

# Fundamentos de GTK

<https://youtube.com/@makigas>

<https://www.makigas.es>

# ¿Qué es GTK?

- GTK es una tecnología para hacer aplicaciones gráficas
- GTK es libre (LGPL) y portable (X11, Wayland, Windows, macOS, web)
- GTK se puede usar desde una multitud de lenguajes de programación

[Donaciones](#) [Crear una cuenta](#) [Acceder](#) ...

## GTK

[42 idiomas](#)[Artículo](#) [Discusión](#)[Leer](#) [Editar](#) [Ver historial](#) [Herramientas](#)

GTK (conocido hasta febrero de 2019 como **GTK+**)<sup>1 2</sup> o *The GIMP Toolkit*<sup>3 4:3</sup> es una biblioteca de componentes gráficos multiplataforma para desarrollar interfaces gráficas de usuario (GUI).<sup>5 4:3</sup> Está licenciado bajo los términos de la GNU LGPL,<sup>5</sup> por lo que permite la creación de tanto software libre como software privativo. GTK es parte del proyecto GNU,<sup>5</sup> siendo usado por proyectos tan importantes como GIMP, Inkscape, GNOME, Xfce, entre otros.

Junto a Qt, es uno de los kits de herramientas de widgets más popular para el sistema operativo GNU/Linux, teniendo un amplio soporte para Wayland y XOrg. Además, se puede emplear para desarrollar aplicaciones gráficas que funcionen sobre otros sistemas Unix-like o sistemas operativos como Microsoft Windows, Mac OS, entre otros.

Historia y Desarrollo [\[ editar \]](#)



# Portabilidad

- GTK está escrito en C para ser fácil de portar
- Cualquier lenguaje de programación con soporte para funciones externas admite GTK
- Bindings más importantes: gtkmm, pygobject, gjs, gtk-rs.
- Vala.

# Funcionamiento técnico

## GTK es orientado a objetos

- Hay **clases** (Window, Button, Label...)
- Las clases tienen **métodos** (set\_visible, destroy, open\_file...)
- También hay **atributos, callbacks, propiedades...**

## Esto incluye C.

- GTK está hecho por encima de **GObject**, un framework de orientación a objetos para C.
- Hay más tecnologías de GObject: GLib, GIO...
- Otros lenguajes usan lo que más se le parezca

# GObject (1/6)

## Sistema de tipos

- **Espacios de nombres:** organizan clases. Gtk, GLib, Pango, Adw...
- **Clases:** representan clases. Incluyen herencia, interfaces...
- **Tipo:** representa clases, interfaces, uniones, estructuras...
- **Señales:** representan eventos (útil para crear callbacks).

## GObject (2/6)

### Adaptación a cada lenguaje

- Se extrae el sistema de tipos del lenguaje de programación en el que se escribe la biblioteca (p.ej., C).
- Se fabrica un esquema de datos que describe los elementos de la API (preferentemente .typelib)
- Con ese esquema luego se puede fabricar un binding para cada lenguaje.
  - En tiempo de ejecución / compilación si el runtime lo soporta (por ejemplo, Python, JavaScript).
  - Pre-generar código fuente en otro lenguaje de programación (por ejemplo, Rust).

# GObject (3/6)

docs.gtk.org

Click, or press 's' to search

## GLIB

API Version: 2.0

Library Version: 2.83.0

### Type

Bytes

### Constructors

`new`

`new_static`

`new_take`

`new_with_free_func`

### Instance methods

`compare`

`equal`

`get_data`

`get_region`

`get_size`

`hash`

`new_from_bytes`

`ref`

## Struct

## GLib ▸ Bytes

since: 2.32

### Description #

[src]

```
struct GBytes {  
    /* No available fields */  
}
```

A simple refcounted data type representing an immutable sequence of zero or more bytes from an unspecified origin.

The purpose of a `GBytes` is to keep the memory region that it holds alive for as long as anyone holds a reference to the bytes. When the last reference count is dropped, the memory is released. Multiple unrelated callers can use byte data in the `GBytes` without coordinating their activities, resting assured that the byte data will not change or move while they hold a reference.

A `GBytes` can come from many different origins that may have different procedures for freeing the memory region. Examples are memory from `g_malloc()`, from memory slices, from a `GMappedFile` or memory from other allocators.

`GBytes` work well as keys in `GHashTable`. Use `g_bytes_equal()` and `g_bytes_hash()` as parameters to `g_hash_table_new()` or `g_hash_table_new_full()`. `GBytes` can also be used

## GObject (4/6)

### Simplificación de C

```
struct app_document {  
    app_parent parent_type;  
  
    char *filename;  
    guint cursor_line;  
    guint cursor_column;  
};  
  
app_document *app_document_new();  
void app_document_free(app_document *document);  
void app_document_open(app_document *document, char *path);  
void app_document_goto(app_document *document, guint line, guint col);
```



## GObject (5/6)

### Cosas que aporta GObject a C

- Sistema de tipos.
- Representación de la orientación a objetos.
- Gestor de memoria propio con soporte para smart pointers y weak pointers.

# GObject (6/6)

## Otros lenguajes

- Mediante introspección se puede crear un binding de una API de GObject en un lenguaje.
- Cada lenguaje interpreta un tipo de GObject según la semántica de su lenguaje.
- Si el lenguaje es orientado a objetos, ese sistema de tipos podría ser nativo.
- Si el lenguaje no es orientado a objetos, lo adaptará como pueda.

- GTK es una biblioteca hecha por encima de GObject.
- Incorpora una serie de **Widgets** para describir los elementos de una interfaz.
- Los widgets tienen propiedades (título, texto, tooltip...)
- Los widgets tienen señales y eventos (clicked, changed...)

# Stack tecnológico de una aplicación GTK

Resumen (<https://www.gtk.org/docs/architecture/>)



Figura 3: Arquitectura de un programa GTK

# Stack tecnológico de una aplicación GTK

## Componentes más habituales

- GObject: framework de orientación a objetos y sistema de tipos.
- GLib: biblioteca estandar con estructuras de datos, colecciones, hilos...
- GIO: biblioteca de entrada y salida (archivos, conexión de red, settings, IPC...)

## Componentes gráficos más habituales

- GDK y GSK: capas intermedias que permiten simplificar GTK.
- Pango: biblioteca para trabajar con tipografía.
- Cairo: biblioteca para trabajar con gráficos.

# Stack tecnológico de una aplicación GTK

## Componentes extra para una aplicación gráfica

- Libadwaita: conjunto extra de componentes gráficos que siguen la HIG de GNOME.
- WebKitGTK: navegadores web.
- Libsoup: cliente HTTP.
- VTE: consolas y terminales.
- GtkSourceView: editores de código fuente.
- Shumate: mapas.

# ¿Por qué elegir GTK?

- Aplicación nativa: no hace falta empaquetar un navegador web.
- Iteración rápida durante el desarrollo.
- Integración con el estilo de cada plataforma. (Al menos en X11 y Wayland).

(Bajo mi punto de vista, consulte siempre con su farmacéutico.)

# GTK frente a Qt

- Ambas tienen bindings para otros lenguajes de programación.
- Ambas tienen su propio conjunto de widgets.
- Ambas son multiplataformas.
- Qt siempre ha tenido un modelo de licencias un poco peculiar.
- GTK es LGPL.

(Bajo mi punto de vista, consulte siempre con su farmacéutico.)



# Componentes de una aplicación GTK

## Application (Gtk.Application)

- Evita gestionar cosas como la entrada de parámetros y la inicialización de la aplicación.
- De gratis: gestión de ventanas en programas multidocumento.
- Se ocupa del event loop principal.
- También gestiona eventos de sistema (abrir archivo, activar...)