

Integración en el escritorio GNOME

Carlos García Campos

carlosgc@gnome.org

Resumen El número de aplicaciones disponibles para el escritorio GNOME no para de crecer, lo cual es bueno ya que hace de GNOME un sistema de escritorio mucho mas rico. Pero muchas de estas aplicaciones fallan en el planteamiento inicial. En general tendemos a planificar el desarrollo de una aplicación como si ésta fuese a estar aislada del mundo. Incluso en ocasiones ponemos en práctica muchos de los conceptos aprendidos en ingeniería del software, realizamos un análisis de requisitos, elaboramos los casos de uso, etc. y nos olvidamos de una parte fundamental: nuestra aplicación va a ejecutar sobre un sistema de escritorio y va a convivir con otras aplicaciones. Si además de disponer de una amplia variedad de aplicaciones para GNOME éstas muestran una consistencia y una integración el resultado es un escritorio mas rico aun.

1. Integración

La integración en el escritorio puede entenderse, al menos, de dos formas:

- Integración de las aplicaciones con el sistema de escritorio
- Integración entre las aplicaciones disponibles en el escritorio

La integración de las aplicaciones con el sistema de escritorio hace referencia a que todas las aplicaciones sean consistentes de manera que, aunque cada una esté diseñada para un propósito distinto, su manejo sea similar e incluso exactamente igual para las tareas que son comunes. Del mismo modo deben ofrecer un aspecto similar y cumplir con los objetivos generales del sistema de escritorio, como la usabilidad, simplicidad, etc.

La integración entre las aplicaciones disponibles en el escritorio se consigue mediante la comunicación entre ellas, ejecutando otra aplicación para hacer una tarea relacionada, etc.

En GNOME disponemos de los recursos necesarios para añadir o mejorar la integración de nuestras aplicaciones con el escritorio.

2. Integración con el sistema de escritorio

2.1. Ficheros .desktop

Uno de los problemas mas comunes es la dificultad de encontrar el ejecutable de un programa instalado. No parece lógico que los usuarios tengan que saberse de memoria como se llama el archivo ejecutable de cada aplicación instalada para poder ejecutarla. Es por ello que es muy importante “publicar” nuestra aplicación en el sistema de escritorio. Para ello disponemos de los ficheros .desktop, utilizados para añadir una entrada al menú, crear un lanzador para el panel, etc.

Los ficheros .desktop son ficheros de texto plano que contienen información acerca de la aplicación. En GNOME se usa la especificación estándar de Freedesktop.

```
[Desktop Entry]
Encoding=UTF-8
Name=File Browser
Name[en_GB]=File Browser
Name[es]=Examinador de archivos
Comment=Browse the filesystem with the file manager
Comment[en_GB]=Browse the filesystem with the file manager
Comment[es]=Examinar el sistema de archivos con el administrador de archivos
TryExec=nautilus
Exec=nautilus --no-desktop --browser %U
Icon=file-manager
Terminal=false
StartupNotify=true
Type=Application
Categories=GNOME;Application;System;Utility;Core;
OnlyShowIn=GNOME;
X-GNOME-Bugzilla-Bugzilla=GNOME
X-GNOME-Bugzilla-Product=nautilus
X-GNOME-Bugzilla-Component=general
X-Gnome-Bugzilla-OtherBinaries=nautilus-adapter;nautilus-content-loser;nautilus-sidebar-loser;nautilus-text-view;nautilus-throbber;
```

Este ejemplo corresponde al fichero .desktop del examinador de ficheros Nautilus. Está compuesto por una sección principal llamada Desktop Entry y un conjunto de pares Clave=valor. Tan solo son obligatorias Type, Encoding y Name, aunque muchas otras

son muy aconsejables como Comment, Icon, Exec, Terminal, etc. Algunas pueden ser traducibles añadiendo el código del idioma entre corchetes a continuación del nombre de la clave.

La mayoría son bastante descriptivas:

- TryExec: es el nombre del binario en el sistema de ficheros. Se utiliza para comprobar que el binario está disponible para ser ejecutado, de lo contrario el programa no será mostrado en los menús, escritorio, panel, etc.
- Exec: representa el comando completo, con los posibles argumentos
- Terminal: indica si el programa debe ser lanzado en una terminal

2.2. El selector de ficheros

El selector de ficheros es un elemento común en todas las aplicaciones que necesitan abrir y guardar ficheros o bien acceder al sistema de ficheros. Usar el selector de ficheros de GNOME tienen numerosas ventajas:

- Consistencia entre las aplicaciones
- El selector de ficheros está bien integrado con el sistema de escritorio
- Facilidad de uso, el usuario usa el mismo selector de ficheros, con el que ya está familiarizado, en todas las aplicaciones
- Mayor seguridad, ya que el código del selector de ficheros está mucho más probado y trabajado
- Reutilización de código

El selector de ficheros es además personalizable según las necesidades de cada aplicación, de manera que es posible añadir opciones extra, opciones de previsualización, etc.



Figura 1. Selector de ficheros personalizado de file-roller

Otra característica importante es la posibilidad de usar el selector de ficheros embebido en nuestra aplicación permitiendo un acceso directo al sistema de ficheros. Aplicaciones como los visores de imágenes, grabadores de CD, etc, suelen incluir acceso directo al sistema de ficheros.

Hacer un simple visor de imágenes utilizando el selector de ficheros podría ser tan sencillo como crear un `GtkFileChooserWidget` e insertarlo en cualquier contenedor de la aplicación:

```
file_chooser = gtk_file_chooser_widget_new (
    GTK_FILE_CHOOSER_ACTION_OPEN);
filter = gtk_file_filter_new ();
gtk_file_filter_add_pixbuf_formats (filter);
gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (file_chooser),
    filter);
gtk_box_pack_start (GTK_BOX (hbox), file_chooser,
    TRUE, TRUE, 0);
gtk_widget_show (file_chooser);
```

Después bastaría con capturar el evento “selection-changed” del selector de ficheros y actualizar una imagen, el manejador podría ser tan simple como:

```
static void
on_file_chooser_selection_changed (GtkWidget *widget,
    gpointer gdata)
{
    GtkImage *image = GTK_IMAGE (gdata);
    gchar *filename = NULL;

    filename = gtk_file_chooser_get_filename (
        GTK_FILE_CHOOSER (widget));

    if (filename) {
        gtk_image_set_from_file (image, filename);
        gtk_widget_show (GTK_WIDGET (image));
        g_free (filename);
    }
}
```

2.3. Sistema de preferencias centralizado

GNOME dispone de un sistema de almacenamiento de configuraciones y preferencias centralizado llamado GConf. El tener todas las preferencias y configuraciones centralizadas permite a los usuarios mas avanzados el acceso a dichas preferencias sin tener que conocer con profundidad cada aplicación para saber dónde y cómo almacena sus configuraciones. Por otro lado permite tener ciertas configuraciones compartidas por varias aplicaciones o configuraciones globales al escritorio.

GConf ofrece a los desarrolladores un mecanismo sencillo de almacenamiento de las preferencias. Los administradores y usuarios avanzados disponen de aplicaciones como gconftool o gconf-editor para el acceso y mantenimiento del sistema GConf.

El sistema GConf permite a las aplicaciones consultar sus preferencias, actualizarlas e incluso ser notificadas ante cambios.

2.4. Archivos recientes

Las aplicaciones que manejan ficheros, ya sea un documento de texto, un archivo de audio o vídeo, etc. pueden mantener un acceso rápido a los últimos ficheros utilizados mas recientemente por cada usuario, tanto internamente desde la propia aplicación como desde el sistema de escritorio de una forma global.

El código para el manejo de los archivos recientes se encuentra de momento en libegg, por lo que es común, copiar directamente los archivos necesarios a nuestro directorio de fuentes y usarlos como si fuesen parte de nuestra aplicación.

El manejo es bastante sencillo, lo mas común es querer mantener una lista actualizada de los archivos que se abran en el menú principal de la aplicación. Para ello es necesario crear un modelo donde guardar la lista de los ficheros

```
EggRecentModel *recent_model;

recent_model = egg_recent_model_new (EGG_RECENT_MODEL_SORT_MRU);
egg_recent_model_set_limit (recent_model, 5);
egg_recent_model_set_filter_groups (recent_model,
    "TestRecent", NULL);
```

Podemos, entonces, construir una vista para ese modelo en el menú de la aplicación:

```
EggRecentViewGtk *recent_view;

recent_view = egg_recent_view_gtk_new (menu, menu_item);
egg_recent_view_gtk_set_trailing_sep (recent_view, TRUE);
egg_recent_view_set_model (EGG_RECENT_VIEW (recent_view),
    recent_model);
```

Al crear la vista indicamos el menú donde va a ser insertada y el ítem del menú después del cual se insertará. Ahora cada vez que nuestra aplicación abra un fichero lo incluiremos en el modelo y la vista será actualizada:

```
EggRecentItem *item;

item = egg_recent_item_new_from_uri (uri);
egg_recent_item_add_group (item, "TestRecent");
egg_recent_model_add_full (recent_model, item);
```

Para manejar las pulsaciones del usuario sobre las entradas del menú correspondientes a los ficheros recientes, podemos añadir un manejador de la siguiente forma:

```
g_signal_connect (recent_view, "activate",
                  G_CALLBACK (on_recent_file_activate),
                  NULL);
```

Y en el manejador recibiremos tanto la vista como el ítem seleccionado:

```
static void
on_recent_file_activate (EggRecentViewGtk *view,
                        EggRecentItem *item,
                        gpointer gdata)
{
    gchar *uri;
    uri = egg_recent_item_get_uri (item);

    . . .

    g_free (uri);
}
```

2.5. Guía de Interfaces Humanas

La guía de interfaces humanas (HIG) es un documento que guía al desarrollador o al diseñador de interfaces en la creación de interfaces gráficas de usuario. La manera en que el usuario se comunica con el ordenador en un sistema de escritorio como GNOME es interactuando mediante las interfaces gráficas. Es por ello que deben estar

muy cuidadas y seguir unas directrices comunes que hagan al usuario sentirse cómodo con las aplicaciones. Si todas las aplicaciones siguen las HIG obtenemos, en conjunto, un sistema de escritorio en el que todas los programas tienen un aspecto similar y se comportan igual en los puntos que son comunes. Esto hace que el usuario no tenga que aprender como usar cada uno de los programas del escritorio sino que aprende un comportamiento global que puede aplicar a cada una de las aplicaciones que usa.

Hoy en día es un requisito imprescindible para todas las aplicaciones que forman parte de GNOME, el seguir de la forma mas estricta posible las HIG.

2.6. El área de notificación

El área de notificación es otro espacio común del sistema de escritorio que las aplicaciones pueden utilizar para enviar notificaciones, información del estado, etc.

Es importante no confundir los applets con los iconos en el área de notificación.

3. Integración entre las aplicaciones

3.1. Nautilus

Nautilus es el eje central del sistema GNOME. Es quien comunica al usuario con el sistema de ficheros, permitiendo un acceso no solo local sino también remoto. Nautilus ofrece a los desarrolladores un sencillo sistema de extensiones mediante el cual es posible añadirle funcionalidades desde las propias aplicaciones.

3.1.1. Hojas de propiedades Dada la naturaleza de cada archivo no es posible tener una hoja de propiedades común para todos los tipos. Un archivo de audio, por ejemplo, tendrá información como la duración que no tendría sentido en un documento de texto. Es posible extender la hoja de propiedades que muestra nautilus para un archivo añadiendo la información específica de dicho archivo. Así por ejemplo, el visor de documentos pdf podría extender la hoja de propiedades de los archivos de tipo pdf para mostrar información específica sin necesidad de cambiar el código de nautilus.

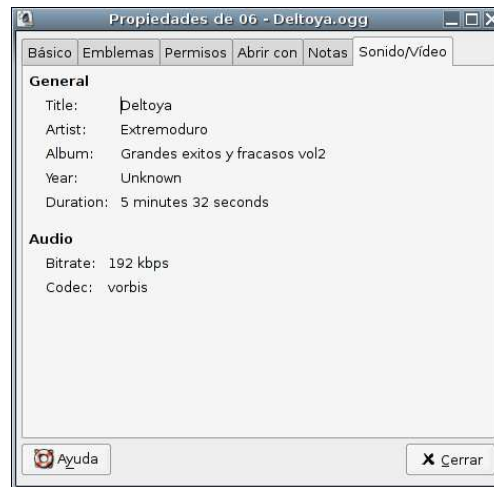


Figura 2. Hoja de propiedades de Nautilus para un fichero de audio

3.1.2. Menú contextual De la misma forma es posible añadir elementos al menú contextual de nautilus que son específicos para un tipo de archivo concreto. Esta extensión es implementada por las aplicaciones sin necesidad de modificar el código de nautilus.

Esta funcionalidad es utilizada por aplicaciones como file-roller, rhythmbox, nautilus-sendto o el módulo shares-admin de GNOME System Tools.

3.2. Arrastrar y soltar

Es una acción muy común en todos los sistemas de escritorio que ofrece una manera muy sencilla e intuitiva de comunicación entre aplicaciones. Es posible hacer que elementos de nuestra aplicación sean “arrastrables” a otras aplicaciones y/o que elementos de otras aplicaciones puedan ser “arrastrados” a la nuestra. Así, por ejemplo, si nuestra aplicación maneja algún tipo de fichero es común permitir el acceso a dichos ficheros arrastrándolos desde nautilus.

Aplicaciones como nautilus permiten realizar una gran cantidad de acciones arrastrando y soltando elementos de unos puntos a otros.

3.3. Ejecución de otras aplicaciones

En ocasiones puede ser necesario realizar desde una aplicación tareas que están relacionadas con ella, pero que no son funcionalidades propias de dicha aplicación. Es

importante saber que hay otras aplicaciones en el escritorio que son capaces de realizar la tarea requerida. Para este tipo de acciones es posible ejecutar otros programas de forma síncrona o asíncrona desde nuestra aplicación. Así es posible ejecutar el reproductor de películas para ver un vídeo, el navegador web para ver una página web, etc. Por otro lado es importante que los programas implementen una completa interfaz a través de la línea de comandos que permita su ejecución con las opciones necesarias.

Rhythmbox, por ejemplo, es una aplicación de catalogación y reproducción de audio. La extracción de un CD de audio en ficheros no es una funcionalidad propia de la aplicación, pero si está relacionada, ya que Rhythmbox trabaja con ficheros de audio. Por tanto, es lógico pensar que puede resultar útil disponer de una opción de extracción de audio desde un CD en Rhythmbox, para luego añadir los ficheros resultantes a nuestra biblioteca musical. Puesto que existe una aplicación especializada en ello, es posible añadir dicha funcionalidad a Rhythmbox ejecutando dicha aplicación especializada y dejando que ella haga el trabajo para el que ha sido diseñada.

Existen muchos otros ejemplos en el escritorio GNOME donde una aplicación llama a otra para que realice una acción, como Nautilus, evolution, epiphany, etc.

En la librería GLib tenemos disponibles una serie de funciones que nos permiten la ejecución tanto síncrona como asíncrona de otros programas. Ejecutar el reproductor de películas totem para ver un dvd podría ser tan sencillo como:

```
gchar *player;
gchar *command;

if ((player = g_find_program_in_path ("totem"))) {
    command = g_strdup_printf ("%s dvd://", player);
    g_spawn_command_line_async (command, NULL);
    g_free (player);
    g_free (command);
}
```

3.4. Servidores de datos

Otra forma de comunicación, que proporciona integración, es a través de servidores de datos. La idea es separar el almacenamiento y acceso a los datos de nuestra aplicación en un servidor de datos. De esta manera es posible, no solo desde nuestra aplicación

sino desde cualquier otra, acceder a la información siempre actualizada. Es el caso de evolution-data-server que permite a otros programas acceder a los contactos, citas del calendario, tareas pendientes, etc. de evolution.

Así, por ejemplo, el applet del reloj es capaz de mostrar un calendario con las citas y tareas pendientes del calendario de evolution. El applet de búsqueda de contactos y nautilus-sendto son ejemplos de aplicaciones que consultan la libreta de direcciones de evolution.

El acceso a los datos se realiza a través de funciones de librería, por lo que solo es necesario conocer la API de dichas librerías. El acceso a las tareas pendientes, por ejemplo, requiere de unas pocas líneas:

```
ECal    *cal;
GList   *tasks;
icalcomponent *ical;
icalproperty *prop;

cal = e_cal_new_system_tasks ();
e_cal_open (cal, FALSE, NULL);
e_cal_get_object_list (cal, "#t", &tasks, NULL);

while (tasks) {
    ical = tasks->data;
    prop = icalcomponent_get_first_property (ical,
                                             ICAL_SUMMARY_PROPERTY);
    if (prop)
        g_print ("%s\n", icalproperty_get_summary (prop));
    tasks = g_list_remove (tasks, ical);
}
```

4. Conclusiones

Para llevar a cabo estas técnicas o posibilidades de integración se utilizan tanto tecnologías que son propias de GNOME, como muchas otras que son un estándar de Freedesktop. La tendencia actual es utilizar, siempre que sea posible estándares, principalmente de Freedesktop. En GNOME, por tanto, tenemos la tecnología necesaria para

implementar, prácticamente, cualquier técnica de integración entre aplicaciones y con el escritorio. Aunque es cierto que la integración es algo que cada vez se tiene mas presente a la hora de desarrollar aplicaciones, hay mucho que mejorar todavía en GNOME. Nuevas tecnologías como Dbus de Freedesktop ayudan a mejorar la integración facilitando la comunicación entre las aplicaciones.

5. Referencias

- Desarrollo GNOME: <http://developer.gnome.org>
- Freedesktop: <http://www.freedesktop.org>