Índice General.

Manual del Desarrollador.

1. Información Básica.

Este programa es una interfaz gráfica para la administración y gestión de grupos scouts. Esta diseñado específicamente para cumplimentar las reglas administrativas correspondientes al funcionamiento que suelen tener estas instituciones. Las funcionalidades que se contemplan en este son:

- Registro e Informes de miembros de un grupo, incluyendo almacenamiento de datos básicos, datos de sus padres y datos referidos a históricos médicos del mismo.
- Visualización y búsqueda con filtros que actúan en tiempo real, para que la consulta de información sea sencilla y ágil.
- Creación de Eventos Económicos con posibilidad de establecer ganancias para el miembro por unidad.
- Posibilidad de establecer Costos Anuales Fijos que se incluyen automáticamente dentro de la deuda de los miembros activos.
- Gestión Contable individual del Miembro, incluyendo detalles de cada movimiento que realice y resúmenes de la totalidad de ellos.
- Gestión Contable de Grupo, con informe detallado de los movimientos contables del grupo, incluyendo la totalidad de los movimientos de los miembros.
- Registro de los materiales de los que dispone el grupo con visualización detallada del listado total.
- Impresión de Informes en cada sección, para poder exteriorizar los contenidos que se almacenan.

En el desarrollo de este programa se utilizaron tecnologías como:

- Escrito enteramente en C++, basado en la plataforma Qt (QT SDK - http://qt.nokia.com).
- Liberado bajo la licencia GNU GPL v3 (http://www.gnu.org/licenses/gpl.html).
- Portabilidad sencilla a otras plataformas (multiplataforma).
- Distribuido desde la plataforma SourceForge con instaladores para Windows y Debian/Ubuntu Linux.
- Git para el controlo de versiones.
- Impresiones de documentos hechos enteramente en HTML para un diseño de calidad profesional.
- Motor de base de datos SQLite.
- GitHub como plataforma de hosting para el desarrollo descentralizado.
- Inno Installer para realizar el instalador de Windows y Debreate en el caso de Linux.

La totalidad del programa fue realizada en Ubuntu Linux 11.10 utilizando el software Qt Creator 2.2.1 (Qt 4.7.3).

Para poder expresar la metodología de trabajo que se utilizo a lo largo del desarrollo del programa, es necesario explicar la forma de trabajo que propone el programa IDE que se utilizando, QtCreator. A continuación se describen brevemente algunos de los mecanismos utilizados y algunas de las tecnologías que forman parte del programa.

2. Mecanismo Signal - Slot.

Los signals y slots o señales y ranuras, son un mecanismo fundamental de todos los programas realizados en qt. Permite la comunicación entre objetos sin que ellos sepan nada entre ellos mismos. Un ejemplo típico es cuando queremos que cambiar algo en una widget, cambie algo en otra widget distinta, cuando se realiza alguna acción con ella. Las widget que poseen el mecanismo de signal-slots son aquellas que heredan de la clase QObject directa o indirectamente El concepto se basa en que QObject puede enviar señales que contienen información sobre el evento que se realizo en el widget, lo que puede hacer que algún otro widget reaccione al mismo recibiéndolo en alguno de sus slots.

Los Slots pueden recibir señales, pero también son funciones ordinarias de C++ por lo tanto no siempre es necesario que se ejecuten para reaccionar a otro widget que le envía una señal, sino que pueden ser invocados en cualquier momento de la ejecución del programa. Cada widget tiene sus propios slots, pero a su vez hereda los slots estándar de la clase QObject, los cuales están en todos los widgets.

3. Mecanismo Model-View.

El mecanismo model/view es una arquitectura para administrar la relación entre información de la base de datos y la forma en que es presentada al usuario, este deriva directamente del MVC (Modelo vista controlador), pero con la vista y el controlador integrados . El ejemplo típico es asociar una tabla de la base de datos con una vista de tabla o tableview (widget). Esto permite personalizar la visualización de los datos, sin modificar nada de la tabla que tiene asociada. En el programa se usan modelos asociados a tableview, a combobox, etc. los cuales hacen referencia a una tabla o consulta sobre la base de datos, por lo general cada ves que se introduce un cambio en la base de datos es necesario actualizar su modelo asociado así los cambios se reflejan en la vista asociada al modelo concretamente. También en algunas secciones se una dinámica inversa, haciendo modificaciones sobre el modelo y reflejando los cambios en la base de datos. Existen varios tipos de modelos, en el programa se utiliza por lo general los QSqlQueryModel y QSqlTableModel. Para más información consultar link.

4. Sistema de Archivos de Recursos.

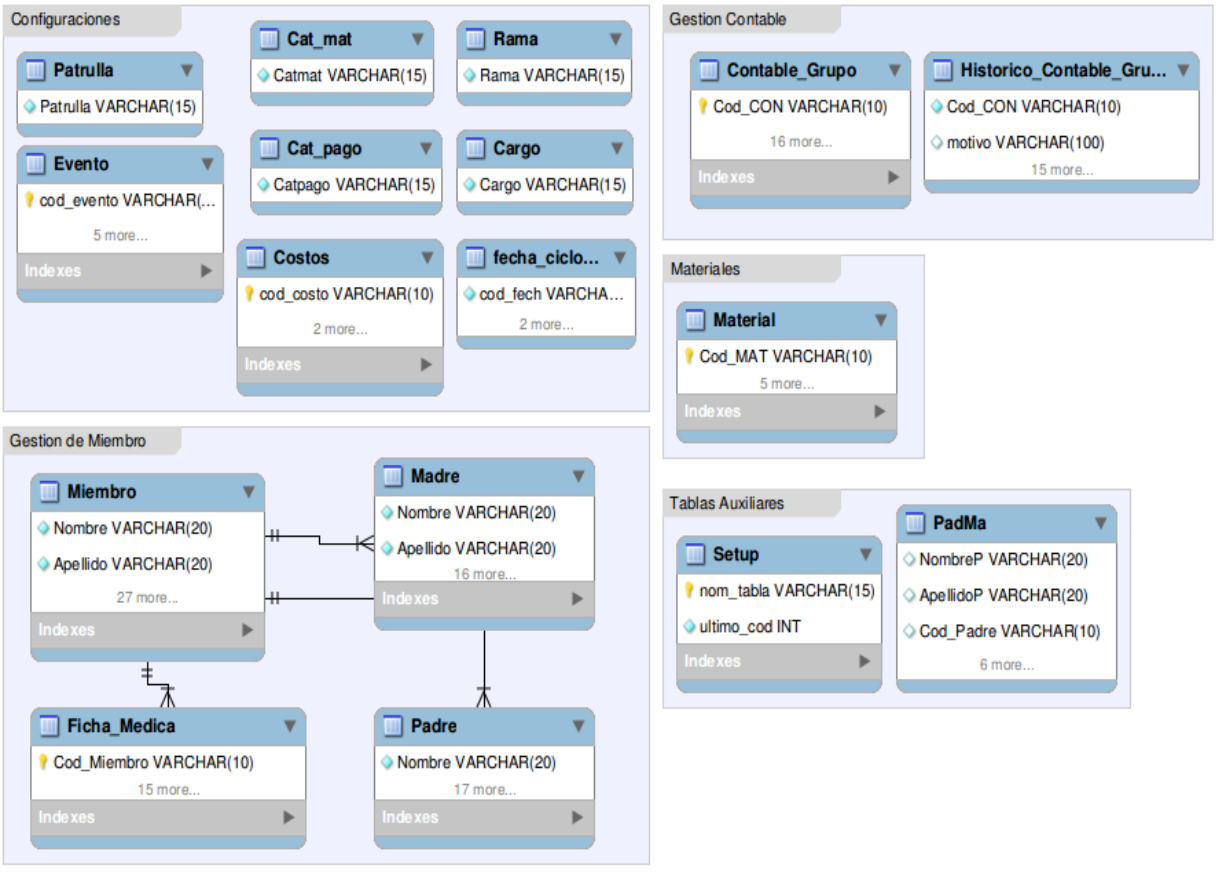
Es una forma de incluir archivos binarios dentro del mismo ejecutable del programa, esto es útil si la aplicación necesita incluir archivos como podrían ser iconos, manuales, etc. dentro del mismo ejecutable, evitando el riesgo de que se pierdan esos archivos y se produzcan errores en el programa. En el programa se incluyen los iconos, splashscreen y manuales dentro de un archivo con extensión *qrc el cual es el archivo de recurso mismo (en nuestro caso icons.qrc). Para más información dirigirse a link.

5. Base de Datos SQLite.

Se utilizo para almacenar los datos que se manejan en software, el motor de bases de datos SQLite ya que este brinda las siguientes características convenientes:

- Varios procesos o hilos pueden acceder a la misma base de datos sin problemas.
- SQLite es un proyecto de dominio público.
- SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones
- Permite la inclusión de campos tipo BLOB (archivos binarios de gran tamaño, ej: un archivo jpeg).
- Robusto y popular en la industria de desarrollo de software

Como se nombro anteriormente, cada clase tiene sus mecanismos específicos para administrar las tablas correspondientes en la base de datos, y la clase que se encargaba de conectarse/crear la base de datos era conectarconbd. SQLite crea un archivo de tipo *.sqlite (En nuestro caso BD.sqlite) en el directorio del programa, el cual almacena toda la información que se maneja en el programa, esto nos da la ventaja que es muy sencillo hacer copias de backup de la misma, y quita la necesidad de instalar bibliotecas complementarias para su funcionamiento.



Nota: La base de datos posee la siguiente Estructura. Para más detallo ver pdf adjunto.

6. Estructura de la Aplicación.

La aplicación se estructura en base a la forma que tiene de trabajar el entorno de desarrollo que se utilizo, en este caso Qt Creator. Hay 3 clases que son específicas de los widgets que forman la aplicación, en este caso:

MainWindow (Ventana Principal): Es esta clase se contemplan todas las reglas de interfaz para la ventana principal, variables auxiliares para procesar datos, y la programación de los slots.
Configuraciones: Igual que la ventana principal, pero para la ventana que contempla las configuraciones del programa.

Rautomaticpadres: Es un widget auxiliar que tiene la funcionalidad de auto completado de datos en caso de redundancia de los mismos en la Base de Datos, tiene las mismas características que las anteriores descritas.

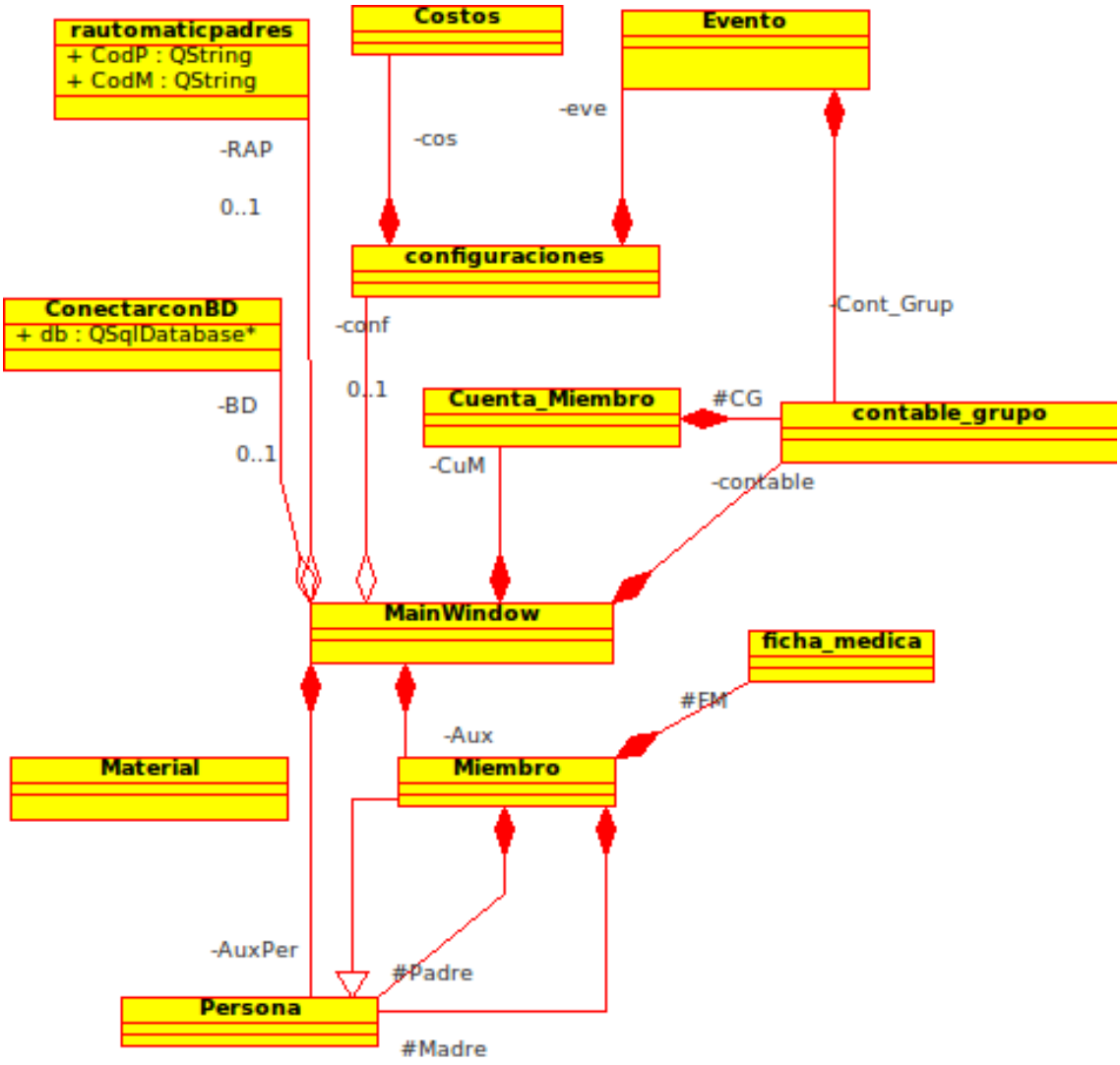
Estas son clases que tienen asociadas un archivo de tipo ui que representa la interfaz misma del widget, especifico, en el caso de la clase MainWindow, un widget de tipo QMainwindow y en el caso de Configuraciones y Rautomaticpadres un Qdialog, de hecho, estas son clases que heredan de las estándares de Qt, por lo tanto poseen las funcionalidades que estas ofrecen mas las programadas específicamente para como se diseño la interfaz. Los Interfaces en si son un conjunto de widgets que interactúan entre si para cumplir las funcionalidades que requiere el software, por ejemplo, labels, combobox, lineedit, etc. Cada uno de estos widgets a su vez son clases estándares de qt que brindan ya operaciones básicas para interactuar. Entre las cosas que brindan se encuentran los signals y slots (Ver signals y slots), los cuales responden a lo programado en la clase, es decir, la dinámica de trabajo en estas clases que hacen referencia a la interfaz es la de especificar que debe hacerse en el caso de que ejecute un slot determinado.

Luego están las clases especificas modeladas para los requerimientos del software, estas son:

Miembro: Clase que describe los datos necesarios para un miembro y las operaciones necesarias del mismo. Esta clase heredad datos de la clase persona, que es mas general.
Persona: Clase que describe a una persona, con los datos que se requieren específicamente para este software, y que posee las operaciones necesarias para las mismas.
Contable grupo: Clase que actúa de interfaz para las operaciones contables del grupo en la base de datos. Se incluyen las funciones y datos necesarios para que esto sea posible.
Contable_Miembro: Ídem Contable_grupo, pero específicamente para un miembro individual.
Evento: Clase que describe un Evento económico, y que tienes los métodos necesarios para procesarlos.
Costo: Clase que describe los costos anuales fijos y que posee los métodos necesarios para procesarlos.
ficha_médica: Clase que contempla los datos de una ficha médica de un miembro y que tiene los métodos para procesar los mismos.

Material: Clase que modela un material (herramientas) y que tiene los métodos necesarios para administrarlo.

Luego se encuentra la clase que se encarga de la declaración, creación y contemplación de errores de la base de datos, esta se llama, Conectarconbd, y tiene los métodos para crear las tablas y abrir y cerrar la base de datos en cada apertura del programa. A su vez tiene funciones auxiliares con reglas para la metodología interna de códigos primary key para ciertas tablas.



Nota: Ejemplo de estructura de una clase Qt. Para más detalle ver el pdf adjunto.

7. Funcionamiento interno del programa.

Sección Miembro:

En esta sección, se permite la carga y edición de cada uno de los datos de un miembro. Internamente, se trabaja llenando una clase de tipo Miembro donde se almacenan cada uno de los datos de cada sección (datos miembros, padres y ficha médica). Esta clase ejecuta los métodos de llenado una vez que se presiona el botón de guardar. Una vez llena la clase se ejecuta el método para realizar una inserción en la tabla miembro de la base de datos.

Para la actualización de cualquiera de estos datos, se consulta internamente la tabla y se cargan en los lugares correspondientes los datos para modificar. Todo esto sucede a través de métodos internos de la clase miembro (guardar, actualizar, leer).

En la sección de Padres hay un botón especial “Automático”, que es una función auxiliar que se ejecuta para reconocer los datos de los padres, deduciéndolos a partir de los datos cargados de miembro (apellido, dirección, teléfono, etc.), o consultando el apellido del miembro en la tabla de Padre o Madre de la base de datos, de esta manera se evita tener que cargar 2 veces los mismos padres. En este ultimo caso, se hace una consulta compleja sobre la unión de las tablas de padres, cargando temporalmente, los datos de los padres en una tabla auxiliar que se muestra en forma de dialogo para que el usuario elija entre los resultados posibles. Una vez que se selecciona una opción cada uno de los datos de esos padres seleccionados se cargan el los espacios disponibles, con lo que ya se podría guardar normalmente el miembro sin duplicar la información de los padres.

En la pestaña de Ficha contable del miembro, se administran la cosas de forma diferente, ya que no se esta manejando información fija del miembro sino, solo la información contable del mismo. En esta sección hay dos tablas con un modelo asociado cada una, estos modelos hacen referencia a la tabla contable grupo, pero con un filtrado para que solo se vean las filas que pertenecen al miembro que se consulta, en cada caso. La tabla de costos, permite hacer ingresos de los pago que va haciendo el miembro a lo largo del año, esto actúa insertando información en el modelo. Se permite hacer edición sobre cualquiera de las filas con un funcionamiento similar al de la edición del miembro, se consultan los datos se los carga en los lugares correspondientes y una vez que se guardan se modifican en el modelo. En la tabla de eventos solo se permite la edición de la columna cantidad y monto pagado, que internamente es lo mismo que en la tabla de costos. Los cambios realizados se reflejan en los resúmenes, gracias a funciones auxiliares que consultan las tablas y hacen las operaciones correspondientes.

La búsqueda de miembros funciona a través de filtros que actúan sobre un modelo auxiliar que se despliega apenas de ingresa una letra en la línea de búsqueda. Este filtro es seleccionar una columna para que el método interno del modelo filtre. También se utiliza un Autocompletado mediante un Qcompleter, el cual permite desplegar el modelo y hacer la búsqueda en tiempo real.

Sección Contable Grupo:

Esta sección gira en torno a un modelo asociado al tableview principal que permite hacer inserciones, ediciones y borrado sobre los datos de la tabla. Esto funciona a través de la manipulación del modelo. Sin embargo en esta sección hay reglas especiales. No se puede modificar ni los eventos ni los ingresos de los miembros en la tabla. Para controlar internamente esto, cada vez que se hace doble clic sobre la tabla, primero se verifica que la columna categoría, y en el caso que sea General, se permite hacer la edición cargando los datos del elemento de la tabla en la sección de carga. Cuando se presiona el botón borrar, se elimina la fila en el modelo y también en la base de datos.

Sección Configuraciones:

En las configuraciones se trabaja con modelos (QSqlTableModel) en cada una de las secciones que incluyen una tableview.
Para la sección de Base de Datos hay 5 modelos que hacen referencia a cada una de las tablas que guardan las categorías en la base de datos (Rama, Patrulla, Cargo, categoría de pago y categoría de material). Las operaciones posibles son el agregado y la eliminación de categoría en cada una, modificando el modelo internamente, por lo que se ven reflejados los cambios en la base de datos. Una vez que se agregaron nuevas categorías y se cierra el dialogo de configuraciones, se ejecuta una función auxiliar que actualiza los modelos asociados a los combo box de la ventana principal.

Para la sección Eventos, internamente es la misma dinámica de modelo que se usa en la sección Base de Datos, la diferencia es que también se permite hacer modificación sobre cualquier Evento realizado. Para esto se vuelve cargar los datos de el evento seleccionado, en la sección de agregar un nuevo evento, una vez que se guarda, el modelo se actualiza y a su vez la base de datos también. Otra diferencia es que cada vez que se agrega un nuevo evento, se ejecuta una función auxiliar que agrega a cada miembro activo el evento para que pueda participar en el.

Esto es agregar por cada miembro dos filas en la tabla de contable grupo, para registrar las ganancias individuales de miembro y así también las de grupo por miembro.
En la sección Costo, sucede lo mismo que en la sección evento, pero en este caso los costos que se agreguen se suman al “debe” de cada miembro activo.
La sección Ciclo contable es una sección clave para el funcionamiento interno del programa. Aquí se inicia y termina cada ciclo anual contable.

Al iniciarse un ciclo, se copia toda la información contable del ciclo anterior a una tabla auxiliar “histórico contable”. La información que queda luego en esta tabla se puede imprimir o exportar en un archivo separado por comas (*cvs) para su edición posterior en cualquier programa de edición de hojas de calculo, cada una de estas opciones es una función auxiliar que actúa a través de un modelo sobre esa tabla histórico. Otra de las cosas que se hacen internamente al iniciar un ciclo es poner todos los miembros como inactivos, para que se pueda hacer la reinscripción.

Todas las tablas de las configuraciones se vacían para comenzar todo nuevamente y se hace un resumen del año anterior incluyéndolo en el nuevo año como una nueva entrada en la tabla de contable grupo, y con esto registro si quedan ganancias o deudas pendientes del año anterior. Una vez que se comienza un ciclo, se habilitan todos los ingresos en todas las secciones, menos en la sección de materiales que es independiente a el resto del programa.

Al finalizar un ciclo, se inhabilitan todas las posibilidades de ingreso de datos de todas las secciones, pero se permite realizar impresiones.
Las fechas son guardadas en una tabla en la base de datos, cuando se inicia un ciclo, se guarda esa fecha como la fecha de inicio del ciclo actual y las fechas de inicio y finalización del ciclo anterior pasan a guardarse como fechas anteriores. Cuando se finaliza un ciclo, se guarda esa fecha como fecha de finalización del ciclo actual en la tabla de fechas de la base de datos.

Sección Material:

En esta sección se trabaja con un modelo (QSqlTableModel) asociado a la tableview del dialogo, lo que permite hacer transacciones en tiempo real sobre la base de datos. Cada vez que se agrega un material nuevo, se agrega una fila al modelo, que a su vez esta configurado para hacer un insert en la tabla Material de la base de Datos. De forma similar sucede al querer hace una actualización, cuando se hace doble clic sobre cualquier elemento de la tabla que se quiera modificar, se copian los datos en la sección de carga, habilitándose la modificación de los mismos, y dando la opción de guardado para finalizar la operación. Se hace una actualización del modelo y a su vez nuevamente sobre la base de datos. Distinto es el caso de Borrar o dar de baja un material, ya que en este caso no se hace una operación delete sobre la base de datos, sino que se cambia el valor de la columna estado, quedando como “Dado de baja”, es decir solo se actualiza el valor de esa columna, en el modelo, y en consecuencia, en la base de datos.

La búsqueda de cualquier elemento, funciona haciendo un filtrado sobre el modelo, dejando de lado la base de datos. Según la elección de filtro en la interfaz, columna que se eligiera internamente para el filtrado. También se puede activar o desactivar la visualización de los elementos dados de baja, que internamente es permitir que en la columna de estado se agrega la opción de “Dados de baja” como parte de los estados posibles. Los dados de baja pueden ser activados o desactivados independientemente de la búsqueda que se esté haciendo, ya que es solo una inclusión de el string “Dado de baja” para la columna estado, dentro de la cadena final filtro.

8. Distribución de paquetes binarios.

Uno de los objetivos principales del programa es que se pueda utilizado en diferentes sistemas operativos. Para esto, se describirá de manera breve las herramientas que se utilizaron para empaquetar el programa en las diferentes plataformas (Sistema operativos - arquitectura).

Windows

Para empaquetar la versión de Windows, se utilizo el binario compilado con la versión de la IDE para Windows. Luego, se utilizo el programa Dendency Walker para averiguar los dll necesarios para el funcionamiento de la aplicación. Una vez obtenidos los dlls, se utilizó el programa Inno Installer para hacer el instalador indicado. Una vez compilado el script del instalados se obtuvo un archivo *.exe para su posterior instalación.

Linux

Similar a la versión de Windows, se utilizo el binario compilado con la IDE. Se empaqueto el binario con el programa debreate, con previo strip (compresión del binario) del binario por consola (comando strip en la terminal de Linux). Una vez establecidas las dependencias y compilado el paquete se obtuvo un archivo *.deb que es similar a un *.exe de Windows.

Distribución

Para la distribución de los binarios y el código fuente, se creo una página en SourceForge con los archivos correspondientes. Para acceder al repositorio dirigirse ha:

Se puede acceder al código fuente del programa accediendo al repositorio de GitHub que se utilizo durante el desarrollo.(https://github.com/lucianodato/GAGS)