Universidad de las Ciencias Informáticas

Facultad 1

Título: Guía para la implementación de interfaces gráficas de usuario para la suite LibreOffice.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:

Dayani Quesada Mora

Tutores:

Msc. Maidely Calderón Montero

Ing. Grettel Barrio Marshall

Consultante:

Ing. Adriam Delgado Rivero

La Habana, Cuba, Junio 2014.

"Año 56 de la Revolución"

"No hacen falta alas para hacer un sueño, basta con las manos, basta con el pecho, basta con las piernas y con el empeño…"

Silvio Rodríguez

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al departamento SIMAYS de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo el presente a los ____ días del mes de Junio del año 2014.

Dayani Quesada Mora

_____

Firma del autor

Msc. Maidely Calderón Montero

Firma del tutor

Ing. Grettel Barrio Marshall

Firma del tutor

Dedicatoria

A mi familia, por sembrar en mí el amor y por enseñarme que debo tener la fortaleza de continuar hacia delante sin importar las circunstancias que la vida nos presenta.

Agradecimientos

A Maidely, Grettel, Adriam y Antuan por su ayuda incondicional, por su tiempo, porque sin ellos este trabajo no se hubiera terminado.

A mi madre bella Carmen, la principal protagonista en mi vida, le agradezco infinitamente por tanto amor y cariño, gracias por existir y ser la madre más maravillosa sobre la faz de la tierra.

A mis primos Regla (la preferida), Daylin, Yanisleydis, Yaneysi, Raulito, a mis tías Silvia, Estela, a mis abuelitos queridos María y Pedro (Cuco) por toda la confianza depositada en mí y los consejos brindados, por el empuje que siempre me han dado para seguir avanzando y por todo el cariño y aprecio que me han brindado.

A mis tíos Juan Alberto (Flaco), Raúl y a mi padrastro Carlos por formar parte de mi vida.
A mi compañera de la vida Jenny por su apoyo incondicional y cariño.

A mi novio Juan Manuel por estar siempre presente, por tanto cariño y dedicación, por ser tan comprensivo, por quererme tanto; gracias por enseñarme a ver la vida de otra manera.

A mis compañeros de batalla Elka (elkcho), Leonor, Lien, Leonar, Yurien, Adriana; a mis chicos del café Tapita, Dario, Leonel, Laura; a todos gracias por darme la oportunidad de conocerlos, nunca los olvidaré.

A mis compañeros de aula en especial a mis chicos de soporte técnico Dany, Gustavo y Hanoi.
A todas las amistades que he cultivado en estos cinco años.
A todos los que alguna vez me preguntaron: ¿y la tesis cómo va?

Resumen

En la presente investigación se llevó a cabo el desarrollo de una guía de implementación de interfaces gráficas de usuario para la *suite* LibreOffice, con el fin de obtener un mecanismo que permitiera a través de un conjunto de pasos lógicos, facilitar la creación e integración con las interfaces existente para la personalización cubana de la *suite.* La guía está enfocada en aspectos como: ubicación de las interfaces dentro del código, diseño, implementación, integración y compilación, teniendo en cuenta que la integración para cada tipo de aplicación de la *suite* se realiza de diferentes formas. Para desarrollar las interfaces de la investigación se utilizó como lenguajes de programación C++ y XML, entorno de desarrollo integrado Anjuta y herramienta de diseño Glade, basándose en la biblioteca GTK+.

Palabras claves: *Interfaces gráficas de usuario, código fuente.*

Índice de Contenido

Índice de Tablas

Índice de Figuras

Índice de Anexos

Introducción

Con el paso de los años, la humanidad se ha visto involucrada en crecientes cambios debido a las necesidades del hombre en cuanto al manejo de grandes volúmenes de información, lo que conllevan a la necesidad de almacenar información de manera digital, permitiendo su eficiente manejo mediante procesos automatizados que provean una mejor gestión de la misma.

La ofimática, proporciona un conjunto de técnicas informáticas que brindan facilidad en la gestión de trabajos en el campo de la producción y automatización de documentos, además de facilitar la creación, transmisión y almacenamiento de la información. Con la agilización de estos procesos, se ha logrado un adecuado uso de las tecnologías de la información, haciendo a un lado máquinas de escribir, calculadoras y otros equipos de contabilidad que permiten realizar las tareas de forma manual (1).

En la actualidad la humanidad vive en un mundo prácticamente privatizado donde el *Software* Libre (SWL) y el Código Abierto han cobrado auge como alternativa a los países e instituciones que abogan por su soberanía tecnológica. El SWL brinda ventajas como la colaboración y el trabajo en equipo, la no privatización del código fuente y la libertad de acceso a la información. Constituye además, un amplio movimiento tecnológico, por lo que en Cuba, su implementación es una alternativa en aras de impulsar el desarrollo de soluciones informáticas (2).
Ante la necesidad de migrar a SWL y valorando la importancia de lograr una correcta adaptación de los usuarios de las empresas nacionales y de la administración pública, el departamento de Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS) del Centro de *Software* Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI), asumió la responsabilidad de crear un grupo destinado a la investigación y desarrollo de componentes relativos a la herramienta ofimática LibreOffice en el entorno libre para satisfacer las necesidades de las empresas cubanas; previendo una posible descontinuación de sus desarrollos en la actualidad.
La *suite* ofimática LibreOffice cuenta con múltiples aplicaciones tales como: LibreOffice Writer para procesamiento de texto; LibreOffice Impress para crear presentaciones en diapositivas; LibreOffice Calc que es una hoja de cálculo para el trabajo con datos mediante gráficas y herramientas de análisis; LibreOffice Base como gestor de bases de datos; LibreOffice Draw que es un programa para el diseño de gráficos vectoriales y LibreOffice Math que es un editor de fórmulas matemáticas (3).

Para lograr una personalización de LibreOffice el equipo de desarrollo debe trabajar en la adición de nuevas funcionalidades. Gran parte de estas soluciones dependen de las interfaces gráficas de

usuario (en inglés *Graphic User Interface*, GUI), que como principal función proporcionan un entorno visual sencillo para permitir la comunicación con la suite. El insuficiente conocimiento sobre la integración de las GUIs de LibreOffice con sus respectivas funcionalidades, trae como consecuencia: poco enriquecimiento de la suite, evitando que se adicionen o modifiquen funcionalidades en la misma; y demora en el desarrollo de la personalización, que atenta contra la satisfacción de las necesidades de los usuarios que acceden a ella.

Dada la problemática anterior se propone el siguiente **problema de investigación**: ¿Cómo obtener interfaces gráficas de usuario que faciliten su integración con las funcionalidades básicas de LibreOffice en la personalización cubana?

Para dar solución al problema se propone como **objetivo general** desarrollar una guía para la implementación de interfaces gráficas de usuario para facilitar su integración con las funcionalidades existentes en la personalización cubana de LibreOffice. Definiendo como **objeto de estudio** el proceso de implementación de interfaces gráficas de usuario, enmarcando como su **campo de acción** el proceso de implementación de interfaces gráficas de usuario para la s*uite* ofimática LibreOffice.

Para darle cumplimiento al objetivo general se plantean los siguientes **objetivos específicos**:

1. Realizar un estudio del arte relacionado con el desarrollo de las interfaces gráficas de usuario para LibreOffice.
2. Confeccionar la guía para la implementación de las interfaces gráficas de usuario de LibreOffice.
3. Probar la guía confeccionada para la implementación de interfaces gráficas de usuario para LibreOffice.

Se plantea como **idea a defender** de la presente investigación que el desarrollo de una guía para la implementación de interfaces gráficas de usuario facilitará su integración con las funcionalidades básicas de LibreOffice en la personalización cubana.

Los objetivos específicos anteriores se concretan en las siguientes **tareas de investigación**:

1. Revisión del código fuente de las interfaces gráficas de usuario de la *suite* LibreOffice.
2. Revisión de la bibliografía asociada a la implementación de interfaces gráficas de usuario de la *suite* LibreOffice.
3. Análisis y selección de las herramientas y tecnologías existentes para el desarrollo de interfaces gráficas de usuario.

4. Definición de la guía para la implementación de las interfaces gráficas de usuario de *suite* LibreOffice.
5. Implementación de las interfaces gráficas de usuario del Asistente para Calendarios de LibreOffice Writer y el Conversor Chino de la *suite* empleando la guía confeccionada.

En el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- **Analítico-Sintético:** posibilitó el análisis del código fuente de las versiones de la *suite* LibreOffice. A partir del cual se sintetizaron los datos obtenidos en posición de obtener los resultados necesarios para el cumplimiento del objetivo general.
- **Histórico-Lógico:** permitió analizar la evolución de las versiones de la *suite* LibreOffice.

El presente trabajo de diploma está estructurado en tres capítulos de los cuales se muestra una breve descripción.

Capítulo 1: Fundamentación teórica: En este capítulo se analiza el código fuente de LibreOffice buscando los directorios relacionados a las GUIs en la *suite*; además se identifican varias bibliotecas para el desarrollo de interfaces gráficas de usuario. Se realiza un estudio de las herramientas y lenguajes de programación que se utilizan en la solución propuesta.

Capítulo 2: Guía para la implementación de interfaces de usuario para la suite LibreOffice: En este capítulo se confecciona una guía que permite, mediante una secuencia de pasos lógicos, orientar a los desarrolladores en la implementación de las interfaces gráficas de usuario para la *suite* LibreOffice.

Capítulo 3: Pruebas a la guía confeccionada: En este capítulo se pone en práctica la guía confeccionada, el desarrollo de cada uno de los pasos definidos en la misma se lleva a cabo a través de la integración del Asistente para Calendarios de LibreOffice Writer (4) y el Conversor Chino de la *suite*, con el objetivo de validar su correcto funcionamiento.

Capítulo 1. Fundamentación teórica

En este capítulo se analiza el código fuente de LibreOffice buscando los directorios relacionados con las GUIs en la *suite*; además se identifican varias bibliotecas para el desarrollo de interfaces gráficas de usuario. Se realiza un estudio de las herramientas y lenguajes de programación que se utilizan en la solución propuesta.

1.1 Suite Ofimática LibreOffice

El 28 de septiembre de 2010, surge la *suite* ofimática libre y gratuita LibreOffice, compatible con *Microsoft Windows*, Mac y GNU/Linux. Fue creada por *The Document Foundation* como una bifurcación de la *suite* OpenOffice.org. La bifurcación tuvo lugar ante los temores de que *Oracle Corporation*, después de comprar a *Sun Microsystems* -anterior patrocinador de OpenOffice.org- descontinuara la *suite*, como hizo con *OpenSolaris*. A finales de octubre del 2010 la mayor parte de los desarrolladores dejaron OpenOffice.org y se trasladaron a *The Document Foundation* para apoyar a LibreOffice, además recibieron ayuda de parte de la antigua comunidad de OpenOffice.org, incluyendo a las empresas Novell, Red Hat, Canonical y Google. Su objetivo es producir una *suite* ofimática independiente de cualquier proveedor, compatible con el formato de archivo *OpenDocument*.

The Document Foundation se dirige a lanzar nuevas versiones de LibreOffice a un ritmo de una cada seis meses, enfocándose los desarrolladores principalmente en eliminar el código no utilizado e insertar interfaces gráficas de usuario que apoyen el perfeccionamiento de la *suite* (5).

1.2 Análisis del código fuente de la suite LibreOffice

El código fuente de la *suite* LibreOffice cuenta aproximadamente con 2302 archivos agrupados en 149 directorios, en la que se identificaron 22 directorios relacionados directamente con el desarrollo de las GUIs.

- **Basctl:** Contiene controles y cuadros de diálogo para *Basic* y el IDE *Basic*.
- **Chart2:** Contiene la implementación gráfica de LibreOffice Calc.
- **Cui:** Contiene las GUIs comunes de más de una aplicación.
- **Dbaccess:** Contiene las herramientas de acceso a bases de datos, para la aplicación LibreOffice Base.

- **Desktop:** Contiene el escritorio en StarOffice 5.
- **Extensions:** Contiene los p*lugins* para el navegador o las aplicaciones.
- **Filter:** Contiene algunos filtros de registros simples.
- **Fpicker:** Contiene los recolectores de archivo nativo de *OS/X*[1] y *Windows* (diálogo de abrir archivo).
- **Framework:** Contiene la reescritura de la interfaz de usuario, barras de herramientas, menús, incluyendo aceleradores y la interacción.
- **Icon-themes:** Contiene el repositorio de íconos para las aplicaciones.
- **Officecfg:** Contiene la configuración de esquema y predeterminados para la base de datos de configuración de LibreOffice.
- **Reportdesign:** Contiene parte del código de la aplicación LibreOffice Base.
- **Sc:** Contiene el código de la aplicación LibreOffice Calc.
- **Sd:** Contiene el código de la aplicación LibreOffice Impress y LibreOffice Draw.
- **Starmath:** Contiene el código de la aplicación LibreOffice Math.
- **Svl:** Contiene código auxiliar no gráfico para aplicaciones de oficina.
- **Svtools:** Contiene interfaces de uso común.
- **Svx:** Contiene gráficos relacionados con el código ayuda de las aplicaciones LibreOffice Impress y LibreOffice Draw.
- **Sw:** Contiene el código de la aplicación LibreOffice Writer.
- **Uui:** Contiene un controlador de interacción para el directorio “ucb”. Funciona a través de la Biblioteca de Componentes Visuales.
- **Vcl:** Contiene los *widgets* (ventanas, botones, controles y archivos recolectores) de abstracción del sistema operativo, incluyendo la prestación básica (por ejemplo, el dispositivo de salida).
- **Xmlsecurity:** Contiene elementos para la firma de documentos.

Dentro de cada directorio se almacenan diferentes subdirectorios como:

1. **Inc:** Contiene archivos para configurar e integrar las GUIs a la *suite* LibreOffice.

1 **OS/X**: Es un sistema operativo basado en Unix, desarrollado, comercializado y vendido por Apple Inc.

2. **Sdi:** Contiene archivos para configurar e integrar las GUIs a la *suite* LibreOffice.
3. **Uiconfig:** Contiene todo lo relacionado con las GUIs desarrolladas con la biblioteca GTK de la aplicación.
    - **Ui:** Se almacenan todas las GUIs creadas con la biblioteca GTK usando la extensión .ui.
    - **ToolBar:** Se almacenan todos los elementos que conformarán la barra de herramientas de la aplicación usando la extensión .xml.
    - **Menubar:** Contiene la barra de menú de la aplicación usando la extensión .xml, en donde se añaden las interfaces de usuario.
4. **Source:** Contiene código funcional de las GUIs en C++.
    - **Ui:** Contiene el código fuente de las interfaces de la aplicación, usando las extensiones .cxx y .hxx.

1.3 Bibliotecas de desarrollo de Interfaces Gráficas de Usuario

LibreOffice cuenta con una gran variedad de GUIs que representan un conjunto de formas y métodos que posibilitan la interacción de la *suite* con los usuarios, utilizando formas gráficas e imágenes (6). La GUI de la *suite* es de sencillo uso y fácil personalización. Tras personalizar la *suite* ofimática se identificaron varias bibliotecas existentes para el desarrollo de las GUIs como se muestra a continuación.

1.3.1 Biblioteca de Componentes Visuales

La Biblioteca de Componentes Visuales (en inglés *Visual Components Library*, VCL) es un marco de trabajo desarrollado por la compañía de *software* Borland[2] basado en objetos visuales que tienen como finalidad diseñar marcos (*frames)* para las aplicaciones que se han programado para *Windows* y diseñada bajo el concepto de componente: propiedades, métodos y eventos. Está estructuralmente y visualmente sincronizado con *Windows*, pues el aspecto de las ventanas, los botones o los accesos a Internet, son similares. Este tipo de biblioteca basa su programación en el paradigma orientado a objeto, pues los componentes son clases predeterminadas que hacen su uso más sencillo, visual y cómodo. La misma cuenta con dos tipos de componentes:

2 **Borland Software Corporation** es una compañía de *software*, ubicada en Austin, Texas, Estados Unidos. Fue fundada en 1983 por Niels Jensen, Ole Henriksen, Mogens Glad y Philippe Kahn.

- **Los componentes visuales** son aquellos que, al utilizarlos, muestran algún elemento (o dibujo) en la pantalla y el usuario puede interactuar con él. Hay muchos componentes de este tipo, como son los botones, etiquetas de texto, formas.
- **Los componentes no visuales** son aquellos que no aparecen en la ventana, y se insertan en un formulario para que el programador los utilice. Son más fáciles de programar que los componentes visuales, ya que no tienen ningún tipo de interfaz gráfica. Se pueden mencionar como ejemplos de componentes no visuales a un temporizador, una tabla o una conexión a base de datos.

La VCL hace uso extensivo del concepto de herencia, con el objetivo de crear clases que representan a componentes. Aunque algunas clases no hacen referencia a componentes concretos, realizan tareas de gestión interna y se emplean como clases bases para derivar otras clases mediante herencia (7).

1.3.1.1 Tipos de clases en la VCL

Las clases que conforman la parte superior de la jerarquía de la VCL (*TObject*, *TPersistent* y *TComponent*) se denominan clases abstractas porque sirven para estructurar, y agrupar comportamientos comunes de las clases de la biblioteca. No se suelen crear objetos directamente a partir de ellas.

- ***TObject:*** Encapsula el comportamiento común de los objetos en *C++ Builder*, como puede ser información de la clase e instanciación. Directamente de *TObject* heredan aquellas clases que no son componentes, y no necesitan ser almacenadas en disco.
- ***TPersistent:*** Esta clase tiene que ver con la habilidad de un objeto de almacenarse en disco o en memoria, asignarse a otros objetos, así como otros detalles internos de *C++ Builder* que no es preciso conocer.
- ***TComponent:*** Esta es una de las clases más importantes, ya que la mayoría de los objetos que se manejan en una aplicación son componentes. Esta clase proporciona toda la funcionalidad que requiere un componente básico. La funcionalidad de *TComponent* permite que los objetos aparezcan en la paleta de componentes y que sean manipulados por el diseñador de formularios, además de otras capacidades comunes a los componentes. Los componentes no visuales derivan directamente de *TComponent* mientras que los componentes visuales derivan de *TControl*, que a su vez deriva de *TComponent*. Por esta razón se suelen denominar controles a los componentes visuales.
- ***TControl:*** Proporciona la funcionalidad de los componentes visuales (controles). Esta funcionalidad es principalmente el aspecto visual que deben ofrecer los componentes en

tiempo de ejecución. Proporciona mayor funcionalidad que la que requieren los componentes visuales, ya que los componentes individuales derivan de *TGraphicControl* o de *TWinControl*, clases derivadas de *TControl*.

- ***TGraphicControl:*** Generaliza a los controles que tienen una representación visual, pero no pueden recibir el foco, el usuario podrá verlos pero no interactuar con ellos. Suelen ser controles para visualizar texto (no editable en tiempo de ejecución), gráficos o dibujos.
- ***TWinControl:*** Son los controles típicos de *Windows*, que pueden recibir el foco, contener otros controles, y además poseen un manejador de ventana. Un manejador de ventana es un identificador que proporciona *Windows* para el control, por lo que podemos decir que *Windows* tiene un conocimiento directo de la existencia del mismo.

Existen otros controles que no aparecen en la paleta de componente que son de gran importancia:

- ***TApplication:*** Encapsula una aplicación *Windows* por lo que caracteriza las operaciones fundamentales de un programa en *Windows*. Simplifica la interfaz entre el programador y el sistema operativo, ocupándose de tareas como gestionar el paso de mensajes, proporcionar la ayuda contextual, establecer los textos de sugerencia de los botones y barras de estado, gestión de excepciones y ejecutar cuadros de mensajes.
- ***TForm:*** *C*aracteriza a los formularios en la VCL. Los formularios se emplean como ventanas principales, cuadros de diálogo, ventanas secundarias o cualquier otro tipo de ventana que se pueda imaginar (7).

1.3.1.2 Restricciones al uso de la VCL en C++ Builder

Como la *VCL* se desarrolló originariamente para *Delphi,* construida en *Object Pascal,* esto no supone normalmente consideraciones especiales, pero existen algunos aspectos que deben ser tenidos en cuenta:

- Los objetos de la *VCL* deben crearse siempre dinámicamente.
- La *VCL* no asigna valores por defecto a las funciones.
- No se puede usar herencia múltiple con clases *VCL* (7).

1.3.2 GIMP Toolkit (GTK+ 3.2.3)

Conjunto de bibliotecas multiplataforma para crear GUIs, desarrollada bajo la Licencia Pública General Reducida (LGPL). Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente existen otros tipos de programas en los sistemas

GNU/Linux que lo usan. GTK+ se ha diseñado para permitir programar con lenguajes como C, C++, Java, Ruby, Perl, PHP o Python (8).

1.3.2.1 GTK+ Orientado a Objeto

GTK+ está basado en una jerarquía de clases, por lo que las instancias asociadas a una clase utilizan punteros a estructuras de C, a estos se les llama "*widgets*". Los *widgets* son componentes gráficos con el cual el usuario interactúa, como por ejemplo, una ventana o una caja de texto (9).
Unas clases heredan de otra, con fin de aprovechar las funcionalidades comunes de otros w*idgets*, agruparla en una clase padre, y añadirla a las clases hijas. La herencia se lleva a cabo con la inclusión de un campo con la copia de las estructuras de las clases padres en la de la hija.
La comprobación de tipos utiliza un método propio, basado en un *casting* (macros de conversión) de C mejorado, que se utilizan para convertir un w*idget* genérico en cualquier otro tipo de w*idget* GTK+ y para asegurarse de que ésta se puede realizar. La privacidad debe ser controlada por el desarrollador, por lo que el encapsulamiento se llevará a cabo con la interfaz única de cada objeto (w*idget*).
En las señales y retrollamadas se gestionan los eventos o sucesos que ocasionan las interacciones con los usuarios. Estas señales de las clases que se derivan también son heredadas, por lo que deben ser registradas de manera adecuada, al igual que las retrollamadas para así procesar aquellas señales que se quieran.
En el polimorfismo, los procedimientos no podrán admitir parámetros por defecto, como se harían en C++, por lo que no se harán sobrecargas de operadores, ni de funciones. El tratamiento de excepciones se lleva a cabo con el tratamiento de errores estándar de C, que se incluirá (de forma alterada) en la biblioteca asociada a GTK+ llamada Glib (10).

1.3.2.2 Bibliotecas que soporta GTK+

GTK+ se basa en varias bibliotecas desarrolladas por el equipo de GTK+ y de GNOME, se relacionan de la siguiente manera:

1. **Glib** es la base del proyecto GTK+ y GNOME. Provee las estructuras de datos en el lenguaje C. Contiene la infraestructura necesaria para construir interfaces para tales funciones como un bucle de ejecución de eventos, hilos, carga dinámica de módulos y un sistema de objetos. Siendo así, una biblioteca de utilidades de propósito general, la cual provee tipos de datos útiles, así como macros, conversiones de tipo, utilidades de cadenas y muchas más. Se encarga de interactuar con el SO anfitrión sin distraer al programador de las diferencias más significativas entre las distintas variedades de estos, reuniendo las siguientes características:

- **Tipos de datos:** Los tipos de datos (char, int, float, double), pueden diferir entre los diferentes SO o arquitecturas de *hardware*. Glib libera al programador de prestar atención a estos detalles y en su lugar ofrece un conjunto de tipos propios (gchar, gint, gint32, guint, guint32, guchar).
- **Gestión de memoria:** Tarea especialmente delicada, por lo que Glib la administra automáticamente. Si el programador necesita mayor control de la memoria entonces tendrá a su disposición gran cantidad de métodos que le permitirán un control más exhaustivo de ésta.
- **Estructuras de datos:** Las estructuras de datos más comunes como listas enlazadas, arreglos dinámicos, tablas de claves, pilas y colas, ya están disponibles en Glib.
- **Sistema de objetos:** Se ha creado con la intención de integrarse fácilmente con otros lenguajes de programación diferentes de C. Todos los objetos de GTK+ se derivan de la clase fundamental de Glib: Gobject.
- **Bucle de ejecución:** En lugar de crear un bucle de eventos nosotros mismos, podemos dejarle la tarea a Glib para centrarnos en el diseño y desarrollo de nuestras aplicaciones asíncronas. Glib se encarga de la distribución de señales y mensajes a las diferentes partes de nuestro programa. También se encarga de administrar alarmas (temporizadores para aplicaciones asíncronas), momentos de inactividad de la aplicación, eventos de entrada y salida en tuberías o descriptores de archivos, así como hilos.

2. **GObject** es la parte de Glib que implementa las características de lenguajes orientados a objetos que no están disponibles intrínsecamente en el lenguaje C. De esta manera GTK+ puede ser una caja de herramientas orientada a objetos sin tener que implementarse en C++ u otro lenguaje.
3. **GDK** interactúa con las primitivas gráficas de cada sistema operativo y las prepara para su uso con GTK+. El programador de aplicaciones no notará esto y, salvo raras excepciones, un programa escrito para GTK+ en Linux, por ejemplo, puede compilarse en *Windows* sin mayor modificación.
4. **GdkPixbuf** contiene convenientes rutinas para el manejo de imágenes (png, gif y jpeg), tales como abrir y guardar, trabajar con mapas de imágenes en memoria, escalado y animaciones entre otras características.
5. **Pango** es un entorno de trabajo para el procesamiento y despliegue de texto internacional utilizando 8-bit *Unicode Transformation Format* (UTF-8) como código base; está integrado

por completo en GTK+ y permite desplegar caracteres en distintos alfabetos y estilos de escritura, texto bidireccional y con atributos como color, tamaño y estilo o rotación.

6. **ATK** proporciona un conjunto de interfaces de accesibilidad. Si uno soporta interfaces ATK en un programa, tendremos la posibilidad de utilizar diferentes herramientas como magnificadores de pantalla, texto a diálogo o diferentes métodos de entrada para personas con capacidades diferentes (8).

1.3.2.3 Aplicaciones que utilizan GTK+ en el desarrollo de sus GUIs

Existen algunas aplicaciones que usan GTK+ para desarrollar sus interfaces de usuario, como:

- **AbiWord:** Procesador de textos
- **Evolution:** Cliente de correo electrónico
- **Firefox:** Navegador web
- **GIMP:** Editor de gráficos
- **LibreOffice:** *Suite* ofimática
- **Gnumeric:** Programa de hoja de cálculo
- **Chromium:** Navegador Web desarrollado en gran medida por Google
- **GRAMPS:** *Software* de genealogía
- **Inkscape:** Editor de gráficos vectoriales
- **K-3D:** Programa de modelado 3D libre
- **Marionnet:** Simulador de red interactivo
- **Nero Linux:** Programa para la edición de discos
- **Pidgin:** Cliente de mensajería instantánea
- **VMware Player:** Máquina virtual
- **Wireshark:** Capturador y analizador de paquetes de redes computacionales (11)

1.3.3 Qt+ 5.2.0

Es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con GUI, así como para el desarrollo de programas sin GUI, como herramientas para la línea de comandos y consolas para servidores. Es desarrollada como un *software* libre y de código abierto a través de *Qt Project* bajo la licencia LGPL, Licencia Pública General (GPL) y Propietaria, donde participa tanto la comunidad, como desarrolladores de Nokia, Digia y otras empresas (12).

Qt es utilizada en el Entorno de Desarrollo K (KDE), un entorno de escritorio para sistemas operativos libre como GNU/Linux o FreeBSD. Utiliza el lenguaje de programación C++ de forma

nativa, adicionalmente puede ser utilizado en varios lenguajes de programación a través de *bindings*[3].

Tipos de *bindings* a utilizar en Qt:

- **PySide:** LGPL *bindings* para Python de OpenBossa (subsidiario de Nokia)
- **PythonQt:** LGPL *bindings* para Python
- **Qyoto:** *Bindings* para C# u otros lenguajes .NET
- **QtRuby:** *Bindings* para Ruby
- **Qt Jambi:** *Bindings* para Java
- **QtAda:** *Bindings* para Ada
- **FreePascal Qt:** *Bindings* para Pascal
- **Perl Qt:** *Bindings* para Perl
- **PHP-Qt:** *Bindings* para PHP (13)

1.3.3.1 Qt Orientado a Objeto

Gracias a que Qt se desarrolla directamente en C++, no tiene ninguna dificultad en adaptar los w*idgets*, la calidad en tiempo de ejecución es más alta, ya que no necesita hacer uso de referencias innecesarias y además, no depende de ninguna otra biblioteca, aparte de sí misma. También se pueden añadir las siguientes características:

1. Tiene un mecanismo de comunicación entre objetos llamado Señales y Ranuras (S*ignals -Slots*[4]), el cual hace visible la programación basada en componentes.
2. Las propiedades de cada objeto se pueden diseñar y consultar.
3. Contiene potentes eventos y filtros de eventos.
4. Contiene cadenas contextuales de traducción para una mejor internacionalización.
5. Incluye un sofisticado soporte de temporizadores que hacen posible integrar elegantemente muchas tareas en una GUI orientada a eventos.
6. Incluye un árbol jerárquico de objetos ordenados de un modo natural y con una fácil interpretación.
7. Incluye punteros seguros que son automáticamente puestos vacíos cuando el objeto referenciado es destruido, en contraposición con los punteros normales de C++ que no cambian cuando sus objetos son destruidos.

---

3 **B*indings.*** *A*daptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

4 **Signals-Slots.** Mecanismo de comunicación entre objetos seguro, flexible y totalmente orientado a objetos.

Muchas de estas características de Qt están implementadas con las técnicas estándar de C++, basadas en la herencia. Algunas otras, como el mecanismo de comunicación entre objetos y el sistema dinámico de propiedades requieren del *Meta Object System*, el cual se basa en el *Meta Object Compiler* (MOC), programa que lee el fichero fuente de C++ (10).

1.3.3.2 Módulos de Qt

La biblioteca C++ Qt provee un rico conjunto de componentes para la construcción de aplicaciones, distribuidas en módulos como:

- **QtCore:** Clases no gráficas utilizadas por los otros módulos
- **QtGui:** Componentes para las GUI
- **QtNetwork:** Programación de redes
- **QtOpenGL:** Soporte a OpenGL
- **QtScript:** Evaluación de Qt *Scripts*
- **QtScriptTools:** Componentes Qt *Script* adicionales
- **QtSql:** Integración con SQL
- **QtSvg:** Clases para mostrar el contenido de ficheros SVG (Gráficos Vectoriales Redimensionables)
- **QtWebKit:** Clases para mostrar y editar contenido web
- **QtXml:** Manejo del *Extensible Markup Language* (XML)
- **QtXmlPattern:** Motores XQuery y XPath para XML
- **Phonon:** Clases del *framework*[5] mutimedia
- **Qt3Support:** Soporte de compatibilidad a clases de Qt 3 (14)

1.3.3.3 Aplicaciones que utilizan Qt en el desarrollo de sus GUIs

- **Álbum de Adobe Photoshop:** Aplicación para organizar imágenes
- **Avidemux:** Programa libre para la edición y procesamiento de video
- **Doxygen:** Interfaz de programación de aplicaciones generadora de documentación
- **Gadu-Gadu:** Cliente polaco de mensajería instantánea
- **KDE:** Entorno de escritorio para sistemas operativos *Unix*.
- **Launchy:** Programa de código abierto para ejecutar aplicaciones para Windows
- **MythTV:** Grabador de vídeo digital de código abierto

5 **Frameworks**. Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*.

- **Psi:** Cliente de mensajería instantánea para XMPP[6]
- **Qt Creator:** Entorno de desarrollo integrado, *software* libre y multiplataforma de Nokia
- **Scribus:** Aplicación para la publicación de escritorio
- **TeamSpeak:** Aplicación multiplataforma para la comunicación con voz
- **VirtualBox:** Aplicación de código abierto para la virtualización
- **VLC Media Player:** Reproductor multimedia de código abierto (13)

1.3.4 Biblioteca wxWidgets 3.0.0

wxWidgets (W para *Windows* y X para *X-Unix*) son bibliotecas multiplataforma y libres, para el desarrollo de GUIs programadas en lenguaje C++. Está distribuido bajo una licencia personalizada *wxWindows License*, similar a la Licencia Pública General Reducida de GNU, con la excepción de que trabajos derivados en forma binaria pueden ser distribuidos bajo los términos del usuario. Esta licencia está aprobada por la Iniciativa *Open Source (Open Source Initiative)*, haciendo de *wxWidgets* un *software* libre (15).

Las *wxWidgets* proporcionan una interfaz gráfica basada en las bibliotecas ya existentes en el sistema (nativas), con lo que se integran de forma óptima y resultan muy portables entre distintos sistemas operativos. Están disponibles para las siguientes plataformas:

- **Windows:** wxMSW (Windows 95/98/ME, NT, 2000, XP, y Vista)
- **Linux/Unix:** wxGTK+, wxX11, wxMotif
- **Mac OS X:** wxMac (10.3 usando Carbon)
- **OS/2:** wxOS2, wxPM, wxWidgets para GTK+.

También pueden ser utilizadas desde otros lenguajes de programación, aparte del C++: Java, JavaScript, Perl, Python, Smalltalk, Ruby.

wxWidgets se describe como un conjunto de herramientas que utilizan un control nativo en las plataformas existentes lo que permite conseguir mejores resultados visuales para la GUI que otras bibliotecas como *Swing* (para Java), además de ofrecer mejor rendimiento y otros beneficios. No solo se restringe al desarrollo de interfaces gráficas, ya que la biblioteca cuenta con una capa de inter-procesos de comunicación y funcionalidades para la red como *sockets* (16).

1.3.4.1 wxWidgets orientada a objeto

Como la mayoría de los *frameworks* modernos para GUIs, w*xWidgets* se beneficia del uso de la programación orientada a objetos. Cada ventana es representada como un objeto en C++. Estos objetos tienen un comportamiento bien definido, y pueden recibir y responder a eventos. Lo que el

6 **XMPP**. Protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea.

usuario ve es la manifestación visual de este sistema interactivo. El trabajo del desarrollador es organizar el comportamiento colectivo de todos estos objetos, lo que se vuelve más fácil con el comportamiento por defecto que w*xWidgets* implementa (17).

1.3.4.2 Aplicaciones desarrolladas con wxWidgets

Existen un conjunto de aplicaciones que utilizan esta biblioteca para desarrollar sus GUIs, entre las más conocidas se encuentran:

- **Amaya:** Desarrollo web
- **Amule:** Aplicación de intercambio de archivos
- **Audacity:** Editor de audio
- **BitTorrent:** Aplicación de intercambio de archivos
- **Code::Blocks:** Entorno de desarrollo integrado (IDE) de C/C++
- **CodeLite:** Editor simple para C++ (Colección de herramientas Gratuitas, implementadas mediante extensiones)
- **Digsby:** Aplicación multiprotocolo de mensajería instantánea
- **FileZilla:** Cliente FTP
- **RapidSVN:** Cliente de *Subversion* (18)

1.3.5 Comparación entre las diferentes bibliotecas para el desarrollo de GUIs

Aspectos	Bibliotecas			
	VCL	**GTK+ 3.2.3**	**Qt+ 5.2.0**	**wxWidgets 3.0.0**
Entorno de desarrollo	Delphi, C++Builder	Multiplataforma	Multiplataforma	Multiplataforma
Licencia	LGPL	LGPL	LGPL, GPL, Propietaria	*wxWindows Library Licence*
Lenguaje de programación	Varios	Varios	Varios	Varios
Soporte con otras bibliotecas	Si	Si	No	No
GUI de LibreOffice	Si	Si	No	No
Herramienta de diseño	No	Si	Si	Si

Tabla 1: *Comparación de las diferentes bibliotecas de desarrollo GUIs. Fuente: Elaboración propia.*

Tras el análisis de las bibliotecas para el desarrollo de GUIs de la tabla anterior, se identifica que solamente las bibliotecas VCL y GTK+ son usadas para la creación de las interfaces de usuario de LibreOffice; para facilitar el diseño e implementación de las mismas en la presente investigación se utilizará GTK+, ya que cuenta con una herramienta propia para el diseño.

1.4 Herramientas y lenguajes a utilizar

1.4.1 Lenguaje de programación

En LibreOffice, para el desarrollo del código de las GUIs se emplean como lenguajes de programación C++ y XML, por lo que en la presente investigación se utilizan los mismos lenguajes para facilitar su integración en la *suite*.

C++, es un lenguaje imperativo orientado a objetos derivado del C. En realidad un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al *hardware* subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se le han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción (19).

XML: Es un lenguaje de programación usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el consorcio *World Wide Web*, en colaboración con un panel que incluye representantes de las principales compañías productoras de *software* (20).

1.4.2 Entorno de desarrollo integrado (IDE)

Anjuta 3.4.0, es un IDE para C y C++ en GNU/Linux. Ha sido escrito para GTK/GNOME. Tiene licencia GPL y cuenta con una serie de instalaciones avanzadas de programación como la gestión de proyectos, asistentes para aplicaciones, un depurador interactivo y un poderoso editor de código fuente con la navegación y resaltado de sintaxis. Se centra en proporcionar la interfaz de usuario simple y utilizable, pero de gran alcance para el desarrollo eficiente.

Características:

Contiene varias extensiones como:

- Arquitectura revisada y extensible
- Nuevo intérprete de comandos propio y documentación del API
- Integrado un nuevo sistema de ayuda
- Un diseñador gráfico de interfaces de usuario con Glade

- Diversas mejoras en el editor de programación (edición remota, mejor coloreado de la sintaxis)
- Nuevo administrador de tareas
- Extensión para añadir macros, insertar texto predefinido o personalizado
- Plantilla fácilmente extensible para proyectos mediante asistente
- Extensión para *Subversion*
- Actualizada la extensión para CVS
- Administrador de sesiones de trabajo (21)

Netbeans 7.4, es un entorno de desarrollo integrado libre para desarrolladores de *software*. Es un proyecto exitoso con una gran base de usuarios, una comunidad en constante crecimiento, está escrito en java, pero soporta una gran cantidad de lenguajes de programación como el propio java, C/C++, PHP, JavaScript y Groovy. Existe además un número importante de módulos para extender el Netbeans, como su módulo para Python, haciéndolo uno de los mejores IDE para desarrollar en ese lenguaje (22).

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario con la biblioteca *Swing*
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- *Framework* basado en asistentes (diálogo paso a paso) (23)

Eclipse 3.7.2, es un entorno de desarrollo integrado de código abierto, multiplataforma; es desarrollado por la Fundación Eclipse. Cuenta con un desarrollo muy activo, y una buena cantidad de módulos que lo hacen una excelente herramienta para el desarrollo, con soporte actualmente para una gran variedad de lenguajes de programación como Java, Python, Perl, PHP y muchos otros. A la plataforma base de Eclipse se le pueden añadir extensiones (*plugins*) para extender la funcionalidad (24).

Características:

- Dispone de un editor de texto con resaltado de sintaxis donde se puede ver el contenido del fichero en el que se está trabajando.
- Contiene una lista de tareas y otros módulos similares.
- La compilación es en tiempo real.

- Tiene pruebas unitarias con *Junit.*
- Integración con asistentes (*wizards*) para creación de proyectos, clases, refactorización (25).

1.4.2.1 Comparación de los IDE

Aspectos	IDE		
	Anjuta	**Netbeans**	**Eclipse**
Sistema operativo	GNU/Linux	Multiplataforma	Multiplataforma
Licencia	GPL	GPL	GPL
Idioma	Multilingüe	Multilingüe	Multilingüe
Lenguaje de programación	Varios	Varios	Varios
Integración con Glade	Si	No	No

Tabla 2: *Comparación de los diferentes entornos de desarrollo integrado. Fuente: Elaboración propia.*

Tras el análisis de los IDE anteriores se identifica que para el desarrollo de la presente investigación se utiliza como IDE Anjuta, ya que es un entorno para desarrollar GUIs en C++ utilizando la biblioteca GTK y se integra con la herramienta de diseño Glade.

1.4.3 Herramienta de diseño

En LibreOffice, para el diseño de las GUIs se emplea la herramienta Glade, por lo que en la presente investigación se utiliza la misma con el fin de facilitar su proceso de integración en la *suite*.

Glade 3.12.1, desarrollador de interfaces que permite construir de forma gráfica e interactiva interfaces de usuario gráficos para GNOME/GTK, publicado bajo la licencia GNU GPL. Es capaz de generar código para implementar la interfaz visual, que se describe en XML, en C, C++, Ada 95, Perl y Eiffel (26).

1.4.4 Herramienta para el Control de Versiones

Subversion **(SVN)**, es un *software* de sistema de control de versiones. Se llama control de versiones a la gestión de cambios que se realizan sobre los elementos de algún producto. Presenta varias ventajas:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.

- Se envían sólo las diferencias en ambas direcciones.
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez (27).

RapidSVN 0.12.0, es un cliente gráfico para el sistema *Subversion* escrito en el lenguaje C++, con código abierto y *software* libre bajo la licencia GNU GPL. Está disponible en varios idiomas y actúa de cliente gráfico para el acceso al repositorio SVN, tanto si éste es remoto como si es local. Es de fácil manejo para los usuarios principiantes pero lo suficientemente potente y con herramientas interesantes para los usuarios avanzados. El programa se puede ejecutar en Linux, *Windows*, Mac OS / X, Solaris (28).

1.5 Conclusión parcial

Una vez realizado el análisis del código fuente de LibreOffice, se determinó utilizar la biblioteca GTK así como su herramienta de diseño Glade, con lo cual se podrán crear y modificar las interfaces pertinentes, sin necesidad de acceder a su código fuente. Para llevar a cabo el objetivo propuesto se escogieron los lenguajes de programación C++ y XML, los cuales por sus características facilitarán el correcto desarrollo de la solución, empleando para ello el IDE Anjuta.

Capítulo 2. Guía para la implementación de interfaces de usuario para LibreOffice

En este capítulo se confecciona una guía que permite, mediante una secuencia de pasos lógicos, orientar a los desarrolladores en la implementación de las interfaces gráficas de usuario para la *suite* LibreOffice. Para esto se lleva a cabo un análisis de lo investigado en el capítulo anterior teniendo como resultado un conjunto de buenas prácticas a tener en cuenta para obtener las GUIs.

Guía para la implementación de GUIs para LibreOffice

2.1 Introducción

La presente guía constituye un mecanismo que permite la creación e integración de las GUIs en LibreOffice, con el objetivo de encaminar a los desarrolladores en el funcionamiento e implementación de las mismas en la *suite*. A continuación se detalla gráficamente cómo está estructurada.

2.1.1 Estructura de la guía

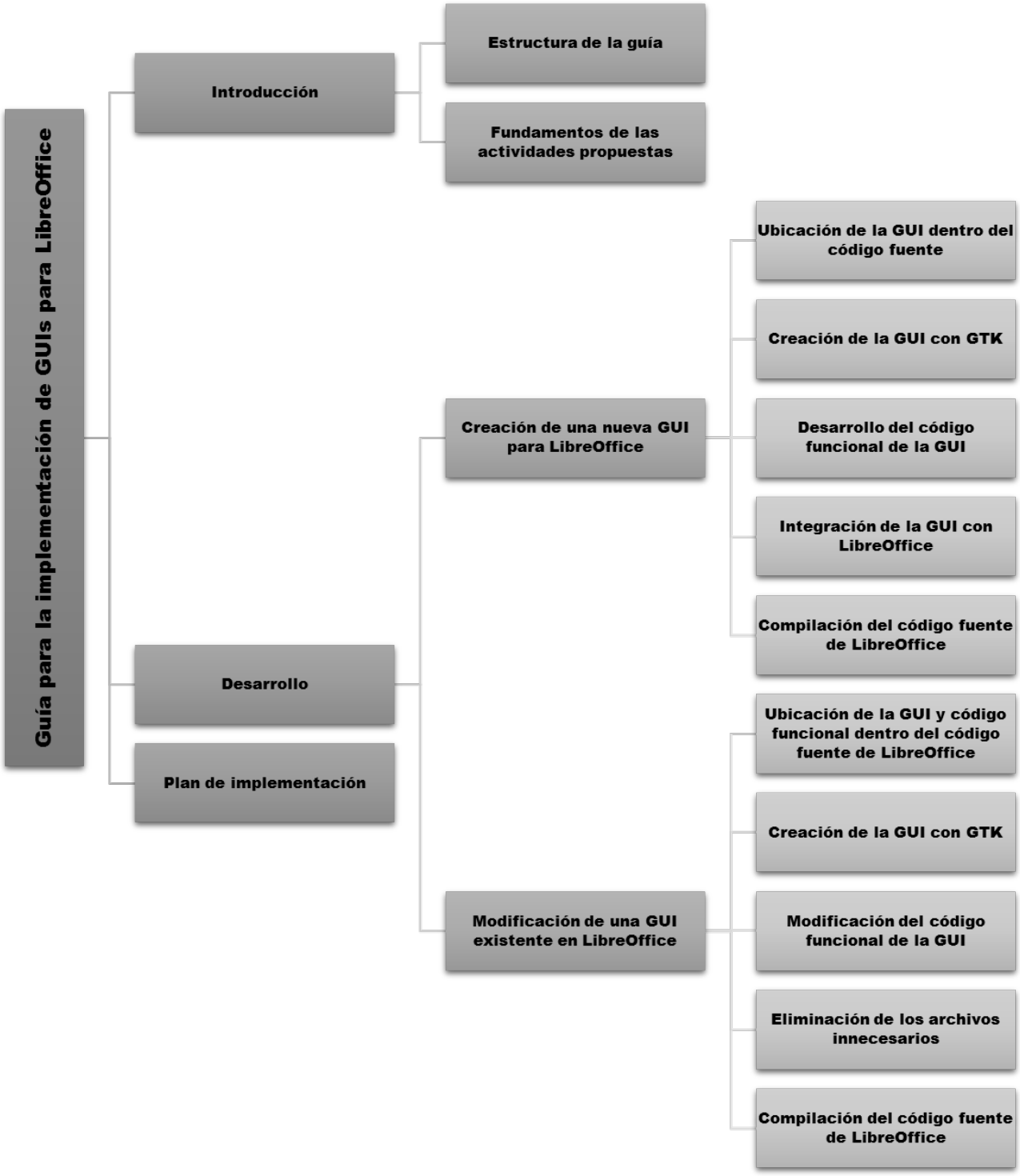

Figura 1: *Estructura de la guía*

2.1.2 Fundamentos de las actividades propuestas

En el desarrollo de la guía para la implementación de GUIs en LibreOffice se identifican los epígrafes: Introducción, Desarrollo y Plan de implementación respectivamente, que son los elementos a tener en cuenta en la investigación para obtener una GUI en esta *suite*. En el primer epígrafe se define la estructura y fundamentos de las actividades propuestas de la guía.

En el segundo epígrafe se determinan dos tipos de procesos para el desarrollo de las GUIs en la *suite*: Creación de una nueva GUI y Personalización de una GUI existente.

- En el primer proceso se identifica como actividad inicial la ubicación de la GUI dentro del código fuente de la *suite*, ya que para desarrollar estas interfaces se debe primeramente estar situado en el directorio necesario, para así facilitar la adicción o modificación de archivos. Posteriormente se pasa a la creación de la GUI con GTK, permitiendo obtener un diseño visual de la funcionalidad que se adicionará y todos los elementos indispensables, para agilizar el desarrollo del código funcional de la GUI, la cual se define como tercera actividad. Por consiguiente, se realiza la integración de la GUI en la *suite* que es el paso fundamental de la guía, ya que conlleva a incluir la interfaz funcional al código fuente de LibreOffice. Para concluir se compila el código fuente, actividad común para cada uno de los procesos, que permiten la visualización y correcto funcionamiento de la GUI en la *suite*.
- En el segundo proceso se evidencia como paso inicial la ubicación de la GUI y código funcional dentro del código fuente de la *suite*, ya que sin tener identificado y posicionado la GUI que se desea adaptar, no se puede comenzar el proceso de personalización. Seguidamente se crea la GUI con GTK, la cual se define como segundo paso, ya que su creación se basa en la GUI antes identificada. Luego se pasa a la modificación del código funcional de la GUI, ya que esta actividad se apoya en los elementos insertados en el diseño visual de la interfaz. Después se suprimen todos los archivos innecesarios, porque el código fuente de LibreOffice tiene una capacidad de almacenamiento bastante grande y si no se eliminan estos archivos indispensables, el código aumentaría sin necesidad.

Como tercer epígrafe se tiene el plan de implementación donde se definen los requisitos que deben tener los desarrolladores para hacer uso de la guía.

Además, se identifica en cada uno de los procesos la entrada y salida de las actividades en cuestión, con el objetivo de mejorar la comprensión de la guía.

2.2 Desarrollo

2.2.1 Creación de una nueva GUI para la suite LibreOffice

2.2.1.1 Ubicación de la GUI dentro del código fuente de la suite

Para desarrollar una nueva GUI para LibreOffice se tiene que definir dónde estará ubicado todo el código funcional y diseño de la interfaz en el código fuente de la *suite*. Determinando como **entrada** de la actividad el tipo de aplicación para la cual se va a desarrollar la GUI en la *suite* LibreOffice.
Dicha actividad se fundamentó con el análisis del código fuente de dicha *suite*, identificando los siguientes directorios para cada una de las aplicaciones.

- **Dbaccess:** Herramientas de acceso a bases de datos, para la aplicación LibreOffice Base.
- **Sc:** Código de la aplicación LibreOffice Calc.
- **Sd:** Código de la aplicación LibreOffice Impress y LibreOffice Draw.
- **Starmath:** Código de la aplicación LibreOffice Math.
- **Sw:** Código de la aplicación LibreOffice Writer.

Se tiene como **salida** de la actividad la ubicación en el código fuente del código funcional y diseño de la GUI.

2.2.1.2 Creación de la GUI con GTK

Para la creación de una GUI en LibreOffice primeramente se tiene como **entrada** de dicha actividad la funcionalidad a insertar en LibreOffice. Donde en ella se identifica que son GUIs que utilizan un estándar “ui” basado en el lenguaje de programación XML, por lo que su desarrollo se puede realizar de dos formas:

- Con la herramienta Glade
- Con el lenguaje de programación XML

Para que se tenga un desarrollo menos complejo y más cómodo se recomienda utilizar la herramienta Glade, que brinda facilidad y agilidad en la realización de las GUIs, ya que cuenta con una paleta de trabajo conformada por una gran variedad de *widgets* destinados al diseño. Esta relación de *widgets* se puede identificar concretamente en los **Anexos 2**.
Se define como **salida** de la actividad el diseño de la GUI correspondiente a la funcionalidad a insertar.

2.2.1.3 Desarrollo del código funcional de la GUI

Para el desarrollo del código funcional de la GUI en LibreOffice se evidencia como **entrada** de la actividad el diseño de la GUI correspondiente a la funcionalidad a insertar. Donde se tiene que para la implementación de este código se utiliza como lenguaje de programación C++, utilizando dos tipos de archivos para su creación. Estos archivos se identifican como:

- .hxx: Es donde se tiene la relación de todos punteros a los *widgets* con que cuenta la interfaz que necesitan ser funcionales, es decir, que se le debe implementar alguna funcionalidad, como los botones, *checkbox* (botón de verificación), *radiobutton* (botón de opción) y declarar sus respectivos eventos.
- .cxx: Es donde se implementan todas las funcionalidades de cada *widgets* de la interfaz a través de algún evento.

Para la implementación de estos archivos se identificaron varios aspectos fundamentales a tener en cuenta:

1. Creación de los punteros de los *widgets* de la interfaz en el archivo .hxx :

   <nombre_widget>* **<variables_widget>;**

   - nombre_widget: Se especifica según el tipo de *widget* utilizado en GTK pero el nombre que se pone es el correspondiente con el de la VCL.

     Nota: Ver tabla de equivalencia entre la biblioteca GTK y VCL. **(Anexo 1)**

   - variable_widget: Se especifica la variable que identificará a ese *widget* en todo el código.

2. Declaración de los eventos de la interfaz en el archivo .hxx:

   DECL_LINK(**<nombre_evento>**, **<tipo_método>** *);

   - nombre_evento: Se especifica el nombre que va a tener el evento de cada botón.
   - tipo_método: Se especifica el tipo de método (void, bool).

3. Creación del constructor de la clase en el archivo .cxx.

   <nombre_clase_principal>::**<nombre_clase_principal>**(**<parámetros>**):
   ModalDialog (**<variable_diálogo>**, "**<nombre_interfaz>**", "**<dirección_interfaz>**")

Nota: Entre los parámetros se debe tener en cuenta la variable que hace llamada al diálogo: Window ***<variable_diálogo>**, ya que es la variable que interviene en el método de ejecución de la interfaz

- variable_diálogo: Se especifica el tipo de variable de la llamada a la interfaz.
- nombre_interfaz: Se especifica el nombre que tiene el archivo .ui creado.
- dirección_interfaz: Se inserta la dirección de ubicación del archivo .ui dentro del código fuente de la aplicación.

4. Obtención de los punteros de los *widgets* creados en .hxx:

```
get(<variable_widget>, "<id_interfaz>");
```

- variable_widget: Variable asignada en el archivo .hxx para determinado *widget*.
- id_interfaz: es el identificador usado cuando se crea la interfaz con Glade.

5. Asignación del evento a cada *widget* de la interfaz:

```
<variable_widget> → <tipo_evento>( LINK(this, <nombre_clase_principal>, <nombre_evento>));
```

- tipo_evento: Se puede especificar de varias formas SetClickHdl (clic). SetSelectHdl (selección), SetModifyHdl (modificación).
- nombre_evento: Se asigna el nombre del evento identificado en el archivo .hxx.

6. Implementación de las funcionalidades de los *widgets*.

```
IMPL_LINK_NOARG(<nombre_clase_principal>, <nombre_evento>)
```

Se determina como **salida** para esta actividad la GUI funcional desarrollada para la funcionalidad.

2.2.1.4 Integración de la GUI en la suite

Para la integración de la GUI en la *suite*, teniendo como **entrada** de la actividad la GUI funcional desarrollada para la funcionalidad, se deben modificar 13 archivos dentro del directorio de la aplicación del código fuente de la *suite.* Estos archivos permiten de manera general definir: el método de ejecución de la GUI, si va a pertenecer a la barra de menú, de estado o de herramientas, en caso de que pertenezca a la barra de herramientas dónde va a estar ubicado y qué nombre llevará en dicha ubicación.

A continuación se identifican cada unos de los archivos a modificar para cada una de las aplicaciones de la *suite*.

1. Archivo en el cual se hace constancia de la existencia de la nueva GUI creada para la aplicación, donde se le debe asignar un número de ranura que la identifique. Este archivo está ubicado en: <directorio>/inc/. Definiendo para cada aplicación su archivo correspondiente:
   - Sw → cmdid.hrc
   - Sc → sc.hrc
   - Sd → app.hrc
   - Starmath → starmath.hrc

```
#define <nombre_ranura>          (<ranura_directorio>_START+ID)
```

2. El archivo .xml ubicado en el directorio "MenuBar", permite posicionar la GUI dentro de la barra de menú de la aplicación; teniendo en cuenta que para añadirlas a dicha barra existen varias etiquetas con diferentes funcionalidades:
   - Para añadir un nuevo campo en la barra de menú.

```
<menu:menu menu:id=".uno:<campo_insertado>">
```

   - Para añadir un nuevo ítem en el campo creado en la barra de menú.

```
<menu:menuitem menu:id=".uno:<nombre_interfaz>"/>
```

   - Para añadir un separador entre los ítems en la barra de menú.

```
<menu:menuseparator/>
```

   - Para añadir un nuevo menú a partir de un ítem en la barra de menú.

```
<menu:menupopup>
```

3. Archivos ubicados en: <directorio>/sdi/, que permiten definir el método de ejecución y de estado de la GUI dentro de la *suite*. Para cada una de las aplicaciones se definen como:
   - Sw → view.sdi
   - Sc → docsh.sdi
   - Sd → drviewsh.sdi
   - Starmath → smslots.sdi

```
<nombre_ranura> [
<aspectos> ]
```

Aspectos:

- ExecMethod: Método de ejecución (Execute, FuTemporary).
- StateMethod: Método de estado (GetState, GetMenuState).

4. Archivos ubicados en: <directorio>/sdi/, que brindan opciones para personalizar la GUI de la *suite*, contando con los siguientes archivos para cada una de las aplicaciones:

- Sw → swriter.sdi
- Sc → scalc.sdi
- Sd → sdraw.sdi
- Starmath → smath.sdi

SfxVoidItem **<nombre_interfaz> <nombre_ranura>** () [**<Elementos_a_configurar>**]

Se tienen varios elementos a configurar, donde, en su mayoría son variables booleanas (*True* o *False*/Verdadero o Falso):

- AutoUpdate: Actualización automática
- FastCall: Llamada rápida
- HasCoreId: Código identificado
- HasDialog: Interfaz
- ReadOnlyDoc: Documento de sólo lectura
- Toggle: Alternar
- Container: Contenedor
- RecordAbsolute: Récord absoluto
- AccelConfig: Configuración de aceleradores
- MenuConfig: Configuración de menú
- StatusBarConfig: Configuración de barra de estado
- ToolBoxConfig: Configuración de la caja de herramientas
- GroupId: Identificación del Grupo

5. Archivo .h ubicado en: <directorio>/inc/, donde se define un nombre para el comando que hace llamada a la GUI en la barra de menú y en los archivos de configuración. Para cada aplicación se tienen identificados los siguientes archivos:

- Sw → swcommands.h
- Sc → sccommands.h
- Sd → sdcommands.h

- Starmath → smcommands.h

```
#define CMD_<nombre_ranura>      ".uno:<nombre_interfaz>"
```

6. Estos archivos identificados en: <directorio>/source/ui/, permiten definir una función en la cual se especifica la cantidad de parámetros y la estructura del constructor de la GUI. Para eso se tienen que modificar los siguientes archivos relacionados con cada aplicación:
   - Sw → dialog/swdlgfact.hxx – swdlgfact.cxx – swabstdlg.hxx
   - Sc → attrdlg/scdlgfact.hxx – scdlgfact.cxx – scabstdlg.hxx
   - Sd → dlg/sddlgfact.hxx – sddlgfact.cxx – sdabstdlg.hxx

   En el archivo .hxx:

```
virtual VclAbstractDialog* Create<nombre_clase_principal> (<parámetros>);
```

   En el archivo .cxx:

```
VclAbstractDialog * <directorio>AbstractDialogFactory_Impl::
Create<nombre_clase_principal> (<parámetros>)
```

   Nota: No se puede olvidar incluir el archivo .hxx de la interfaz que se está insertando en este archivo.

```
#include "<nombre_interfaz>.hxx"
```

7. El archivo que se encuentra ubicado en <directorio>/source/, se implementa el método de ejecución de la GUI declarado en archivos anteriores, donde se identifica para cada aplicación los siguientes:
   - Sw → ui/uiview/view2.cxx
   - Sc → ui/view/cellsh3.cxx
   - Sd → ui/view/drviews2.cxx
   - Starmath → view.cxx

```
case <nombre_ranura>
<funcionalidades>
break;
```

8. Los archivos de configuración ubicados en: officecfg/registry/data/org/openoffice/Office/UI, permiten darle nombre al nodo donde aparecerá la GUI en la barra de menú. Se definieron los siguientes archivos para cada una de las aplicaciones:

- Sw → WriterCommands.xcu
- Sc → CalcCommands.xcu
- Sd → DrawImpressCommands.xcu
- Starmath → MathCommands.xcu

```
<node oor:name=".uno:<nombre_interfaz>" oor:op="replace">
    <prop oor:name="Label" oor:type="xs:string">
        <value xml:lang="en-US"><nombre_nodo_agregado></value>
    </prop>
    <prop oor:name="Properties" oor:type="xs:int">
        <value>1</value>
    </prop>
</node>
```

9. Los archivos .mk permiten la compilación de los archivos en la suite, donde se concretan que para el código funcional (.cxx) se tiene el archivo Library_**<directorio>**ui.mk y para las GUI (.ui) el archivo UIConfig_**<módulo>**.mk. Se identifican los siguientes módulos para cada aplicación:
    - Sw → swriter
    - Sc → scalc
    - Sd → simpress – sdraw
    - Starmarth → smath

Se define como **salida** de dicha actividad la GUI funcional integrada a la *suite* LibreOffice.

2.2.1.5 Compilación del código fuente de la suite LibreOffice

Después de tener todos los archivos ubicados en sus directorios e insertados en los archivos de compilación, se da paso a la última actividad de la guía, la compilación de código de LibreOffice, que permite la visualización de la GUI en la suite. Para llevar a cabo esta técnica se utilizará la guía de compilación para la *suite* LibreOffice desarrollada en el centro (29). Definiendo como **entrada** de la actividad la GUI funcional integrada a la *suite*, y como **salida** la visualización de la GUI en LibreOffice.

2.2.2 Personalización de una GUI existente en la suite LibreOffice

2.2.2.1 Ubicación de la GUI y código funcional dentro del código fuente de la suite

En el código fuente de la *suite* LibreOffice existen GUIs que están desarrolladas con la biblioteca VCL con formatos .src/.hrc, las cuales necesitan ser convertidas en GUIs de la biblioteca GTK de extensión .ui, por la facilidad que brinda esta en futuras modificaciones. Identificando como **entrada** de la actividad la funcionalidad a personalizar desarrollada con la biblioteca VCL, y como **salida** la ubicación de la GUI en formato .src/.hrc y código funcional correspondiente dentro del código fuente de LibreOffice.

2.2.2.2 Creación de la GUI con GTK

En la creación de una GUI se determina como **entrada** de la actividad la GUI con formato .src/.hrc de la funcionalidad a personalizar. Se desarrolla con la herramienta Glade apoyándose del formato .src/.hrc ya existente, haciendo uso de la misma relación de *widgets*; obteniendo como **salida** el diseño de la GUI desarrollada con la biblioteca GTK en la herramienta Glade. Para obtener conocimiento de la correspondencia de los *widgets* VCL con los de GTK ver **Anexo 1**.

2.2.2.3 Modificación del código funcional de la GUI

Para modificar el código funcional de una GUI que se desea personalizar se tiene como **entrada** de la actividad el código funcional de la funcionalidad a personalizar. Donde solo se tienen que modificar varios aspectos que permitan que el código trabaje bajo la biblioteca GTK y no la VCL:

- Los punteros de los *widgets* que están implementado, donde solo se declaran los que se les implementa alguna funcionalidad.

  <nombre_widget>* **<variables_widget>;**

- El constructor definido ya que se le tiene que pasar la Gui creada con GTK.

  ModalDialog (**<variable_diálogo>**, "**<nombre_interfaz>**", "**<dirección_interfaz>**")

- La llamada de los *widgets* declarados en el archivo .hxx, señalando que una vez cambiada la variable del *widget*, el mismo debe ser modificado en todo el código.

  get(**<variable_widget>**, "**<id_interfaz>**");

Se definen como **salida** de la actividad la GUI funcional personalizada de la funcionalidad.

2.2.2.4 Supresión de los archivos innecesarios

Una vez diseñado y modificado el código funcional de la GUI, se tienen que eliminar los archivos que no son necesarios en el código fuente de LibreOffice, como los archivos .src/.hrc ya que han sido sustituidos por la nueva GUI. Además, se borra la llamada de la misma en el archivo de compilación de los .src, evitando que se produzca algún error en la compilación de la *suite.* Esto conlleva a que se determine como **entrada** de la actividad la GUI con formato .src/.hrc de una funcionalidad a personalizar, y como **salida** la eliminación de la GUI con formato .src/.hrc de una funcionalidad a personalizar.

2.2.2.5 Compilación del código fuente de la suite LibreOffice

Después de tener todos archivos necesarios modificados, se da paso a la compilación del código utilizando la técnica definida en el epígrafe 2.2.1.5. Se define como **entrada** de la actividad la GUI funcional personalizada de la funcionalidad y como **salida** la visualización de la GUI personalizada en LibreOffice.

2.3 Plan de implementación

Para hacer uso de la guía se debe tener conocimiento previo de C++, del trabajo con la herramienta Glade; además tener todo el código fuente de LibreOffice descargado.

Conclusión parcial

Finalizado este capítulo, se obtuvo una guía para la implementación de GUIs para LibreOffice, en la cual se brinda un mecanismo que permite a los desarrolladores llevar a cabo la creación de GUIs para cualquier aplicación de la *suite*, donde se concretaron aspectos relacionados con ubicación en el código fuente, diseño, implementación, integración y compilación.

Capítulo 3. Prueba de la guía confeccionada

En este capítulo se pone en práctica la guía confeccionada, el desarrollo de cada uno de los pasos definidos en la misma se lleva a cabo a través de la integración del Asistente para Calendarios de LibreOffice Writer y el Conversor Chino de la *suite*, con el objetivo de validar su correcto funcionamiento.

3.1 Creación de la GUI del Asistente para Calendario para LibreOffice Writer

El Asistente para Calendarios de LibreOffice Writer cuenta con GUIs desarrolladas con la biblioteca *Swing* para el lenguaje de programación Java, donde en este epígrafe se lleva la GUI principal del asistente a la biblioteca GTK, utilizándola como caso de estudio para probar la guía confeccionada.

3.1.1 Ubicación del Asistente para Calendario para LibreOffice Writer

La GUI creada se almacenará dentro del código fuente en el directorio de la aplicación LibreOffice Writer (sw/uiconfig) (**Ver Figura 2**) y el código funcional en sw/source/ui/asistente (**Ver Figura 3**).

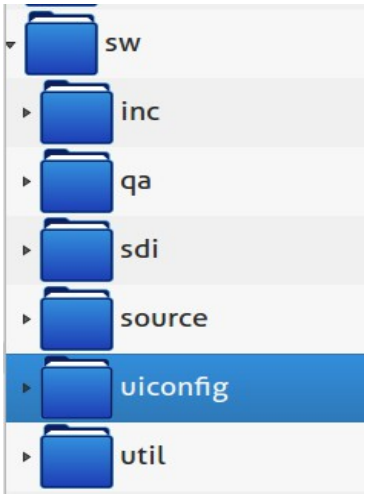

Figura 2: *Contenedor de las GUIs .ui de LibreOffice Writer.*

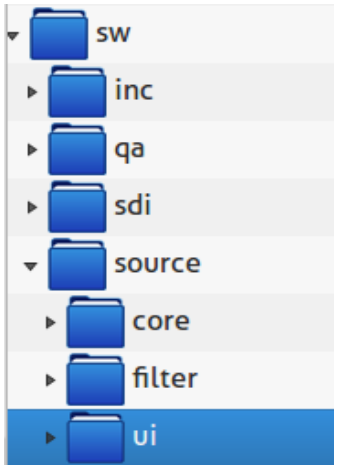

Figura 3: *Contenedor del código funcional de las GUIs de LibreOffice Writer.*

3.1.2 Creación de la GUI del Asistente para Calendario con GTK

1. Crear un nuevo proyecto y guardarlo dentro del directorio “ui” con el nombre: Asistente.ui **(Ver Figura 4)**.

2. Seleccionar el elemento ventana (GtkWindows) y modificar sus propiedades: Comunes/Visible: Si, General/Nombre: Asistente, General/Título_de_la_Ventana: Asistente de Calendario **(Ver Figura 5)**.

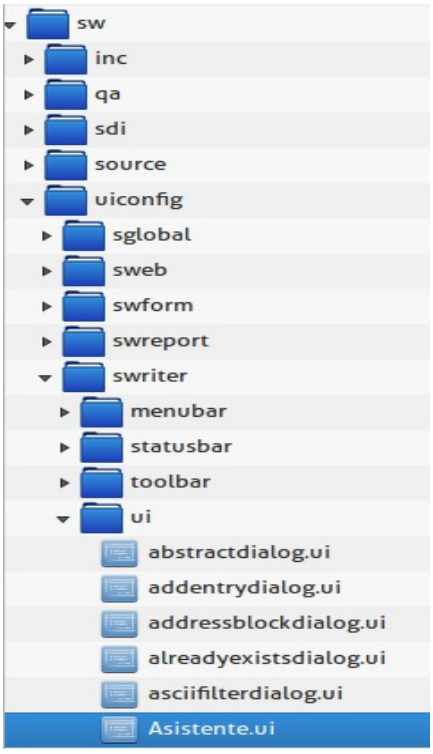

Figura 4: *Interfaz del asistente almacenada en el directorio de LibreOffice Writer.*

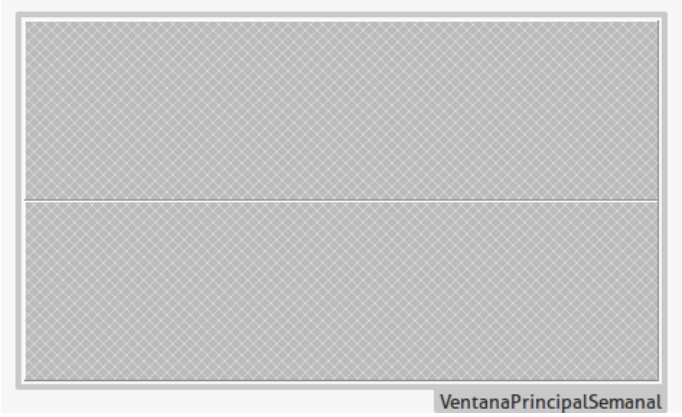

Figura 5: *Ventana con GtkBox1.*

3. Seleccionar el elemento caja (GtkBox1) y dar como valor de ítem 2, para que se divida en dos secciones. La primera sección añadir un contenedor de pestañas (GtkNotebook) y en el segundo una caja de botones (GtkButtonBox).
    - GtkNotebook: Expandir el contenedor de pestañas: Empaquetado/Expander: Si. Modificar las etiquetas (GtkLabel): En la primera: General/Nombre: label_plantilla, General/Edita_la_apariencia_de_la_etiqueta: Plantillas. En la segunda: General/Nombre: label_tarea, General/Edita_la_apariencia_de_la_etiqueta: Tareas.

      Nota: Como aparecen tres pestañas por defecto en el contenedor, se tiene que eliminar una para crear esta interfaz.

    - GtkButtonBox: Valor de ítem:3, en General/Estilo_de_la_distribución: *Spread*, General/Distribución: horizontal y añadir tres botones: Plantillas, Tareas y Generar Calendario **(Ver Figura 7)**.

4. Dentro del GtkNotebook en la sección de **Plantillas**, añadir un GtkBox2 con: Valor de ítem: 2, Expandir: Si y Orientación: Horizontal.
    - En el primer ítem añadir otro GtkBox3 con tres ítems:

      1º- Insertar una imagen (GtkImage) y añadirla en General/File_Name. Ajustar la margen en Comunes: Margen_Izquierdo:10, Margen_Arriba:5, Margen_Abajo:5 **(Ver Figura 6)**.

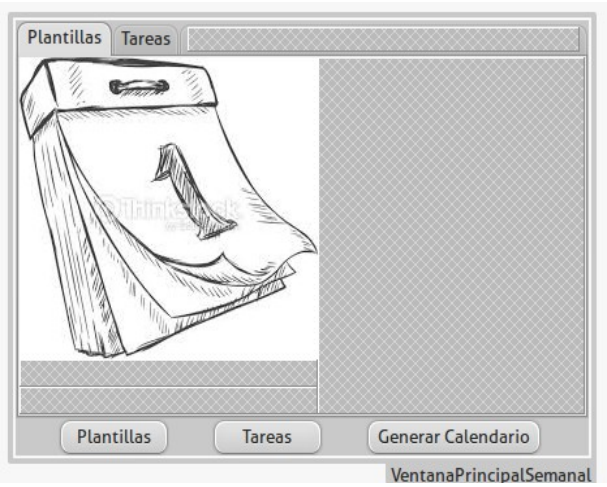

Figura 7: *Ventana con GtkImage.*

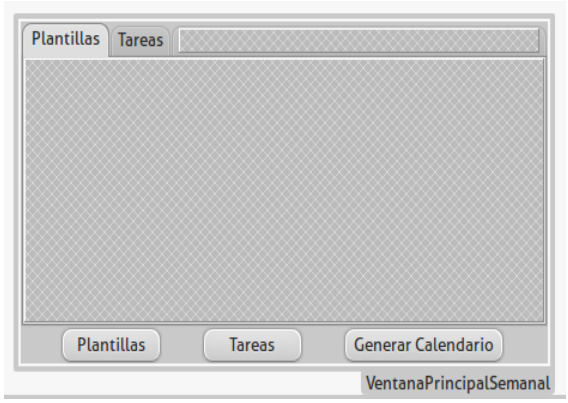

Figura 6: *Ventana con GtkNotebook y GtkButtonBox.*

2º- Insertar un GtkBox4 con valor de ítems: 4, donde:

- GtkLabel: Disposición con Margen_Derecho:130, Margen_Arriba:5, Margen_Abajo:5.
- GtkRadioButton: Etiqueta: Calendario Semanal, Nombre: radio_sem, Margen_Izquierdo: 90, Margen_Arriba: 5, Margen_Abajo: 5.
- GtkRadioButton: Etiqueta: Calendario Mensual, Nombre: radio_men, Margen_Izquierdo: 90, Margen_Arriba: 5, Margen_Abajo: 5 y en General/Grupo: radio_sem.
- GtkRadioButton: Etiqueta: Calendario Anual, Nombre: radio_an, Margen_Izquierdo: 90, Margen_Arriba: 5, Margen_Abajo: 5 y en General/Grupo: radio_sem. **(Ver Figura 8)**

3º- Insertar GtkBox5 con valor de ítem: 4, donde:

- GtkLabel: Etiqueta: Seleccione la fecha inicial, Margen_Derecho: 50, Margen_Arriba: 5, Margen_Abajo: 5.

- GtkBox6: Valor de ítem: 2, Orientación: Horizontal, Margen_Derecho: 30, Margen_Arriba: 5, Margen_Abajo: 5. En el primero se añade una entrada de texto (GtkEntry). En el segundo un GtkButton: Nombre: boton_cal1, General/Añadir_contenido_de_botón_personalizado y se añade un GtkImage de un ícono de calendario.
- GtkLabel: Etiqueta: Seleccione la fecha final, Margen_Derecho: 53, Margen_Arriba: 5, Margen_Abajo: 5.
- GtkBox7: Valor de ítem: 2, Orientación: Horizontal, Margen_Derecho: 30, Margen_Arriba: 5, Margen_Abajo: 5. En el primero se añade una entrada de texto (GtkEntry). En el segundo un GtkButton: Nombre: boton_cal2, General/Añadir_contenido_de_botón_personalizado y se añade un GtkImage de un ícono de calendario. **(Ver Figura 9)**

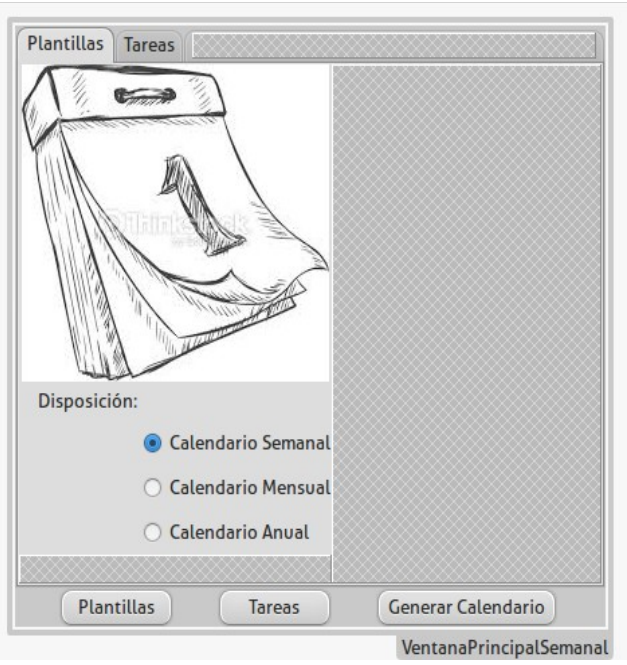

Figura 9: *Ventana con GtkRadioButton.*

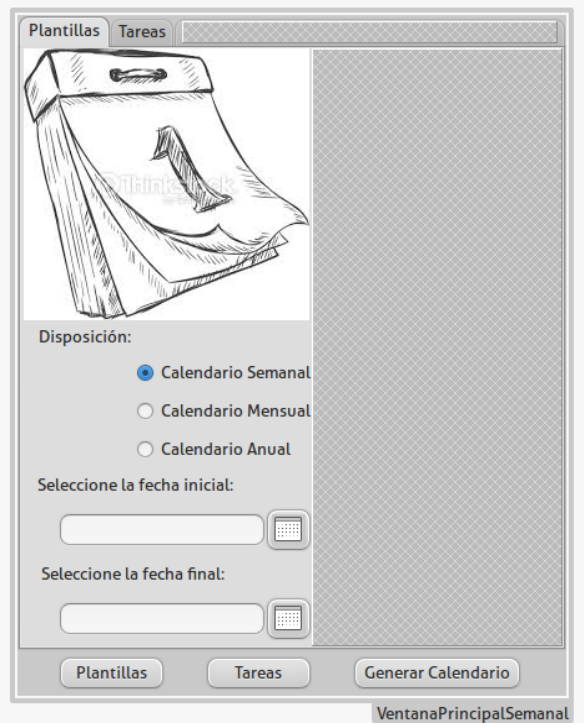

Figura 8: *Ventana con GtkEntry y GtkButton.*

- En el segundo ítem del GtkBox3 se añade un marco (GtkFrame) con Margen_Izquierdo: 5, Margen_Derecho: 5, Margen_Arriba: 5, Margen_Abajo: 5. El marco cuenta con un GtkLabel, donde se modifica: Etiqueta: Vista Previa del Calendario Semanal, Margen_Izquierdo: 30, Margen_Derecho: 30, Margen_Arriba:

5, Margen_Abajo: 5. Dentro del marco insertar un GtkImage donde se inserta una imagen de la vista previa del calendario semanal. **(Ver Figura 10)**

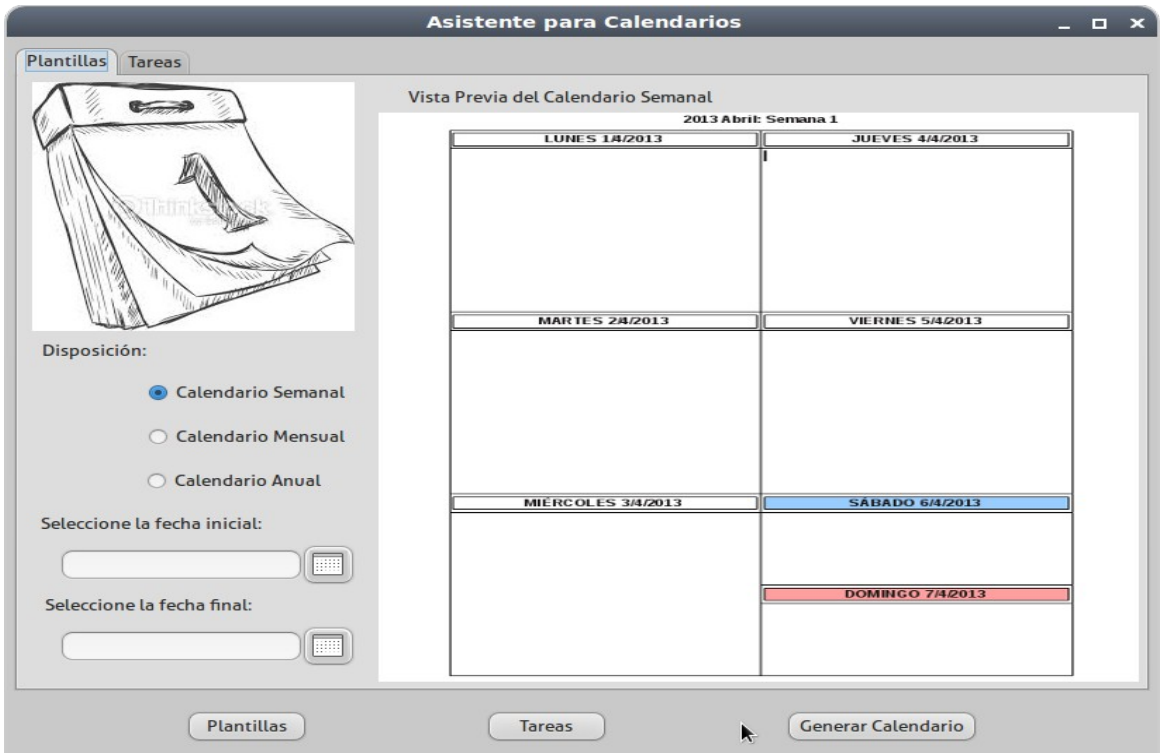

Figura 10: *Ventana Principal del Asistente para Calendario en la pestaña de Plantillas.*

5. Dentro del GtkNotebook en la sección de **Tareas**, añadir un GtkBox8 con valor de ítem: 2, donde:

1º- Insertar una ventana de desplazamiento (GtkScrolledWindow) con Margen_Arriba: 5. Dentro de esta ventana, añadir una vista de árbol (GtkTreeView), donde hay que añadir los elementos de las columnas de la vista: Edit/Jerarquía, tienen como Nombre y Título para cada una: Fecha y Hora, Título, Lugar, Repetir Tarea, Descripción.**(Ver Figura 11)**, luego añadir un GtkCellRendererText a cada cabecera y modificar sus propiedades.**(Ver Figuras 12 y 14)** Además crear una lista de almacenamiento (GtkListStore) con Nombre: ListaCalendario para las columnas de la vista y añadírsela al GtkTreeView en General/Modelo_TreeView: ListaCalendario.**(Ver Figura 13)**

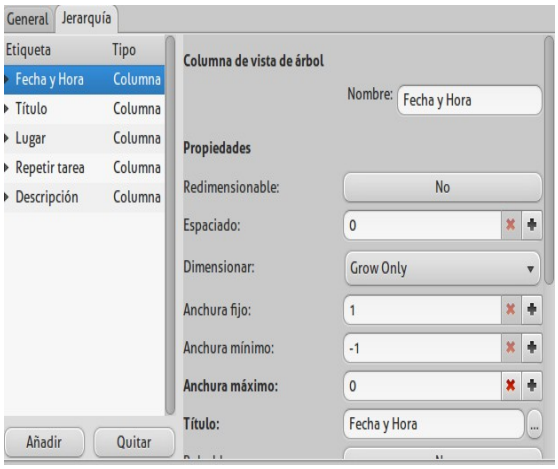

Figura 11: *Añadir columnas al GtkTreeView.*

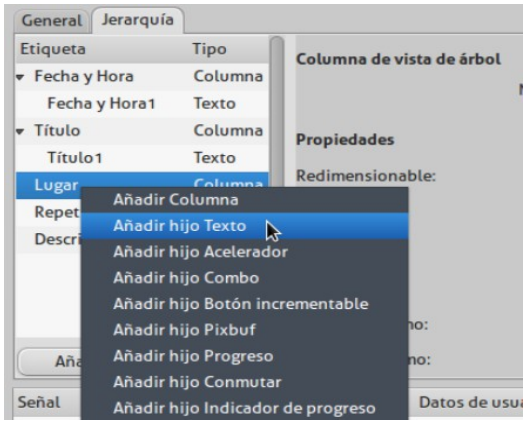

Figura 12: *Añadir GtkCellRendererText a las columnas del GtkTreeView*

Figura 14: *Editar propiedades del GtkCellRendererText.*

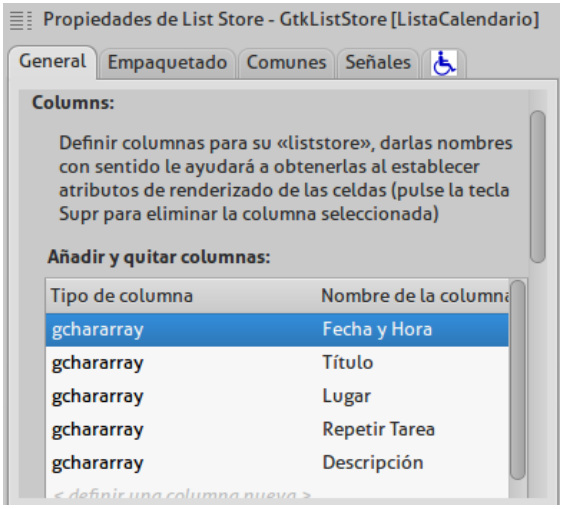

Figura 13: *Editar propiedades del GtkListStore.*

2º- Añadir una barra de herramientas (GtkToolBar) y modificar sus propiedades: General/Estilo_de_barra_de_herramientas: Icons Only, Empaquetado/Separación: 4. Ubicarse en editar la jerarquía del GtkToolBar y añadir dos botones (Añadir y Eliminar).**(Ver Figura 15)**

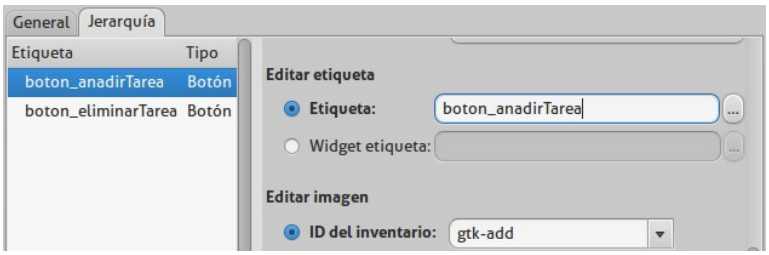

Figura 15: *Añadir botones al GtkToolBar.*

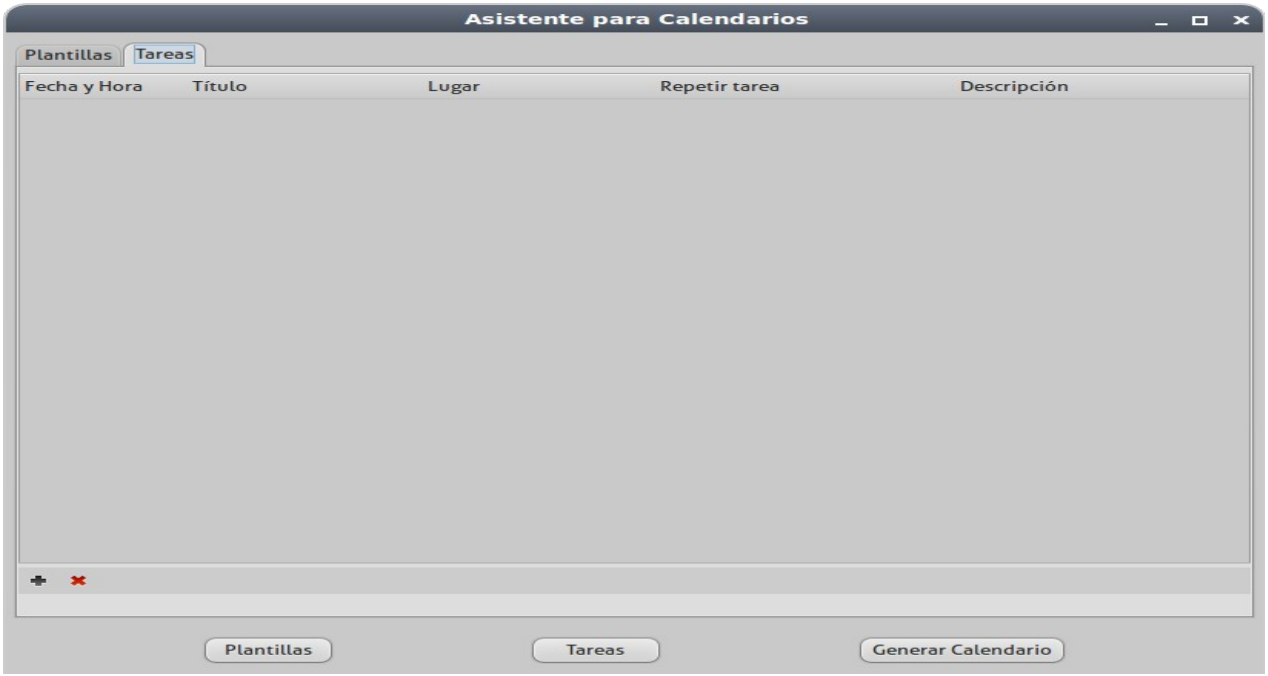

Figura 16: *Ventana Principal del Asistente para Calendario en la pestaña de Tareas.*

3.1.3 Desarrollo del código funcional de la GUI

1. Crear los archivos .hxx y .cxx. Salvarlos dentro de los archivos correspondientes.

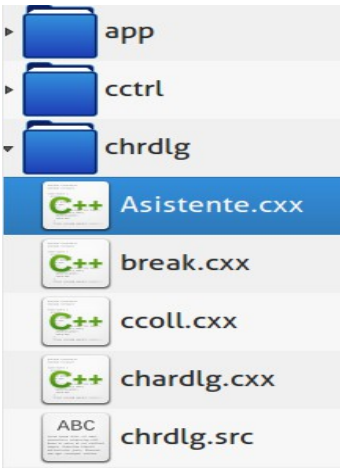

Figura 17: *Archivo .cxx de la GUI para el Asistente para Calendario.*

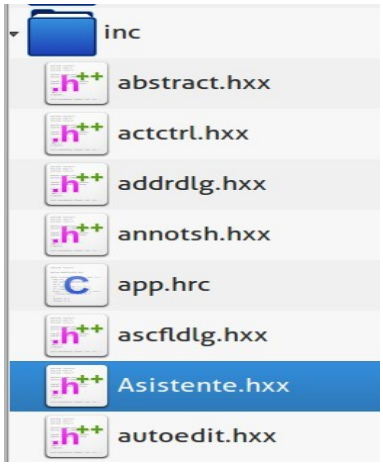

Figura 18: *Archivo .hxx de la GUI para el Asistente para Calendario.*

2. Implementar archivo .hxx.

```
#ifndef _SW_ASISTENTE_HXX
#define _SW_ASISTENTE_HXX
#include "tools/link.hxx"
#include <vcl/button.hxx>
#include <vcl/dialog.hxx>
class Window;
class SwWrtShell;
class SwPageDesc;
class SwAsistente: public ModalDialog
{
public:
        SwAsistente(Window *pParent);
        ~SwAsistente();
        virtual short Execute();
private:
            PushButton* m_pBtnGenerar;
            DECL_LINK(GenerarBtnHdl, void *);
};
#endif // _SW_ASISTENTE_HXX
```

3. Implementar archivo .cxx.

```
#include "Asistente.hxx"
```

```
#include "cmdid.h"
#include <chrdlg.hrc>
#include <sfx2/request.hxx>
#include <svl/stritem.hxx>
#include <vcl/msgbox.hxx>
SwAsistente::SwAsistente(Window *pParent)
: ModalDialog(pParent, "Asistente", "modules/swriter/ui/Asistente.ui")
  {
        get(m_pBtnGenerar, "boton_generar");
        m_pBtnGenerar->SetClickHdl(LINK(this, SwAsistente, GenerarBtnHdl));
  }
SwAsistente::~SwAsistente()
{
}
short SwAsistente::Execute()
{
  return ModalDialog::Execute();
}
IMPL_LINK_NOARG(SwAsistente, GenerarBtnHdl)
{
  InfoBox aInfo(this, OUString("Calendario Generado"));
  aInfo.Execute();
  return 0;
}
```

3.1.4 Integración de la GUI en la suite

1. Guardar el archivo .ui (sw/uiconfig/swriter/ui/Asistente.ui), .cxx (sw/source/ui/chrdlg) y .hxx (sw/source/ui/inc).
2. Modificar el archivo sw/inc/cmdid.h.

```
#define SID_ASISTENTE          (FN_FILE + 100 )
```

3. Modificar la barra de menú sw/uiconfig/swriter/menubar/menubar.xml.

```
<menu:menubar xmlns:menu="http://openoffice.org/2001/menu"
menu:id="menubar">
 <menu:menu menu:id=".uno:PickList">
```

```
<menu:menupopup>
 <menu:menuitem menu:id=".uno:AddDirect"/>
 <menu:menuitem menu:id=".uno:OpenFromWriter"/>
 <menu:menuitem menu:id=".uno:RecentFileList"/>
 <menu:menuseparator/>
 <menu:menuitem menu:id=".uno:AutoPilotMenu"/>
 <menu:menuitem menu:id=".uno:asistente"/>
 <menu:menuseparator/>
```

4. Modificar el archivo sw/sdi/viewsh.sdi.

```
SID_ASISTENTE [
    ExecMethod = Execute;
    StateMethod = GetState;
]
```

5. Modificar el archivo sw/sdi/swriter.sdi.

```
SfxVoidItem ASISTENTE SID_ ASISTENTE () [
    /* flags: */
    AutoUpdate = FALSE,
    Cachable = Cachable,
    FastCall = TRUE,
    HasCoreId = FALSE,
    HasDialog = TRUE,
    ReadOnlyDoc = FALSE,
    Toggle = FALSE,
    Container = FALSE,
    RecordAbsolute = FALSE,
    RecordPerItem;
    Synchron;
    /* config: */
    AccelConfig = FALSE,
    MenuConfig = TRUE,
    StatusBarConfig = FALSE,
    ToolBoxConfig = TRUE,
    GroupId = GID_FORMAT; ]
```

6. Modificar el archivo sw/inc/swcommands.h

```
#define CMD_SID_ASISTENTE      ".uno: asistente"
```

7. Modificar los archivos:

   - sw/source/ui/dialog/swdlgfact.hxx

```
virtual VclAbstractDialog *        CreateSwAsistente(Window *pParent);
```

   - sw/source/ui/dialog/swdlgfact.cxx

```
#include "Asistente.hxx"

VclAbstractDialog * SwAbstractDialogFactory_Impl::CreateSwAsistente(Window*
pParent)
{
   Dialog* pDlg = new SwAsistente( pParent );
   return new VclAbstractDialog_Impl( pDlg );
}
```

8. Modificar el archivo sw/inc/swabstdlg.hxx.

```
virtual VclAbstractDialog *        CreateSwAsistente (Window *pParent) = 0;
```

9. Modificar el archivo sw/source/ui/uiview/view2.cxx

```
case SID_ASISTENTE:
      {
         SfxViewFrame* pTmpFrame = GetViewFrame();
         SwAbstractDialogFactory* pFact = SwAbstractDialogFactory::Create();
         OSL_ENSURE(pFact, "Dialogdiet fail!");
            VclAbstractDialog* pDlg = pFact→CreateSwAsistente(&pTmpFrame→
GetWindow());
         if (pDlg)
         {
               pDlg→Execute();
               delete pDlg;
         }
         rReq.Ignore ();
      }
      break;
```

10. Modificar el archivo officecfg/registry/data/org/openoffice/Office/UI/WriterCommands.xcu.

```
<node oor:name=".uno:asistente" oor:op="replace">
    <prop oor:name="Label" oor:type="xs:string">
        <value xml:lang="en-US">Calendar Wizard</value>
    </prop>
    <prop oor:name="Properties" oor:type="xs:int">
        <value>1</value>
    </prop>
</node>
```

11. Modificar los archivos:

- sw/Library_swui.mk.

  sw/source/ui/chrdlg/Asistente \
- sw/UIConfig_swritermk.

  sw/uiconfig/swriter/ui/Asistente \

3.1.5 Compilación del código fuente de la suite LibreOffice

Una vez compilado el código fuente de la *suite* con la interfaz insertada, se ejecuta la *suite* y queda todo integrado para hacer uso de dicha GUI **(Ver Figura 19 y 20)**.

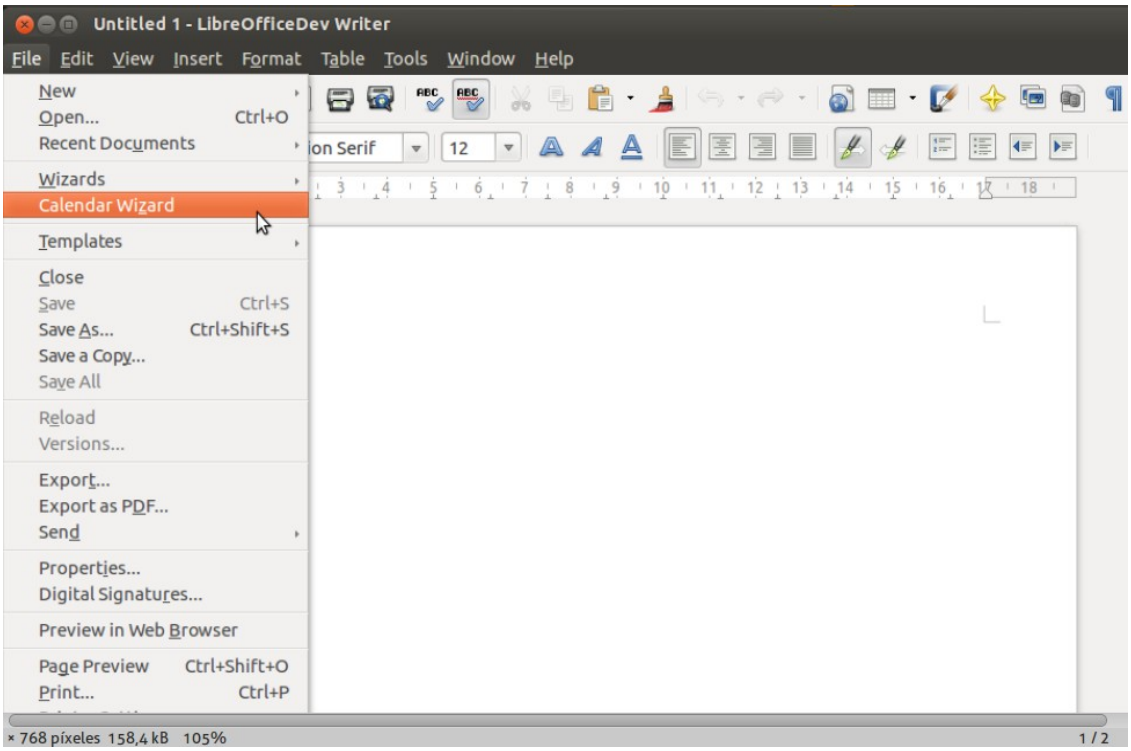

Figura 19: *Asistente para Calendario integrado en el menú "Archivo".*

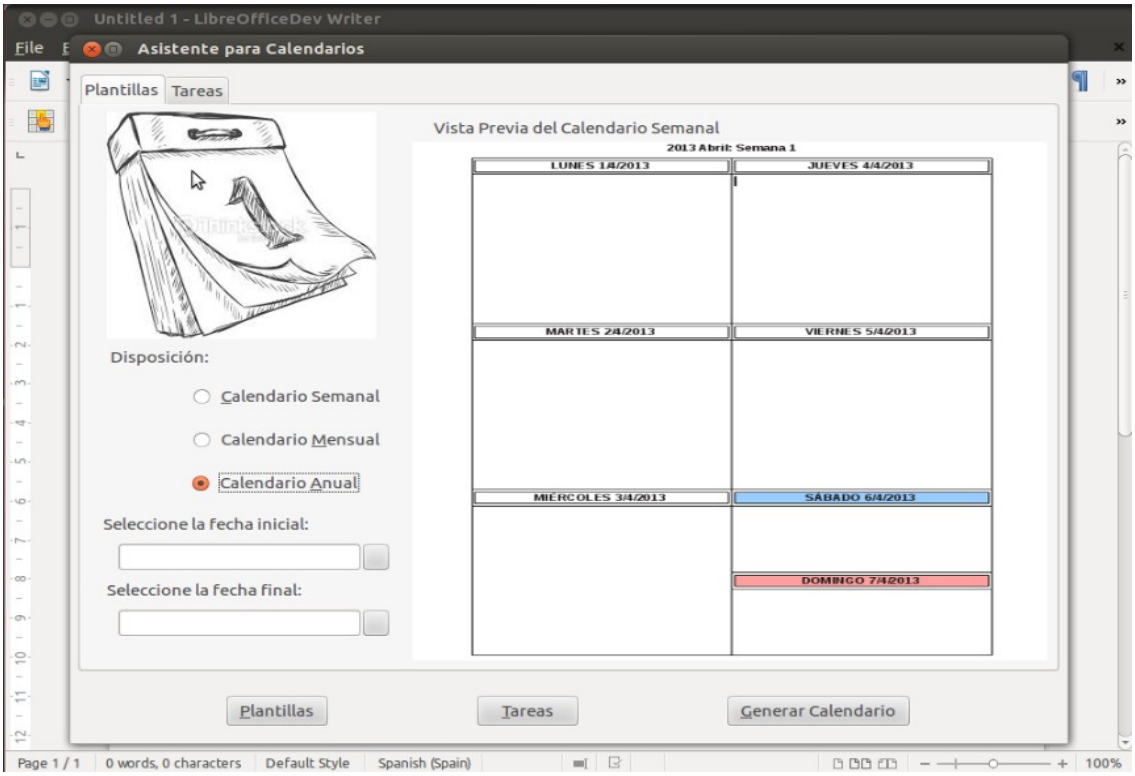

Figura 20: *Asistente para Calendario para la suite LibreOffice.*

3.2 Personalización de la GUI del Conversor Chino existente en la suite LibreOffice

En el proceso de prueba de la personalización de la GUI se utiliza el Conversor Chino existente en la *suite* que se encuentra desarrollado con la biblioteca VCL, la cual a través de la guía confeccionada se transforma a la biblioteca GTK. En este epígrafe se mostrarán los elementos del código que deben ser eliminados (en color gris) e insertados (en color negro).

3.2.1 Ubicación del Conversor Chino dentro del código fuente de la suite

1. Ubicar los archivos con formato .hrc/.src.
    - svx/source/unodialogs/textconversiondlgs/chinese_dialogs.src
    - svx/source/unodialogs/textconversiondlgs/chinese_translationdialog.hrc
2. Ubicar los archivos .hxx y .cxx de la GUI.
    - svx/source/unodialogs/textconversiondlgs/chinese_translationdialog.cxx
    - svx/source/unodialogs/textconversiondlgs/chinese_translationdialog.hxx

3.2.2 Creación de la GUI del Conversor Chino con GTK

1. Crear la GUI en Glade.

Figura 21: *La GUI del Conversor Chino creado con Glade*

2. Guardar la GUI creada en:
    - Directorio uiconfig: svx/uiconfig/ui/chineseconversiondialog
    - Archivo compilación: svx/UIConfig_svx.mk

3.2.3 Modificación del código funcional de la GUI

1. Dentro del archivo .hxx, modificar los *widgets*.

```
private:
        RadioButton m_aRB_To_Simplified;
        RadioButton m_aRB_To_Traditional;
        RadioButton* m_pRB_To_Simplified;
        RadioButton* m_pRB_To_Traditional;
        CheckBox m_aCB_Translate_Commonterms;
        PushButton m_aPB_Editterms;
        CheckBox* m_pCB_Translate_Commonterms;
        PushButton* m_pPB_Editterms;
        OKButton m_aBP_OK;
        OKButton* m_pBP_OK;
```

2. Dentro del archivo .cxx:
    - Constructor:

```
ChineseTranslationDialog::ChineseTranslationDialog( Window* pParent )
```

```
: ModalDialog( pParent,
TextConversionDlgs_ResId( DLG_CHINESETRANSLATION ) )
: ModalDialog(pParent, "ChineseConversionDialog",
"svx/ui/chineseconversiondialog.ui")
```

- Llamar a los *widgets* declarados en .hxx y eliminar la llamada a recursos:

```
ChineseTranslationDialog::ChineseTranslationDialog( Window* pParent )
: ModalDialog(pParent, "ChineseConversionDialog",
"svx/ui/chineseconversiondialog.ui"),
m_aRB_To_Simplified(this,TextConversionDlgs_ResId( RB_TO_SIMPLIFIED )),
m_aRB_To_Traditional(this,TextConversionDlgs_ResId( RB_TO_TRADITIONAL)),
m_aCB_Translate_Commonterms(this,TextConversionDlgs_ResId( CB_TRANSLA
TE_COMMONTERMS)),
m_aPB_Editterms( this, TextConversionDlgs_ResId( PB_EDITTERMS)),
m_pDictionaryDialog(0)
{
get(m_pBP_OK, "ok");
get(m_pPB_Editterms, "editterms");
get(m_pRB_To_Simplified, "tosimplified");
get(m_pRB_To_Traditional, "totraditional");
get(m_pCB_Translate_Commonterms, "commonterms");
FreeResource();
```

3.2.4 Supresión de los archivos innecesarios

1. Eliminar los archivos .hrc/.src:
   - svx/source/unodialogs/textconversiondlgs/chinese_dialogs.src
   - svx/source/unodialogs/textconversiondlgs/chinese_translationdialog.hrc
2. Eliminar el enlace del archivo .src:
   - svx/AllLangResTarget_svx.mk.

3.2.5 Compilación del código fuente de la suite LibreOffice

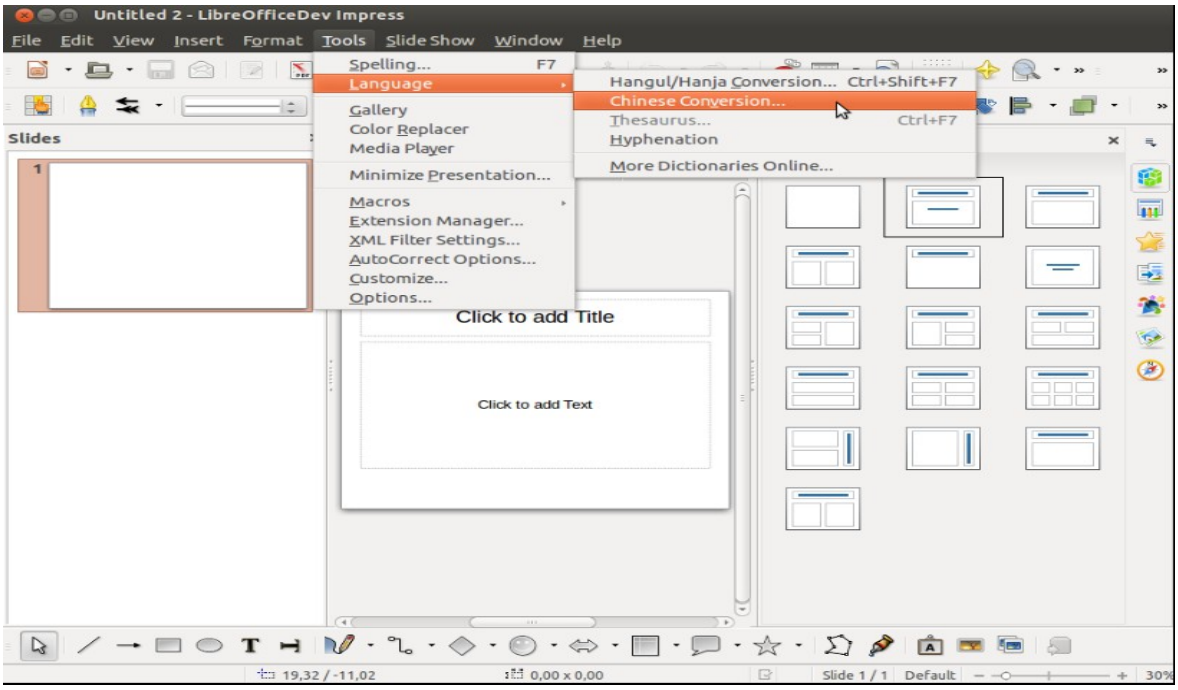

Figura 22: *Conversor Chino integrado en el menú "Herramientas"*

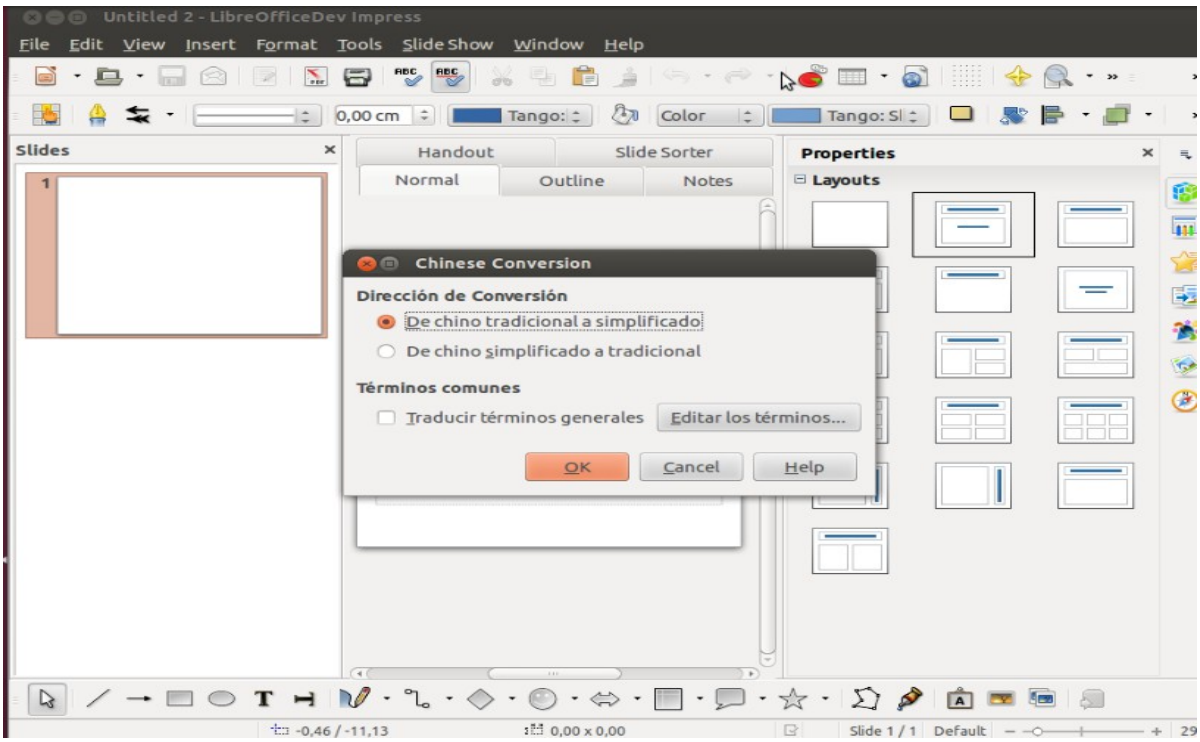

Figura 23: *Conversor Chino integrado en la suite LibreOffice*

3.3 Conclusión parcial

La guía de implementación de GUIs se probó a través del Asistente para Calendarios de LibreOffice Writer y el Conversor Chino de la *suite*, con lo cual se constató el correcto funcionamiento de la misma. De esta forma, se verificó el cumplimento del objetivo general de la presente investigación.

Conclusiones

El desarrollo del presente trabajo de diploma permitió confeccionar una guía de implementación para la *suite* LibreOffice, dando cumplimiento a los objetivos trazados, destacándose de manera general los siguientes aspectos:

- Se realizó un estudio del código fuente de la *suite* LibreOffice y de diferentes bibliotecas para el desarrollo de GUIs en entornos libres, arrojando como resultados la selección de la biblioteca que se utiliza en LibreOffice para la creación de las GUIs.
- La correcta utilización de las herramientas y lenguajes descritos, hicieron posible la obtención del diseño e implementación de forma tal que las GUIs se integraran fácilmente a la *suite* LibreOffice.
- Las pruebas que se le realizaron a la guía de implementación, corroboraron el correcto funcionamiento de la misma, por lo que se puede utilizar para la creación de cualquier GUI para la *suite* LibreOffice.

Recomendación

Una vez cumplido con el objetivo general propuesto, se recomienda:

- Incorporar a la guía cómo integrar de una GUI para la aplicación LibreOffice Base.

Glosario de siglas y términos

Código Abierto: Es el término con el que se conoce al *software* distribuido y desarrollado libremente. El Código Abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado *Software* Libre.

GNOME: Es un entorno de escritorio e infraestructura de desarrollo para Sistemas operativos Unix y derivados *Unix* como GNU/Linux, BSD o Solaris; compuesto enteramente de *software* libre.

GNU: Es un sistema operativo desarrollado por el Proyecto GNU. Está formado en su totalidad por *software* libre. Tiene como objetivo ser un sistema de *software* completo compatible con Unix.

GPL: Es una licencia creada por la Fundación de SWL a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de *software*.

IDE: Es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

KDE. Es un proyecto de *software* libre para la creación de un entorno de escritorio e infraestructura de desarrollo para diversos Sistemas operativos como GNU/Linux, Mac OS X, Windows.

LGPL: Es una licencia de *software* creada por la Fundación de *Software* Libre que pretende garantizar la libertad de compartir y modificar el *software* cubierto por ella, asegurando que el *software* es libre para todos sus usuarios.

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de *software*, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en *Windows* en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86 (solo para equipos *Apple*) o en un *PowerPC*.

Propietaria: Según la Fundación para el *Software* Libre (FSF), es un término que se aplica a cualquier programa informático que no es libre o que sólo lo es parcialmente (semilibre), porque su redistribución o modificación está prohibida, es decir requiere permiso expreso del titular del *software*.

SWL: Es la denominación del *software* que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la Fundación de *Software* Libre, el SWL se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el *software* y distribuirlo modificado.

UTF-8: Es un formato de codificación de caracteres Unicode e ISO 10646 utilizando símbolos de longitud variable. Creado por Robert C. Pike y Kenneth L. Thompson. Está definido como estándar por la RFC 3629 de la *Internet Engineering Task Force* (IETF).1 Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes web, o cuatro en ISO 10646.

Referencias Bibliográficas

1. TECNOLOGÍA: OFIMÁTICA. [en línea] 2011. [Consulta: 11 abril 2014]. Disponible en: http://tecnologia-ofimatica.blogspot.com/2011/04/ofimatica.html.

2. SEN, T.E.T. 2012. *Software Libre y Abierto: comunidades y redes de producción digital de bienes comunes.* [en línea]. S.l.: Universidad Nacional Autónoma de México. Disponible en: http://flosshub.org/sites/flosshub.org/files/Tesis.pdf.

3. Características » LibreOffice. [en línea] 2014. [Consulta: 11 abril 2014]. Disponible en: http://es.libreoffice.org/caracteristicas/.

4. JIMÉNEZ, M. del C.R. y RIVERO, A.D. 2013. *Asistente para la elaboración de calendarios en LibreOffice Writer.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas.

5. JORQUERA, I.M.Á. 2013. Herramientas Ofimáticas Libres. *Especial de Ofimática de Comunicación y Pedagogía* [en línea]. Disponible en: http://www.centrocp.com/herramientas-libres-de-ofimatica/.

6. *Microsoft Word-mem7.doc-PROYECTO%2FCapitulo3.pdf* [en línea], [sin fecha]. S.l.: s.n. [Consulta: 2 abril 2014]. Disponible en: http://bibing.us.es/proyectos/abreproy/11300/fichero/PROYECTO%252FCapitulo3.pdf.

7. *VCL* [en línea]. [Consulta: 11 Febrero 2014]. Disponible desde: http://elvex.ugr.es/decsai/builder/intro/4.html

8. The GTK+ Project. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: http://www.gtk.org/.

9. FUNDACIÓN BOLIVARIANA DE INFORMÁTICA Y TELEMÁTICA, 2009. *Manual Glade y Anjuta.* [en línea]. S.l.: s.n. Disponible en: http://portaleducativo.edu.ve/Recursos_didacticos/manuales/documentos/Manual_Glade_Anjuta.pdf.

10. GONZÁLEZ, R.G. y GONZÁLEZ, R.S., 2009. *Comparativa entre las bibliotecas gráficas GTK+ y Qt.* [en línea]. 2009. S.l.: s.n. Disponible en: http://es.cyclopaedia.net/wiki/Qt-%28biblioteca%29.

11. Libro Gtk. [en línea] 2013. [Consulta: 27 mayo 2014]. Disponible en: http://es.scribd.com/doc/69925989/Libro-Gtk.

12. BLANCHETTE, J. y SUMMERFILED, M. 2010. *C++ GUI Programming with Qt 3.* [en línea]. S.l.: Prentice Hail. Disponible en: http://www.etnassoft.com/biblioteca/c-gui-programming-with-qt-3/.

13. About Qt (biblioteca)-Portal TOL On-Line Technology. [en línea] 2014. [Consulta: 27 mayo 2014]. Disponible en: http://articles.portal-tol.net/english-language-es/Qt%20(biblioteca).

14. *python qt.* [en línea] 2011. [Consulta: 27 mayo 2014]. Disponible en: http://es.scribd.com/doc/41258653/python-qt.

15. *wxWidgets en Españo-Capítulo 1-Introducción*. [en línea] [sin fecha]. [Consulta: 6 marzo 2014]. Disponible en: http://wxwidgets.wikispaces.com/Cap%C3%ADtulo+1+-+Introducci%C3%B3n.

16. *wxWidgets: Cross-Platform GUI Library.* [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: http://www.wxwidgets.org/.

17. NOVARA, I.P. 2009. Programación Orientada a Objetos. *Desarrollo de un proyecto con wxWidgets* [en línea]. S.l.: s.n. Disponible en: http://zinjai.sourceforge.net/wxTutorial.pdf.

18. CARRASCO, M.A.M. y QUINTANILLA, A.E.J. 2011. *Análisis comparativo de bibliotecas multiplataforma para el desarrollo de aplicaciones de escritorio, aplicado a la escuela de diseño gráfico* [en línea]. RIOBAMBA–ECUADOR: ESCUELA DE INGENIERÍA EN SISTEMAS. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/1520/1/18T00464.pdf.

19. *cplusplus.com-The C++ Resources Network*. [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.cplusplus.com/.

20. *Focus Area News | XML.org*. [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.xml.org/.

21. *Anjuta DevStudio: Integrated Development Environment*. [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.anjuta.org/.

22. *Welcome to NetBeans.* [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: https://netbeans.org/.

23. *NetBeans*. [en línea] [sin fecha]. [Consulta: 23 mayo 2014 a]. Disponible en: http://netjava.bligoo.com/netbeans.

24. *Eclipse-The Eclipse Foundation open source community website.* [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.eclipse.org/.

25. *IDE Eclipse-PicandoJava.com.* [en línea] [sin fecha]. [Consulta: 23 mayo 2014]. Disponible en: https://sites.google.com/site/picandojavacom/mundo-java/ide-eclipse.

26. *Glade-A User Interface Designer*. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: https://glade.gnome.org/.

27. JAIME, L.A.G. y ORTEGA, R.J. [sin fecha]. *Subversión. Un enfoque práctico*. [en línea]. S.l. Disponible en: http://www.webmastergranada.es/descargas/Charlas/Subversion.pdf.

28. *rapidsvn.tigris.org.* [en línea] 2009. [Consulta: 3 abril 2014]. Disponible en: http://rapidsvn.tigris.org/.

29. CÁCERES, J.C.S. 2014. *Herramienta de Empaquetamiento de LibreOffice.* [en línea]. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Univesidad de las Ciencias Informáticas. Disponible en: “soporte digital”.

30. *GTK+ 3 Reference Manual: GTK+ 3 Reference Manual*. [en línea] 2014. [Consulta: 15 mayo 2014]. Disponible en: https://developer.gnome.org/gtk3/3.10/.

Bibliografía

1. About Qt (biblioteca) - Portal TOL On-Line Technology. [en línea] 2014. [Consulta: 27 mayo 2014]. Disponible en: http://articles.portal-tol.net/english-language-es/Qt%20(biblioteca).

2. ALVAREZ, A. y VALERIO, E., 2014. *Generadores de Interfaces de Usuario: Qt Designer, NetBeans y Windows Forms Designer.* [en línea]. abril 2014. S.l.: s.n. Disponible en: http://www.di-mare.com/adolfo/cursos/2007-1/pp-GenGUI.pdf.

3. Andrés Tortolero-Página Personal. [en línea] [sin fecha]. [Consulta: 10 abril 2014]. Disponible en: http://www.ie.uia.mx/acad/atortole/progra2/gtk.html.

4. Anjuta DevStudio: Integrated Development Environment. [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.anjuta.org/.

5. Bienvenido a la Comunidad Hispana de LibreOffice » LibreOffice. [en línea] 2014. [Consulta: 22 enero 2014]. Disponible en: http://es.libreoffice.org/.

6. BLANCHETTE, J. y SUMMERFILED, M. 2010. *C++ GUI Programming with Qt 3.* [en línea]. S.l.: Prentice Hail. Disponible en: http://www.etnassoft.com/biblioteca/c-gui-programming-with-qt-3/.

7. CÁCERES, J.C.S. 2014. *Herramienta de Empaquetamiento de LibreOffice.* [en línea]. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Univesidad de las Ciencias Informáticas. Disponible en: “soporte digital”.

8. CARRASCO, M.A.M. y QUINTANILLA, A.E.J. 2011. *Análisis comparativo de bibliotecas multiplataforma para el desarrollo de aplicaciones de escritorio, aplicado a la escuela de diseño gráfico* [en línea]. RIOBAMBA–ECUADOR: ESCUELA DE INGENIERÍA EN SISTEMAS. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/1520/1/18T00464.pdf.

9. cplusplus.com-The C++ Resources Network. [en línea] 2014. [Consulta: 19 mayo 2014]. Disponible en: http://www.cplusplus.com/.

10. Programación visual de aplicaciones con C++ Builder. [en línea] 2009. [Consulta: 2 abril 2014]. Disponible en: http://elvex.ugr.es/decsai/builder/index.html.

11. Documentacion/Desarrollo/Gtk-GNOME Hispano. [en línea] 2011. [Consulta: 15 mayo 2014]. Disponible en: http://www.es.gnome.org/Documentacion/Desarrollo/Gtk.

12. FUNDACIÓN BOLIVARIANA DE INFORMÁTICA Y TELEMÁTICA, 2009. *Manual Glade y Anjuta.* [en línea]. Disponible en: http://portaleducativo.edu.ve/Recursos_didacticos/manuales/documentos/Manual_Glade_Anjuta.pdf.

13. GONZÁLEZ, R.G. y GONZÁLEZ, R.S., 2009. *Comparativa entre las bibliotecas gráficas GTK+ y Qt.* [en línea]. S.l.: s.n. Disponible en: http://es.cyclopaedia.net/wiki/Qt-%28biblioteca%29.

14. Glade-A User Interface Designer. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: https://glade.gnome.org/.

15. Glade User Interface Designer Reference Manual. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: https://developer.gnome.org/gladeui/3.12/.

16. GTK+ 3 Reference Manual: GTK+ 3 Reference Manual. [en línea] 2014. [Consulta: 15 mayo 2014]. Disponible en: https://developer.gnome.org/gtk3/3.10/.

17. JIMÉNEZ, M. del C.R. y RIVERO, A.D. 2013. *Asistente para la elaboración de calendarios en LibreOffice Writer.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas.

18. LEIGH, R., 2009. *An introduction to programming with GTK+ and Glade in ISO C, ISO C++ and Python* [en línea]. S.l.: s.n. Disponible en: http://people.debian.org/~rleigh/gtk/ogcalc/ogcalc.pdf.

19. LibreOffice Modules. [en línea] 2014. [Consulta: 10 marzo 2014]. Disponible en: http://docs.libreoffice.org/.

20. Libro Gtk. [en línea] 2013. [Consulta: 27 mayo 2014]. Disponible en: http://es.scribd.com/doc/69925989/Libro-Gtk.

21. NetBeans. [en línea] [sin fecha]. [Consulta: 23 mayo 2014 a]. Disponible en: http://netjava.bligoo.com/netbeans.

22. NOVARA, I.P. 2009. Programación Orientada a Objetos. *Desarrollo de un proyecto con wxWidgets* [en línea]. S.l.: s.n. Disponible en: http://zinjai.sourceforge.net/wxTutorial.pdf.

23. python qt. [en línea] 2011. [Consulta: 27 mayo 2014]. Disponible en: http://es.scribd.com/doc/41258653/python-qt.

24. *rapidsvn.tigris.org.* [en línea] 2009. [Consulta: 3 abril 2014]. Disponible en: http://rapidsvn.tigris.org/.

25. SEN, T.E.T. 2012. *Software Libre y Abierto: comunidades y redes de producción digital de bienes comunes.* [en línea]. S.l.: Universidad Nacional Autónoma de México. Disponible en: http://flosshub.org/sites/flosshub.org/files/Tesis.pdf.

26. The Document Foundation Wiki. [en línea] 2014. [Consulta: 14 mayo 2014]. Disponible en: https://wiki.documentfoundation.org/Main_Page.

27. The GTK+ Project. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: http://www.gtk.org/.

28. VCL. [en línea]. [Consulta: 11 febrero 2014]. Disponible en: http://elvex.ugr.es/decsai/builder/intro/4.html.

29. wxWidgets: Cross-Platform GUI Library. [en línea] 2014. [Consulta: 10 abril 2014]. Disponible en: http://www.wxwidgets.org/.

30. wxWidgets en Español-Capítulo 1-Introducción. [en línea] [sin fecha]. [Consulta: 6 marzo 2014]. Disponible en: http://wxwidgets.wikispaces.com/Cap%C3%ADtulo+1+-+Introducci%C3%B3n.

Anexos

Anexo 1: Equivalencia entre la biblioteca GTK y VCL.

Biblioteca GTK.	Biblioteca VCL.
GtkHBox	VclHBox
GtkVBox	VclVBox
GtkHButtonBox	VclVButtonBox
GtkVButtonBox	VclVButtonBox
GtkGrid	VclGrid
GtkFrame	VclFrame
GtkAlignment	VclAlignment
GtkExpander	VclExpander
GtkNoteBook	TabControl
GtkEventBox	VclEventBox
GtkSizeGroup	VclSizeGroup
GtkLabel	FixedText
GtkImage	FixedImage
GtkSeparator	FixedLine
GtkEntry	Edit
GtkDrawingArea	Windows
GtkDialog	Dialog
GtkButton	PushButton
GtkRadioButton	RadioButton
GtkCheckButton	CheckBox
GtkSpinButton	NumericField
GtkComboBox	ListBox
GtkTreeView	ListBox
GtkComboBoxText	ComboBox
GtkScrollbar	ScrollBar

GtkTextView	VclMultiLineEdit
GtkLinkButton	FixedHyperlink

Anexo 2: Paleta de trabajo de Glade.

Ícono	Nombre del *Widget*	Descripción
Acciones		
	GtkActionGroup	**Grupo de Acciones:** Organiza las acciones (GtkAction) en grupos.
	GtkAction	**Acción:** Representa operaciones que el usuario puede ejecutar. Cada acción proporciona métodos para crear iconos, elementos de menú y elementos de barra que representan en sí.
	GtkToggleAction	**Acción Alterna:** Acción que cuenta con un estado "activo" que especifica si la acción se ha comprobado o no.
	GtkRadioAction	**Acción Radio:** Acción en el cual un solo grupo puede estar activado.
	GtkRecentAction	**Acción Reciente:** Acción que representa una lista de archivos utilizados recientemente.

Tabla de Anexo 1:*Widgets de acciones (30).*

Ícono	Nombre del *Widget*	Descripción
Ventanas		
	GtkWindows	**Ventana:** Crea una ventana a la que se le puede añadir otros *widgets*.
	GtkOffscreenWindows	**Ventana *Offscreen:*** Está estrictamente destinado a ser utilizado para la obtención de *widgets* instantáneos que no son parte de una jerarquía normal de *widgets*.
	GtkDialog	**Ventana Emergente:** Crea una ventana emergente que permita al usuario entrar una pequeña cantidad de elementos, por ejemplo, para mostrar un mensaje, hacer

		una pregunta, o cualquier otra cosa que no requiere gran esfuerzo por parte del usuario.
	GtkAboutDialog	**Ventana Acerca:** Crea una ventana que brinde información de la aplicación.
	GtkColorSelectionDialog	**Ventana de Colores:** Crea una ventana predeterminada de colores.
	GtkFileChooserDialog	**Ventana Abrir/Guardar:** Crea una ventana predeterminada para abrir o salvar archivo.
	GtkFontSelectionDialog	**Ventana de Tipografía:** Crea una ventana predeterminada para darle formato al texto.
	GtkMessageDialog	**Ventana Mensaje:** Crea una ventana con una imagen que representa el tipo de mensaje (Error o pregunta), junto con un poco de texto del mensaje. Se trata simplemente de un *widget* de conveniencia.
	GtkRecentChooserDialog	**Ventana de Archivos Recientes:** Crea una ventana archivos abiertos recientemente.
	GtkAssistant	**Ventana de Asistente:** Crea una ventana para representar una operación generalmente compleja dividido en varios pasos, guiando al usuario a través de sus páginas y controlar el flujo de la página para recopilar los datos necesarios.
	GtkAppChooserDialog	**Ventana de Selección de Aplicaciones:** Crea una ventana para seleccionar la aplicación para abrir un archivo.

Tabla de Anexo 2: *Widgets ventanas (30).*

Ícono	Nombre del *Widget*	Descripción
Contenedores		
	GtkBox	**Caja de Trabajo:** Organiza sus *widgets* hijos en un área rectangular. La caja de trabajo basa su funcionamiento en el concepto de embalaje, es decir, la adición de los *widgets* usando como referencia una posición en

		particular. Para una GtkBox, hay dos posiciones de referencia: el inicio y el final del cuadro. Para una GtkBox vertical (GtkVBox), el inicio se define como la parte superior de la caja y al final se define como la parte inferior. Para una GtkBox horizontal (GtkHBox) del comienzo se define como el lado izquierdo y el extremo se define como el lado derecho.
	GtkGrid	**Cuadrícula:** Es un contenedor cuadricular que trabaja parecido al GtkBox a diferencia que organiza sus *widgets* hijos en filas y columnas.
	GtkNotebook	**Contenedor de Pestañas:** Cuenta con pestañas hijas (páginas) que se puede cambiar entre el uso de las etiquetas de la ficha a lo largo de un borde.
	GtkFrame	**Marco:** El control de marco es una papelera que rodea a su hijo con un marco decorativo y una etiqueta opcional. Si está presente, la etiqueta se dibuja en un hueco en el lado superior del marco.
	GtkAspectFrame	**Marco Proporcional:** Se deriva de GtkFrame y se utiliza cuando se desea empaquetar un *widget* para que pueda cambiar el tamaño.
	GtkMenuBar	**Barra de Menú:** Crea una barra de menú estándar que puede contener muchos elementos de menú. Estos se pueden añadir a través de la clase base abstracta GtkMenuShell, que contiene uno o más GtkMenuItems.
	GtkToolBar	**Barra de Herramientas:** Una barra de herramientas puede contener instancias de una subclase de GtkToolItem. Para agregar un botón a la barra de herramientas, agregue una instancia del GtkToolButton. Los elementos de barra de herramientas se pueden agrupar visualmente mediante la adición de instancias de GtkSeparatorToolItem a la barra de herramientas.
	GtkToolPalette	**Paleta de Herramientas:** Permite añadir GtkToolItems a un contenedor de paleta como con diferentes categorías y arrastrar y soltar. GtkToolItems no se pueden agregar directamente a una GtkToolPalette, sino que se añaden a un GtkToolItemGroup que los que se pueden agregar a un

		GtkToolPalette.
	GtkPaned	**Panel:** Está compuesto por dos paneles: horizontal (GtkHPaned) y vertical (GtkVPaned). La división entre los dos paneles es ajustable por el usuario.
	GtkButtonBox	**Caja de Botones:** Se utiliza para proporcionar un diseño coherente de los botones de toda la aplicación. Se pueden posicionar de dos formas: verticalmente (GtkVButtonBox) o horizontalmente (GtkHButtonBox).
	GtkLayout	**Contenedor de Diseño:** Es un área de desplazamiento infinito que contiene *widgets* hijos y/o dibujo personalizado. Es similar a GtkDrawingArea, pero es compatible con el desplazamiento (GtkScrolledWindow), y puede contener *widgets* hijos, ya que es un contenedor. Sin embargo, si sólo vas a dibujar, una GtkDrawingArea es una mejor opción, ya que tiene una menor carga.
	GtkFixed	**Contenedor Fijo:** Coloca *widgets* hijos en posiciones fijas y con tamaños fijos, dado en píxeles. No realiza ninguna gestión automática de la distribución. Este contenedor no es recomendable utilizar en la mayoría de las aplicaciones, así se evita tener que aprender sobre los otros contenedores GTK+.
	GtkEventBox	**Caja de Eventos:** Es una subclase de GtkBin que también tiene su propia ventana. Esto es útil ya que permite poner eventos a los *widgets* que no tienen su propia ventana.
	GtkExpander	**Expansor:** Permite al usuario ocultar o mostrar su *widget* hijo haciendo clic en un triángulo de expansión similar a los triángulos utilizados en un GtkTreeView.
	GtkViewport	***Puerto de Vista*:** El *widget* de GtkViewport actúa como una clase adaptador, implementando la capacidad de desplazamiento de *widgets* hijos que carecen de su propia capacidad de desplazamiento. Se utiliza para desplazar los *widgets* hijos como GtkGrid, GtkBox, y así sucesivamente. El GtkViewport empezará a desplazar el contenido sólo si se le asigna menos del mínimo del *widget* hijo en una

		orientación determinada.
	GtkScrolledWindow	**Ventana de Desplazamiento:** Añade barras de desplazamiento para el control hijo y, opcionalmente, dibuja un marco biselado alrededor del *widget* hijo. Algunos *widgets* tienen soporte nativo de desplazamiento como GtkTreeView, GtkTextView y GtkLayout. Para los *widgets* que carecen de apoyo de desplazamiento nativo, GtkViewport actúa como una clase adaptador para desplazar *widgets* hijos como GtkGrid y GtkBox.
	GtkAlingment	**Alineamiento:** Controla la alineación y el tamaño de su control hijo. Tiene varios ajustes: Ajuste de Escala (Xscale y Yscale) y Ajuste de Alineación (xalign y yalign). Los ajustes de escala se utilizan para especificar la cantidad de control hijo debería expandirse para llenar el espacio asignado a la GtkAlignment. Los valores pueden oscilar entre 0 (no se expande en absoluto) a 1 (se expande para llenar todo el espacio disponible). Los ajustes de alineación se utilizan para colocar el *widget* hijo dentro de la zona disponible. Los valores van desde 0 (parte superior o izquierda) a 1 (parte inferior o derecha). Si los ajustes de la balanza se encuentran ambos a 1, los ajustes de alineación no tienen ningún efecto.

Tabla de Anexo 3: *Widgets contenedores (30).*

Ícono	Nombre del *Widget*	Descripción
Control y	**Visualización**	
	GtkButton	**Botón:** Generalmente se usa para activar una función de devolución de llamada que se invoca cuando se presiona el botón. Puede contener cualquier control hijo válida, es decir, puede contener casi cualquier otro *widget* estándar. El *widget* más utilizado es el GtkLabel.
	GtkToggleButton	**Botón de Dos Estados:** Es un GtkButton que permanecerá “en presionado” cuando se hace clic. Al hacer clic de nuevo hará que el botón de activación vuelva a su estado normal.

	GtkCheckButton	**Botón de Verificación:** Coloca un GtkToggleButton discreto al lado de un *widget*, por lo general un GtkLabel.
	GtkRadioButton	**Botón de Opción:** Es un único botón de opción que realiza la misma función básica que GtkCheckButton. Pero cuando múltiples botones de radio se agrupan juntos se convierten en un componente de interfaz de usuario diferente en su propio derecho. Cada botón de opción es miembro de algún grupo de botones de radio. Cuando se selecciona uno, todos los demás botones de opción del mismo grupo se desactivan. A GtkRadioButton es una manera de dar al usuario una opción entre muchas opciones.
	GtkSwitch	**Interruptor:** Es un *widget* que tiene dos estados: activo o inactivo. El usuario puede controlar qué estado debe estar activo haciendo clic en el área vacía, o arrastrando el *mouse*.
	GtkEntry	**Entrada de Texto:** Permite la entrada de texto de una línea. Si el texto que ha introducido es mayor que la asignación del *widget*, se desplazará de manera que la posición del cursor es visible.
	GtkSpinButton	**Selector Cíclico:** Es una forma ideal para que el usuario pueda ajustar el valor de algún atributo. En lugar de tener que teclear directamente un número en una GtkEntry, GtkSpinButton permite que el usuario haga clic en una de las dos flechas para aumentar o disminuir el valor mostrado. El valor se puede escribir, pero también puede ser comprobado para asegurar que esté en un rango determinado.
	GtkComboBox	**Caja Combo:** Permite al usuario elegir entre una lista de opciones válidas y muestra la opción seleccionada. Cuando se activa, él muestra una ventana emergente que le permite al usuario realizar una nueva elección. El estilo en el que se muestra el valor seleccionado, y el estilo de la ventana emergente se determinan por el tema actual. Puede ser similar a un cuadro combinado de estilo de ventana.

	GtkComboBoxText	**Caja Combo con Texto:** Es una variante del GtkComboBox el cual puede tener una entrada de texto.
	GtkImage	**Imagen:** Muestra una imagen, pero existen varios tipos de objetos se pueden mostrar como una imagen, más típicamente, debería cargar un GdkPixbuf ("búfer de píxeles") a partir de un archivo, y luego mostrarlo.
	GtkLabel	**Texto:** Muestra una pequeña cantidad de texto y así como el nombre implica, la mayoría de las etiquetas se utilizan para etiquetar otro *widget* tal como un GtkButton o un GtkMenuItem.
	GtkAccelLabel	**Etiqueta de Aceleración:** Es una subclase de GtkLabel que también muestra una tecla de aceleración a la derecha del texto de la etiqueta, por ejemplo, 'Ctrl + S'. Se utiliza comúnmente en los menús para mostrar los atajos de teclado para los comandos. La tecla de aceleración para mostrar no se establece explícitamente. En cambio, el GtkAccelLabel muestra los aceleradores que se han añadido a un widget en particular.
	GtkFileChooserButton	**Botón Abrir Archivo:** Permite al usuario seleccionar un archivo. Implementa la interfaz GtkFileChooser. Visualmente es un nombre de archivo con un botón para que aparezca un GtkFileChooserDialog. El usuario puede entonces utilizar ese diálogo para cambiar el archivo asociado a ese botón. Este *widget* no permite establecer la propiedad "seleccionar múltiples" en TRUE.
	GtkColorButton	**Botón Selector de Color:** Es un botón que muestra el color seleccionado actualmente, permite abrir un diálogo de selección de color para cambiar el color. Es el *widget* adecuado para seleccionar un color en un cuadro de diálogo de preferencias.
	GtkFontButton	**Botón Selector de Tipografía:** Es un botón que muestra la fuente seleccionada actualmente, permite abrir un diálogo selector de fuente para cambiar la fuente. Es *widget* adecuado para seleccionar una fuente en un cuadro de diálogo de preferencias.

	GtkLinkButton	**Botón Hipervínculo:** Es un GtkButton con un hipervínculo, similar a la utilizada por los navegadores web, lo que desencadena una acción cuando se hace clic. Es útil para mostrar enlaces rápidos a recursos.
	GtkScaleButton	**Botón Escala:** Proporciona un botón que abrirá un control de escala. Este tipo de control se utiliza comúnmente para el control de volumen en aplicaciones multimedia, y GTK+ ofrece una subclase GtkVolumeButton que se adapta para este caso de uso.
	GtkVolumeButton	**Botón Volumen:** Crea un botón volumen con un rango entre 0,0 y 1,0, con un escalonamiento de 0,02. Los valores del volumen se pueden obtener y modificar utilizando las funciones de GtkScaleButton.
	GtkAppChooserButton	**Botón Selector de Aplicaciones:** Es un *widget* que permite al usuario seleccionar una aplicación. Implementa la interfaz GtkAppChooser.
	GtkScale	**Escala:** Es un control deslizante que se utiliza para seleccionar un valor numérico. Esto es útil para aplicaciones que quieren mostrar un valor determinante en la escala, sin necesidad de cambiar el diseño de la aplicación (como reproductores de películas o música). Este se puede posicionar de dos formas: horizontal (GtkHScale) o vertical (GtkVScale).
	GtkScrollbar	**Barra de Desplazamiento:** Es una barra de desplazamiento horizontal (GtkHScrollbar) o vertical (GtkVScrollbar), dependiendo del valor de la propiedad "orientación".
	GtkProgressBar	**Barra de Progreso:** Se utilizar para mostrar el progreso de una operación de larga duración de dos modos diferentes: modo de porcentaje y el modo de actividad. Proporciona una indicación visual de que el procesamiento está en marcha.
	GtkSpinner	**Cargando:** Muestra una animación giratoria que a menudo se utiliza como una alternativa a un GtkProgressBar para la visualización de la actividad por tiempo indefinido, en lugar del progreso real.

	GtkTextView	**Vista de Texto:** Es un *widget* que puede mostrar un GtkTextBuffer.
	GtkTreeView	**Vista o Lista de Árbol:** Muestra cualquier objeto que implemente la interfaz GtkTreeModel.
	GtkIconView	**Vista de Ícono:** Se muestra el modelo como una cuadrícula de íconos con etiquetas. Permite seleccionar uno o varios elementos. Tenga en cuenta que si el modelo de árbol está respaldado por una tienda de árbol real (en oposición a una lista plana donde la asignación a los iconos es evidente), GtkIconView sólo mostrará el primer nivel del árbol y pasa por alto las ramas del árbol.
	GtkHandleBox	**Caja Manipulable:** Permite que una porción de una ventana para ser "arrancada". Se trata de un GtkBin, que representa su hijo y un mango que el usuario puede arrastrar al arrancar una ventana separada (la ventana flotante) que contiene el control hijo. No es recomendable utilizar en aplicaciones modernas, es muy especializado, carece de características para que sea útil, por tal motivo, ha quedado obsoleto.
	GtkStatusBar	**Barra de Estado:** Se coloca generalmente a lo largo de la parte inferior de GtkWindow principal de una aplicación. Puede proporcionar un comentario periódica de la situación de la aplicación (como suele ser el caso en un navegador web, por ejemplo), o se puede usar simplemente para generar un mensaje cuando cambia el estado, (cuando una carga se ha completado en un cliente de FTP, por ejemplo). Las barras de estado en GTK+ mantienen un montón de mensajes. El mensaje en la parte superior de la pila de cada barra es el que se muestra actualmente.
	GtkCalendar	**Calendario:** Crea un calendario que muestra un mes a la vez.
	GtkMenu	**Menú:** Es un GtkMenuShell que implementa un menú desplegable que consiste en una lista de objetos GtkMenuItem en el que se pueda navegar y son activados

		por el usuario para realizar funciones de aplicación.
	GtkSeparator	**Separador:** Se utiliza de manera horizontal (GtkHSeparator) o vertical (GtkVSeparator), en función del valor de la propiedad "orientación", se utiliza para agrupar los *widgets* de dentro de una ventana. Se muestra una línea con una sombra para hacer que aparece hundido en la interfaz.
	GtkArrow	**Flecha:** Es utilizado para dibujar flechas simples que deben apuntar en una de las cuatro direcciones cardinales (arriba, abajo, izquierda o derecha). El estilo de la flecha puede ser de sombra: sombra afuera, grabada, o grabada a cabo.
	GtkDrawingArea	**Zona de Dibujo:** Se utiliza para la creación de elementos de interfaz de usuario personalizados. Básicamente se trata de un *widget* en blanco, se puede dibujar en ella.
	GtkInfoBar	**Barra de Información:** Se puede utilizar para mostrar mensajes al usuario sin mostrar un cuadro de diálogo. A menudo se muestra temporalmente en la parte superior o inferior de un documento. En contraste con GtkDialog, que tiene un área de acción horizontal en la parte inferior, GtkInfoBar tiene un área de acción vertical en el lado.

Tabla de Anexo 4: *Widgets control y visualización (30).*

Ícono	Nombre del *Widget*	Descripción
Widgets	**Compuestos**	
	GtkColorSelection	**Selector de Color:** Se utiliza para seleccionar un color. Se compone de una rueda de color y el número de deslizadores y cajas de entrada para los parámetros de color, como matiz, saturación, valor, rojo, verde, azul, y la opacidad. Se encuentra en el cuadro de diálogo de selección de color estándar GtkColorSelectionDialog.
	GtkFontSelection	**Selector de Tipografía:** Enumera las fuentes disponibles, estilos y tamaños, lo que permite al usuario seleccionar una fuente. Se utiliza en el *widget* GtkFontSelectionDialog para proporcionar un cuadro de diálogo para la selección de

		fuentes.
	GtkRecentChooser	**Selector de Archivos Recientes:** Es una interfaz que puede ser implementado por los *widgets* que muestran la lista de archivos usados recientemente. En GTK+, los principales objetos que implementan esta interfaz son GtkRecentChooserWidget, GtkRecentChooserDialog y GtkRecentChooserMenu.
	GtkFileChooserWidget	**Selector de Archivos:** Es un *widget* adecuado para seleccionar los archivos. Es el principal bloque de construcción de un GtkFileChooserDialog. Tenga en cuenta que GtkFileChooserWidget solo utiliza las funciones que trabajan en un GtkFileChooser.
	GtkAppChooserWidget	**Selector de Aplicaciones:** Es un *widget* para seleccionar aplicaciones. Es el principal bloque de construcción para GtkAppChooserDialog. La mayoría de las aplicaciones sólo necesitan usar GtkAppChooserDialog; pero puede utilizar este *widget* como parte de un control más grande si usted tiene necesidades especiales.

Tabla de Anexo 5: *Widgets compuestos* (30).

Ícono	Nombre del *Widget*	Descripción
Diversos		
	GtkListStore	**Lista de Almacenamiento:** Se utiliza para modelar una lista de *widget* con GtkTreeView. Implementa la interfaz GtkTreeModel puede utilizar todos los métodos disponibles allí. También implementa la interfaz GtkTreeSortable así que puede ser ordenada por la vista.
	GtkTreeStore	**Árbol de Almacenamiento:** Se utiliza para modelar una lista de árboles con GtkTreeView. Implementa la interfaz GtkTreeModel puede utilizar todos los métodos disponibles allí. También implementa la interfaz GtkTreeSortable así que puede ser ordenada por la vista.

	GtkTreeModelFilter	**Modelo de Árbol de Filtro:** Es un modelo de árbol que envuelve otro modelo de árbol, en el que se puede filtrar filas específicas con base en datos de una columna visible, modificar la apariencia del modelo utilizando una función de modificación y establecer un nodo raíz diferente, también conocida como una raíz virtual.
	GtkTreeModelSort	**Modelo de Árbol Ordenado:** Es un modelo que implementa la interfaz GtkTreeSortable. El propósito principal de este modelo es proporcionar una manera de ordenar un modelo diferente y sin modificarlo. Tenga en cuenta que la función de clasificación utilizado por GtkTreeModelSort no se garantiza que sea estable.
	GtkEntryBuffer	**Entrada de Bufffer:** Contiene el texto real que se muestra en un *widget* GtkEntry. Un solo objeto GtkEntryBuffer puede ser compartida por varios *widgets* GtkEntry se podrán compartir el mismo contenido de texto, pero no la posición del cursor, atributos de visibilidad e ícono.
	GtkTextBuffer	**Texto Buffer:** Representa un texto que se está editando para luego ser mostrado en el GtkTextView.
	GtkTextTag	**Texto de la Etiqueta:** Una etiqueta está representada por un objeto GtkTextTag, se puede aplicar a cualquier número de intervalos de texto en cualquier número de memorias intermedias.
	GtkTextTagTable	**Tabla de variables de texto:** Se define un conjunto de etiquetas que se pueden utilizar juntos. Cada búfer tiene una tabla de etiquetas asociado con él; sólo etiquetas de tabla de variables que se pueden utilizar con el búfer. Sin embargo, una tabla de variables solo se puede compartir entre varios buffers.
	GtkSizeGroup	**Tamaño del Grupo:** Proporciona un mecanismo para agrupar una serie de *widgets* juntos y que todos pidan la misma cantidad de espacio. Este suele ser útil cuando se desea una columna de *widgets* que tiene el mismo tamaño, pero no se puede utilizar un *widget* GtkGrid.

	GtkWindowGroup	**Grupo de ventanas:** Los objetos GtkWindowGroup son referenciados por cada ventana en el grupo, así que una vez que haya agregado todas las ventanas a un GtkWindowGroup, se puede eliminar la referencia inicial al grupo de ventanas. Si las ventanas en el grupo de ventanas se destruyen posteriormente, entonces van a ser eliminados del grupo de ventanas y dejan caer sus referencias sobre el grupo de ventanas, y cuando se han eliminado todas las ventanas, el grupo de ventanas se liberará.
	GtkAccelGroup	**Grupo de aceleradores:** Representa un grupo de aceleradores de teclado, típicamente unido a un GtkWindow de nivel superior. Por lo general, usted no tendrá que crear un GtkAccelGroup directamente, sino utilizando GtkUIManager, GTK + configura automáticamente los aceleradores para sus menús en GtkAccelGroup del ui.
	GtkAdjustment	**Ajustamiento:** Representa un valor que tiene un límite inferior y superior asociado, junto con incrementos de los pasos y de página, y un tamaño de página. Se utiliza dentro de varios *widgets* GTK+, incluyendo GtkSpinButton, GtkViewport y GtkRange (clase base para GtkHScrollbar, GtkVScrollbar, GtkHScale y GtkVScale).
	GtkEntryCompletion	**Finalización de Entrada:** Es un objeto auxiliar para ser utilizado en conjunción con GtkEntry para proporcionar la funcionalidad finalización, que significa que cuando el usuario modifica el texto de la entrada, se chequee en el GtkEntryCompletion qué filas en el modelo coinciden con el contenido actual de la entrada, y muestra una lista de coincidencias.
	GtkIconFactory	**Fábrica de Íconos:** Navegar por los iconos comunes disponibles en la lista de identificadores de valores se encuentra aquí.
	GtkStatusIcon	**Ícono de Estado:** La "bandeja del sistema" o área de notificación se utiliza normalmente para los iconos transitorios que indican algún estado especial. Por ejemplo, un icono de la bandeja de sistema puede decirle

		al usuario que tienen un correo nuevo, o tiene un mensaje instantáneo entrante. La idea básica es que la creación de un icono en el área de notificación es menos molesto que aparecer un cuadro de diálogo. Un objeto GtkStatusIcon se puede utilizar para mostrar un icono en una "bandeja del sistema". El icono puede tener un texto de ayuda, y el usuario puede interactuar con él mediante la activación o desplegar un menú contextual.
	GtkFileFilter	**Filtro de Archivos:** Se puede utilizar para restringir los archivos que se muestran en un GtkFileChooser.
	GtkRecentFilter	**Filtro Reciente:** Se puede utilizar para restringir los archivos que se muestran en un GtkRecentChooser.
	GtkRecentManager	**Gestor Reciente:** Proporciona una facilidad para añadir, eliminar y buscar los archivos utilizados recientemente.

Tabla de Anexo 6: *Otros widgets (30).*

Ícono	Nombre del *Widget*	Descripción
Ventanas	**Imprimir**	
	GtkPageSetupUnixDialog	**Ventana de Configuración de Página:** Implementa un cuadro de diálogo de configuración de página para las plataformas que no ofrecen un cuadro de diálogo de configuración de página nativa, como Unix[7].
	GtkPrintUnixDialog	**Ventana de Imprimir:** Implementa un cuadro de diálogo de impresión para las plataformas que no ofrecen un cuadro de diálogo de impresión nativo, como Unix.

Tabla de Anexo 7: *Widgets ventana para imprimir (30).*

7 **Unix:** Es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969, por un grupo de empleados de los laboratorios Bell de AT&T