Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Informe final de Proyecto Terminal:

Herramienta generadora de código XUL para la implementación de componentes software tipo *TOOLBAR* para el web browser Mozilla Firefox

Jorge Armando Dotor Vázquez

Matrícula: 203301481

Trimestre 13-I

4 de enero 2013

Dr. Oscar Herrera Alcántara

Número económico: 24709

ÍNDICE DE CONTENIDOS

OBJETIVOS

Objetivo general

Desarrollar una herramienta que genere código *XUL* necesario para implementar complementos de tipo *Toolbar* en el *web browser Mozilla Firefox*.

Objetivos particulares

- Elaborar el documento de requerimientos del proyecto.
- Elaborar el modelo de casos de uso de la aplicación.
- Diseñar el modelo de clases de software de la aplicación
- Diseñar el control y lógica de la aplicación con sus módulos internos:
  - Entrada de datos
  - Análisis de datos
  - Generación de código
  - Empaquetador
- Diseñar la interfaz de usuario de la aplicación.
- Diseñar los módulos correspondientes a las librerías de la aplicación.
- Seleccionar los algoritmos más adecuados para implementar la lógica de la aplicación.
- Implementar los módulos de la aplicación.
- Integrar los módulos de la aplicación.
- Realizar pruebas con la aplicación.
- Documentar el desarrollo del proyecto.
- Redactar el informe final sobre el proyecto terminal.

ANTECEDENTES

Un *web browser* es un programa cuya función básica consistía en descargar documentos *HTML*[5] y mostrarlos en pantalla. En la actualidad, no solamente descarga este tipo de documentos, sino que muestra imágenes, sonidos e incluso *streamings* de vídeo en diferentes formatos y protocolos.

Analizando lo anterior, sería difícil ver a un *web browser* como un *framework* de desarrollo de aplicaciones, no obstante, sí se le puede considerar así, ya que permite a los usuarios acceder a aplicaciones como: *e-mail*, comercio electrónico, sitios de noticias, *blogs*, contenidos colaborativos, suites ofimáticas en línea, almacenamiento de archivos en línea, agendas, calendarios y redes sociales. Todas estas aplicaciones han sido creadas utilizando elementos de la diversa gama de tecnologías que existen para la creación de contenido *web* y han aprovechado el uso de una interfaz común con el usuario: el *web browser*.

Mozilla[6] ha llevado más allá esa idea[1,2,3]. El proyecto Mozilla fue iniciado en Marzo de 1998 con el propósito de desarrollar el sucesor de **Netscape's Communicator 4.x browser suite**[7]. Sin embargo, Mozilla no sólo es un *web browser,* también es un *framework* que permite construir aplicaciones multiplataforma o componentes que brinden al *web browser* mayor funcionalidad. El *front-end* multiplataforma de Mozilla (**Cross-Platform Front-End, XPFE**[8]), como también se le conoce al *framework* de Mozilla, es un conjunto de tecnologías y estándares para el desarrollo de aplicaciones[1,2,3], entre los cuales están:

- ***XML-based User-interface Language (XUL***[2,9]***)***. Lenguaje basado en **XML**[10] para la interfaz de Usuario. Se encarga de la estructura y el contenido de una aplicación.
- ***Cascading Style Sheets (CSS***[11]***)***. Hojas de estilo en cascada. Controlan la apariencia (*look and feel*) de una aplicación.
- ***JavaScript***[12]*.* Lenguaje encargado de crear la funcionalidad de una aplicación, sin embargo, otros lenguajes *script*, como *Python*, *Perl* o *Ruby*, pueden ser usados en lugar de *JavaScript*.
- ***Cross-Platform Install (XPInstall***[13]***).*** Instalador Multiplataforma. Empaqueta las aplicaciones para que puedan ser instaladas en cualquier plataforma.
- ***eXtensible Binding Language (XBL***[14]***).*** Lenguaje Extensible de Asociaciones. Usado para crear *widgets* reutilizables con la combinación de *XUL* y *JavaScript*.
- ***Cross-Platform Component Model (XPCOM***[15]***)***. Modelo de Componentes Multiplataforma. Interfaz que permite acceder y utilizar librerías en otros lenguajes.
- ***Cross-Platform Connect (XPConnect***[16]***).*** Permite la conexión entre los componentes *XPCOM* y *JavaScript*.
- ***Resource Description Framework (RDF***[4, 17]***).*** *Framework* de Descripción de Recursos. Usado para

almacenar y transmitir información.

Algunas de las aplicaciones que han sido desarrolladas con el *framework* de Mozilla son:

- **Thunderbird**[18]. Lector de *e-mail*, *blogs*, noticias y contenido RSS.
- **Camino**[19]. *Web browser.*
- **Seamonkey**[20]. *Web browser*, lector de *e-mail*, noticias y cliente de *chat;*
- **Sunbird**[21]. Aplicación de calendario y agenda.
- **Songbird**[22]. Reproductor de música.
- **Firefox**[23]. *Web browse*r

El *web browser* **Mozilla Firefox** utiliza básicamente tres tipos de complementos[1, 2, 3, 30]:

- **Plugins**[24] – piezas de software que interactúan con el *browser* para darle funciones muy específicas. Por ejemplo, *plug-ins* para contenido de formato multimedia.
- **Extensiones**[25] – Agregados que brindan nueva funcionalidad a las aplicaciones. Permiten personalizar el *browser* para ajustarse a las necesidades del usuario.
- **Temas**[26] – También conocidos como *skins*. Permiten cambiar el aspecto de la interfaz de usuario y personalizarla a nuestros gustos.

Un tipo de extensión para Firefox es el complemento **Barra de Herramientas (*toolbar*)**, que es un componente que puede contener menús para presentar una lista de opciones y/o botones que proveen rápido acceso a las tareas más usadas. De esta manera, facilita la navegación por medio de accesos directos a páginas que son visitadas con frecuencia.

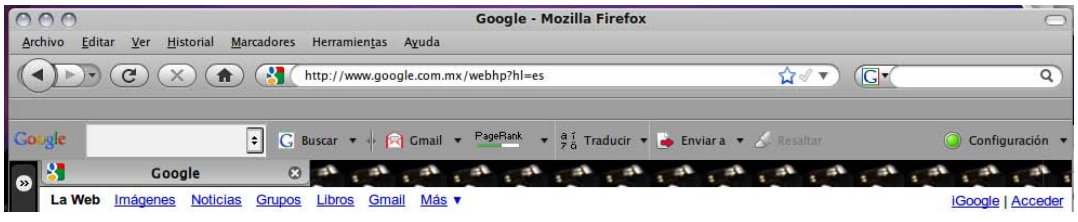

Figura 1. *Web browser* Mozilla Firefox con **Google Toolbar**[27], la cual contiene acceso directo al buscador **Google**[28], servicio de correo electrónico **Gmail**[29] y traductores, entre otros.

Para implementar un *toolbar,* es necesario utilizar muchos elementos del *framework* de Mozilla para generar archivos que describan el contenido, la apariencia y el control o manejo de los eventos del *toolbar*. Además, se realiza una estructura de

directorios que contenga dichos archivos generados, cada uno en una carpeta específica. Finalmente se empaqueta toda la estructura de directorios en un archivo ***XPI***, el cual permite la instalación del complemento en Firefox. Es fácil observar que para implementar una nueva *toolbar*, el proceso antes descrito tiene que ser realizado nuevamente. Y así, para cada toolbar.

El Dr. Oscar Herrera, profesor de la licenciatura en Ingeniería en Computación en la UAM, ha implementado *toolbars*, realizando el proceso para cada una de ellas. El *toolbar* que realizó para el ITESM[30] y el *toolbar* de la UAEM Centro Universitario Zumpango[44], son ejemplos de cómo un complemento agrega funcionalidad al *browser* para acceder fácilmente a los sitios más importantes de dichas instituciones. Además, otorgan un carácter de identificación y personalización de *Firefox* con la comunidad universitaria, ya que los alumnos sólo tienen que instalar un componente personalizado con su universidad.

En el sitio del grupo de desarrolladores del proyecto Mozilla existe un *plug-in*[31] para la *IDE* **Eclipse**[32], el cual genera los archivos vacíos y los directorios necesarios para desarrollar una aplicación *XUL*. Sin embargo, sólo es utilizable para desarrolladores, ya que el código interno de los archivos tiene que ser implementado por el usuario.

Actualmente, no existe una herramienta que permita generar automáticamente un paquete *XPI* para implementar un *toolbar* sin que el usuario tenga conocimientos del *framework*. Tampoco existe un *toolbar* personalizado para los alumnos de la UAM que contenga acceso directo a los sitios de mayor interés en la vida universitaria o la integración de algunos servicios que proporciona nuestra Universidad.

El objetivo de esta propuesta de proyecto terminal es desarrollar una herramienta que permita automatizar, de una forma transparente, la generación de los archivos y la estructura de carpetas, necesarios para implementar una *toolbar* a partir de la entrada de datos del usuario, que básicamente consiste en imágenes para los iconos y los enlaces (*links*) a las páginas *web* destino.

Cabe mencionar que ya se cuenta con conocimientos del lenguaje *JavaScript*, *XML* y *CSS*, que se requieren en el *framework* de Mozilla, sin embargo, se carece de experiencia en las tecnologías restantes, mas allá del análisis necesario para la realización de esta propuesta.

JUSTIFICACIÓN

Como se mencionó anteriormente, el *framework* de Mozilla permite el desarrollo de aplicaciones que pueden ser independientes (*standalone*), o componentes de alguna aplicación existente. Para el caso de esta propuesta, serán componentes para el *web browser* Mozilla Firefox.

Ya se analizó, en forma breve, el proceso que se requiere para implementar un complemento de tipo *toolbar* para el *web browser* Mozilla Firefox. También, se hizo notar la necesidad de realizar todas las acciones para cada *toolbar* que se desee implementar, remarcando que en la actualidad no existe una herramienta específica que permita ahorrar el tiempo para el desarrollo de un *toolbar* y el *plug-in* que existe con este propósito, sólo es utilizable para desarrolladores, ya que dicha herramienta es complemento del *IDE* Eclipse. Sin embargo, un usuario final, sin conocimientos en desarrollo de aplicaciones, no podría personalizar un *toolbar* de acuerdo a sus necesidades o actividades más frecuentes en el *web browser*.

El proyecto propone simplificar la tarea de implementar un tipo de complemento para dicho *web browser*, el *toolbar*. Un usuario final, podrá implementar un *toolbar* con sólo proporcionar algunos datos de los sitios *Web* de su interés y el programa proporcionará un paquete *XPI* para instalar la *toolbar* correspondiente a sus datos.

Es necesario mencionar que, actualmente, los *web browser* cuentan con **marcadores** o **favoritos**, los cuales permiten almacenar en el navegador los sitios que son visitados con mayor frecuencia. Sin embargo, la funcionalidad de un marcador está restringida a una posición en el menú del *web browser* que, generalmente, no es visible en la interfaz del navegador. Además, no existe integración con algunos servicios *web*, es decir, un marcador no puede ser notificado de la llegada de nuevos mensajes en la bandeja del *e-mail*, o integrar un servicio de *chat*, *blog*, *microblog,* contenido, red social o cualquier contenido *RSS*. Se planea, como continuación a futuro, que la aplicación pueda integrar algunos de estos servicios para tener control sobre ellos directamente en el *toolbar*.

Mozilla Firefox es un navegador que actualmente posee el segundo lugar de popularidad entre los *web browsers* utilizados a nivel mundial[33]. Es una aplicación de software libre, gratuita, apegada a estándares internacionales. Es un proyecto en constante desarrollo y ha tenido menos vulnerabilidades que su principal rival, el ***Internet Explorer(IE)***[34] de **Microsoft**[35]. De esta forma, Mozilla se muestra como un *framework* adecuado para personalizar al *web browser* Firefox de acuerdo a las necesidades del usuario, y en los alcances de este proyecto, no se considerará al *web browser* IE.

Algunos usos de la aplicación en la UAM darían como resultado que los alumnos podrían contar con acceso a sitios *web* y servicios de la universidad en línea, algunas oficinas podrían contar con *toolbar* de accesos directos a los principales servidores de la UAM y/o servicios *web* que proporcionen sus mismos u otros departamentos en la institución. Todo esto brindaría a los usuarios un carácter de identidad con la universidad y el *web browser*.

Los conocimientos necesarios para desarrollar este proyecto (programación, interoperabilidad e integración de aplicaciones, análisis y diseño de sistemas, ingeniería de software), son parte de la formación de los alumnos de la Licenciatura en Ingeniería en Computación impartida en la UAM, y por lo mismo es un proyecto que sólo podría ser realizado por un ingeniero con dicho perfil de conocimientos.

Descripción Técnica

Se va a desarrollar una aplicación que permita a los usuarios:

- Especificar el nombre para el toolbar que se va a implementar.
- Especificar el número de botones que va a incluir el toolbar
- Especificar para cada botón: nombre, icono y *link*.
- Especificar, si es el caso, el uso de un menú con sus respectivos botones.
- Especificar apariencia del *toolbar* (color, posición de botones, etc).

Con estos datos proporcionados la aplicación generará:

- Un paquete *XPI*, con el nombre del *toolbar*, que contenga principalmente:
  - Estructura de directorios necesaria para el paquete *XPInstall.*
  - Archivos con código *XUL* necesario para describir la estructura y contenido del *toolbar.*
  - Archivos con código *CSS* que describan el *look and feel* del *toolbar* y sus botones.
  - Archivos con código *JavaScript* que controlarán la lógica del *toolbar* y todas las acciones de los botones.
  - Archivos con código RDF utilizados para registrar la instalación del *toolbar* y las hojas con estilo CSS.
- Un archivo README.txt con instrucciones de instalación del complemento *toolbar.*

Se ha considerado la siguiente distribución por módulos para la implementación de la aplicación:

- **Módulo Interfaz de Usuario.** Interacción con el usuario por medio de un archivo de entrada.
- **Módulo Control.** Lógica y control de la aplicación. Contiene:
  - **Módulo Entrada.** Entrada de datos de la Interfaz de Usuario.
  - **Módulo Analizador.** Analiza la entrada de los datos.
  - **Módulo Generador.** Genera el código correspondiente para cada archivo.
  - **Módulo Empaquetador.** Empaqueta las carpetas en el *XPI.*

- **Módulo *SAX*[36].** Biblioteca para manejo *XML*.
- **Módulo librerías.** Biblioteca de la aplicación.

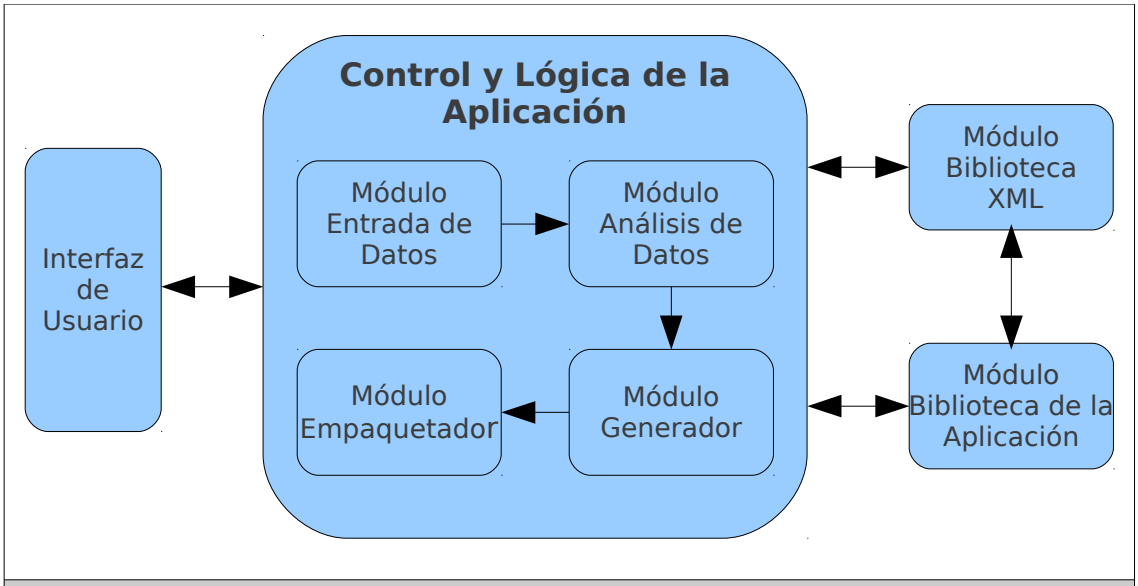

Figura 2. Módulos en los que se dividirá la aplicación a desarrollar.

Todos los módulos serán implementados por el alumno, con excepción del módulo correspondiente a las bibliotecas *XML*, las cuales se obtendrán desde el sitio del proyecto *SAX*. En dado caso, sólo se implementará una interfaz para poder adaptar las librerías con la aplicación.

Se espera que en el toolbar puedan ser integrados, como continuación de proyecto, algunos enlaces a servicios *web* como *e-mail*, buscadores, calendarios o bien algunos servicios proporcionados por la UAM como los sistemas **MOODLE**[37], **SAI**[38], **COSEI**[39], correo de la UAM[40], etc. Por lo cual, se consideraría agregar módulos que permitan la interoperabilidad entre estos componentes, así como las librerías necesarias para tal propósito.

La aplicación será desarrollada bajo el sistema operativo Linux y se hará uso del siguiente software:

- *Mozilla Firefox*
- *BlueFish*[41], editor de documentos
- *Screem*[42], editor documentos *XML*
- *IDE* NetBeans[43]
- *IDE* Eclipse
- *Java SDK*[44]
- *SAX* para el manejo de *XML* en *Java*

El software escogido no requiere pago por uso, por lo que se cuenta con todas las licencias para poder utilizarlo.

Especificaciones Técnicas

La herramienta será implementada con el lenguaje *Java*, de modo que los módulos internos serán implementados con el mismo lenguaje, vigilando la integración e interoperabilidad de los módulos en la herramienta. La herramienta estará en un paquete ejecutable **JAR**.

En una primera etapa, la entrada de datos será por medio de un archivo que contendrá en forma tabular los datos correspondientes a cada botón: **nombre**, **icono** y **link**. Entonces, la interfaz permitirá al usuario escoger el archivo por medio de un selector de Archivo implementado en *Java***.**

Se espera que un avance futuro sea la posibilidad de integrar una interfaz que permita, de forma visual, diseñar el *toolbar* con elementos arrastrables, sin embargo, se evaluará durante el proyecto la vialidad, beneficios y alcances de esta opción.

La conexión entre la interfaz de usuario y el módulo de control y lógica de la aplicación, será realizada mediante clases escritas en el lenguaje *Java*, de modo que éstas permitan una óptima integración de los módulos. Lo anterior se realizará utilizando algunas clases escritas en la biblioteca de *Java* y algunas que será necesario implementar. De igual forma, el módulo que analiza los datos, se comunicará con el módulo generador de archivos por medio de clases y librerías *Java*. Finalmente, el módulo empaquetador se encargará de hacer el paquete *XPI* con todos los archivos y directorios, y notificará, por medio de la interfaz de usuario, que se ha construido el paquete correspondiente al *toolbar*.

Respecto al número de botones que el *toolbar* podrá contener, aún no se ha definido el alcance, pero estará limitado por el tamaño de cada botón y el aspecto visual que tenga el *toolbar* en el navegador, con el fin de no recargar el contenido en el *toolbar.*

El proyecto se dará por concluido, más no estará limitado, cuando se cumpla con las siguientes pruebas sin restricción con respecto al número:

- Archivo ejecutable de la aplicación que corra en una PC cualquiera, que cuente con una versión de la máquina virtual de *Java*.
- Interfaz de Usuario que reciba los datos del *toolbar*.
- Paquete ***XPI*** con archivo **README.txt**, que correspondan al *toolbar* elaborado.

Se entregará un reporte final correspondiente a los lineamientos del Consejo Divisional que incluirá:

- Reporte final impreso
- CD que contendrá:
  - Reporte final (PDF)
  - Manual del usuario (PDF)
  - Manual del programador (PDF)
    - Documentos de análisis y diseño de la aplicación
    - Diagramas de la aplicación

DESARROLLO

Del diseño realizado al inicio de la propuesta de este proyecto, se evaluó modificar el uso de módulos o las tecnologías involucradas para lograr la optimización de código y utilizar librerías que ya pertenecen a la estandarización de Java.

Para la creación del archivo ejecutable JAR, se creó una clase principal(MozFire.java), la cual se encarga de conectar los diversos módulos de la aplicación, así como inicializar las propiedades variables y constantes. Para el manejo de propiedades se utilizó la librería java.util.Properties, la cual lee un archivo properties que contiene un diccionario llave-valor y utiliza una estructura de datos optimizada para buscar las propiedades por la llave obteniendo el valor que se lee desde el archivo. El uso de propiedades permite que la aplicación tenga menos valores constantes inicializados desde código, por lo que facilita modificar algún valor en particular y que la aplicación lea los parámetros en cada ejecución sin necesidad de compilar nuevamente cada que se realiza un cambio.

La clase principal también se encarga de recibir los parámetros de entrada: nombre del archivo de entrada y nombre del toolbar a crear, validando que los parámetros de entrada coincidan con el número definido de argumentos que se deben recibir en la línea de comandos.

Como la estructura de carpeta dentro del archivo XPI contiene diversos archivos, el módulo generador es el encargado de escribir dichos archivos, considerando que el contenido es tomado directamente de los valores de las propiedades que son manejadas por el Properties. Para dicha función de escritura se decidió utilizar una lista de instancias de escritores con búfer(BufferedWriter), cada uno es encargado de escribir cada uno de los archivos dentro del árbol de directorios. La clase principal permite que los demás módulos escriban una cadena de caracteres, String, indicando el BufferedWriter deseado. Al terminar de escribir los archivos, la clase principal cierra todos los escritores. Esto permite que los BufferedWriters permanezcan abiertos para escribir secuencialmente bajo peticiones aleatorias.

Una vez que se han generado los archivos y que los escritores se han cerrado, la clase principal utiliza el módulo empaquetador que está encargado de crear los archivos comprimidos jar y xpi. También es encargado de eliminar los archivos sobrantes para dejar como resultado final el archivo XPI que el usuario final utilizará para instalar el toolbar.

Para los módulos encargados de la manipulación de código XUL, se escogió utilizar las liberías JAXB, Java Architecture for XML Binding. Dichas librerías forman parte de la tecnología Java tanto para la edición Standard como para la empresarial. Estas librerías permiten la creación y el parseo de XML por medio de la utilización de clases Java conocidas como POJO(Plain Old Java Object), las que sólo tienen propiedades privadas que pueden ser accedidas o configuradas por medio de los métodos get y set. JAXB se vale del uso de anotaciones que permiten al programador añadir metadatos a la clase para que la librería de JAXB identifique los elementos que serán utilizados en XML. Las anotaciones también forman parte de la tecnología Java en las más recientes versiones. Cabe mencionar que estos conocimientos fueron adquiridos después de la elaboración de la propuesta, es por

esto que se decidió cambiar las librerías SAX por las librerías JAXB, ya que, éstas últimas permiten ver el modelo de negocio con clases e instancias, y el parseo o creación de código XML es transparente para el programador y permiten que el código esté más ordenado y estandarizado con el uso de POJO's o beans que representen un elemento o atributo XML.

De esta forma, el módulo de análisis y generación de XUL es formado por clases que corresponden directamente a los elementos XML y RDF. La orientación a objetos es un factor clave del uso de las liberías JAXB ya que se requiere la abstracción de los elementos xml en objetos que serán mapeados al xml.

En el anexo de código fuente se puede observar los paquetes org.polesino.mozfire.xul y org.polesino.mozfire.rdf, contienen las clases que son mapeadas a elementos xml utilizando las librerías JAXB. Al tener un control directo sobre la instanciación de elementos xml permite que los módulos de lectura y análisis de datos puedan estar perfectamente integrados, ya que la interpretación del archivo de entrada produce tantas instancias de elementos del toolbar como sean necesarias.

Por último se desarrolló la parte encargada de copiar los archivos de imagen que serán utilizados como íconos en el toolbar. Esta parte toma como ruta de archivo el valor obtenido por el módulo de lectura de datos. Debido a la conexión y dependencia de la lectura de datos con el copiado de archivos de imagen, se decidió integrar estos métodos dentro del módulo encargado para la lectura de datos.

Conclusiones

El desarrollo de este proyecto me permitió aplicar los conocimientos adquiridos en diversas asignaturas de la carrera de Ingeniería en Computación y facilitó el aprendizaje de nuevas tecnologías, dentro de las especificaciones de la edición estándar de *Java*, que permitieron la simplificación y optimización de código para la implementación de diversos módulos de la aplicación, principalmente con la generación de *XML*.

La realización del proyecto cumplió el objetivo principal, ya que se logró desarrollar una herramienta generadora de código *XUL* que es necesario para la creación de complementos de tipo *toolbar* en el *web browser* Mozilla Firefox, así como la estructura básica de carpetas dentro de un archivo comprimido *XPI* que permita la instalación del complemento en el *web browser* Mozilla Firefox.

Para el análisis y diseño de la aplicación, se elaboraron los documentos necesarios para identificar los requerimientos del proyecto, así como, el modelo de casos de uso de la aplicación, mismos que permitieron diseñar el modelo de clases de software de la aplicación con los que se dividió la lógica y control de la aplicación en módulos internos encargados de funcionalidad específica como: entrada y análisis de datos, generación de código, archivos y carpetas, y por último el módulo empaquetador que crea el archivo comprimido *XPI* para la instalación del *toolbar*.

Para la interfaz de usuario, se escogió la entrada de datos por medio de un archivo con los parámetros necesarios para cada elemento del *toolbar*, ya que se buscó una mayor facilidad de ejecución para el usuario. Así mismo, se escogió una estructura muy sencilla para el archivo de entrada, lo que garantiza que un usuario pueda crear fácilmente un archivo sin conocimientos en *XML*, *XUL* o cualquiera de las tecnologías utilizadas por el *framework* de Mozilla.

Se diseñó e implementó el uso de librerías propias de la aplicación para el manejo de *namespaces*, así como constantes y variables buscando que fueran configurables gracias al manejo de las utilerías de *java* para configuración de propiedades. Esto permitirá que se puedan modificar los valores por *default*, si el *framework* de Mozilla lo requiriera en un futuro.

También, durante el diseño y análisis se buscó que la codificación y los algoritmos usados en la lógica de la aplicación siguieran una estandarización y estilo recomendados como buenas prácticas de Java, principalmente en nomenclatura de clases, propiedades, métodos y paquetes.

BIBLIOGRAFÍA

1 BOSWEL, David et.al. "*Creating Applications with Mozilla*".
First Edition. O'Reilly September(2002).

2 BULLARD, Vaughn et.al. "*Essential XUL Programming*".
First Edition. John Wiley & Sons. (2001).

3 FELDT, Kenneth C. "*Programming Firefox*".
First Edition. O'Reilly. April (2007).

4 POWERS, Shelley. "*Practical RDF*".
First Edition. O'Reilly. July (2003).

5 HyperText Markup Language (HTML)
http://www.w3.org/TR/html401
Visitado el 11 de marzo de 2009.

6 The Mozilla Project
http://www.mozilla.org/
Visitado el 11 de marzo de 2009.

7 The Netscape Archive
http://browser.netscape.com/
Visitado el 11 de marzo de 2009.

8 Mozilla Cross-Platform Front End (XPFE). https://developer.mozilla.org/en/
Visitado el 11 de marzo de 2009.

9 XUL. https://developer.mozilla.org/en/XUL
Visitado el 11 de marzo de 2009.

10 eXtensible Markup Language (XML). http://www.xml.com/
Visitado el 11 de marzo de 2009.

11 Cascading Style Sheets (CSS). http://www.w3.org/Style/CSS/
Visitado el 11 de marzo de 2009.

12 JavaScript. https://developer.mozilla.org/en/JavaScript
Visitado el 11 de marzo de 2009.

13 XPInstall. https://developer.mozilla.org/en/XPInstall
Visitado el 11 de marzo de 2009.

14 XBL. https://developer.mozilla.org/en/XBL
Visitado el 11 de marzo de 2009.

15 XPCOM. https://developer.mozilla.org/en/XPCOM
Visitado el 11 de marzo de 2009.

16 XPConnect. https://developer.mozilla.org/en/XPConnect
Visitado el 11 de marzo de 2009.

17 RDF. http://www.w3.org/RDF/
Visitado el 11 de marzo de 2009.

18 Thunderbird. http://www.mozillamessaging.com/en-US/thunderbird/
Visitado el 11 de marzo de 2009.

19 Camino. http://caminobrowser.org/
Visitado el 11 de marzo de 2009.

20 Seamonkey. http://www.seamonkey-project.org/
Visitado el 11 de marzo de 2009.

21 Sunbird. http://www.mozilla.org/projects/calendar/sunbird/
Visitado el 11 de marzo de 2009.

22 Songbird. http://www.getsongbird.com/
Visitado el 11 de marzo de 2009.

23 Firefox. http://www.mozilla.com/en-US/
Visitado el 11 de marzo de 2009.

24 Complementos para Firefox – Plugins. https://addons.mozilla.org/es-ES/firefox/browse/type:7
Visitado el 11 de marzo de 2009.

25 Complementos para Firefox – Extensiones. https://addons.mozilla.org/es-ES/firefox/
Visitado el 11 de marzo de 2009.

26 Complementos para Firefox – Temas. https://addons.mozilla.org/es-ES/firefox/browse/type:2
Visitado el 11 de marzo de 2009.

27 Google Toolbar.
http://www.google.com/tools/firefox/toolbar/FT5/intl/es/index.html
Visitado el 11 de marzo de 2009.

28 Google. http://www.google.com.mx/intl/es/about.html
Visitado el 11 de marzo de 2009.

29 Gmail. http://mail.google.com/mail/help/intl/en/about.html
Visitado el 11 de marzo de 2009

30 HERRERA, Oscar. "*On implementing a Mozilla Firefox Extension: The ITESM Toolbar*"
5o Festival GNU/Linux y Software Libre organizado por la Universidad de Guadalajara
Guadalajara, México. Septiembre (2006)

31 CODEGEN for XUL a plugin for Eclipse. http://www.firedictionary.com/dev/index.html
Visitado el 11 de marzo de 2009.

32 Eclipse.org Home. http://www.eclipse.org/
Visitado el 11 de marzo de 2009.

33 Uso actual de los navegadores.
http://es.wikipedia.org/wiki/Browser#Uso_actual_de_navegadores_web
Visitado el 11 de marzo de 2009.

34 Internet Explorer
http://www.microsoft.com/latam/windows/internet-explorer/download-ie.aspx
Visitado el 11 de marzo de 2009.

35 Microsoft. http://www.microsoft.com/en/us/default.aspx
Visitado el 11 de marzo de 2009.

36 SAX. http://www.saxproject.org/
Visitado el 11 de marzo de 2009.

37 MOODLE. http://moodle.org/
Visitado el 11 de marzo de 2009.

38 SAI. http://sainet.uam.mx/moodle/login/index.php
Visitado el 11 de marzo de 2009.

39 Correo electrónico de la UAM. https://alumnos.azc.uam.mx/cgi-bin/openwebmail.pl
Visitado el 11 de marzo de 2009.

40 BlueFish. http://bluefish.openoffice.nl/
Visitado el 11 de marzo de 2009.

41 Screem. http://www.screem.org/
Visitado el 11 de marzo de 2009.

42 NetBeans. http://www.netbeans.org/
Visitado el 11 de marzo de 2009.

43 Java SDK. http://java.sun.com/javase/downloads/index.jsp
Visitado el 11 de marzo de 2009.

44 RAMIREZ, Sonia. "*Implementación de una barra de herramientas en Mozilla Firefox para el Centro Universitario Zumpango: La CUZ-Toolbar*", Tesis de Licenciatura.
Universidad Autónoma del Estado de México, Centro Universitario Zumpango. (En desarrollo). Abril, 2009

45 JAXB, Java Architecture for XML Binding
http://www.oracle.com/technetwork/articles/javase/index-140168.html
Visitado el 14 de diciembre de 2012

Recursos

El alumno financiará el proyecto ya que ninguno de los requisitos excede a los recursos disponibles. Los recursos con los que el alumno dispone son:

- Computadora personal
- **Linux** – Sistema Operativo
- **Eclipse** – *IDE* para *Java*
- **Netbeans** – IDE para *Java*
- **BlueFish** – editor de textos para archivos *XML*, *CSS* y *JavaScript*.
- **Umbrello** – diseño en *UML*
- **Mozilla Firefox** – *web browser*

Todas las aplicaciones están bajo las licencias de Software Libre, por lo que no hay que comprar licencias privadas.

Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Manual de programador

Herramienta generadora de código XUL para la implementación de componentes software tipo *TOOLBAR* para el web browser Mozilla Firefox

Jorge Armando Dotor Vázquez

Matrícula: 203301481

Trimestre 13-I

4 de enero 2013

Dr. Oscar Herrera Alcántara

Número económico: 24709

ÍNDICE DE CONTENIDOS

Código fuente

MozFire.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package proyectoTerminal;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.lang.annotation.Annotation;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import org.polesino.mozfire.input.Reader;
import org.polesino.mozfire.output.Packer;
import org.polesino.mozfire.rdf.Description;
import org.polesino.mozfire.rdf.Rdf;
import org.polesino.mozfire.rdf.TargetApplication;
import org.polesino.mozfire.xml.annotation.XmlStylesheet;
import org.polesino.mozfire.xul.Overlay;

/**
 * Clase principal para ejecución de la aplicación
 * @author polesino
 */
public class MozFire {

    private static final List<BufferedWriter> bufferedWriters = new ArrayList<>();
    private static final Properties properties = new Properties();
    private static final String PROPERTIES_FILE = "./conf/mozfire.properties";
    private static final String MOZFIRE_USAGE = "mozfire.usage";
    /**
     * Constante para identificador del BufferedWriter para Archivo chrome.manifest
     */
    public static final int WRITER_CHROMEMANIFEST = 0;
    /**
     * Constante para identificador del BufferedWriter para chrome/archivo.js
     */
    public static final int WRITER_JAVASCRIPT_CHROME = 1;
    /**
     * Constante para identificador del BufferedWriter para chrome/content/archivo.js
     */
    public static final int WRITER_JAVASCRIPT_CONTENT = 2;
    /**
     * Constante para identificador del BufferedWriter para chrome/skin/archivo.css
     */
    public static final int WRITER_CSS_CHROME = 3;
    /**
     * Constante para identificador del BufferedWriter para skin/archivo.css
     */
    public static final int WRITER_CSS_SKIN = 4;
    /**
     * Constante para identificador del BufferedWriter para content/archivo.xul
     */
    public static final int WRITER_XUL_CONTENT = 5;
    /**
     * Constante para identificador del BufferedWriter para chrome/content/archivo.xul
     */
    public static final int WRITER_XUL_CHROME = 6;
    /**
     * Constante para directorio content
     */
    public static String DIR_CONTENT = "dir.content";
    /**
     * Constante para directorio chrome
     */
    public static String DIR_CHROME = "dir.chrome";
    /**
     * Constante para directorio skin
     */
    public static String DIR_SKIN = "dir.skin";
    /**
     * Constante para valor del archivo chrome.manifest
     */
    public static String FILE_CHROMEMANIFEST = "chrome.manifest.file";
    /**
     * Constante para extensión archivo CSS
     */
    public static String EXT_CSS = "css.ext";
```

```
/**
 * Constante para valor de extensión javascript
 */
public static String EXT_JAVASCRIPT = "javascript.ext";
/**
 * Constante para valor de extensión XUL
 */
public static String EXT_XUL = "xul.ext";
/**
 * Constante para valor de extensión XPI
 */
public static String EXT_XPI = "xpi.ext";
/**
 * Constante para valor de extensión de JAR
 */
public static String EXT_JAR = "jar.ext";
/**
 * Constante para valor de extensión RDF
 */
public static String FILE_RDF = "rdf.file";
/**
 * Constante para valor de contenido Javascript
 */
public static String JAVASCRIPT_CONTENT = "javascript.content";
/**
 * Constante para valor de contenido chrome.manifest
 */
public static String CHROME_MANIFEST_CONTENT = "chrome.manifest.content";
/**
 * Constante para valor de overlay del chrome.manifest
 */
public static String CHROME_MANIFEST_OVERLAY = "chrome.manifest.overlay";
/**
 * Constante para valor de skin del chrome.manifest
 */
public static String CHROME_MANIFEST_SKIN = "chrome.manifest.skin";
/**
 * Constante para valor de xmlheaders
 */
public static String MARSHALLER_XMLHEADERS = "marshaller.xmlHeaders";
/**
 * Constante para valor de contenido css
 */
public static String CSS_CONTENT = "css.content";
/**
 * Constante para valor de tóken a reemplazar toolbar
 */
public static String REPLACE_TOOLBAR = "replace.toolbar";
/**
 * Constante para valor de token a reemplazar url
 */
public static String REPLACE_URL = "replace.url";
/**
 * Constante para valor de button
 */
public static String TYPE_BUTTON = "type.button";
/**
 * Constante para valor de menu
 */
public static String TYPE_MENU = "type.menu";
/**
 * Constante para valor de src para elemento overlay
 */
public static String OVERLAY_SCRIPT_SRC = "overlay.script.src";
/**
 * Constante para valor de type para elemento overlay
 */
public static String OVERLAY_SCRIPT_TYPE = "overlay.script.type";
/**
 * Constante para valor de elemento about para Description
 */
public static String RDF_DESCRIPTION_ABOUT = "rdf.description.about";
/**
 * Constante para valor de elemento creator para Description
 */
public static String RDF_DESCRIPTION_CREATOR = "rdf.description.creator";
/**
 * Constante para valor de elemento description para Description
 */
public static String RDF_DESCRIPTION_DESCRIPTION = "rdf.description.description";
/**
 * Constante para valor de elemento id para Description
 */
public static String RDF_DESCRIPTION_ID = "rdf.description.id";
/**
 * Constante para valor de elemento name para Description
 */
public static String RDF_DESCRIPTION_NAME = "rdf.description.name";
/**
 * Constante para valor de elemento url para Description
 */
public static String RDF_DESCRIPTION_URL = "rdf.description.url";
/**
 * Constante para valor de elemento version para Description
 */
```

```
public static String RDF_DESCRIPTION_VERSION = "rdf.description.version";
/**
 * Constante para valor de elemento id para TargetApplication
 */
public static String RDF_TARGETAPPLICATION_ID = "rdf.targetAplication.id";
/**
 * Constante para valor de maxVersion para TargetApplication
 */
public static String RDF_TARGETAPPLICATION_MAX = "rdf.targetAplication.max";
/**
 * Constante para valor de minVersion para TargetApplication
 */
public static String RDF_TARGETAPPLICATION_MIN = "rdf.targetAplication.min";
/**
 * Constante para valor de sufijo toolbar
 */
public static String SUFFIX_TOOLBAR = "suffix.toolbar";
/**
 * Constante para valor del nombre del toolbar a crear
 */
public static String TOOLBAR_NAME;

/**
 * Carga de properties desde archivo.
 */
static {
    try {
        properties.load(new FileReader(PROPERTIES_FILE));
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
}

/**
 * inicializar variables desde properties
 */
private static void initializeVariables() {
    CSS_CONTENT = get(CSS_CONTENT);
    REPLACE_TOOLBAR = get(REPLACE_TOOLBAR);
    REPLACE_URL = get(REPLACE_URL);
    TYPE_BUTTON = get(TYPE_BUTTON);
    TYPE_MENU = get(TYPE_MENU);
    DIR_CHROME = get(DIR_CHROME);
    DIR_CONTENT = get(DIR_CONTENT);
    DIR_SKIN = get(DIR_SKIN);
    EXT_CSS = get(EXT_CSS);
    EXT_JAVASCRIPT = get(EXT_JAVASCRIPT);
    EXT_XUL = get(EXT_XUL);
    FILE_CHROMEMANIFEST = get(FILE_CHROMEMANIFEST);
    FILE_RDF = get(FILE_RDF);
    OVERLAY_SCRIPT_SRC = get(OVERLAY_SCRIPT_SRC);
    OVERLAY_SCRIPT_TYPE = get(OVERLAY_SCRIPT_TYPE);
    RDF_DESCRIPTION_ABOUT = get(RDF_DESCRIPTION_ABOUT);
    RDF_DESCRIPTION_CREATOR = get(RDF_DESCRIPTION_CREATOR);
    RDF_DESCRIPTION_DESCRIPTION = get(RDF_DESCRIPTION_DESCRIPTION);
    RDF_DESCRIPTION_ID = get(RDF_DESCRIPTION_ID);
    RDF_DESCRIPTION_NAME = get(RDF_DESCRIPTION_NAME);
    RDF_DESCRIPTION_URL = get(RDF_DESCRIPTION_URL);
    RDF_DESCRIPTION_VERSION = get(RDF_DESCRIPTION_VERSION);
    RDF_TARGETAPPLICATION_ID = get(RDF_TARGETAPPLICATION_ID);
    RDF_TARGETAPPLICATION_MAX = get(RDF_TARGETAPPLICATION_MAX);
    RDF_TARGETAPPLICATION_MIN = get(RDF_TARGETAPPLICATION_MIN);
}

/**
 * Reemplaza la cadena por el nombre del toolbar
 * @param replaceable cadena a reemplazar.
 * @return Cadena reemplazada
 */
public static String replaceToolbar(String replaceable) {
    return replaceable.replaceAll(REPLACE_TOOLBAR, TOOLBAR_NAME);
}

/**
 * Inicio de BufferedWriters
 * @throws IOException
 */
private static void startBufferWriters() throws IOException {
    bufferedWriters.add(WRITER_CHROMEMANIFEST,
            new BufferedWriter(new FileWriter(FILE_CHROMEMANIFEST)));
    bufferedWriters.add(WRITER_JAVASCRIPT_CHROME,
            new BufferedWriter(new FileWriter(DIR_CHROME + DIR_CONTENT + TOOLBAR_NAME + EXT_JAVASCRIPT)));
    bufferedWriters.add(WRITER_JAVASCRIPT_CONTENT, new BufferedWriter(
            new FileWriter(DIR_CONTENT + TOOLBAR_NAME + EXT_JAVASCRIPT)));
    bufferedWriters.add(WRITER_CSS_CHROME,
            new BufferedWriter(new FileWriter(DIR_CHROME + DIR_SKIN + TOOLBAR_NAME + EXT_CSS)));
    bufferedWriters.add(WRITER_CSS_SKIN,
            new BufferedWriter(new FileWriter(DIR_SKIN + TOOLBAR_NAME + EXT_CSS)));
    bufferedWriters.add(WRITER_XUL_CONTENT,
            new BufferedWriter(new FileWriter(DIR_CONTENT + TOOLBAR_NAME + EXT_XUL)));
    bufferedWriters.add(WRITER_XUL_CHROME,
            new BufferedWriter(new FileWriter(DIR_CHROME + DIR_CONTENT + TOOLBAR_NAME + EXT_XUL)));
}

/**
```

```
 * Cerrar BufferedWriters
 */
private static void shutdownWriters() {
    for (BufferedWriter bw : bufferedWriters) {
        if (bw != null) {
            try {
                bw.flush();
                bw.close();
                bw = null;
            } catch (IOException ex) {
                ex.printStackTrace(System.out);
            }
        }
    }
    bufferedWriters.clear();
}

/**
 * Escribe un String para el BufferedWriter deseado
 * @param bufferedWriter identificador del BuffereWriter
 * @param content Cadena a escribir
 * @throws IOException
 */
public static void write(int bufferedWriter, String content) throws IOException {
    BufferedWriter bw = bufferedWriters.get(bufferedWriter);
    bw.write(content);
    bw.newLine();
}

/**
 * Obtiene la propiedad contenida en Properties.
 * @param key clave a buscar
 * @return valor encontrado
 */
public static String get(String key) {
    return properties.getProperty(key);
}

/**
 * Obtiene la propiedad contenida en Properties, sino se encuentra regresa el valor default
 * @param key clave a buscar
 * @param defaultValue valor por default
 * @return valor encontrado o por defuault
 */
public static String get(String key, String defaultValue) {
    return properties.getProperty(key, defaultValue);
}

/**
 * Crea directorio
 * @param directory directorio a ser creado
 */
private static void makeDirectory(String directory) {
    System.out.println("Creating [" + directory + "] ");
    File dir = new File(directory);
    if (!dir.mkdir()) {
        if (dir.isDirectory()) {
            System.out.println("Directory already exists...");
        }
        if (dir.isFile()) {
            System.out.println("Already exists a file with same name.");
        }
    } else {
        System.out.println("directory created: " + dir.getAbsolutePath());
    }
}

/**
 * Crea la estructura de archivos necesaria para el XPI
 * <b>chrome/</b>
 * <b>content/</b>
 * <b>skin/</b>
 * <b>chrome/content</b>
 * <b>chrome/skin</b>
 */
private static void createFileStructure() {
    makeDirectory(DIR_CHROME);
    makeDirectory(DIR_CONTENT);
    makeDirectory(DIR_SKIN);
    makeDirectory(DIR_CHROME + DIR_CONTENT);
    makeDirectory(DIR_CHROME + DIR_SKIN);
}

/**
 * Escribe el contenido del archivo chrome.manifest
 * @throws IOException
 */
private static void writeChromeManifest() throws IOException {
    StringBuilder sb = new StringBuilder(get(CHROME_MANIFEST_CONTENT));
    sb.append(get(CHROME_MANIFEST_OVERLAY)).append(get(CHROME_MANIFEST_SKIN));
    String content = replaceToolbar(sb.toString());
    write(WRITER_CHROMEMANIFEST, content);
}

/**
```

```
     * Escribe el contenido del archivo javascript de control del xpi
     * @throws IOException
     */
    private static void writeJavascript() throws IOException {
        String content = replaceToolbar(get(JAVASCRIPT_CONTENT));
        write(WRITER_JAVASCRIPT_CHROME, content);
        write(WRITER_JAVASCRIPT_CONTENT, content);
    }

    /**
     * Escribe el archivo RDF
     * @throws JAXBException
     */
    private static void writeRdf() throws JAXBException {
        JAXBContext jaxbc = JAXBContext.newInstance(Rdf.class);
        Marshaller m = jaxbc.createMarshaller();
        m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        Rdf rdf = new Rdf();
        Description descRdf = new Description(RDF_DESCRIPTION_ABOUT,
                RDF_DESCRIPTION_ID, RDF_DESCRIPTION_NAME, RDF_DESCRIPTION_VERSION);
        Description descTa = new Description(RDF_TARGETAPPLICATION_ID,
                RDF_TARGETAPPLICATION_MIN, RDF_TARGETAPPLICATION_MAX);
        descRdf.setTargetApplication(new TargetApplication(descTa));
        rdf.setDescription(descRdf);
        descRdf.setCreator(RDF_DESCRIPTION_CREATOR);
        descRdf.setDescription(RDF_DESCRIPTION_DESCRIPTION);
        descRdf.setHomepageUrl(RDF_DESCRIPTION_URL);
        m.marshal(rdf, new File(FILE_RDF));
    }

    /**
     * Escribe el archivo XUL
     * @param source entrada de datos del usuario
     * @throws JAXBException
     * @throws FileNotFoundException
     * @throws IOException
     */
    private static void writeXUL(String source) throws JAXBException,
            FileNotFoundException, IOException {
        JAXBContext jaxbc = JAXBContext.newInstance(Overlay.class);
        Marshaller m = jaxbc.createMarshaller();
        m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        Annotation annotation = Overlay.class.getAnnotation(XmlStylesheet.class);
        if (annotation instanceof XmlStylesheet) {
            XmlStylesheet xsl = (XmlStylesheet) annotation;
            m.setProperty(get(MARSHALLER_XMLHEADERS), replaceToolbar(get(xsl.value())));
        }
        Reader reader = new Reader(new File(source));
        Overlay overlay = reader.read();
        m.marshal(overlay, bufferedWriters.get(WRITER_XUL_CHROME));
        m.marshal(overlay, bufferedWriters.get(WRITER_XUL_CONTENT));
    }

    /**
     * Método main que ejecuta la aplicación.
     * Debe recibir 2 argumentos:
     * <b>1</b>: archivo de entrada de datos del usuario
     * <b>2</b>: nombre del toolbar a crear
     * @param args argumentos de línea de comandos
     */
    public static void main(String[] args) throws JAXBException,
            FileNotFoundException, IOException {
        if (args.length == 2) {
            TOOLBAR_NAME = args[1];
            initializeVariables();
            createFileStructure();
            startBufferWriters();
            writeChromeManifest();
            writeJavascript();
            writeRdf();
            writeXUL(args[0]);
            shutdownWriters();
            Packer packer = new Packer();
            packer.pack();
        } else {
            System.out.println(get(MOZFIRE_USAGE));
        }
    }
}
```

Clase Reader.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.input;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import org.polesino.mozfire.xul.MenuItem;
import org.polesino.mozfire.xul.Overlay;
import org.polesino.mozfire.xul.Script;
import org.polesino.mozfire.xul.Toolbar;
import org.polesino.mozfire.xul.ToolbarButton;
import org.polesino.mozfire.xul.Toolbox;
import proyectoTerminal.MozFire;

/**
 * Clase que sirve de interfaz para leer la entrada de datos del usuario. Lee de
 * un archivo <b>file</b> los detalles de la toolbar que será creada.
 *
 * @author polesino
 */
public class Reader {

    private static final String TOKEN_BUTTON = "button:";
    private static final String TOKEN_MENUITEM = "menuitem:";
    private static final String TOKEN_MENU = "menu:";
    private static final String SPLIT_REGEX = "\\|";
    private static final String REPLACE_IMG = "<IMG>";
    private static final String REPLACE_ID = "<ID>";
    private static final int BUFFER_SIZE = 2048;
    private File file;

    /**
     *
     * @param file Archivo de entrada de datos para la creación de la
     * <b>toolbar</b>.
     */
    public Reader(File file) {
        this.file = file;
    }

    /**
     * Lee el archivo de entrada que contiene los detalles de la toolbar
     * deseada. Parsea 3 diferentes tókens: MENU, MENUITEM & BUTTON.
     *
     * @return the overlay xml root element for the XUL file of the toolbar
     * @throws FileNotFoundException
     * @throws IOException
     */
    public Overlay read() throws FileNotFoundException, IOException {
        BufferedReader br = new BufferedReader(new FileReader(file));
        String line;
        Toolbar toolbar = new Toolbar(MozFire.TOOLBAR_NAME);
        Script script = new Script(MozFire.TOOLBAR_NAME);
        Toolbox toolbox = new Toolbox(toolbar);
        Overlay overlay = new Overlay(script, toolbox);
        ToolbarButton toolbarButton = null;
        while ((line = br.readLine()) != null) {
            if (line.length() > 0) {
                System.out.println("line [" + line + ']');
                if (line.startsWith(TOKEN_MENUITEM)) {
                    String menuItem = line.replaceFirst(TOKEN_MENUITEM, "");
                    String[] split = menuItem.split(SPLIT_REGEX);
                    toolbarButton.addMenuItem(new MenuItem(split[0], split[1]));
                    if (split.length > 2) {
                        setImageAndCss(split[0], split[2]);
                    }
                } else if (line.startsWith(TOKEN_MENU) || line.startsWith(TOKEN_BUTTON)) {
                    String buttonLine = line.replaceFirst(TOKEN_BUTTON, "").replace(TOKEN_MENU, "");
                    String[] split = buttonLine.split(SPLIT_REGEX);
                    if (line.startsWith(TOKEN_BUTTON)) {
                        toolbarButton = new ToolbarButton(split[0], split[1]);
                        if (split.length > 2) {
                            setImageAndCss(toolbarButton.getId(), split[2]);
                        }
                    } else {
                        toolbarButton = new ToolbarButton(split[0]);
                    }
                    toolbar.addToolbarButton(toolbarButton);
                } else {
                    System.out.println("Unrecognized line [" + line + ']');
                }
            }
        }
        return overlay;
```

```
    }

    /**
     * Establece los parámetros correspondientes a la imagen que acompañará al
     * MENUITEM. Escribe en el archivo css y chrome los valores asociados a la
     * imagen.
     *
     * @param id identifica al menuitem que corresponde la imagen
     * @param img Ruta origen del archivo de la imagen.
     * @throws IOException
     */
    private void setImageAndCss(String id, String img) throws IOException {
        File image = new File(img);
        if (image.exists()) {
            String content = MozFire.replaceToolbar(MozFire.CSS_CONTENT);
            content = content.replaceAll(REPLACE_IMG, image.getName()).replaceAll(REPLACE_ID, id);
            MozFire.write(MozFire.WRITER_CSS_CHROME, content);
            MozFire.write(MozFire.WRITER_CSS_SKIN, content);
            copyImage(image);
        }
    }

    /**
     * Copia una imagen en dos diferentes destinos: <b>skin</b> y
     * <b>chrome/skin</b>, para tener el archivo de la imagen que será usada
     * como ícono en la TOOLBAR
     *
     * @param image Archivo de la imagen que será utilizado como ícono
     * @throws FileNotFoundException
     * @throws IOException
     */
    private void copyImage(File image) throws FileNotFoundException, IOException {
        File destSkin = new File(MozFire.DIR_SKIN + image.getName());
        File destChrome = new File(MozFire.DIR_CHROME + MozFire.DIR_SKIN + image.getName());
        OutputStream outSkin;
        OutputStream outChrome;
        try (InputStream in = new FileInputStream(image)) {
            System.out.println("Copying " + image.getPath() + " to " + destSkin.getPath());
            outSkin = new FileOutputStream(destSkin);
            outChrome = new FileOutputStream(destChrome);
            byte[] buffer = new byte[BUFFER_SIZE];
            int lenght;
            while ((lenght = in.read(buffer)) > 0) {
                outSkin.write(buffer, 0, lenght);
                outChrome.write(buffer, 0, lenght);
            }
        }
        outSkin.close();
        outChrome.close();
    }
}
```

Interface Namespace

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.namespaces;

import proyectoTerminal.MozFire;

/**
 *
 * @author polesino
 */
public interface Namespace {

    /**
     * xmlns
     * <b>http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul</b>
     */
    public static final String URI_XUL =
            "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul";
    /**
     * xmlns <b>http://www.w3.org/1999/02/22-rdf-syntax-ns#</b>
     */
    public static final String URI_RDF = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
    /**
     * xmlns <b>http://www.mozilla.org/2004/em-rdf#</b>
     */
    public static final String URI_EM_RDF = "http://www.mozilla.org/2004/em-rdf#";
    /**
     * String vacío ""
     */
    public static final String PREFIX_EMPTY = "";
    /**
     * prefijo de namespace rdf: em
     */
    public static final String PREFIX_EM_RDF = "em";
    /**
     * Valor para src de tipo script
     */
    public static final String SCRIPT_SRC = MozFire.OVERLAY_SCRIPT_SRC;
    /**
     * Valor para type de tipo script
     */
    public static final String SCRIPT_TYPE = MozFire.OVERLAY_SCRIPT_TYPE;
    /**
     * Elemento xml para stylesheet de xpi
     */
    public static final String XML_STYLESHEET = "overlay.xml.stylesheet";
    /**
     * Elemento XML de tipo menu
     */
    public static final String TYPE_MENU = MozFire.TYPE_MENU;
    /**
     * Elemento xml de tipo button
     */
    public static final String TYPE_BUTTON = MozFire.TYPE_BUTTON;
    /**
     * Código javascript para abrir una URL
     */
    public static final String ONCOMMAND = "<TOOLBAR>_LoadURL('<URL>')";
}
```

Clase Cleaner.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.output;

import java.io.File;
import java.util.List;

/**
 *
 * @author polesino
 */
public class Cleaner {

    /**
     * Borra los archivos innecesarios en las carpetas.
     * @param files Lista de archivos a borrar
     */
    public static void delete(List<File> files) {
        for (File file : files) {
            String name = file.getName();
            if (file.delete()) {
                System.out.println('[' + name + "] DELETED");
            }
        }
    }
}
```

Clase Packer.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.output;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
import proyectoTerminal.MozFire;

/**
 *
 * @author bemobile
 */
public class Packer {

    private BufferedOutputStream outputStream;
    private List<File> files;
    private String jar;
    private String xpi;
    private static final String EXT_JAR = MozFire.get(MozFire.EXT_JAR, ".jar");
    private static final String EXT_XPI = MozFire.get(MozFire.EXT_XPI, ".xpi");
    private static final int BUFFER = 2048;

    public Packer() {
        this.jar = MozFire.DIR_CHROME + MozFire.TOOLBAR_NAME + EXT_JAR;
        this.xpi = MozFire.TOOLBAR_NAME + EXT_XPI;
    }

    /**
     * Empaqueta las carpetas en los archivos jar y crea el xpi para instalación
     * @throws FileNotFoundException
     */
    public void pack() throws FileNotFoundException {
        createJar();
        createXpi();
    }

    /**
     * Empaqueta los archivos y crea el archivo comprimido jar
     * @throws FileNotFoundException
     */
    private void createJar() throws FileNotFoundException {
        files = new ArrayList<>();
        File contentDir = new File(MozFire.DIR_CONTENT);
        File skinDir = new File(MozFire.DIR_SKIN);
        try {
            outputStream = new BufferedOutputStream(new FileOutputStream(jar));
            files.addAll(Arrays.asList(contentDir.listFiles()));
            files.add(contentDir);
            files.addAll(Arrays.asList(skinDir.listFiles()));
            files.add(skinDir);
            zip(files);
        } catch (IOException ex) {
            ex.printStackTrace(System.out);
        }
        closeOutput();
        Cleaner.delete(files);
    }

    /**
     * Empaqueta los archivos y crea el archivo comprimido XPI
     * @throws FileNotFoundException
     */
    private void createXpi() throws FileNotFoundException {
        files = new ArrayList<>();
        File contentDir = new File(MozFire.DIR_CHROME + MozFire.DIR_CONTENT);
        File skinDir = new File(MozFire.DIR_CHROME + MozFire.DIR_SKIN);
        File chromeDir = new File(MozFire.DIR_CHROME);
        try {
            outputStream = new BufferedOutputStream(new FileOutputStream(xpi));
            files.addAll(Arrays.asList(contentDir.listFiles()));
            files.addAll(Arrays.asList(skinDir.listFiles()));
            files.addAll(Arrays.asList(chromeDir.listFiles()));
            files.add(chromeDir);
            files.add(new File(MozFire.FILE_CHROMEMANIFEST));
            files.add(new File(MozFire.FILE_RDF));
            zip(files);
        } catch (IOException ex) {
            ex.printStackTrace(System.out);
        } finally {
            closeOutput();
        }
```

```
        Cleaner.delete(files);
    }

    /**
     * Empaqueta los archivos contenidos en una carpeta. Ocupa el OutputStream de la instancia.
     * @param files Lista de archivos para ser comprimidos
     * @throws FileNotFoundException
     * @throws IOException
     */
    private void zip(List<File> files) throws FileNotFoundException, IOException {
        try (ZipOutputStream out = new ZipOutputStream(outputStream)) {
            out.setMethod(ZipOutputStream.DEFLATED);
            BufferedInputStream bis;
            for (File file : files) {
                byte buffer[] = new byte[BUFFER];
                if (file.isFile()) {
                    System.out.println("Compressing [" + file + "]");
                    bis = new BufferedInputStream(new FileInputStream(file), BUFFER);
                    ZipEntry entry = new ZipEntry(file.getPath());
                    out.putNextEntry(entry);
                    int count;
                    while ((count = bis.read(buffer, 0, BUFFER)) != -1) {
                        out.write(buffer, 0, count);
                    }
                    bis.close();
                }
            }
        }
    }

    /**
     * Cierra el OutputStream de la instancia.
     */
    private void closeOutput() {
        if (outputStream != null) {
            try {
                outputStream.close();
                outputStream = null;
            } catch (IOException ex) {
                ex.printStackTrace(System.out);
            }
        }
    }
}
```

Clase Description.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.rdf;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase POJO para el elemento XML Description
 * @author polesino
 */
@XmlType(propOrder = {"id", "name", "version", "targetApplication", "creator",
    "description", "homepageUrl", "minVersion", "maxVersion"})
public class Description implements Namespace {

    private String about;
    private String id;
    private String name;
    private String version;
    private TargetApplication targetApplication;
    private String minVersion;
    private String maxVersion;
    private String creator;
    private String description;
    private String homepageUrl;

    /**
     * Constructor por default
     */
    public Description() {
    }

    /**
     * Constructor de Description
     * @param about Nodo XML about
     * @param id Nodo XML id
     * @param name Nodo XML name
     * @param version Nodo XML version
     */
    public Description(String about, String id, String name, String version) {
        this.about = about;
        this.id = id;
        this.name = name;

        this.version = version;
    }

    /**
     * Constructor
     * @param id Nodo XML id
     * @param minVersion Nodo XML minVersion
     * @param maxVersion Nodo XML maxVersion
     */
    public Description(String id, String minVersion, String maxVersion) {
        this.id = id;
        this.minVersion = minVersion;
        this.maxVersion = maxVersion;
    }

    @XmlAttribute
    public String getAbout() {
        return about;
    }

    public void setAbout(String about) {
        this.about = about;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getVersion() {
        return version;
```

```
    }

    public void setVersion(String version) {
        this.version = version;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public TargetApplication getTargetApplication() {
        return targetApplication;
    }

    public void setTargetApplication(TargetApplication targetApplication) {
        this.targetApplication = targetApplication;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getCreator() {
        return creator;
    }

    public void setCreator(String creator) {
        this.creator = creator;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    @XmlElement(name = "homepageURL", namespace = URI_EM_RDF)
    public String getHomepageUrl() {
        return homepageUrl;
    }

    public void setHomepageUrl(String homepageUrl) {
        this.homepageUrl = homepageUrl;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getMinVersion() {
        return minVersion;
    }

    public void setMinVersion(String minVersion) {
        this.minVersion = minVersion;
    }

    @XmlElement(namespace = URI_EM_RDF)
    public String getMaxVersion() {
        return maxVersion;
    }

    public void setMaxVersion(String maxVersion) {
        this.maxVersion = maxVersion;
    }
}
```

Clase Rdf.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.rdf;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase para el Root XML: RDF
 * @author polesino
 */
@XmlRootElement(name = "RDF", namespace = Namespace.URI_RDF)
public class Rdf implements Namespace {

    private Description description;

    public Rdf() {
    }

    @XmlElement(name = "Description", namespace = URI_RDF)
    public Description getDescription() {
        return description;
    }

    public void setDescription(Description description) {
        this.description = description;
    }
}
```

Clase TargetApplication.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.rdf;

import javax.xml.bind.annotation.XmlElement;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase POJO para el elemento XML TargetApplication
 * @author polesino
 */
public class TargetApplication implements Namespace {

    private Description description;

    public TargetApplication() {
    }

    public TargetApplication(Description description) {
        this.description = description;
    }

    @XmlElement(name = "Description", namespace = URI_RDF)
    public Description getDescription() {
        return description;
    }

    public void setDescription(Description description) {
        this.description = description;
    }
}
```

package-info.java org.polesino.mozfire.rdf

```
/**
 * package-info para declarar los Namespaces a utilizar en el paquete
 * <b>org.polesino.mozfire.rdf</b>
 */
@XmlSchema(namespace = Namespace.URI_RDF,
elementFormDefault = XmlNsForm.QUALIFIED,
xmlns = {
    @XmlNs(namespaceURI = Namespace.URI_RDF,
    prefix = Namespace.PREFIX_EMPTY),
    @XmlNs(namespaceURI = Namespace.URI_EM_RDF,
    prefix = Namespace.PREFIX_EM_RDF)})
package org.polesino.mozfire.rdf;

import javax.xml.bind.annotation.XmlNs;
import javax.xml.bind.annotation.XmlNsForm;
import javax.xml.bind.annotation.XmlSchema;
import org.polesino.mozfire.namespaces.Namespace;
```

Anotación de Interface XmlStylesheet.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xml.annotation;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * <b>interface</b> para la anotación XmlStylesheet.
 * Aplicable sólo a las clases.
 * @author polesino
 */
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(value = ElementType.TYPE)
public @interface XmlStylesheet {

    /**
     *
     * @return Valor del XmlStylesheet
     */
    public String value();
}
```

package-info.java org.polesino.mozfire.xul

```
/**
 * package-info para declarar los Namespaces a utilizar en el paquete
 * <b>org.polesino.mozfire.xul</b>
 */
@XmlSchema(
        namespace = Namespace.URI_XUL,
elementFormDefault = XmlNsForm.QUALIFIED,
xmlns = {
    @XmlNs(namespaceURI = Namespace.URI_XUL,
    prefix = Namespace.PREFIX_EMPTY),})
package org.polesino.mozfire.xul;

import javax.xml.bind.annotation.XmlNs;
import javax.xml.bind.annotation.XmlNsForm;
import javax.xml.bind.annotation.XmlSchema;
import org.polesino.mozfire.namespaces.Namespace;
```

Clase MenuItem.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

/**
 * Clase POJO para el elemento XML menuitem
 * @author polesino
 */
public class MenuItem extends Widget {

    public MenuItem() {
    }

    /**
     * Constructor
     * @param label etiqueta del menuitem
     * @param url URL del menuitem
     */
    public MenuItem(String label, String url) {
        this.label = label;
        this.oncommand = url;
    }
}
```

Clase Overlay.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import org.polesino.mozfire.namespaces.Namespace;
import org.polesino.mozfire.xml.annotation.XmlStylesheet;

/**
 * Clase POJO para Root XML:  Overlay
 * @author polesino
 */
@XmlRootElement(namespace = Namespace.URI_XUL)
@XmlStylesheet(value = Namespace.XML_STYLESHEET)
public class Overlay implements Namespace {

    private Script script;
    private Toolbox toolbox;

    /**
     * Constructor default
     */
    public Overlay() {
    }

    /**
     * Constructor
     * @param script Elemento XML script
     * @param toolbox Elemento XML toolbox
     */
    public Overlay(Script script, Toolbox toolbox) {
        this.script = script;
        this.toolbox = toolbox;
    }

    @XmlElement(namespace = URI_XUL)
    public Script getScript() {
        return script;
    }

    @XmlElement(namespace = URI_XUL)
    public Toolbox getToolbox() {
        return toolbox;
    }
}
```

Clase Script.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase POJO para el elemento XML script
 * @author polesino
 */
@XmlType(propOrder = {"type", "src"})
public class Script implements Namespace {

    private String src;

    public Script() {
    }

    public Script(String src) {
        this.src = src;
    }

    @XmlAttribute()
    public String getType() {
        return SCRIPT_TYPE;
    }

    @XmlAttribute()
    public String getSrc() {
        return SCRIPT_SRC.replace("<TOOLBAR>", src);
    }
}
```

Clase Toolbar.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase POJO para el elemento XML toolbar
 * @author polesino
 */
@XmlType(propOrder = {"id", "accesskey", "chromeClass", "context", "toolbarname",
    "hidden", "persist", "toolbarButtons"})
public class Toolbar implements Namespace {

    private List<ToolbarButton> toolbarButtons;
    private String id;

    /**
     * Constructor default
     */
    public Toolbar() {
    }

    /**
     * Constructor
     * @param id attributo id
     */
    public Toolbar(String id) {
        this.id = id;
    }

    @XmlAttribute
    public String getId() {
        return id;
    }

    @XmlAttribute
    public String getAccesskey() {
        return id.substring(0, 1);
    }

    @XmlAttribute(name = "class")
    public String getChromeClass() {
        return "chromeclass-toolbar";
    }

    @XmlAttribute()
    public String getContext() {
        return "toolbar-context-menu";
    }

    @XmlAttribute()
    public String getToolbarname() {
        return id;
    }

    @XmlAttribute()
    public boolean isHidden() {
        return false;
    }

    @XmlAttribute
    public String getPersist() {
        return "hidden";
    }

    @XmlElement(name = "toolbarbutton")
    public List<ToolbarButton> getToolbarButtons() {
        return toolbarButtons;
    }

    public void addToolbarButton(ToolbarButton toolbarButton) {
        if (toolbarButtons == null) {
            toolbarButtons = new ArrayList<>();
        }
        toolbarButtons.add(toolbarButton);
    }
}
```

Clase ToolbarButton.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;

/**
 * Clase POJO para elemento XML toolbarbutton
 *
 * @author polesino
 */
public class ToolbarButton extends Widget {

    private List<MenuItem> menupopup;

    /**
     * Constructor default
     */
    public ToolbarButton() {
    }

    /**
     * Constructor
     *
     * @param label attribute label
     */
    public ToolbarButton(String label) {
        this.label = label;
    }

    /**
     * Constructor
     *
     * @param label attribute label
     * @param url attribute url
     */
    public ToolbarButton(String label, String url) {
        this.label = label;
        this.oncommand = url;
    }

    @XmlAttribute
    public String getId() {
        return menupopup == null || menupopup.size() < 1
                ? TYPE_BUTTON + '_' + label : TYPE_MENU + '_' + label;
    }

    @XmlAttribute
    public String getType() {
        return menupopup == null || menupopup.size() < 1 ? null : TYPE_MENU;
    }

    @XmlElementWrapper(namespace = URI_XUL)
    @XmlElement(name = "menuitem")
    public List<MenuItem> getMenupopup() {
        return menupopup;
    }

    public void addMenuItem(MenuItem menuItem) {
        if (menupopup == null) {
            menupopup = new ArrayList<>();
        }
        menupopup.add(menuItem);
    }
}
```

Clase Toolbox.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import javax.xml.bind.annotation.XmlElement;
import org.polesino.mozfire.namespaces.Namespace;

/**
 * Clase POJO para elemento XML toolbox
 * @author polesino
 */
public class Toolbox implements Namespace {

    private Toolbar toolbar;

    /**
     * Constructor default
     */
    public Toolbox() {
    }

    /**
     * Constructor
     * @param toolbar elemento xml toolbar
     */
    public Toolbox(Toolbar toolbar) {
        this.toolbar = toolbar;
    }

    @XmlElement(namespace = URI_XUL)
    public Toolbar getToolbar() {
        return toolbar;
    }
}
```

Clase Widget.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.polesino.mozfire.xul;

import javax.xml.bind.annotation.XmlAttribute;
import org.polesino.mozfire.namespaces.Namespace;
import proyectoTerminal.MozFire;

/**
 * Clase POJO para nodos xml
 * @author polesino
 */
public class Widget implements Namespace {

    protected String label;
    protected String oncommand;

    /**
     * Constructor
     */
    public Widget() {
    }

    @XmlAttribute()
    public String getOncommand() {
        if (oncommand != null) {
            return ONCOMMAND.replace("<TOOLBAR>", MozFire.TOOLBAR_NAME).
                    replace("<URL>", oncommand);
        }
        return oncommand;
    }

    @XmlAttribute()
    public String getLabel() {
        return label;
    }
}
```

Diagramas de Modelo de Clases

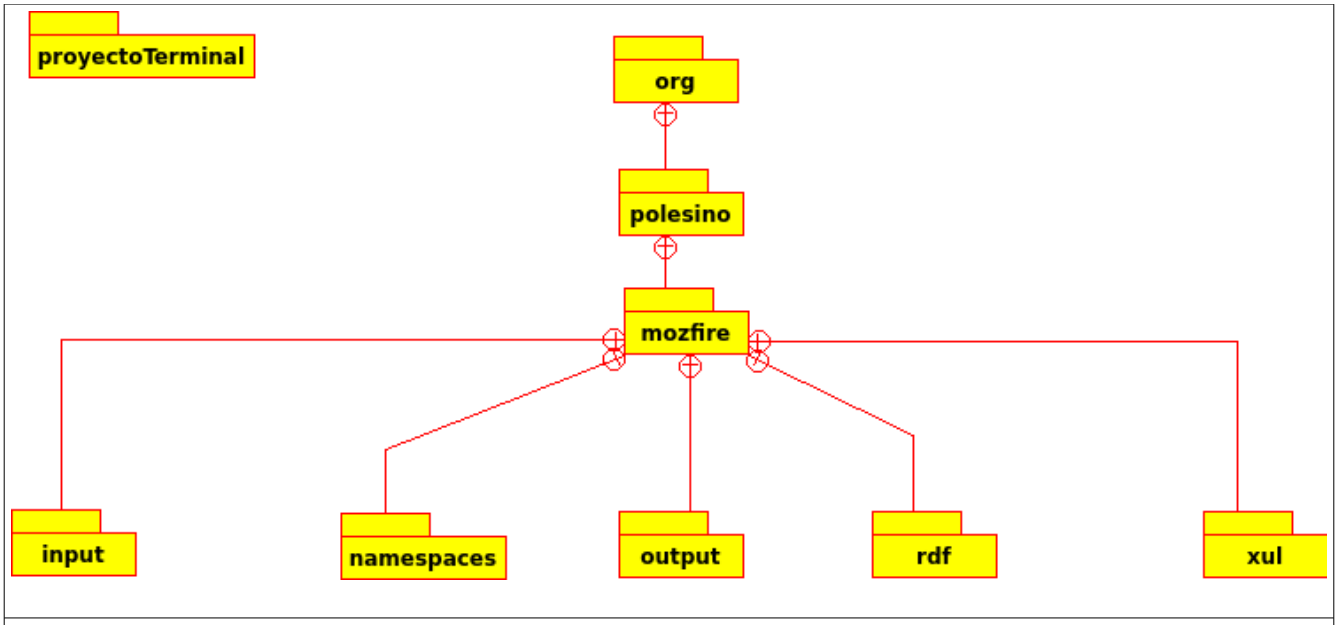

Diagrama de paquetes

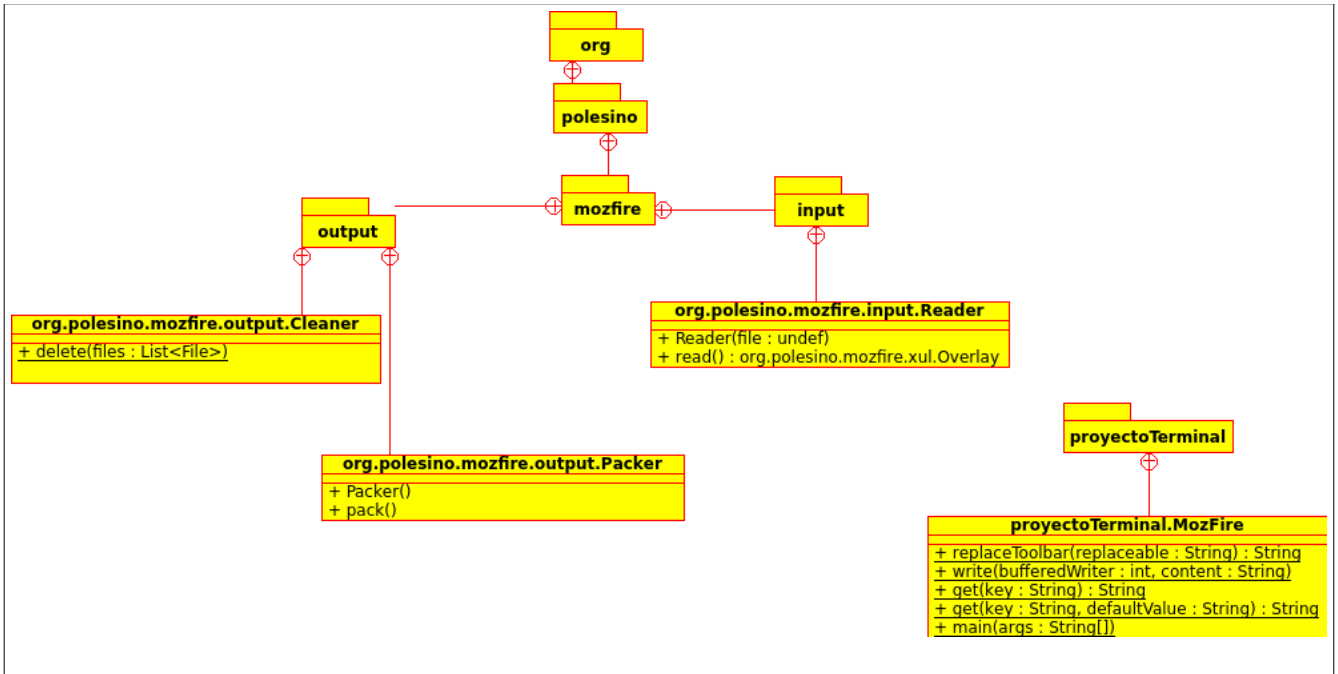

Diagrama de clases:

- org.polesino.mozfire.input
  - Reader
- org.polesino.mozfire.output
  - Cleaner
  - Packer
- proyectoTerminal
  - MozFire (Clase principal)

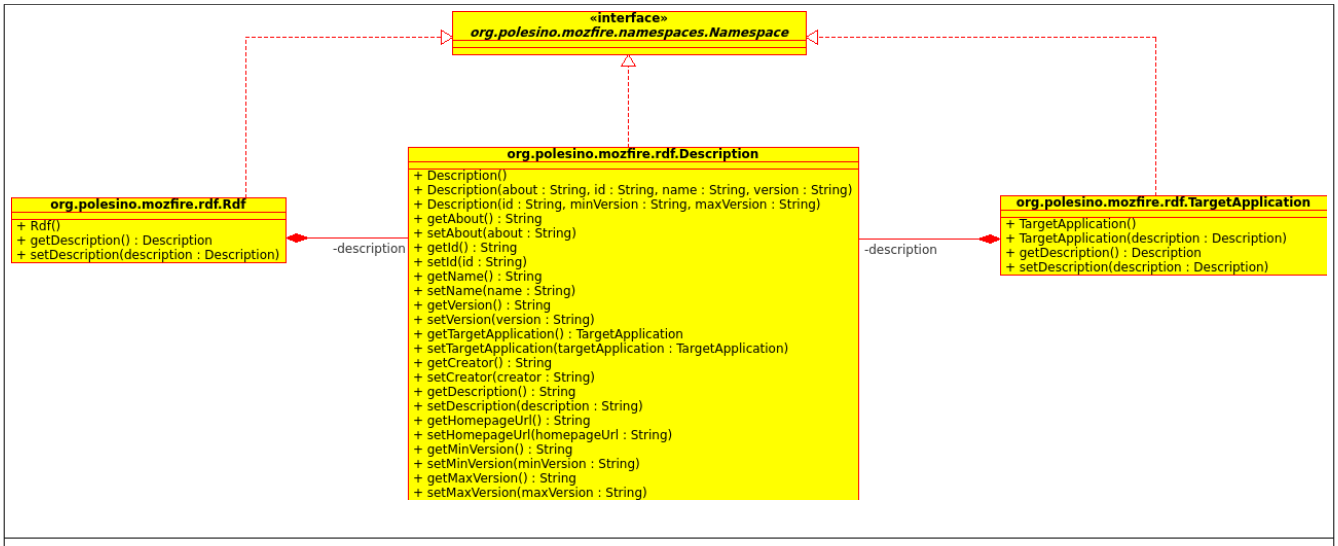

Diagrama de clases:

- org.polesino.mozfire.rdf
  - Rdf
  - Description
  - TargetApplication
- org.polesino.mozfire.namespaces
  - Namespace

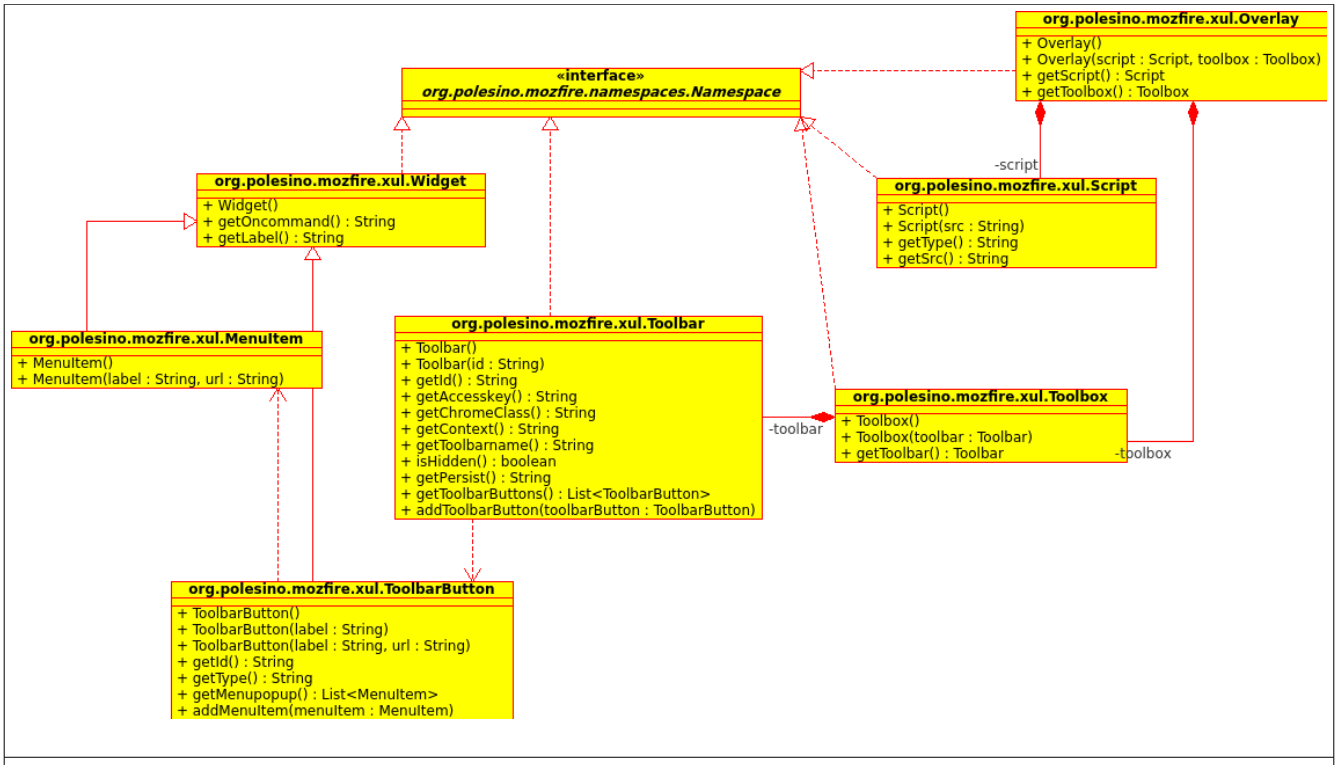

Diagrama de clases:

- org.polesino.mozfire.xul
  - MenuItem
  - Overlay
  - Script
  - Toolbar
  - ToolbarButton
  - Toolbox
  - Widget
- org.polesino.mozfire.namespaces
  - Namespace

Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Manual de usuario

Herramienta generadora de código XUL para la implementación de componentes software tipo *toolbar* para el web browser Mozilla Firefox

Jorge Armando Dotor Vázquez

Matrícula: 203301481

Trimestre 13-I

4 de enero 2013

Dr. Oscar Herrera Alcántara

Número económico: 24709

ÍNDICE DE CONTENIDOS

OBJETIVOS

- Mostrar la forma de uso para que el usuario final pueda crear toolbar a partir de un archivo de entrada de datos
- Describir el funcionamiento básico de la aplicación
- Explicar la estructura del archivo de entrada de datos

DESCRIPCIÓN

La aplicación generadora de código XUL para la implementación de complementos de tipo TOOLBAR, permite que el usuario final pueda crear, a partir de un archivo de entrada de datos, un archivo XPI que contenga la estructura necesaria para instalar un TOOLBAR en el web browser Mozilla Firefox de una forma transparente y sencilla para el usuario final.

Se requiere que la computadora que va a ejecutar la aplicación contenga:

- Java Virtual Machine versión 6 o superior
- WebBrowser Mozilla Firefox versión 3 o superior

El usuario debe crear una archivo txt que contenga los valores del toolbar separados con | de la siguiente forma:

```
button:google|http://www.google.com.mx
menu:uam|logo_uam.jpg
menuitem:cosei|http://espartaco.azc.uam.mx/
```

Los elementos que puede ocupar el usuario son:

- button
  - Es un botón en la barra de herramientas(toolbar). No utiliza imagen como ícono.
  - Utiliza dos elementos que son separados por |
    - etiqueta del botón
    - URL que será utilizada como link al accionar el botón.
- menu
  - Es un botón que despliega un menú(menupopup) que contendrá distintos elementos internos(menuitem). Puede tener una imagen que será utilizada como ícono.
  - Puede utilizar hasta 2 elementos que son separados por |
    - etiqueta del menú
    - ruta del archivo de imagen que se utilizará como ícono. Esta ruta puede ser absoluta o relativa, pero se debe tener en cuenta, que si es relativa, debe serlo en referencia a la ejecución de la aplicación. Se aconseja que se cree una carpeta que contenga las imágenes a utilizar y que el archivo tenga la ruta relativa en el mismo nivel de carpeta que el archivo jar ejecutable.
- menuitem
  - Es el elemento interno de un menu. No contiene ícono
  - Utiliza dos elementos que son separados por |
    - etiqueta del men uitem
    - URL que será utilizada como link al accionar el menuitem.

Se aconseja que el archivo ejecutable jar tenga la siguiente estructura de carpeta

para poder acceder a los archivos de configuración e imágenes de forma sencilla:

```
.
├── conf
│   └── mozfire.properties
├── img
│   ├── facebook.png
│   ├── mail.gif
│   ├── photo.gif
│   └── poke.gif
├── menu.txt
└── proyecto_terminal.jar
```

El archivo mozfire.properties es utilizado como archivo de configuración para la ejecución de la aplicación. Contiene los valores por default y variables que la aplicación utiliza. Siempre debe estar en la carpeta conf/ en el mismo nivel que el ejecutable jar.

El archivo de entrada de datos puede estar en cualquier ruta, pero se aconseja que esté en la misma carpeta que la aplicación. Este es ejemplo de un archivo de entrada:

```
menu:uam
menuitem:sae|http://www.azc.uam.mx/sae
menuitem:cosei|http://www.cosei.azc.uam.mx
menuitem:sai|http://www.sai.azc.uam.mx
button:poke|http://www.yahoo.com.mx|img/poke.gif
button:cinemex|http://www.cinemex.com.mx|img/mail.gif
menu:uam
menuitem:cosei|http://www.cosei.azc.uam.mx
menuitem:sai|http://www.sai.azc.uam.mx
menuitem:sai|http://www.sai.azc.uam.mx
menuitem:sai|http://www.sai.azc.uam.mx
menu:facebook
menuitem:facebook|http://www.facebook.com|img/facebook.png
button:google|http://www.google.com.mx|img/photo.gif
```

La ejecución de la aplicación se realiza desde línea de comandos siguiendo la siguiente sintaxis:

$ java -jar proyecto_terminal.jar menu.txt menu

Donde menu.txt es el archivo de entrada de datos y menu es el nombre del toolbar que será creado.

La salida en la terminal es la siguiente:

Creating [chrome/]
directory created: /home/polesino/Dropbox/uam/proyecto terminal/pt/chrome
Creating [content/]
directory created: /home/polesino/Dropbox/uam/proyecto terminal/pt/content
Creating [skin/]
directory created: /home/polesino/Dropbox/uam/proyecto terminal/pt/skin
Creating [chrome/content/]
directory created: /home/polesino/Dropbox/uam/proyecto terminal/pt/chrome/content
Creating [chrome/skin/]
directory created: /home/polesino/Dropbox/uam/proyecto terminal/pt/chrome/skin
line [menu:uam]
line [menuitem:sae|http://www.azc.uam.mx/sae]
line [menuitem:cosei|http://www.cosei.azc.uam.mx]
line [menuitem:sai|http://www.sai.azc.uam.mx]
line [button:poke|http://www.yahoo.com.mx|img/poke.gif]
Copying img/poke.gif to skin/poke.gif
line [button:cinemex|http://www.cinemex.com.mx|img/mail.gif]
Copying img/mail.gif to skin/mail.gif
line [menu:uam]
line [menuitem:cosei|http://www.cosei.azc.uam.mx]
line [menuitem:sai|http://www.sai.azc.uam.mx]
line [menuitem:sai|http://www.sai.azc.uam.mx]
line [menuitem:sai|http://www.sai.azc.uam.mx]
line [menu:facebook]
line [menuitem:facebook|http://www.facebook.com|img/facebook.png]
Copying img/facebook.png to skin/facebook.png
line [button:lore|http://www.google.com.mx|img/photo.gif]
Copying img/photo.gif to skin/photo.gif
Compressing [content/menu.js]
Compressing [content/menu.xul]
Compressing [skin/photo.gif]
Compressing [skin/menu.css]
Compressing [skin/facebook.png]
Compressing [skin/mail.gif]
Compressing [skin/poke.gif]
Compressing [chrome/content/menu.js]
Compressing [chrome/content/menu.xul]
Compressing [chrome/skin/photo.gif]
Compressing [chrome/skin/menu.css]
Compressing [chrome/skin/facebook.png]
Compressing [chrome/skin/mail.gif]
Compressing [chrome/skin/poke.gif]
Compressing [chrome/menu.jar]
Compressing [chrome.manifest]
Compressing [install.rdf]

Al terminar se creará un archivo XPI:

```
.
├── conf
│   └── mozfire.properties
├── img
│   ├── facebook.png
│   ├── mail.gif
│   ├── photo.gif
│   └── poke.gif
├── menu.txt
├── menu.xpi
└── proyecto_terminal.jar
```

El cual se ocupará como archivo de instalación para Mozilla Firefox:

- Desde el Web Browser, seleccionar Abrir, y escoger la ruta del archivo XPI
- Aparecerá la advertencia de instalación
- Aceptar la instalación y reiniciar el web browser.

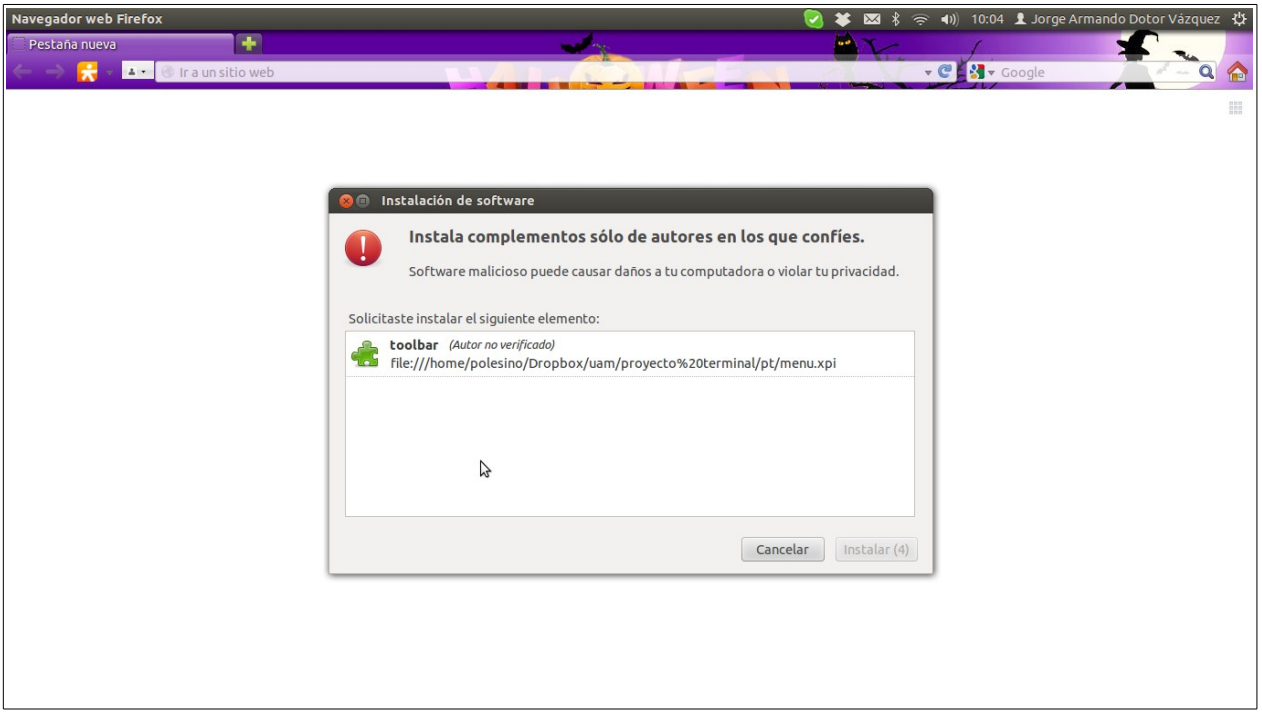

Imagen de Instalación del archivo xpi.

Imagen con la Toolbar instalada y después de accionar el link del menu facebook.