Proyecto fin de carrera
Ingeniería de Telecomunicación

Servicio de interfaz de usuario sobre el framework de OSGi

Autor: Emilio José Maldonado García.
Tutor: Juan M. Vozmediano Torres.

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein.

AGRADECIMIENTOS:

A mis padres y hermana por apoyarme siempre sin condiciones.
A mi familia por estar siempre a mi lado.
A Eli por todo lo que hemos pasado juntos.
A mis amigos, por vivir conmigo "la otra cara de la universidad".
Y… a todos aquellos que pasaron por mi vida durante estos años
y que en cierta manera forman parte de mí.

A todos ellos… Gracias.

Índice

1) INTRODUCCIÓN 7
1.Introducción a OSGi 7
1.1.OSGi como sistema de actualización de software 8
1.2.El Framework de OSGi. El concepto de Bundle 9
1.2.1. Introducción 9
1.2.2. El Framework 9
1.2.3. El Bundle 11
1.2.4. Dependencias de paquetes 12
1.2.5. Uso de librerías nativas 13
1.2.6. Iniciando un bundle 13
1.2.7. El concepto de Dinamismo 13

1.3.El Framework de OSCAR como implementación de OSGi 14
2.XUL: XML User Interface Lenguaje 15
2.1.Introducción a XUL 15
2.2.Estructura de una aplicación XUL 16
2.3.Descripción de una UI usando XUL 16
2.4.Otros lenguajes “like XUL” basados en Java 18

3.Objetivos del proyecto 19

2) IMPLEMENTACIÓN DEL SERVICIO INTERFAZ DE USUARIO 20
1. Introducción al servicio de interfaz de usuario 21
2. Características del servicio de interfaz de usuario 22
2.1. La interfaz con el programador 22
2.2. La interfaz con el usuario 23
3. El entorno de desarrollo 23
4. Esquema general de funcionamiento del servicio 24
4.1. El lenguaje de descripción de la interfaz de usuario 25
4.1.1. Introducción 25
4.1.2. Descripción de etiquetas “LXUL” 26
4.2. El concepto de Widget 32
4.2.1. Interfaz Java de un Widget 32
4.3. El Parseador de XUL 35
4.4. El Motor Gráfico 37

4.5. El Sistema de registro de Widgets 39
4.6. Los manejadores de eventos 39
5. El Concepto de "Vista" 41
6. Implementación de la interfaz Java del servicio 41
6.1. Vista gráfica tipo Swing 41
6.2. Vista gráfica tipo Texto 47
7. Generación del Bundle "Servicio de interfaz de usuario" 50

3) CONCLUSIONES 54
1. Conclusiones finales 55
2. Posibles Mejoras y Ampliaciones 58

ANEXOS 59

1. Instalación del Framework de OSCAR 60
2. Elección de JDOM como parseador de XUL 62
3. El paquete CHARVA 64
4. ¿Cómo añadir un nuevo widget a una vista existente? 66
5. Diagrama de clases 70

Capítulo 1

Introducción

1. Introducción a OSGi

El Open Service Gateway Initiative **(OSGi)** es una especificación para el desarrollo de servicios en las pasarelas de servicios.

El Open Service Gateway Initiative (OSGi) es un grupo de trabajo que surgió en Marzo de 1999, cuyo principal impulsor es Sun MicroSystem, y entre ellos se encuentran algunos socios españoles como Unión FENOSA y Telefónica I+D.

El objetivo es definir y promover un estándar abierto para permitir conectar a los servicios ofrecidos en redes metropolitanas (WAN) a redes locales (LAN) o domóticas (LON). Esto permitirá la conexión de la próxima generación de aparatos inteligentes que se puedan encontrar en un hogar (oficina) con los servicios externos a la casa (oficina) ofrecidos a través de Internet. De esta forma, Proveedores de Internet (ISP), operadores de red y fabricantes de equipos pueden ofrecer una amplia gama de servicios a los usuarios finales utilizando una pasarela común a todos.

Las áreas principales en las que se vuelcan todos los esfuerzos de OSGi son:

- **Servicios:** Se pretende crear una plataforma que sea capaz de procesar y tratar de forma correcta toda la información necesaria para proporcionar servicios de comunicaciones, de entretenimiento, de telecontrol o teledomótica, y de seguridad. Por lo tanto, la especificación OSGi debe tener los interfaces adecuados para soportar todos estos servicios sin incompatibilidades además de permitir gestionarlos de forma adecuada.

- **Métodos de acceso:** La idea es que la pasarela OSGi sea capaz de acceder al mundo exterior (redes de tipo Internet) usando cualquiera de las tecnologías disponibles actualmente. La tendencia actual es llevar a cabo esta tarea usando tecnologías de acceso de banda ancha con conexión permanente a Internet. Destacan el ADSL, el MODEM de cable, o tecnología inalámbricas como UMTS, LMDS. Esta tendencia hacia la banda ancha, es debido, además del aumento de la implantación, a la mejora sustancial en la *"Calidad de Servicio"*, parámetro muy de moda últimamente y que será un requisito necesario para el éxito del mercado de los e-servicios.

- **Redes de datos y control de las viviendas:** Teniendo en cuenta en la variedad de hogares y edificios en donde este tipo de pasarelas deben ser instaladas, esta iniciativa no escoge una única tecnología de conexión entre los múltiples electrodomésticos o dispositivos de las viviendas. Su objetivo es definir un interfaz común para todas ellas, dejando la responsabilidad a los fabricantes de construir los

controladores adecuados para cada una de ellas. Teniendo en cuenta esto, las pasarelas OSGi podrán usar tecnologías de conexión inalámbricas (IrDa, Homero, IEEE 802.11x, Bluetooh), sobre canales telefónicos (HomePNA), sobre la red de baja tensión (HomePlug, Lonworks, EIB/KNX , etc), sobre conexiones Ethernet, USB, etc, y protocolos como el HAVi, el VESA, el Jini, etc. Por lo tanto, la especificación OSGi será la "pasarela" que transforme paquetes de información procedentes del mundo exterior a un paquete de datos de cualquiera de estas tecnologías y viceversa.

1.1 OSGi como sistema de actualización de software:

El software obsoleto incluido en los equipos supone un problema creciente y plantea desafíos tanto a los consumidores, que demandan productos actualizados, como a los proveedores, fabricantes, desarrolladores de aplicaciones y proveedores de servicios. La OSGi Alliance tiene una solución y ya ofrece respuestas innovadoras a los desafíos que supone garantizar que los equipos de la anterior generación sean todavía capaces de responder a los estándares de la generación actual.

OSGi Alliance pretende diseñar una tecnología que mantendrá el software de un automóvil nuevo actualizado durante su ciclo de vida de entre 10 y 20 años.
Una simple descarga de software garantizará que el software integrado de un coche se mantenga actualizado, garantizando de este modo, por ejemplo, que la tecnología de hoy pueda utilizar de modo seguro los avances del futuro. Además, los propietarios podrán añadir nuevos programas al ordenador de a bordo y añadir las últimas versiones de software de diagnóstico y de gestión de maquinaria a lo largo de la vida útil del automóvil.

La meta de OSGi Alliance es que los consumidores puedan actualizar el software en los electrodomésticos actuales y sistemas domésticos de red.

En cuanto a los teléfonos móviles, OSGi Alliance impulsa la posibilidad de que los usuarios actualicen sus dispositivos más fácilmente en el futuro. La actualización de los móviles será cada vez más importante durante los próximos años para poder garantizar la vigencia de plataformas, sistemas operativos y aplicaciones adicionales, tales como juegos y multimedia.

1.2 El Framework de OSGi. El Concepto de Bundle

1.2.1 Introducción.

Para llevar a cabo todas las tareas especificadas en apartados anteriores, OSGi define un Framework por encima de Java, que permite a desarrolladores independientes el desarrollo de aplicaciones que pueden ser remotamente manejadas por ordenadores independientes con diferentes sistemas operativos, procesadores, tamaño y configuraciones.

1.2.2 El Framework:

El Framework realiza la gestión de las necesidades de las aplicaciones de una plataforma en continua ejecución que debe actuar ante los cambios dinámicos del entorno.

Un concepto clave de un Framework es el de **"servicio".** Un servicio es un objeto java que ha sido registrado en el Framework para ser usado por otras aplicaciones.

La funcionalidad de un servicio queda definida por la interfaz que implementa. Esto permite a las diferentes aplicaciones implementar el mismo tipo de "servicio". Por ejemplo un servicio de "registro de usuarios" puede estar optimizado para sistemas con almacenamiento local, o para sistemas de almacenamiento remoto. A pesar de ser servicios diferentes, de cara al usuario no debe haber diferencias apreciables.

El Framework de OSGi ha definido una serie servicios que pueden ser usados por las diferentes aplicaciones:

- **Log Service:** Proporciona un camino para registrar información de las distintas aplicaciones, así como un medio para recibirla cuando estas se producen.
- **Http Service:** Proporciona un servidor web que soporta servlets y ficheros estáticos.
- **Device Access:** Se trata de un modelo que permite la descarga automática de drivers de dispositivos, permitiendo realmente el "plug and play" de los dispositivos.
- **Configuration Management:** Este servicio permite la gestión de sistemas y aplicaciones a la vez que permite establecer parámetros de configuración de las mismas.

- **Preferentes:** Es una base de datos simple que soporta jerarquía de propiedades.

- **User Admin.** : Se trata de un repositorio de usuarios, incluyendo autenticaciones y autorizaciones de estos usuarios. Soporta diferentes sistemas de autenticación.

Las aplicaciones pueden usar además las librerías estándar de Java.

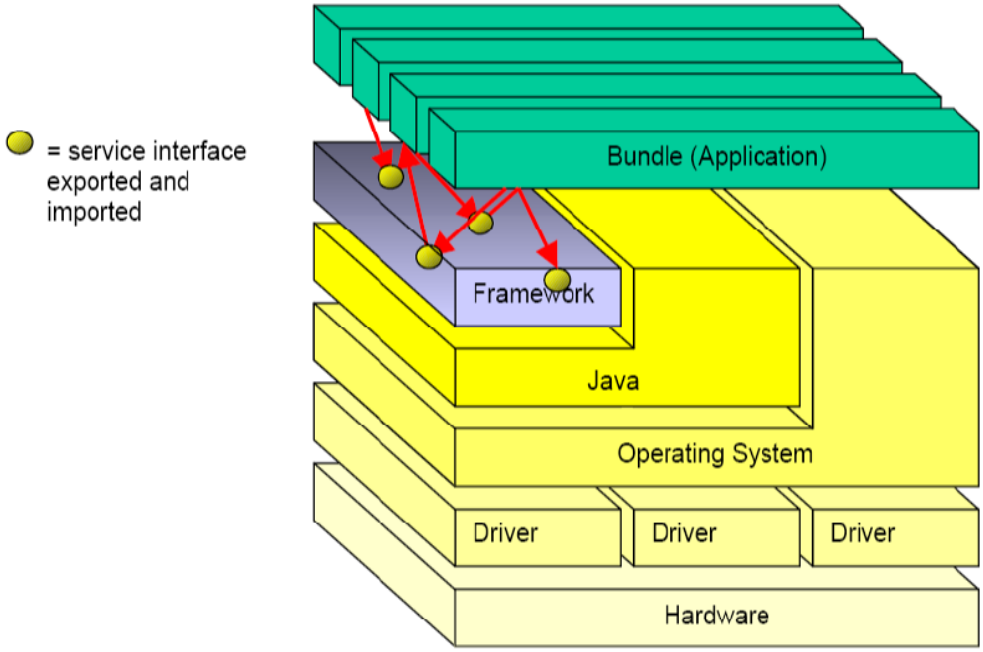

Figura 1. *La figura 1 muestra como están relacionadas las diferentes partes de un sistema basado en OSGi. Se puede observar como los bundles (Aplicaciones OSGi) pueden usar el Framework OSGi, las librerías estándar de Java, o acceder al sistema operativo directamente.*

El Framework es la parte que transforma una máquina virtual de Java, diseñada para ser un entorno para la ejecución de una sola aplicación, en un entorno para la ejecución de múltiples aplicaciones en una sola máquina virtual. Las ventajas de este hecho son muchas:

- Menos intercambio de procesos.
- Una intercomunicación entre aplicaciones más rápida.
- Una considerable reducción en el consumo de memoria.

1.2.3. El bundle:

El concepto de bundle se usa para representar a las múltiples aplicaciones que se ejecutan en la maquina virtual de java. Un bundle es un archivo Java Archive (JAR), que contiene "partes" que son necesarias para la ejecución de una determinada aplicación, y que contiene un archivo de manifiesto en el que declara cuales son las partes que deben estar disponibles cuando la aplicación comience. Estas partes pueden ser provistas por otros bundles o por el propio Framework.

Las dependencias son gestionadas por el Framework. Cuando un bundle comienza, registra y obtiene "servicios". Estos son objetos Java, definidos mediante una interfaz, que puede se encontrada en el servicio de registro del Framework. Los bundles pueden comunicarse únicamente a través de estos servicios.

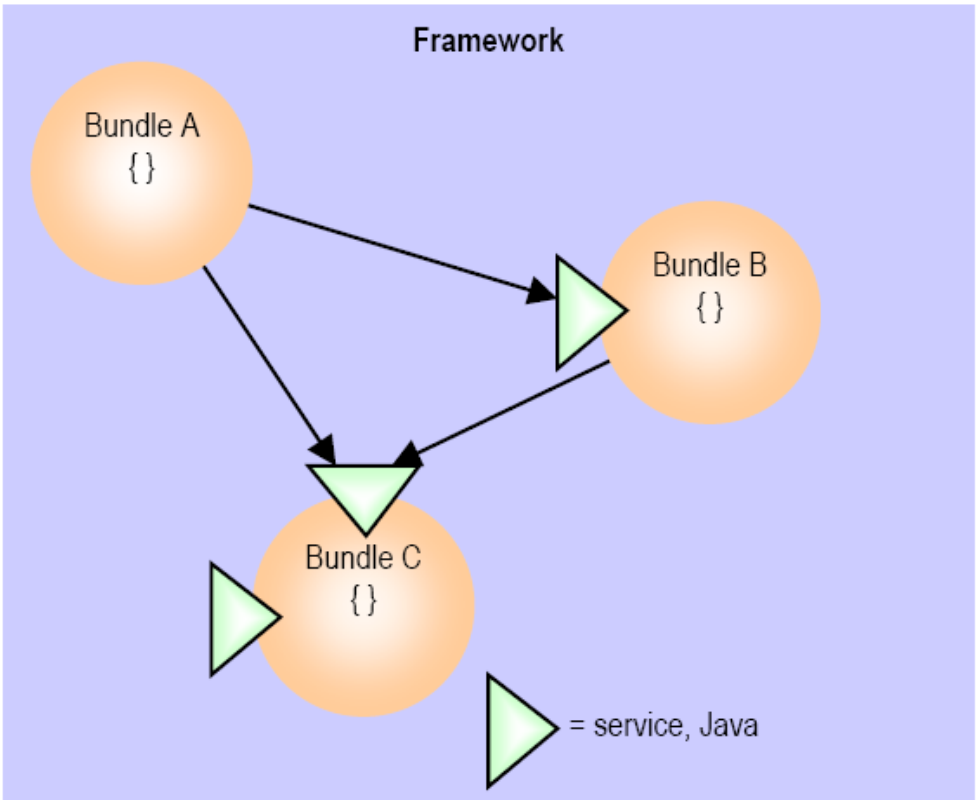

Figura 2: Bundles, servicios en el Framework de OSGi.

1.2.4 Dependencias de paquetes:

Una vez que un bundles es instalado necesita ser iniciado. Sin embargo los paquetes necesitan cumplir una serie de dependencias especificadas en el fichero JAR. Estas dependencias son paquetes que necesita el bundle para poder funcionar. Las clases Java están siempre contenidas en un paquete. El paquete se considera la primera parte del nombre de la clase (hasta el primer "."). Por ejemplo el paquete de la clase java.lang.String es java.lang.

Un bundle puede usar paquetes por tres vías diferentes. En primer lugar puede tener paquetes privados. Todas las clases que un bundle no necesita compartir con otro bundles serán privadas. Diferentes bundles pueden contener entonces el mismo paquete privado.

Las clases que necesitan ser compartidas con otros bundles tienen que ser exportadas. Las interfaces y las clases que son usadas para la comunicación deben ser exportada o si no los bundles que se ejecuten lanzarán un ClassCastExceptions cuando intercambien objetos. Cualquier bundle puede especificar algún paquete para exportar.

El Framework asegurará que sólo uno de los bundles en cada momento de tiempo está exportando un tipo específico de paquete.

Un bundle puede también especificar que necesita ciertos paquetes.

La siguiente figura muestra como los paquetes pueden ser importados desde el Framework, otros bundles o exportados. La resolución de estos paquetes está bajo el control del Framework y debe estar sujeto a verificaciones de seguridad.

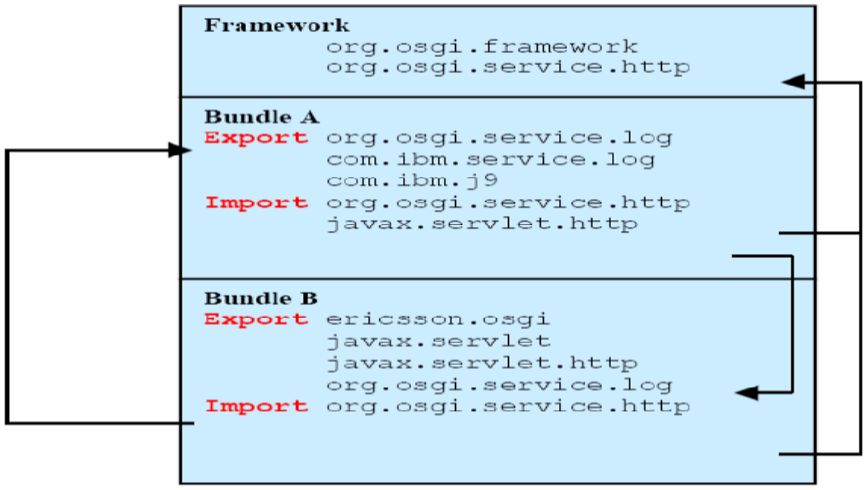

Figura 3. Compartiendo paquetes Java.

1.2.5 Librerías nativas.

Los ficheros JAR pueden contener también librerías nativas. Estas librerías no están restringidas a un hardware o sistema operativo, si no que se puede especificar en el manifiesto qué librería deber usarse para cada arquitectura. El Framework buscará el conjunto de librerías disponibles y cargará la mejor posible en cada caso.

1.2.6 Iniciando un bundle.

Tras ser resueltas todas las dependencias y ser seleccionados los bundles adecuados para exportar, el bundle necesita ser iniciado.

Por esta razón, el manifiesto contiene una cabecera que apunta a una clase especial: **la clase activator**. Un objeto de esta clase es instanciado y se le realiza un cast al tipo BundleActivator. Esta interfaz contiene los métodos **Start** y **Stop** que son usados para iniciar y parar el bundle respectivamente. Por razones de eficiencia, estos métodos deben devolver rápidamente el foco al sistema, ya que si no éste corre el riesgo de bloquearse. Normalmente la manera de usar el método Start es para ejecutar un hilo en background o para registrar algún servicio. El método stop se usa para borrar las partes del Framework que no pueden borrarse directamente y para detener algunos threads.

1.2.7 Dinamismo:

Una de las piezas fundamentales de la arquitectura OSGi es el dinamismo. Los bundles pueden ser instalados, actualizados, iniciados, detenidos o desinstalados en cualquier momento. Los dispositivos también pueden añadirse al Framework en cualquier momento. Esto hace que el entorno sea extremadamente dinámico.

El Framework ofrece un modelo comprensivo para escribir aplicaciones en tal entorno. Todos los cambios importantes producidos en el entorno, son enviados a los “notificadotes” que estén interesados en dicho evento.

La instalación, iniciación, parada, actualización o desinstalación de los bundles son notificadas como eventos de bundle.

1.3 El Framework de OSCAR como implementación de OSGi.

Oscar es una implementación "open source" de la especificación del Framework de OSGi. Su objetivo es proveer de una implementación completamente rigurosa de la especificación del framework de OSGi. Oscar sigue actualmente una gran parte de la especificación 3.0 de OSGi, aunque todavía no existe una implementación completa de la misma.

El framework de OSCAR ha sido diseñado para ser lo suficientemente ligero como para poder ser ejecutado en pequeños dispositivos como PDAs.

2. XUL: XML User Interface Languaje

2.1.Introducción a XUL.

XUL (XML User-interface Lenguaje) fue creado para realizar un desarrollo del navegador web Mozilla más fácil y rápido.
Es un lenguaje basado en XML, por lo que todas las características de este lenguaje están disponibles en XUL.

La mayor parte de las aplicaciones necesitan ser desarrollados usando características específicas de una plataforma, haciendo la construcción de software multiplataforma muy costoso en tiempo.

Muchas han sido las soluciones propuestas en el pasado para solucionar el problema del software multiplataforma. XUL es uno de estos lenguajes diseñado para construir interfaces de usuario portables.

Diseñar una interfaz de usuario es una labor que requiere una gran cantidad de tiempo, incluso para una sola plataforma. El tiempo requerido para compilarla y depurarla puede ser extremadamente largo.

Con XUL una interfaz puede ser implementada y modificada de forma fácil y sencilla.

XUL tiene todas las ventajas del lenguaje XML. Por ejemplo XHTML u otros lenguajes XML tales como MathML o SVG pueden ser insertados en su interior. Además XUL permite una fácil traducción de los textos contenidos en la interfaz de forma fácil y sencilla.

XUL permite aplicar “hojas de estilo” para modificar la apariencia de la interfaz.

XUL permite crear la mayor parte de los elementos encontrados en las interfaces gráficas modernas. Es lo suficientemente genérico para poder ser aplicado a las necesidades especiales de ciertos dispositivos, y es lo suficientemente potente para que los desarrolladores puedan diseñar sofisticadas interfaces con él.

2.2.Estructura de una aplicación XUL

Una interfaz descrita en XUL se estructura generalmente en tres directorios:

- **Content:** En este directorio se almacenan las ventanas de aplicación y los Scripts que se usan en ella.
  Este directorio suele contener varios archivos con extensión .xul, cada uno de ellos conteniendo la descripción de cada una de las ventanas que aparecen en la interfaz.
  Los archivos con extensión .js con los scripts de java que se lanzan en la ejecución de la interfaz.

- **Skin:** Las hojas de estilo, imágenes, y todo aquello relacionado con la apariencia de las ventanas de la interfaz se almacena en este directorio con el fin de facilitar la modificación de la apariencia de la aplicación.

- **Locales:** Aquí se almacenan todas las cadenas de texto que se muestran en la interfaz en diferentes idiomas. Esto facilita enormemente la tarea de generar interfaces gráficas que deben ser desplegadas en distintos idiomas.

2.3 Descripción de una UI usando XUL

En XUL la descripción de las interfaces se realizan usando etiquetas que representan a cada uno de los elementos que queremos que aparezcan en la interfaz. La descripción de una interfaz, puede hacerse usando tantos archivos .xul como se deseen, conteniendo cada uno de ellos cualquier parte de la interfaz.

Cada elemento de la interfaz (ventanas, botones, etiquetas) queda completamente definido por una etiqueta y una serie de atributos que se asocian a cada uno de los elementos.

Veamos un ejemplo sencillo de la descripción de una interfaz usando XUL.

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window
   id="findfile-window"
   title="Find Files"
   orient="horizontal"

  <button id="find-button" label="Find" default="true"/>
  <button id="cancel-button" label="Cancel"/>
</window>
```

Figura 4: Descripción de una UI usando XUL.

<?xml version="1.0"?>

Esta etiqueta declara que nos encontramos ante un archive XML en su versión 1.0

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

Esta etiqueta es la manera que tiene XUL de especificar el lugar donde se encuentran las hojas de estilo que deben ser aplicadas a la intefaz.

<window id="findfile-window" title="Find Files" orient="horizontal"
</window>

Esta etiqueta indica que queremos mostrar una ventana. Los atributos que aparecen en la ventana son: id = "findfile-window" es el identificador de la ventana; title = "Find Files" es el título que aparecerá en la ventana cuando ésta se muestre; orient = "horizontal" representa la orientación que debe tener la ventana.

<button id="find-button" label="Find" default="true"/>

Esta etiqueta añade un botón a la ventana anterior, en el que se leerá la etiqueta "Find".

Como vemos se realiza una descripción del contenido de la interfaz, y no de su apariencia. Esta descripción de la apariencia se realiza en archivos .css o "hojas de estilo", que junto a los archivos de descripción de la interfaz dan una vista global de la misma.

La respuesta a eventos de la interfaz se realiza usando scripts de java. Se usan etiquetas con nombres relativos al evento que se quiere manejar (onclick, onmouse) para especificar la ruta del archivo de script que va a manejar el evento al que esta asociado.

También podemos usar código script insertado directamente en el código xul como valor de la etiqueta del evento en cuestión.

<button id="cancel-button" label="Cancel"
oncommand="window.close();"/>

2.4. Otros lenguajes “like XUL” basados en Java

Son muchos los lenguajes basados en XML destinados a describir interfaces de usuarios. Algunos de los más importantes son:

- **LUXOR:** Luxor es un entorno de desarrollo basado en XML para el desarrollo de interfaces de usuario. Contiene un intérprete de Script para el lenguaje Pitón.

- **THINLET:** Es uno de los entornos de desarrollos de interfaces de usuario en XML más ligeros que existen (su tamaño comprimido es de solo 39kB). Está escrito completamente en Java y su interfaz gráfica no necesita el paquete swing. Permite responder a los eventos en Java (en forma de métodos java), lo que le da una mayor potencia que a los demás entornos que sólo manejan scripts.

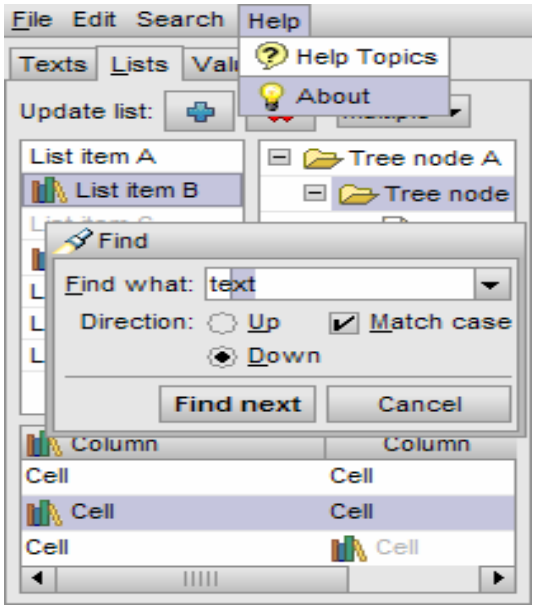

Figura 5: Interfaz gráfica escrita con Thinlet.

- **XALM :** Es el lenguaje de Microsoft para la implementación de interfaces de usuario usando XML. XALM será usado en el desarrollo de la nueva versión de Windows, el “Windows Longhorn”.

3. Objetivos del Proyecto

La meta final del proyecto es conseguir un "bundle" OSGi que exporte el servicio de generación de interfaz de usuario para poder ser usado por todas aquellas aplicaciones que se estén ejecutando en una pasarela residencial sobre el FrameWork de OSCAR.

El servicio de generación de interfaces de usuario recibirá una descripción de la interfaz de usuario de cada aplicación en formato XML y generará la interfaz de usuario de cada aplicación.

Además este servicio deber ser capaz de poder mostrar la misma descripción de interfaz de usuario en distintos tipos de "vistas"; es decir, el servicio de interfaz de usuario será capaz de generar la interfaz para ser mostrada tanto en un pc como en una pda con Terminal monocromo o en un Terminal de tipo carácter.

La misión de este proyecto es abstraer al programador de aplicaciones del tipo de dispositivo en el que finalmente será representada la interfaz de usuario de su aplicación.

Será tarea del servicio de interfaz de usuario obtener la información necesaria del equipo en el que se está ejecutando la aplicación, para poder así decidir qué tipo de "vista" tenemos que representar.

Para llevar a cabo esta tarea se realizan una serie de objetivos intermedios:

- Elección de un lenguaje de descripción de interfaces de usuario independientes de la máquina en que se ejecute.
- Diseño de la interfaz del servicio de generación de interfaces de usuario.
- Implementación de dicha interfaz en varias "vistas" distintas para comprobar la viabilidad del proyecto.
- Creación de una aplicación de ejemplo que haga uso del servicio de generación de interfaz de usuario para la generación de su UI.
- Generar el bundle servicio de generación de interfaz de usuario para que pueda ser usado por cualquier bundle que se ejecute en el Framework de OSCAR.
- Generar el bundle de la aplicación de pruebas para comprobar el correcto funcionamiento del servicio sobre el framework de OSCAR.