
Descripción de la prueba: *handleStartup ().*	
Acción: Chequear	
Resultado esperado: Se chequean las preferencias por defecto	
Evaluación de la prueba: Satisfactoria. Preferencias por defecto chequeadas	

Resultado de las pruebas unitarias

Fueron realizadas dos iteraciones mediante el método de caja blanca aplicando la técnica de camino básico donde no se detectaron errores para el manejo de las preferencias por defecto al iniciar el sistema (*handleStartup ()).***Diseño de los casos de prueba**

Los casos de pruebas fueron definidos a través de los requisitos funcionales descritos en las historias de usuario. La intención que se persigue con estos artefactos es lograr una compresión específica de las condiciones que la solución debe cumplir.

Cada planilla de casos de pruebas contiene la especificación de una historia de usuario, a través de una descripción, condición de ejecución, escenario, respuesta del sistema y flujo central; detallando así, las funcionalidades de la aplicación.

Abreviaturas utilizadas:

CP: Caso de Prueba.

HU: Historia de Usuario.

EC: Escenario.

Tabla 11 :Caso de Prueba de Validación para la Historia de Usuario Gestionar opciones de guardado (Elaboración propia)

Código: CP15_HU15		HU_15: Gestionar opciones de guardado	
Responsable: José Ernesto Cortes Mendez			
Descripción: El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario una serie de opciones para marcar que tipo de información del navegador web desea remover, tales como historial de navegación, cookies, contraseñas guardadas, descargas, sesiones, datos de extensiones; también se presenta el rango de tiempo para remover las opciones seleccionadas tales como: última hora, último día, última semana y desde siempre. El caso de prueba termina cuando el usuario remueve del navegador la información del navegador según las opciones seleccionadas.			
Condiciones de ejecución: 1. El navegador web Mozilla Firefox debe ser una versión mayor o igual a 57.0 2. Debe estar instalada la extensión nova-kiosko en el navegador web Mozilla Firefox 3. El usuario debe haber seleccionado las opciones que desea remover y el rango de tiempo para hacerlo de lo contrario solo removerá las opciones definidas por defecto.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**

EC 15.1 Remover la información del navegador luego de haber seleccionado las opciones deseadas.(Tipo de información y rango de tiempo).	El sistema permite remover correctamente la información del navegador.	Se remueve la información seleccionada en el rango seleccionado y el sistema muestra una notificación de confirmación de la acción "Se ha removido <tipo de información> en <rango de tiempo>"	1.El usuario selecciona la información y el rango de tiempo a remover. 2. Se actualizan las preferencias por defecto a través de el botón "Guardar preferencias" 3.El usuario cliquea la opción remover (botón) 4.El sistema remueve la información del navegador.
EC 15.2 Remover la información del navegador sin haber seleccionado ninguna de las opciones (Tipo de información y rango de tiempo).	El sistema permite remover correctamente la información del navegador.	Se remueve la información en el rango de tiempo definidos por defecto (Historial de navegación y última hora) y el sistema muestra una notificación de confirmación de la acción. "Se ha removido <tipo de información> en <rango de tiempo>"	

EC 15.3 Remover la información del navegado luego de haber desmarcado la opción por defecto del tipo de información (historial de navegación).	El sistema no permite remover la información del navegador.	El sistema muestra un mensaje de error que no hay seleccionada ninguna opción para tipo de información. "No hay tipo de información seleccionada"	

Los restantes casos de pruebas podrán ser consultados en el Anexo 2

3.5 Resultados obtenidos de los casos de prueba

Para probar el correcto funcionamiento de la extensión y cada una de sus funcionalidades se realizaron las pruebas funcionales divididas en tres iteraciones de pruebas. En la primera iteración se detectaron 5 no conformidades, de las cuales 3 son de errores ortográficos y 2 de validación incorrecta. En una segunda iteración se realizaron las pruebas funcionales, para los restantes requisitos que se implementaron y para las funcionalidades que resultaron fallidas donde se detectó 1 no conformidad de validación incorrecta. En una tercera iteración se realizaron las pruebas funcionales donde no se detectaron no conformidades, lo que significa que la extensión funciona correctamente cumpliendo con las expectativas del cliente. A continuación, se muestra cómo quedan reflejados los resultados por cada una de las iteraciones de pruebas funcionales de caja negra realizadas al sistema:

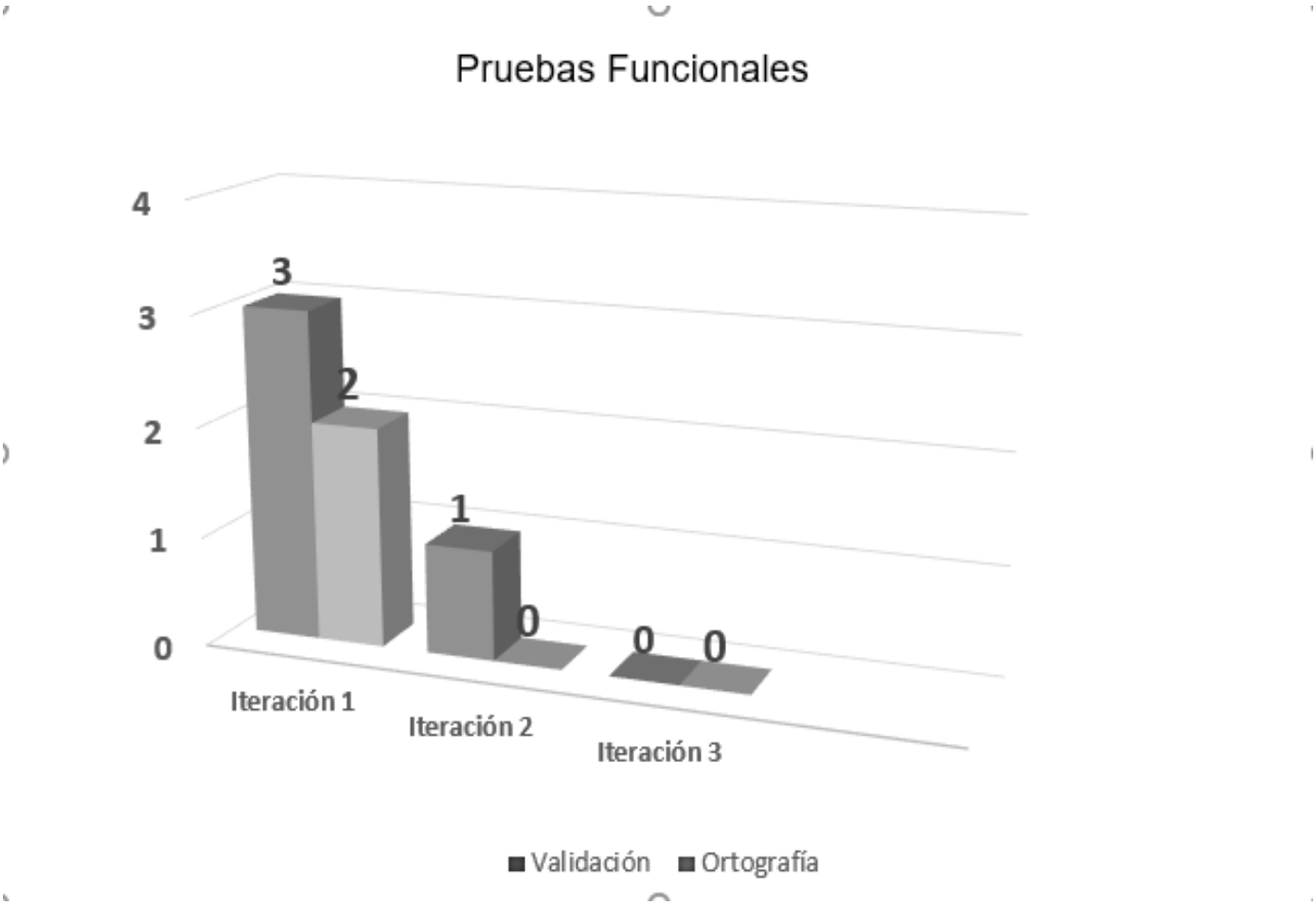


Ilustración 16: Resultado de las pruebas funcionales (Elaboración propia).

3.6 Pruebas de Aceptación

Se escoge realizar la prueba de aceptación debido a que es necesario para la validación de la solución que el cliente esté de acuerdo con el funcionamiento de la extensión desarrollada y así pueda emitir el acta de aceptación del producto. Como técnica se escogen las pruebas alfas, esta se lleva a cabo por un cliente en el lugar de desarrollo, en un entorno controlado y se usa el software de forma natural con el desarrollador como observador del usuario.

Para que tengan validez se debe crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto se procede a realizar las pruebas y a documentar los resultados. El resultado de las pruebas de aceptación se muestra en el Anexo 3 donde fue emitida el acta de aceptación con el cliente.

3.7 Conclusiones del capítulo

En el desarrollo del presente capítulo se realizó el modelo de implementación de la extensión web para la personalización de *Mozilla Firefox* con el propósito de mostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes, lo que permitió una mejor comprensión de la estructura de los componentes que conforman la propuesta de solución. Como parte de la investigación y la solución propuesta se describe la vía para solventar los requisitos que no pudieron ser desarrollados mediante las *API* de *WebExtensions,* lo que posibilitó un mejor entendimiento del desarrollo de la propuesta de solución. Se especificó el uso de los estándares de codificación que posibilitó obtener mayor claridad y organización en el código fuente de la solución. Se realizaron pruebas funcionales para comprobar el correcto funcionamiento de la extensión e identificar fallos, utilizando el método de caja negra basado en la técnica de partición de equivalencia, lo que brindó la posibilidad de suprimir los errores encontrados para que se cumpliese de forma satisfactoria con los requisitos identificados.

Conclusiones

Con la realización de esta investigación se han cumplido los objetivos principales definidos para el desarrollo de una extensión web que permita la personalización del navegador *Mozilla Firefox*. Luego de concluida la primera versión del producto se arribó a las siguientes conclusiones:

1. Se realizó un estudio de las principales extensiones web de similar propósito, lo cual posibilitó determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución.
2. Se realizó el análisis, diseño e implementación de la extensión web obteniendo un archivo .xpi instalable que permite crear personalizaciones en el navegador web *Mozilla Firefox.*
3. Se diseñaron y realizaron las pruebas de *software*, que permitieron la obtención de un *software* completamente funcional y sin errores.

Recomendaciones

Luego de concluida la investigación y el desarrollo de la solución propuesta se recomienda continuar el desarrollo de la extensión web Nova Kiosko a medida que avanzan las tecnologías de desarrollo de *WebExtensions* para agregar nuevas funcionalidades tales como:

1. Definir un pin de seguridad que provea permisos para modificar las preferencias guardadas.
2. Definir una funcionalidad que permita deshabilitar la barra de advertencias del navegador.

Referencias

1. RAMOS, Alicia. *Aplicaciones Web (Novedad 2011)*. Editorial Paraninfo, 2011.
2. MENA DÍAZ, Néstor. Firefox como herramienta para la gestión de información. *Acimed*, 2009, vol. 20, no 4, p. 76-83.
3. CRUZ, Miguel, et al. Interfaz de usuario para un simulador de redes de Petri coloreadas. 2007.
4. Developer.Mozilla. 2017. Developing Extensions. [Online] 2017. [Cited: 11 17, 2017.] https://developer.mozilla.org/en-US/Add-ons.
5. Developer.Mozilla. 2017. Web Extensions. [Online] 2017. [Cited: 11 26, 2017.] https://developer.mozilla.org/en-US/docs/Glossary/WebExtensions.
6. Rodriguez, Gerardo Fernandez. 2010. Repositorio Digital. [Online] 2010. [Cited: 11 26, 2017.] https://repositorio.uci.cu/jspui/handle/ident/TD_03714_10.
7. Gutiérrez, Enrrique González. 2009. aprenderaprogramar.com. *aprenderaprogramar.com.* [En línea] 2009. [Citado el: 8 de enero de 2018.] https://www.aprenderaprogramar.es/attachments/article/435/CU00704B%20Que%20es%20y%20para%20que%20sirve%20HTML%20lenguaje%20web%20mas%20importante.pdf.Quintana, Yoandri Rondón, Camejo , Lianet Domínguez and Díaz , Abel Berenguer. 2011. Diseño de la base de datos para sistemas de digitalización y gestión de medidas. [Online] 2011. [Cited: 12 6, 2017.] http://laboratorios.fi.uba.ar/lie/Revista/Articulos/080815/A3mar2011.pdf.
8. (XP), eXtreme Programming. 2017. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* [Online] 2017. [Cited: 3 2, 2018.] http://www.cyta.com.ar/ta0502/b_v5n2a1.htm..
9. Barbosa Guerrero, Lugo Manuel. 2006. *Arquitectura de software como eje temático de investigación.* 2006.
10. Craig LARMAN. 2003. UML y Patrones. Segunda edición. *UML y Patrones. Segunda edición.* [Online] 2003. [Cited: 2 27, 2018.] http://www.fmonje.com/UTN/ADES%20%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf.
11. DECSAI. 2010. elvex. *elvex.* [Online] 2010. [Cited: 3 4, 2018.] http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

12. definicion.decomplemento. 2018. [Online] 2018. [Cited: mayo 14, 2018.] https://definicion.de/complemento/.

13. developer.mozila. 2017. Que_son_las_WebExtensions. *Que_son_las_WebExtensions.* [Online] septiembre 30, 2017. [Cited: mayo 14, 2018.] https://developer.mozilla.org/es/Add-ons/WebExtensions/Que_son_las_WebExtensions.

14. developer.mozilla. 2017. Anatomia_de_una_WebExtension. *MDN web docs.* [Online] 2017. [Cited: mayo 20, 2018.] https://developer.mozilla.org/es/Add-ons/WebExtensions/Anatomia_de_una_WebExtension.

15. —. 2018. Content_scripts. [Online] 2018. [Cited: mayo 20, 2018.] https://developer.mozilla.org/es/Add-ons/WebExtensions/Content_scripts.

16. —. 2017. Developing Extensions. [Online] 2017. [Cited: 11 17, 2017.] https://developer.mozilla.org/en-US/Add-ons.

17. —. 2018. Learn to style HTML using CSS. *developer.mozilla.org.* [Online] 2018. [Cited: enero 8, 2018.] https://developer.mozilla.org/en-US/docs/Learn/CSS.

18. —. 2017. manifest.json. [Online] 2017. [Cited: mayo 20, 2018.] https://developer.mozilla.org/es/Add-ons/WebExtensions/manifest.json.

19. —. 2017. Web Extensions. [Online] 2017. [Cited: 11 26, 2017.] https://developer.mozilla.org/en-US/docs/Glossary/WebExtensions.

20. —. 2017. web_accessible_resources. [Online] 2017. [Cited: mayo 20, 2018.] https://developer.mozilla.org/en-US/Add-ons/WebExtensions/manifest.json/web_accessible_resources.

21. git-scm. 2018. git-scm. *git-scm.* [Online] 2018. [Cited: enero 9, 2018.] https://git-scm.com/.

22. González, Jesús Rubén Gastón. 2016. repositorio.uam.es. [Online] 2016. [Cited: 16 febrero, 2018.] http://hdl.handle.net/10486/676949.

23. Group, Object Management. 2001. OMG Unified Modeling Language Specification. *OMG Unified Modeling Language Specification.* [Online] 2001. [Cited: 4 10, 2018.] OMG Unified Modeling Language Specification. www.omg.org.

24. Gutiérrez, Enrrique González. 2009. aprenderaprogramar.com. *aprenderaprogramar.com.* [Online] 2009. [Cited: enero 8, 2018.]

https://www.aprenderaprogramar.es/attachments/article/435/CU00704B%20Que%20es%20y%20para%20que%20sirve%20HTML%20lenguaje%20web%20mas%20importante.pdf.

25. Hernández, Enrique Orallo. 2015. El Lenguaje Unificado de Modelado (UML) . [Online] 2015. [Cited: 12 6, 2017.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.
26. Huelga, Yadira García. 2011. Repositorio Digital. [Online] 2011. [Cited: febrero 8, 2018.] https://repositorio.uci.cu/jspui/handle/ident/TD_04987_11.
27. Pressman, Roger. 2002. *Ingeniería del sofware, un enfoque práctico.* 2002.
28. —. 2002. *Ingeniería del sofware, un enfoque práctico.* 2002.
29. Pressman, Roger S. 2009. *Software Engineering. A practitioner´s Approach. New York.* s.l. : McGraw Hill, 2009. ISBN 978-0-07-33759.
30. Pressman, Roger S. 2009. *Software Engineering. A practitioner´s Approach.* New York : McGraw Hill, 2009. ISBN 978-0-07-337597-7.
31. Rodriguez, Gerardo Fernandez. 2010. Repositorio Digital. [Online] 2010. [Cited: 11 26, 2017.] https://repositorio.uci.cu/jspui/handle/ident/TD_03714_10.
32. SOMMERVILLE, IAN. 2005. *Ingeniería de Software.* Madrid : Pearson Addison Wesley, 2005. ISBN:84-7829-074-5..
33. Sommerville, Ian. 2006. *Software Engineering.* 2006. ISBN 0-321-31379-8.
34. Stefanov, Stoyan. 2010. *JavaScript Patterns.* s.l. : O'Reilly, 2010. 978-0-596-80675-0.
35. Synergix, Tecnología y. 2008. Tecnología y Synergix. *Tecnología y Synergix.* [Online] 2008. [Cited: 2 15, 2018.] https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/..
36. Terreros, Ing. Julio Casal. 2018. ComponentSource, What are components? *ComponentSource, What are components?* [Online] 2018. [Cited: mayo 10, 2018.] http://www.componentsource.com/Services/WhatAreComponents.asp?bhcp=1.
37. —. 2017. WebCab Components, About Component Based Development. *WebCab Components, About Component Based Development.* [Online] 2017. [Cited: mayo 10, 2018.] http://webcabcomponents.com/componentization.html.

38. Terreros, Ing. Julio Casal. 2018. ComponentSource, What are components? *ComponentSource, What are components?* [Online] 2018. [Cited: mayo 10, 2018.] http://www.componentsource.com/Services/WhatAreComponents.asp?bhcp=1.

39. —. 2017. WebCab Components, About Component Based Development. *WebCab Components, About Component Based Development.* [Online] 2017. [Cited: mayo 10, 2018.] http://webcabcomponents.com/componentization.html.

40. Quintana, Yoandri Rondón, Camejo , Lianet Domínguez and Díaz , Abel Berenguer. 2011. Diseño de la base de datos para sistemas de digitalización y gestión de medidas. [Online] 2011. [Cited: 12 6, 2017.] http://laboratorios.fi.uba.ar/lie/Revista/Articulos/080815/A3mar2011.pdf.

41. —. 2017. Firefox Releases 57. [Online] 2017. [Cited: junio 4, 2018.] https://developer.mozilla.org/es/Firefox/Releases/57.

Anexos

Anexo 1. Historias de usuarios

Tabla 12: Historia de usuario No.1

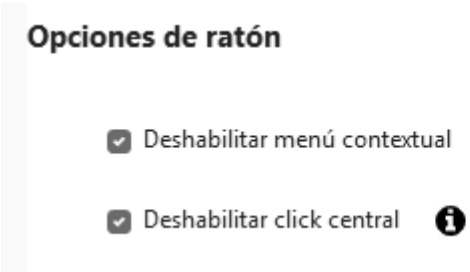
Historia de Usuario	
Número: HU_1	**Nombre del requisito:** Deshabilitar/ habilitar botones del ratón
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: Alto	**Tiempo Real:** 2 días
Descripción: La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar” debajo de la etiqueta “Opciones de ratón”, luego de terminar la acción que indica que los botones del ratón están deshabilitados o habilitados presiona el botón de “Guardar Preferencias” dependiendo de la opción marcada y no permitirá que el usuario utilice estos botones (click derecho, click central) incapacitando sus funciones o simplemente el usuario puede ejecutar cualquier acción con estos botones si están disponibles.	
Prototipo de la interfaz gráfica del usuario 	

Tabla 13: Historia de usuario No.3

Historia de Usuario	
Número: HU_3	**Nombre del requisito:** Personalizar Menú Contextual
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1

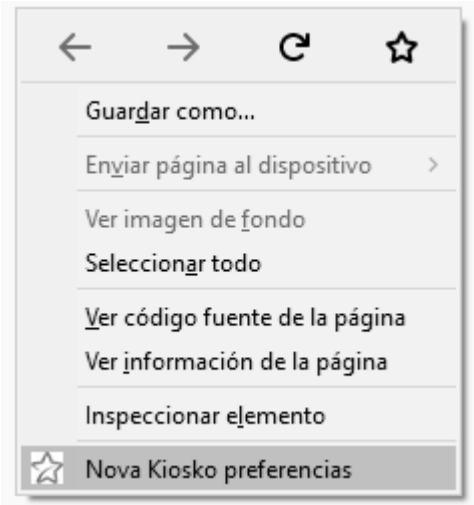
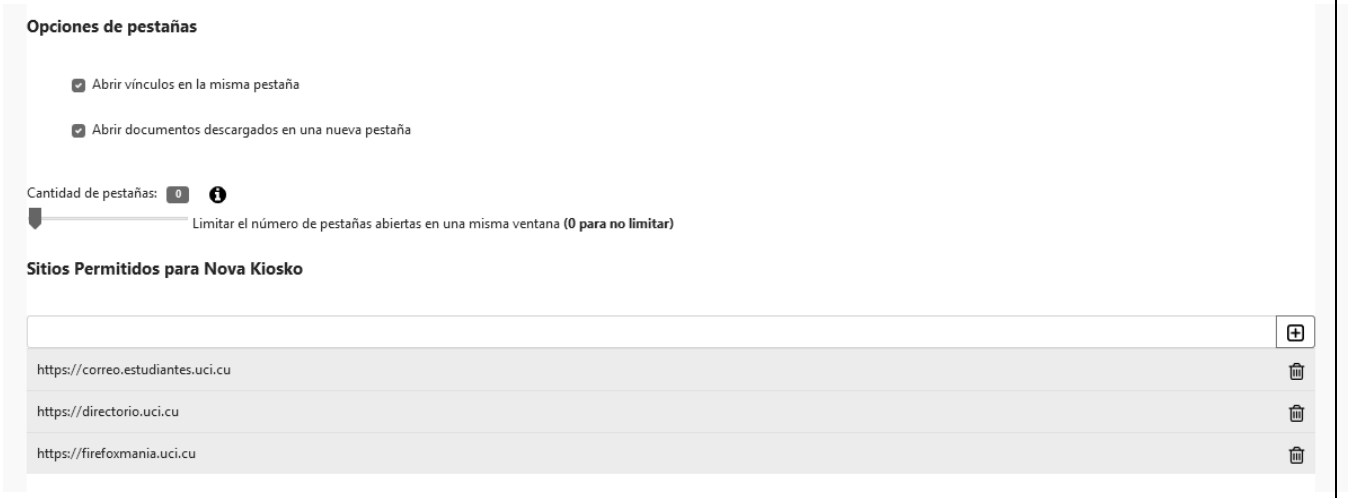
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: Alto	**Tiempo Real:** 24h
Descripción: La HU inicia cuando el usuario accede al menú contextual del navegador este se encuentra personalizado por defecto con dos opciones “Nova Kiosko preferencias” la opción se redirige al usuario a la página de opciones de configuración de la extensión.	
Prototipo de la interfaz gráfica del usuario 	

Tabla 14: Historia de usuario No.7

Historia de Usuario	
Número: HU_7	**Nombre del requisito:** Deshabilitar/ habilitar apertura de documentos descargados
Programador: José Ernesto Cortes Mendez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3 días

Descripción: La HU inicia cuando el usuario accede a las “Opciones de Configuración” de la extensión ubicado como segunda opción al mostrar el menú contextual o en las opciones de configuración de la extensión al poner la url en el navegador “about:addons” y selecciona la opción “Deshabilitar” o “Habilitar” debajo de la etiqueta “Opciones de pestañas”, luego de terminar la acción que indica que los documentos descargados se abrirán en la misma pestaña o en una nueva, presiona el botón “Guardar Preferencias” dependiendo de la opción marcada y no permitirá que el usuario abra los documentos descargados en una nueva pestaña o puede comportarse de forma inversa.

Prototipo de la interfaz gráfica del usuario



Anexo 2. Casos de pruebas para cada historia de usuario

Tabla 15: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar botones del ratón.(Elaboración propia)

<table>
<tr><td colspan="4"></td></tr>
<tr><td colspan="2">Código: CP1_HU1</td><td colspan="2">HU_1: Deshabilitar/Habilitar botones del ratón</td></tr>
<tr><td colspan="4">Responsable: José Ernesto Cortes Mendez</td></tr>
<tr><td colspan="4">Descripción:
El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dentro de la etiqueta “Opciones de ratón” una opción para deshabilitar el menú contextual y otra para deshabilitar el clic central, dependiendo si marcan o no el checkbox con su descripción respectiva será la acción que se ejecute luego de guardar las preferencias mediante el botón ”Guardar Preferencias”</td></tr>
<tr><td colspan="4">Condiciones de ejecución:
1. El navegador web Mozilla Firefox debe ser una versión mayor o igual a 57.0
2. Debe estar instalada la extensión nova-kiosko en el navegador web Mozilla Firefox.
3. El usuario debe haber marcado la opción deseada de lo contrario la extensión por si sola tendrá marcado ambos checkboxs por defecto.</td></tr>
<tr><td>Escenario</td><td>Descripción</td><td>Respuesta del sistema</td><td>Flujo central</td></tr>
<tr><td>EC 1.1 Accionar el clic derecho (menú contextual) o clic central si ambas opciones están marcadas (deshabilitadas).</td><td>El sistema no permite la acción en ninguno de los dos casos.</td><td>Se previene la acción por defecto de estos botones que en este caso no funcionarán.</td><td>1.El usuario marca cuál o cuáles de las dos acciones de los botones desea deshabilitar.
2.Se actualizan las preferencias por defecto a</td></tr>
</table>

EC 1.2 Accionar el clic derecho (menú contextual) estando marcado el *checkbox* (acción deshabilitada) mientras que el clic central está activo dado que la opción esta desmarcada (clic central habilitado).	El sistema permite el clic central en la página, pero no permite el clic derecho (menú contextual).	Se previene la acción por defecto en uno de los casos (menú contextual) dado que no funcionará mientras que en el caso del clic central se permitirá su funcionamiento por defecto.	través de el botón “Guardar preferencias” 3.El sistema permite o no las acciones de los botones antes marcados en el checkbox dependiendo de las opciones seleccionadas.
EC 1.3 Accionar el clic central y el clic derecho (menú contextual) estando los dos desmarcados (es decir habilitados)	El sistema permite el clic central y el clic derecho (menú contextual) en la página.	No se previene la acción por defecto de estos botones que en este caso funcionarán correctamente.	

Tabla 16: Caso de Prueba de Validación para la Historia de Usuario Personalizar Menú Contextual.(Elaboración propia)

Código: CP3_HU3	HU_3: Personalizar Menú Contextual
Responsable: José Ernesto Cortes Mendez	
Descripción: El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dos nuevas acciones en el menú contextual del navegador que permite acceder a las Opciones de Configuración.	

Condiciones de ejecución: 1. El navegador web *Mozilla Firefox* debe ser una versión mayor o igual a 57.0 2. Debe estar instalada la extensión nova-kiosko en el navegador web *Mozilla Firefox.* 3. El usuario debe presionar el clic derecho (menú contextual) en cualquier parte del navegador.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 3.1 Accionar clic derecho en el navegador (menú contextual) y seleccionar la opción “Nova Kiosko preferencias”	El sistema permite la acción de acceder a las opciones de configuración	Redirige a la pestaña donde se encuentran las opciones de configuración de la extensión.	1.El usuario decide accionar el clic derecho (menú contextual). 2.Dependiendo de si la opción “Nova Kiosko preferencias” es seleccionada se llevará a cabo la acción deseada. 3.El sistema permite redirigir a las opciones de configuración a través de “Nova Kiosko preferencias” .

Tabla 17: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar combinaciones de teclas.(Elaboración propia)

Código: CP4_HU4	HU_4:Deshabilitar/Habilitar combinaciones de teclas
Responsable: José Ernesto Cortes Mendez	

Descripción:

El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dentro de la etiqueta “Opciones de teclado” una opción para deshabilitar las todas combinaciones de teclas posibles, dependiendo si es marcado o no este *checkbox* será la acción que se ejecutará luego de guardar las preferencias mediante el botón ”Guardar Preferencias”

Condiciones de ejecución:

1. El navegador web *Mozilla Firefox* debe ser una versión mayor o igual a 57.0
2. Debe estar instalada la extensión nova-kiosko en el navegador web *Mozilla Firefox.*
3. El usuario debe decidir la opción deseada de lo contrario la extensión por si sola tendrá marcado el *checkbox* por defecto.

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 4.1 Presionar cualquier combinación de tecla si el *checkbox* está marcado.	El sistema no permite la acción de la combinación de teclas.	Se previene la acción por defecto de esta combinación de teclas presionada dado que no funcionará	1.El usuario decide marcar o desmarcar la opción de deshabilitar todas las combinaciones de teclas posibles. 2.Se actualizan las preferencias por defecto a través de el botón “Guardar preferencias” 3.El sistema permite o no dependiendo de si la opción está marcada o no la acción por defecto de las combinaciones de teclas
EC 4.2 Presionar cualquier combinación de tecla si el *checkbox* no está marcado	El sistema permite la acción por defecto de la combinación e tecla presionada	No se previene la acción por defecto de ninguna de las combinaciones de teclas presionada, por tanto están funcionaran normalmente.	

EC 4.3 Presionar combinaciones de teclas específicas si el checkbox está marcado. Ejemplos: Ctrl+W, Ctrl+Shift + W, Alt+F4, Ctrl+T, Ctrl+N, y las teclas F5, Esc, Enter, Tab, y flechas de dirección.	El sistema permite la acción por defecto de la combinación e tecla presionada.	No se previene la acción por defecto de estas combinaciones de teclas debido que son combinaciones por defecto del navegador y no pueden se deshabilitadas.	presionadas con las especificidades antes señaladas.

Tabla 18: Caso de Prueba de Validación para la Historia de Usuario Personalizar apertura de vínculos.(Elaboración propia)

Código: CP6_HU6	HU_6: Personalizar apertura de vínculos
Responsable: José Ernesto Cortes Mendez	
Descripción: El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dentro de la etiqueta “Opciones de pestañas” una opción para “abrir vínculos en la misma pestaña” lo que permitirá si el *checkbox* está marcado abrir todos los vínculos en una misma pestaña sin necesidad de usar una nueva pestaña para esto, de lo contrario si la opción no está marcada se hará lo contrario que es la acción por defecto del navegador.	
Condiciones de ejecución: 1. El navegador web *Mozilla Firefox* debe ser una versión mayor o igual a 57.0 2. Debe estar instalada la extensión nova-kiosko en el navegador web *Mozilla Firefox.* 3. El usuario debe haber marcado o desmarcado la opción según su decisión de lo contrario la extensión por si sola tendrá marcado el *checkbox* por defecto.	

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 6.1 Abrir un vínculo sin haber marcado el *checkbox* “abrir todos los vínculos en una misma pestaña”.	El sistema permite la acción por defecto del navegador.	No se previene la acción por defecto del navegador y los vínculos se abrirán en una nueva pestaña.	1.El usuario marca el *checkbox* con la opción “abrir todos los vínculos en una misma pestaña” o no. 2.Se actualizan las preferencias por defecto a través de el botón “Guardar preferencias” 3.El sistema permite o no la acción por defecto del navegador de abrir pestaña en una nueva pestaña dependiendo de la opción deseada (marcar o no el *checkbox*).
EC 6.2 Abrir un vínculo luego de haber marcado el *checkbox* “abrir todos los vínculos en una misma pestaña”.	El sistema no permite la acción por defecto del navegador.	Se previene la acción por defecto del navegador y los vínculos se abrirán en la misma pestaña.	

Tabla 19: Caso de Prueba de Validación para la Historia de Usuario Personalizar apertura de documentos descargados.(Elaboración propia)

Código: CP7_HU7	HU_7: Personalizar apertura de documentos descargados.
Responsable: José Ernesto Cortes Mendez	
Descripción: El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dentro de la etiqueta “Opciones de pestañas” una opción para “abrir documentos descargados en una nueva pestaña” lo que permitirá si el *checkbox* está marcado abrir todos los documentos descargados en una nueva pestaña, de lo contrario si la opción no está marcada se hará lo contrario que es la acción por	

<table>
<tr><td colspan="4">defecto del navegador.</td></tr>
<tr><td colspan="4">Condiciones de ejecución:
1. El navegador web Mozilla Firefox debe ser una versión mayor o igual a 57.0
2. Debe estar instalada la extensión nova-kiosko en el navegador web Mozilla Firefox.
3. El usuario debe haber marcado o desmarcado la opción según su decisión de lo contrario la extensión por si sola tendrá marcado el checkbox por defecto.</td></tr>
<tr><td>Escenario</td><td>Descripción</td><td>Respuesta del sistema</td><td>Flujo central</td></tr>
<tr><td>EC 7.1 Abrir un documento descargados sin haber marcado el checkbox “abrir documentos descargados en una nueva pestaña”.</td><td>El sistema permite la acción por defecto del navegador.</td><td>No se previene la acción por defecto del navegador y los documentos descargados se abrían en una misma pestaña.</td><td rowspan="2">1.El usuario marca el checkbox con la opción “abrir todos los vínculos en una misma pestaña” o no.
2.Se actualizan las preferencias por defecto a través de el botón “Guardar preferencias”
3.El sistema permite o no la acción por defecto del navegador de abrir pestaña en una nueva pestaña dependiendo de la opción deseada (marcar o no el checkbox).</td></tr>
<tr><td>EC 7.2 Abrir un vínculo luego de haber marcado el checkbox “abrir documentos descargados en una nueva pestaña”.</td><td>El sistema no permite la acción por defecto del navegador.</td><td>Se previene la acción por defecto del navegador y los documentos descargados se abrirán en una nueva pestaña.</td></tr>
</table>

Tabla 20: Caso de Prueba de Validación para la Historia de Usuario Deshabilitar/habilitar gestión de las pestañas.(Elaboración propia)

Código: CP14_HU14	HU_14: Deshabilitar/habilitar gestión de las pestañas.		
Responsable: José Ernesto Cortes Mendez			
Descripción: El caso de prueba se inicia al instalarse la extensión. Se presenta al usuario dentro de la etiqueta “Opciones de pestañas” una opción para “limitar la cantidad de pestañas abiertas en una ventana” y otra opción para “Fijar los sitios permitidos para Nova Kiosko” donde la primera permite dependiendo del número marcado por el usuario en la barra restringirá la apertura de pestañas en esa ventana, y la segunda opción es para definir cuál o cuáles serán las URL para iniciar el navegador con estas sin manera alguna de escribir un nueva dirección en el navegador que no esté permitida por la extensión.			
Condiciones de ejecución: 1. El navegador web *Mozilla Firefox* debe ser una versión mayor o igual a 57.0 2. Debe estar instalada la extensión nova-kiosko en el navegador web *Mozilla Firefox.* 3. El usuario debe haber restringido con el número deseado la cantidad de pestañas abiertas para esa ventana. 4. El usuario debe definir las URL del kiosko en el cuadro de texto y agregarlas a la lista de lo contrario están definidas por defecto dos URL.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**

EC 14.1 Abrir una nueva pestaña sin haber restringido la cantidad de pestañas abiertas y las URL del kiosko.	El sistema permite abrir todas las pestañas deseadas por el usuario siempre y cuando estas sean en blanco.	El sistema si no se restringe la cantidad de pestañas en la ventana (límite de cantidad de pestañas = 0, 0 para no limitar) abrirá todas las pestañas que el usuario desee siempre y cuando sean pestañas en blanco debido que por defecto las URL del Kiosko definida por la extensión es *https://correo.estudiantes.uci.cu* y *https://directorio.uci.cu*	1.El usuario define el límite de pestañas o deja el cuadro de texto con los valores por defecto (0 para no limitar) y lo sitios permitidos para Nova Kiosko. 2.Se actualizan las preferencias por defecto a través de el botón “Guardar preferencias” 3.El sistema permite o no la acción de abrir todas las pestañas posibles y cargar las URL que deseen dependen de las opciones seleccionadas.
EC 14.2 Abrir una nueva pestaña sin haber restringido la cantidad de pestañas abiertas pero se definen las URL del kiosko.	El sistema permite abrir nuevas pestañas pero no permite cargar URL que sean diferentes a la definida en el cuadro de texto URL del Kiosko.	Se podrán abrir cuantas pestañas sean necesitas siempre y cuando estas sean en blanco si se intenta cargar alguna URL distinta a las definas en la lista de sitios permitidos para Nova Kiosko se cerrarán automáticamente.	
EC 14.3 Abrir una nueva pestaña luego de haber restringido la cantidad de pestañas abiertas y no definido las URL del kiosko.	El sistema permite abrir nuevas pestañas hasta el límite deseado y si dentro de ese límite se carga una URL esta no se abrirá si es distinta a la	Se podrán abrir cuantas pestañas sean necesarias siempre y cuando estas estén dentro del rango del número de pestañas a limitar, aunque sean en blanco si se intenta cargar alguna URL distinta a las definas en la lista de sitios	

	definida por defecto por la extensión.	permitidos para Nova Kiosko se cerrarán automáticamente .	