



Interfaces Gráficas de Usuario

JAIME ALBERTO GUZMAN LUNA
Universidad Nacional de Colombia

1

1



Interfaces Gráficas de Usuario (GUI)

- Los programas utilizados hasta ahora solo usan la consola para interactuar con el usuario.
- Las GUI permiten al usuario el manejo de ventanas, barras de herramientas, cuadros de diálogo, listas desplegables, botones, y otros elementos que bajo el ambiente de Windows ya estamos muy acostumbrados a tratar.
- En este ambiente, las aplicaciones son **orientadas por eventos** y se crean utilizando las clases que para ello ofrece la librería de Python.
- **Tkinter** y **Tkinter.ttk** son APIs que están compuesta por un conjunto de clases y derivaciones para el desarrollo de interfaces gráficas de usuario

2

2

Tkinter y Tkinter.ttk

Tkinter

- Tkinter, abreviatura de "interfaz Tk", es la interfaz estándar de Python para el kit de herramientas GUI de Tk.
- Proporciona una variedad de widgets, como botones, etiquetas y cuadros de texto, para crear aplicaciones GUI.
- Los widgets de Tkinter son funcionales y fáciles de usar, pero pueden carecer del aspecto moderno que los usuarios esperan de las aplicaciones contemporáneas.

Tkinter.ttk

- El módulo `ttk`, introducido en Tk 8.5 y disponible en la biblioteca estándar de Python, proporciona widgets temáticos. Estos widgets están diseñados para brindar una apariencia más nativa en diferentes sistemas operativos.
- Los widgets `ttk` utilizan temas nativos de la plataforma para mejorar la estética visual de las aplicaciones, haciéndolas lucir más modernas y consistentes con otras aplicaciones en el sistema del usuario.



Uso en el curso actual

- Se usará para el curso Tkinter, dado que es, en general, más fácil de usar para principiantes debido a su enfoque sencillo.

3

3

Se usará para el curso una combinación de Tkinter y Tkinter.ttk, dado que así podremos obtener widgets con estilos nativos y más modernos

Estructura básica de Tkinter (1)

Tkinter con ttk

- La Librería Tkinter tiene muchas clases, métodos y atributos que nos ayudan a crear interfaces de usuarios. Algunos de los elementos principales y básicos para crear una GUI son los siguientes:

- Window (tk)
- Widgets (ttk)
- Frames (ttk)
- Administrador de esquemas (Geometry)

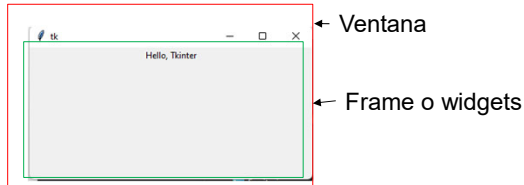
La Librería Tkinter junto con el módulo `ttk` nos proporciona clases, métodos y atributos para crear interfaces de usuario más modernas y personalizables. Algunos de los elementos principales y básicos para crear una GUI con `ttk` son los siguientes:

4

4

Tkinter con ttk

Estructura básica de Tkinter (2)



Una aplicación **Tkinter**, estará compuesta de una ventana (**Window**) de la clase Tk y dentro podrá contener diferentes elementos llamados **Widgets**.

Window (ventana) es el contenedor principal donde se agrupan los diferentes widgets que agreguemos.

Widgets son elementos con funciones exclusivas, como campos de texto, texto, botones entre otros. Incluyen los **Frames**.

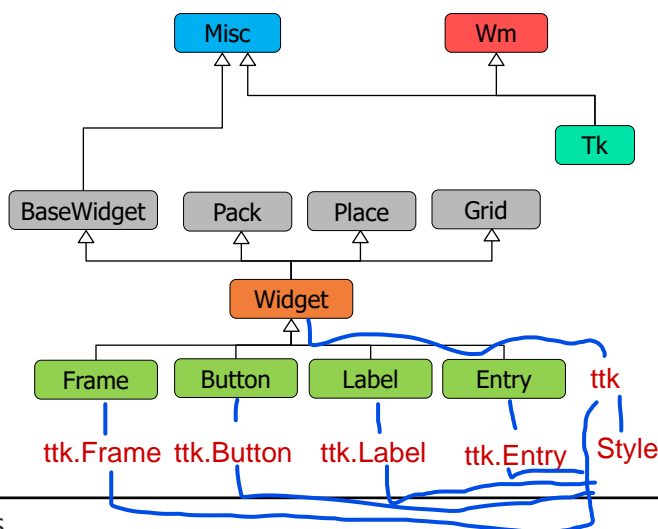
Frame son elementos que nos ayudan a organizar y agrupar los componentes de una interface en una ventana. Son como contenedores

Es importante también agregar que el posicionamiento de un widget en la ventana o frame viene dado por el administrador de esquemas (**Geometry**).

5

5

Diagrama de clases Tkinter



Misc es la clase base que define los métodos comunes que van a existir para los Widgets.

Wm es una clase interna que proporciona las funciones para la comunicación con el administrador de ventanas.

La clase **Tk** hereda los métodos tanto de la clase **Misc** como de **Wm** y **representa la ventana principal de la aplicación**.

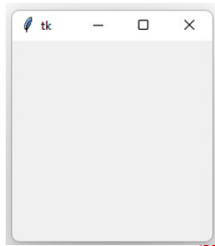
Widget es la clase base de todos los widgets (**Frame**, **Button**, **Label**, etc), los cuales pueden ser posicionados gracias a la herencia de los métodos de los administradores de esquema (**Pack**, **Place**, **Grid**)

6

La clase **Style** en el módulo **ttk** de Tkinter es fundamental para controlar y personalizar la apariencia de los widgets **ttk**. A través de esta clase, los desarrolladores pueden aplicar temas, modificar colores, fuentes, tamaños, y otros aspectos estéticos de los widgets.

La clase **Widget** (y sus variantes en **ttk**, como **ttk.Button**, **ttk.Label**, **ttk.Frame**, etc.) es la base de todos los componentes gráficos que pueden ser posicionados gracias a la herencia de los métodos de los administradores de esquemas (**Pack**, **Place**, **Grid**).

Window o Tk



menús

- Un objeto de la clase **Tk()** es el contenedor base de todos los **widgets** que forman la interfaz.
- En este elemento podemos agregar **menus** o cambiar el nombre o icono de la ventana.

7

7

Ciclo de vida de una aplicación Tkinter

- **Metodo** **mainloop()** **método**
 - Al llamar este **metodo** de la clase **Tk()** se ejecuta el Tkinter y se creará la ventana
 - Este método escucha eventos hasta que la ventana sea cerrada.

- **Metodo** **destroy()**
 - Al llamar este método de la clase **Tk()** se detendrá el método **mainloop()** y **terminara** la ejecución.
terminará

Al llamar este método de la clase Tk(), se detiene el método mainloop() y finaliza la ejecución de la aplicación, cerrando la ventana. Esto aplica tanto para widgets estándar de Tkinter como para los widgets de ttk.

8

8

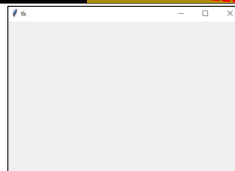
Ejemplo aplicación Tkinter (1)

```
import tkinter as tk

window = tk.Tk()
window.geometry("450x300")

window.mainloop()
```

ejemplo0.py



Para crear una aplicación básica con Tkinter debemos primero importar la librería tkinter, la cual viene incluida en la instalación de Python.

Luego crearemos la ventana creando un objeto de la clase `Tk`.

Usando el método `geometry()` le daremos las dimensiones a nuestra ventana. **Nota:** no confundir con el `geometry` de los administradores de esquemas.

último, Por último usamos el método `mainloop()`, para ejecutar y mostrar la ventana y sus componentes.

El método `mainloop()` es el encargado de estar pendiente de los diferentes eventos que se ejecutan

9

9

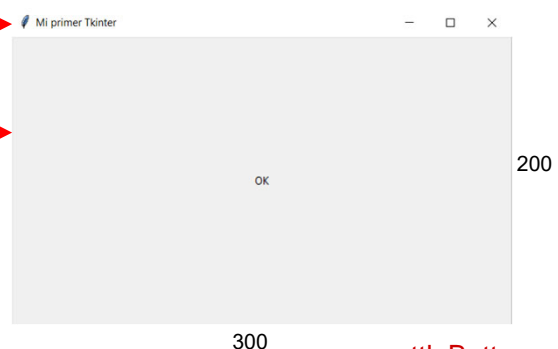
Ejemplo aplicación Tkinter (2)

```
import tkinter as tk
window = tk.Tk()
window.title("Mi primer Tkinter")

button = tk.Button(
    text="OK",
    width=300,
    height=200,
)

button.pack()
window.mainloop()
```

ejemplo1.py



Empaqueta el `button` en la ventana

`Button`

Se crea una ventana que contiene un objeto de tipo `Button` con un tamaño de 300 x 200 píxeles y una etiqueta `OK`.

Nota: cuando no se le define un tamaño a la ventana, se ajusta respecto al tamaño y posición de los componentes que hay contenidos en él.

10

`style.configure("TButton", padding=(300, 200))`: Configura un estilo para los botones (`TButton`). Aquí se establece el `padding`, que es el espacio entre el borde del botón y su contenido.

`style="TButton"`: Aplica el estilo personalizado que se configuró anteriormente.

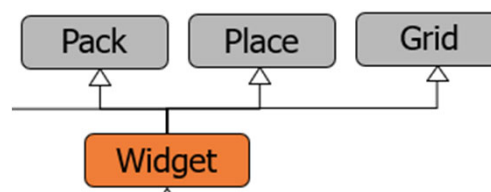
Explicación
código



Administrador de esquemas (Geometry manager)

- En Tkinter todos los Widgets pueden tener un posicionamiento en la ventana. Hay diferentes maneras de posicionar, estas son:

- **Pack**
- **Grid**
- **Place**



11

11



Administrador de esquemas : Pack (1)

- El método `pack()` permite ubicar un widget en una posición específica respecto a su contenedor.
- Similar al `BorderPane` de JavaFX.
- Se define explícitamente en la clase **Pack** y su subclase **Widget** (y el resto de subclases de **Widget**) lo heredan.
- Si colocamos un **widget** con el método **pack** sin argumentos se empaquetará con los atributos por defecto de esta clase. Para conocer estos argumentos por defecto basta con usar en el widget su método `pack_info`.

12

12

Administrador de esquemas : Pack (2)

- La estructura básica es la siguiente:
`widget.pack(side='bottom', anchor='w', padx=0, pady=0, expand=False, fill='x', ...)`
- Los parámetros `side` y `anchor` manipulan directamente el posicionamiento: `side` ubica un widget en cierta posición respecto al centro ('top' para la esquina superior de la ventana, 'bottom' para la esquina inferior, 'left' para la esquina izquierda y 'right' para la esquina derecha); `anchor` tiene la misma función de `side` pero recibe puntos cardinales ('n' para esquina superior, 's' para esquina inferior, 'w' para esquina izquierda, 'e' para esquina derecha y 'c' para el centro). Si se usan ambos parámetros se pueden combinar diferentes posiciones.

Nota: `anchor` permite por sí solo posiciones combinadas. Por ejemplo 'nw', ubicaría el widget en la esquina superior izquierda.

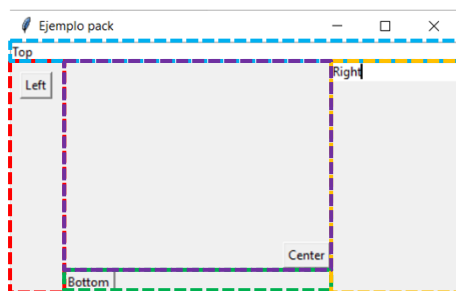
- Los parámetros `padx` y `pady` reciben un entero que corresponde a la distancia mínima (en píxeles) en X y en Y respectivamente que hay entre el widget y otro widget (o de la misma ventana).
- El parámetro `expand` recibe un booleano: si es `True`, el widget se expande en el espacio de la ventana (frame).
- El parámetro `fill` permite extender un widget en una dirección hasta que alcance el tamaño de la ventana o hasta que choque con otro widget ('x' para extenderse horizontalmente, 'y' para extenderse verticalmente y 'both' para ambas direcciones.)

Nota: Este parámetro depende directamente del parámetro `side`: si `side` recibe 'top' o 'bottom', `fill` solo puede extender de manera horizontal; si `side` recibe 'left' o 'right' solo puede extender de manera vertical. La única manera de que se pueda extender en ambas direcciones es que el widget esté en todo el `centro` (es decir, que `expand` reciba un valor de `True`).

13

13

Administrador de esquemas : Pack (3)



Al posicionar widgets, se definen áreas respecto a la posición, las cuales su tamaño dependen de las características del widget que haya ubicado en esta. En el ejemplo, se ve que luego de ubicar 4 widgets en diferentes posiciones, el siguiente que se ubique solo podrá estar dentro de la región morada.

```
import tkinter as tk

ventana = tk.Tk()
ventana.title("Ejemplo pack")
ventana.geometry("400x200")

entryTop=tk.Entry(ventana)
botonLeft = tk.Button(ventana,text="Left")
entryRight=tk.Entry(ventana)
botonBottom=tk.Button(ventana,text="Bottom")
botonCenter=tk.Button(ventana,text="Center")

entryTop.pack(side="top",fill="x")
botonLeft.pack(side="left",anchor="n",padx=10,pady=10)
entryRight.pack(side="right",anchor="n")
botonBottom.pack(side="bottom",anchor="w")
botonCenter.pack(expand=True,anchor="se")

ventana.mainloop()
```

ejemplo2.py

14

14

Administrador de esquemas : Grid (1)

- El **método** `grid()` sirve para posicionar en columnas y filas el widget.
- Similar al `GridPane` de JavaFX.
- Se define explícitamente en la clase **Grid** y su subclase **Widget** (y el resto de subclases de **Widget**) lo heredan.

15

15

Administrador de esquemas : Grid (2)

GRID	Column 0	Column 1	Column 2
Row 0	Column 0 Row 0	Column 1 Row 0	Column 2 Row 0
Row 1	Column 0 Row 1	Column 1 Row 1	Column 2 Row 1
Row 2	Column 0 Row 2	Column 1 Row 2	Column 2 Row 2
Row 3	Column 0 Row 3	Column 1 Row 3	Column 2 Row 3

En cada celda específica se ubica un widget.

La cantidad de filas y columnas se definen al momento de ubicar cada widget.

16

16

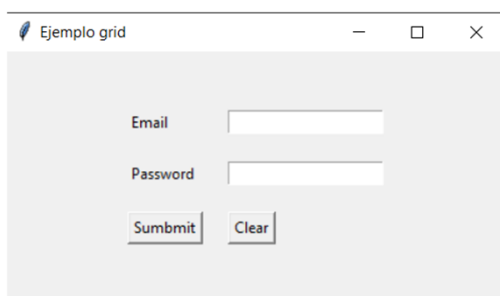
Administrador de esquemas : Grid (3)

- La estructura básica es la siguiente:
`widget.grid(row=0, column=0, columnspan=1, rowspan=1, padx=0, pady=0, sticky='w' ...)`
- Los parámetros *row* y *column* indican la posición de fila y columna respectivamente en la que se ubica el widget.
- Los parámetros *columnspan* y *rowspan* indican cuántas columnas o filas respectivamente ocupa el widget (por defecto el valor es 1).
- Los parámetros *padx* y *pady* reciben un entero que corresponde a la distancia mínima (en píxeles) en X y en Y respectivamente que hay entre el widget y otro widget (o de la misma ventana).
- El parámetro *sticky* recibe un punto cardinal que indica a qué esquina de la celda se debe pegar ('n' esquina superior, 's' esquina inferior, 'w' esquina izquierda y 'e' esquina derecha). También recibe posiciones combinadas.

17

17

Administrador de esquemas : Grid (3)



(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

(row, column)

```
import tkinter as tk

ventana = tk.Tk()
ventana.title("Ejemplo grid")
ventana.geometry("400x200")

frame = tk.Frame(ventana, width=200, height=100)
frame.pack(expand=True)

label1 = tk.Label(frame, text="Email")
entry1 = tk.Entry(frame)
label2 = tk.Label(frame, text="Password")
entry2 = tk.Entry(frame)
boton1 = tk.Button(frame, text="Submit")
boton2 = tk.Button(frame, text="Clear")

label1.grid(row=0, column=0, padx=10, pady=10, sticky="w")
entry1.grid(row=0, column=1, columnspan=2, padx=10, pady=10)
label2.grid(row=1, column=0, padx=10, pady=10, sticky="w")
entry2.grid(row=1, column=1, columnspan=2, padx=10, pady=10)
boton1.grid(row=2, column=0, padx=10, pady=10, sticky="w")
boton2.grid(row=2, column=1, padx=10, pady=10, sticky="w")

ventana.mainloop()
```

ejemplo3.py¹⁸

18



Administrador de esquemas : Place (1)

- El método `place()` sirve para posicionar los widgets por medio de posiciones X y Y.
- Se define explícitamente en la clase **Place** y su subclase **Widget** (y el resto de subclases de **Widget**) lo heredan.

19

19



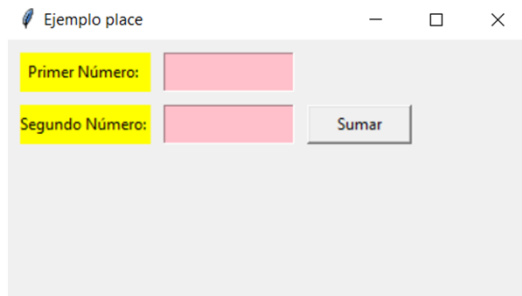
Administrador de esquemas : Place (2)

- La posición que puede tomar puede ser absoluta y quiere decir que su tamaño y posición no varían si se varía el tamaño de su contenedor. La estructura es la siguiente:
`widget.place(x=10,y=10,width=100, height=30,...)`
- Los parámetros de **x** y **y** reciben la posición horizontal y vertical respectivamente (en píxeles). Y **width** y **height** reciben el tamaño del widget (en píxeles).

20

20

Administrador de esquemas : Place (3)



```
import tkinter as tk

vent = tk.Tk()
vent.title("Ejemplo place")
vent.geometry("400x200")

lblNum1 = tk.Label(vent, text="Primer Número: ", bg="yellow")
lblNum1.place(x=10, y=10, width=100, height=30)
txtNum1 = tk.Entry(vent, bg="pink")
txtNum1.place(x=120, y=10, width=100, height=30)

lblNum2 = tk.Label(vent, text="Segundo Número: ", bg="yellow")
lblNum2.place(x=10, y=50, width=100, height=30)
txtNum2 = tk.Entry(vent, bg="pink")
txtNum2.place(x=120, y=50, width=100, height=30)

btn1 = tk.Button(vent, text="Sumar")
btn1.place(x=230, y=50, width=80, height=30)

vent.mainloop()
```

ejemplo4.py 21

21

Un tema en ttk define un conjunto de estilos que controlan la apariencia de los widgets. Esto incluye colores, fuentes, bordes y otros aspectos visuales.

Administrador de esquemas : Place (4)

- También se puede tomar una posición y tamaño relativo, lo cual significa que se toma un porcentaje de ubicación y tamaño de su contenedor, e implica que la posición y tamaño del widget cambia cuando las de su contenedor lo hacen. La estructura es la siguiente:

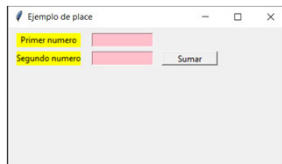
```
widget.place(relx=0.55, rely=0.17, relwidth=0.20, relheight=0.1, ...)
```

- Los parámetros de *relx* y *rely* reciben un porcentaje de ubicación horizontal y vertical respectivamente (entre 0 y 1). Y *relwidth* y *relheight* reciben el porcentaje de tamaño respecto al contenedor (entre 0 y 1).

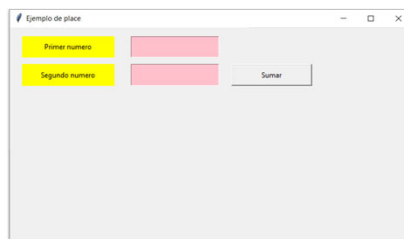
22

22

Administrador de esquemas : Place (5)



↓ Cambio de tamaño



```
import tkinter as tk

vent = tk.Tk()
vent.title("Ejemplo de place")
vent.geometry("400x200")

lbl1 = tk.Label(vent, text="Primer numero", bg="yellow")
lbl1.place(relx=0.03, rely=0.04, relwidth=0.23, relheight=0.1)
txt1 = tk.Entry(vent, bg="pink")
txt1.place(relx=0.3, rely=0.04, relwidth=0.22, relheight=0.1)

lbl2 = tk.Label(vent, text="Segundo numero", bg="yellow")
lbl2.place(relx=0.03, rely=0.17, relwidth=0.23, relheight=0.1)
txt2 = tk.Entry(vent, bg="pink")
txt2.place(relx=0.3, rely=0.17, relwidth=0.22, relheight=0.1)

btn1 = tk.Button(vent, text="Sumar")
btn1.place(relx=0.55, rely=0.17, relwidth=0.20, relheight=0.1)

vent.mainloop()
```

ejemplo5.py

23

23

Frame (1)

- Los **Frame** son widgets especiales que ayudan a contener otros widgets y darles un orden.
- Para declarar un frame se instancia un objeto de la clase **Frame** así:
`miFrame = tk.Frame(master,...)` `miFrame = ttk.Frame(master,...)`
- El parámetro *master* recibe el nombre del contenedor del frame, el cual puede ser la ventana u otro frame.
- Los tres puntos denotan parámetros opcionales tales como: *bg* (color del frame), *height* (la altura en píxeles), *width* (el ancho en píxeles). Más atributos se pueden revisar en: https://www.tutorialspoint.com/python/tk_frame.htm
- El tamaño de un frame se puede definir con los atributos *height* y *width* y a su vez lo define el tamaño del objeto (u objetos) contenidos en el frame.
- Dentro de un **Frame** se puede definir otro **Frame**.

24

24

Los tres puntos denotan parámetros opcionales tales como: *height* (La altura en píxeles), *width* (El ancho en píxeles), *style* (Para aplicar estilos personalizados a través de `ttk.Style`), *padding* (Para agregar espacio alrededor del Frame). Más atributos se pueden revisar en: <https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/ttk-Frame.html>

Frame (2)

```
import tkinter as tk

ventana = tk.Tk()
ventana.title("Ejemplo Frames")
ventana.geometry("300x200")

frame1 = tk.Frame(ventana, bg="red", height=40)
frame1.pack(fill='x')

frame11 = tk.Frame(frame1, bg='blue')
frame11.place(relx=0.5, rely=0.5, relwidth=0.5, relheight=1, anchor="c")

frame2 = tk.Frame(ventana, bg="green", width=40)
frame2.pack(side='right', fill='y')

frame12 = tk.Frame(frame2, bg='orange')
frame12.place(relx=0.5, rely=0.5, relwidth=1, relheight=0.5, anchor="s")

cinco = tk.Button(frame11, text="Cinco")
cinco.place(relx=0.5, rely=0.5, relwidth=0.5, relheight=1, anchor="e")

uno = tk.Button(frame11, text="1")
uno.place(relx=0.6, rely=0.5, relwidth=0.4, relheight=1, anchor="w")

dos = tk.Button(frame12, text="Dos")
dos.pack(anchor="w")

tres = tk.Button(frame12, text="Tres")
tres.pack(anchor="w")

cuatro = tk.Button(frame12, text="Cuatro")
cuatro.pack(anchor="w")

ventana.mainloop()
```

ejemplo6.py

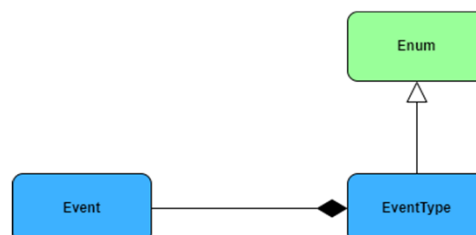
anchor="s" para esquina inferior

25

25

Eventos

- Algunas de las actividades que realiza un usuario al interactuar con una aplicación son:
 - Hacer clic sobre un botón.
 - Mover el mouse sobre algún widget de la ventana
 - Presionar una tecla.
- Tkinter define una clase base para manejar eventos llamada `Event` y un enumerador llamado `EventType`. `Event`, entre otras cosas, contiene un `EventType` y con este se define el tipo de evento que se maneja.



26

26

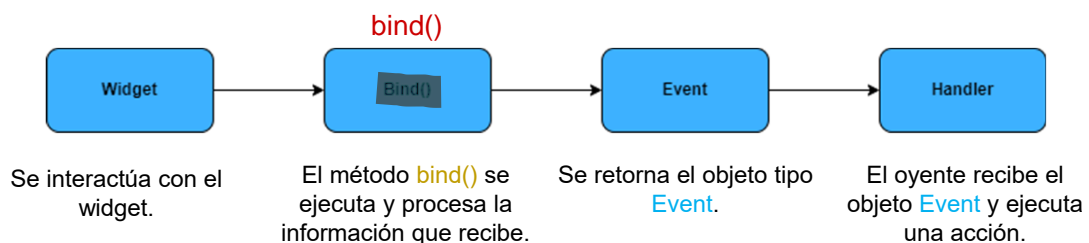
Eventos: Como manejar los eventos (1)

- En la clase Misc se declara un método `bind()`, el cual todas sus subclases heredan, por lo que todas las clases usadas para definir widgets lo implementan.
- La estructura es la siguiente:
`widget.bind(<nombreEvento-valor>, handler)`
- El primer parámetro consta de una etiqueta compuesta por el tipo del evento y un valor entero opcional dependiendo de si es un evento de mouse o no. Internamente, el método `bind()` lee la información de la etiqueta e instancia un objeto de tipo `Event` con esta.
- El segundo parámetro es el nombre de una función previamente definida la cual debe recibir un solo argumento (el cuál será el objeto tipo `Event` que se manda desde el método `bind()`). Esta función servirá como oyente del evento.

27

27

Eventos: Como manejar los eventos (2)



28

28



Eventos: Posibles eventos de Tkinter (1)

Evento	Acción del usuario	Tipo	Detalles
<Button>	Dar click a un widget.	Mouse	En la etiqueta definir -1, -2, -3 para click derecho, mitad o izquierdo respectivamente. Ejm: <Button-1>
<Motion>	Mover el mouse mientras da click a un widget.	Mouse	En la etiqueta definir inicialmente B1, B2 y B3 para click derecho, mitad o izquierdo. Ejm: <B1-Motion>
<ButtonRelease>	Se deja de dar click a un widget.	Mouse	En la etiqueta definir -1, -2, -3 para click derecho, mitad o izquierdo respectivamente. Ejm: <ButtonRelease-1>
<Enter>	El puntero del mouse entra en el widget (no click)	Mouse	
<Leaves>	El puntero del mouse sale del widget.	Mouse	

29

29



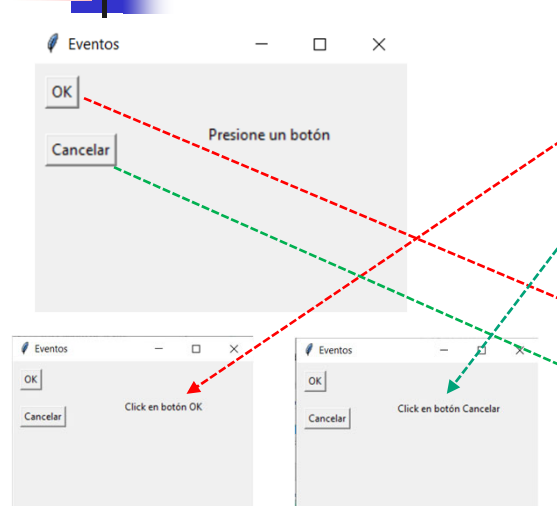
Eventos: Posibles eventos de Tkinter (2)

Evento	Acción del usuario	Tipo	Detalles
<Return>	Presionar la tecla Enter.	Teclado	
<Key>	Presionar cualquier tecla.	Teclado	

30

30

Eventos: Ejemplo



```

import tkinter as tk
ventana = tk.Tk()
ventana.title("Eventos")
ventana.geometry("300x200")

def Ok(evento):
    etiqueta.config(text="Click en botón OK")

def Cancelar(evento):
    etiqueta.config(text="Click en botón Cancelar")

button1 = tk.Button(ventana, text="OK")
button1.pack(side='top', anchor="w", padx=10, pady=10)
button1.bind("<Button-1>", Ok)
button2 = tk.Button(ventana, text="Cancelar")
button2.pack(side='left', anchor="n", padx=10, pady=10)
button2.bind("<Button-1>", Cancelar)
etiqueta = tk.Label(ventana, text="Presione un botón")
etiqueta.pack()

ventana.mainloop()

```

ejemplo7.py

31

Lecturas (1)

- Tutorial útil:
<https://tkdocs.com/tutorial/index.html>
- Listado de eventos:
<https://python-course.eu/tkinter/events-and-binds-in-tkinter.php>
- Código fuente Tkinter:
https://github.com/python/cpython/blob/2fe016fbb7c3b8ec9c759221175971a3f235a68/Lib/tkinter/_init_.py#L151

32

32



Lecturas (2)

- Chapter 1. Introduction to Tkinter
- Chapter 3. Creating Basic Forms with Tkinter and Ttk Widgets
 - Libro: Moore, Alan D. Python GUI Programming with Tkinter. 2da Edición. Edición de Kindle.

