

python

Aplicaciones de Escritorio con

Tkinter



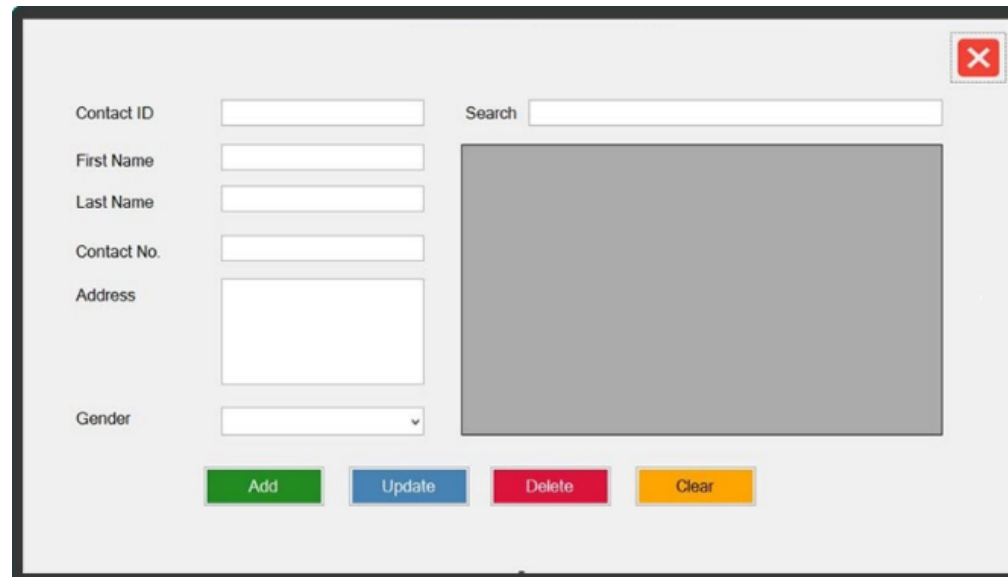
01

Introducción y primer ventana



Desarrollo de aplicaciones gráficas

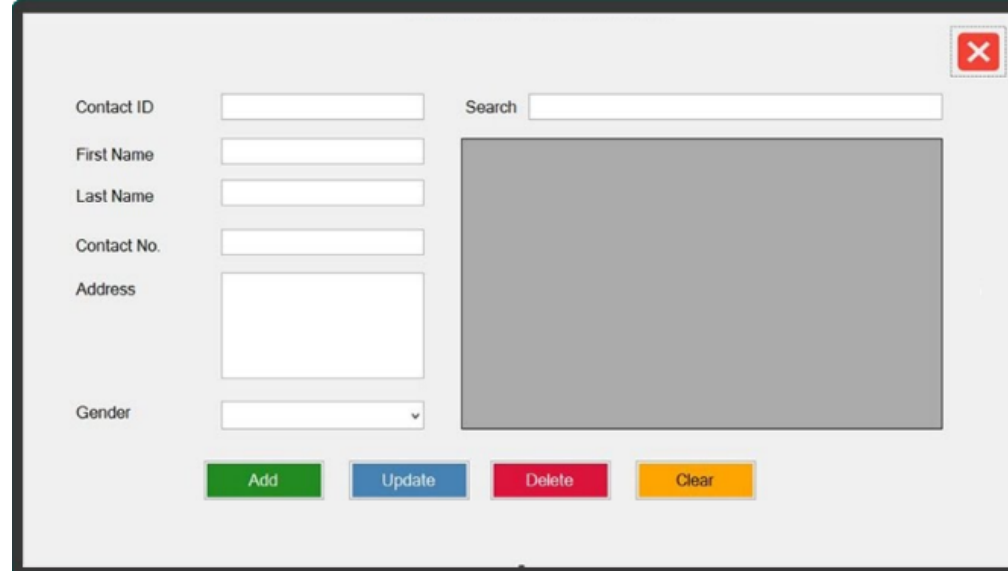
- Desarrollar aplicaciones de escritorio empleando ventanas y controles tales como cajas de texto, etiquetas, botones, listas desplegables, checkbox, menús, etc. en Python



The image shows a Tkinter window titled 'Contact Management' (implied by the context). It features a light gray background and a standard window title bar with a red close button. The window is divided into two main sections. On the left, there are input fields for 'Contact ID', 'First Name', 'Last Name', 'Contact No.', 'Address', and 'Gender' (a dropdown menu). Below these fields are four buttons: 'Add' (green), 'Update' (blue), 'Delete' (red), and 'Clear' (yellow). On the right, there is a 'Search' input field above a large gray rectangular area, likely intended for displaying search results or a list of contacts.

Tkinter

- tkinter es un módulo Python que nos facilita el desarrollo de las GUI, está disponible en la instalación por defecto, soporta diversas plataformas como: Windows, Linux, Mac, y además es muy fácil de usar.



A screenshot of a Tkinter GUI application designed for managing contacts. The window has a light gray background and a standard window control bar with a red close button in the top right corner. On the left side, there are labels for 'Contact ID', 'First Name', 'Last Name', 'Contact No.', 'Address', and 'Gender'. Each label is followed by a corresponding input field: a single-line text box for Contact ID, First Name, Last Name, and Contact No.; a multi-line text area for Address; and a dropdown menu for Gender. To the right of these input fields is a 'Search' label followed by a single-line text box. Below the input fields, there are four buttons: 'Add' (green), 'Update' (blue), 'Delete' (red), and 'Clear' (yellow). A large gray rectangular area is positioned to the right of the input fields, likely intended for displaying a list of contacts.

Primer programa

```
from tkinter import Tk, Label, Button

def mensaje():
    print("Mensaje del boton ")

ventana = Tk()
ventana.geometry("400x280")
ventana.title("VentanaHola mundo")

lbl = Label(ventana, text='Este es un [Label] tkinter')
lbl.pack()

btn = Button(ventana, text='Presiona este [Button] para mensaje', command=mensaje)
btn.pack()

ventana.mainloop()
```



Clase Tk y algunos métodos

- **Tk()** clase para crear una ventana principal donde añadir los widgets
- **pack()** ubica los widgets en una posición que podemos cambiar a través de los correspondientes atributos.
- **mainloop()** inicia el bucle de mensajes, aquí se **monitorea la interacción del usuario a través del ratón o teclado con la aplicación**, cuando se produzca un evento recibiremos la notificación correspondiente y podremos responder a dicho evento, por ejemplo: el usuario hace clic sobre el botón cerrar, respondemos cerrando la aplicación.



Configurar widgets tkinter

- 3 formas:
- Constructor:
 - `miBoton = Button(self, fg="red", bg="blue")`
- Método config:
 - `miBoton.config(fg="red", bg="blue")`
- Accesando a la propiedad (como un diccionario clave/valor)
 - `miBoton["fg"] = "red"`
 - `miBoton["bg"] = "blue"`



Utilizar la extensión .pyw

- Normalmente los script Python son almacenados con la **extensión .py**, para ejecutar este script, en Windows por ejemplo necesitamos abrir una consola de comandos y ejecutar lo siguiente: **python hola_mundo.py**, si deseamos ejecutar una aplicación con GUI por lo general no se requiere enviar argumentos desde la consola, por lo que, **si cambiamos la extensión a .pyw la aplicación se ejecutara automáticamente al hacer doble clic sobre ella.**



python

Aplicaciones de Escritorio con

Tkinter



01

Introducción y primer ventana

