Interfaces gráficas de usuario

Ricardo Pérez López

IES Doñana, curso 2025/2026

Generado el 2025/07/16 a las 21:19:00

Índice

1. Introducción a Tkinter

1.1. ¿Qué es Tkinter?

Tkinter es la biblioteca estándar de Python para crear interfaces gráficas de usuario (GUI).

Proporciona una forma sencilla de construir aplicaciones con ventanas, botones, etiquetas, campos de texto y otros elementos visuales.

Es multiplataforma, lo que significa que las aplicaciones creadas con Tkinter pueden ejecutarse en diferentes sistemas operativos como GNU/Linux, Windows y macOS sin modificaciones importantes.

Es relativamente fácil de aprender y usar, lo que la hace adecuada para principiantes en el desarrollo de interfaces gráficas.

1.2. Instalación y primeras pruebas

Tkinter forma parte de la biblioteca estándar de Python, por lo que debería encontrarse en cualquier instalación normal de Python.

Esa es una de las principales ventajas que tiene respecto a otras bibliotecas de GUI como PyQt.

Al ejecutar el siguiente comando:

```
$ python -m tkinter
```

desde la línea de comandos del sistema operativo, se debería abrir una ventana que demuestre una interfaz Tk simple para saber si tkinter está instalado correctamente en su sistema.

También muestra qué versión de Tcl/Tk está instalada para que pueda leer la documentación de Tcl/Tk específica de esa versión.

1.3. Recursos y documentación

Para saber más sobre Tkinter o resolver dudas técnicas puntuales, se pueden consultar los siguientes enlaces:

- Página de documentación oficial de Tkinter en `python.org`

  Información técnica y oficial de Tkinter.

- TkDocs

  Es un extenso tutorial sobre como crear interfaces de usuario con Tkinter. Explica conceptos clave y muestra enfoques recomendados para usar la API moderna.

- Tkinter 8.5 reference: a GUI for Python

  Documentación de referencia sobre Tkinter 8.5 donde se detallan clases disponibles, métodos y opciones.

1.4. Un primer ejemplo básico

Un *Hola, mundo* muy elemental en Tkinter podría ser el siguiente:

```
import tkinter as tk                          # Importa el módulo tkinter con el nombre tk
raiz = tk.Tk()                                # Crea la ventana principal
tk.Button(raiz, text="Hola, mundo").grid()    # Crea un botón dentro de la ventana principal
raiz.mainloop()                               # Activa el bucle principal
```

Este programa simplemente abre una ventana en la que se muestra un botón con el texto «`Hola, mundo`».

El botón tiene el ancho justo para visualizar el texto que contiene, y la ventana tiene prácticamente el tamaño justo para contener el botón.

Al pulsar ese botón no ocurre nada, y para salir del programa hay que finalizarlo cerrando la ventana o directamente terminando el proceso del intérprete.

1.5. La ventana principal (`Tk`)

La clase `Tk` representa la ventana principal de la aplicación, y esta es importante por varios motivos:

- Toda aplicación Tkinter debe tener una ventana principal que sea instancia de `Tk` (y lo normal es que sólo sea exactamente una).
- Las instancias de `Tk` son contenedores de elementos gráficos (también llamados *widgets*). Salvo casos excepcionales (como las ventanas de diálogo), los *widgets* se deben visualizar siempre dentro de un contenedor y la instancia de `Tk` que creamos para nuestro programa nos sirve como contenedor principal de nuestra aplicación.

  Sin un contenedor, no podríamos visualizar *widgets*, así que la ventana principal es imprescindible en cualquier aplicación Tkinter.

El funcionamiento de la interfaz gráfica realmente comienza cuando activamos el bucle principal de la ventana principal invocando su método `mainloop`.

1.6. El bucle principal (*mainloop*)

En interfaces gráficas, la ejecución del programa está dirigida por eventos (por ejemplo, pulsar un botón o elegir una opción en un menú) y no por un flujo lineal de instrucciones como en los programas de consola.

El bucle principal (*mainloop*) de Tkinter es la función que pone en marcha la aplicación gráfica y la mantiene funcionando hasta que el usuario la cierra.

El bucle principal se activa invocando el método `mainloop` sobre la instancia de `Tk` que representa la ventana principal de la aplicación.

A partir de ese momento, el programa irá atendiendo los eventos que se vayan produciendo ejecutando el código encargado de *manejar* o *gestionar* dicho evento.

Sus funciones principales son:

- Inicia la gestión de eventos:

Tkinter entra en un bucle infinito en el que espera la aparición de eventos (pulsaciones de los botones del ratón, pulsaciones de teclas, redimensionado de ventanas, etc.) y los gestiona según los manejadores de eventos que se hayan definido en el código.

- Mantiene visible la ventana:

  Mientras `mainloop` está activo, la ventana de la aplicación se sigue mostrando y respondiendo a interacciones. Si no se llama a `mainloop`, la ventana puede crearse y cerrarse instantáneamente, porque el programa termina sin esperar eventos.

- Es un bucle de eventos:

  Internamente, comprueba si hay nuevos eventos en la cola de eventos del sistema operativo y los procesa uno a uno, actualizando la interfaz cuando sea necesario.

El bucle principal termina:

- Cuando el usuario cierra la ventana principal, o
- Si el programa llama explícitamente al método `quit` de la ventana principal para finalizarlo.

Aspectos importantes a tener en cuenta:

- El bucle principal es *bloqueante*:

  Cuando se llama a `mainloop`, no se ejecuta nada después de esa línea hasta que el bucle termina.

- No se debe llamar varias veces:

  Normalmente, se llama una sola vez. Si se necesita reiniciar la ventana, se debe crear un nuevo `Tk` o usar `mainloop` tras finalizar la ejecución anterior, pero no se debe intentar mantener dos ejecuciones simultáneas de `mainloop`.

2. Widgets básicos

2.1. `Label`, `Button`, `Entry`, `Text`, `Checkbutton`, `Radiobutton`

2.2. Atributos comunes: texto, color, fuente, tamaño

2.3. Métodos útiles: `get`, `insert`, `delete`

3. *Layout* y organización de la interfaz

3.1. Geometría con `pack`, `grid` y `place`

3.2. Uso de `Frame` para dividir la ventana

3.3. Diseño *responsive* básico

4. Eventos y funciones asociadas

4.1. Asociar funciones a eventos (*callbacks*)

4.2. Uso de `command=`

4.3. Eventos con `bind`

4.4. Variables de control (`StringVar`, `IntVar`, etc.)

Bibliografía

Roseman, Mark. n.d. “TkDocs.com.” https://tkdocs.com/.