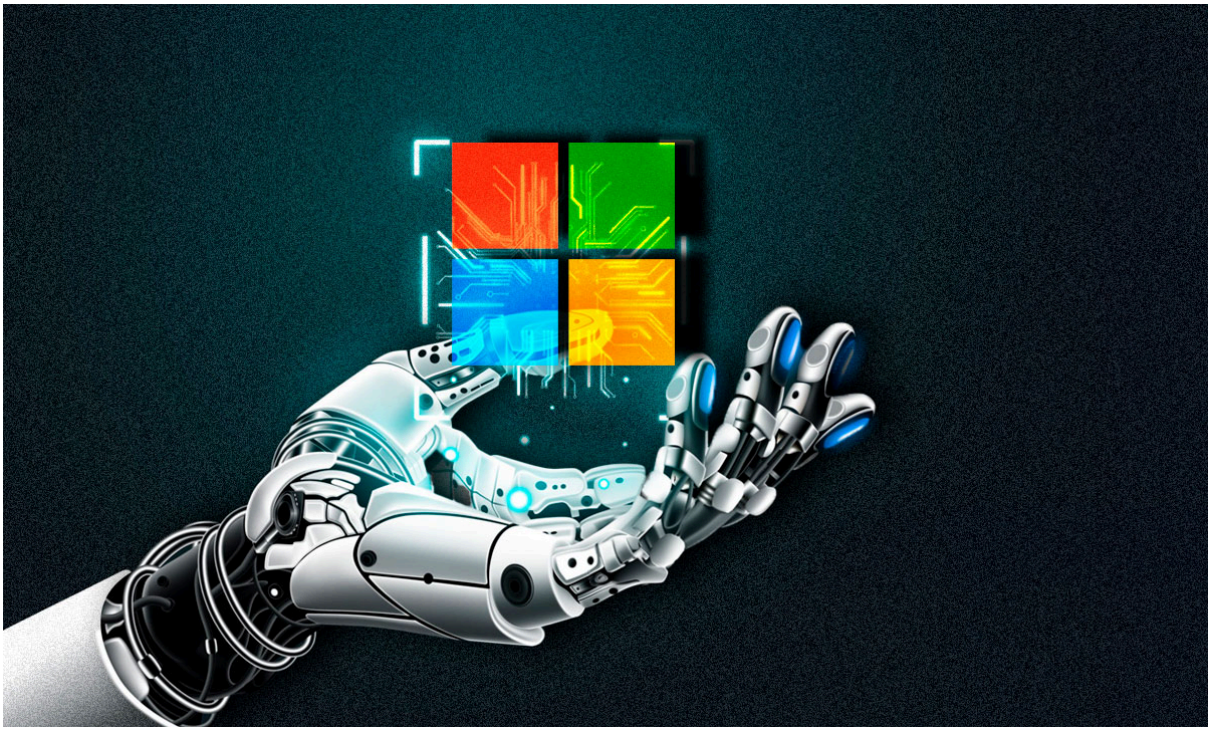
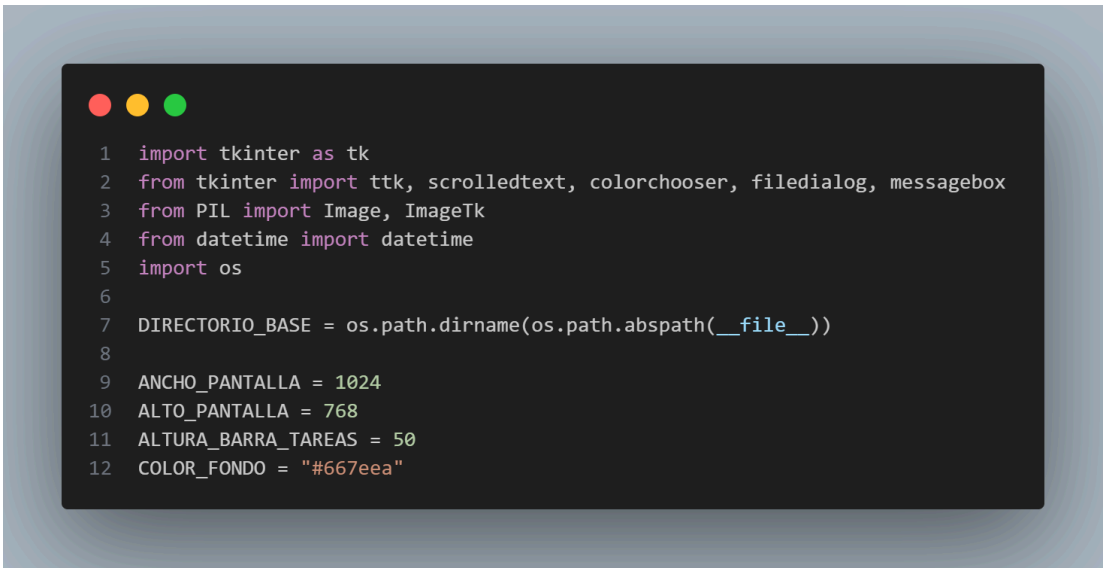
Windows Mediante IAs

Por: Julio Gacia



1. Imports necesarios y configuración del contenedor del programa

En este apartado se muestran las librerías utilizadas para el proyecto (Tkinter, PIL, datetime, OS, etc.) así como las constantes globales y la configuración base de la ventana principal que actúa como contenedor del escritorio



```python
import tkinter as tk
from tkinter import ttk, scrolledtext, colorchooser, filedialog, messagebox
from PIL import Image, ImageTk
from datetime import datetime
import os

DIRECTORIO_BASE = os.path.dirname(os.path.abspath(__file__))

ANCHO_PANTALLA = 1024
ALTO_PANTALLA = 768
ALTURA_BARRA_TAREAS = 50
COLOR_FONDO = "#667eea"
```

2. Lógica principal de la calculadora

En esta sección se describe la clase encargada de gestionar la calculadora. Incluye la creación dinámica de botones, la variable de pantalla y la función que procesa cada pulsación para realizar las operaciones básicas (+, -, *, /, %, etc.).

```python
def clic(self, tecla):
        actual = self.variable_pantalla.get()
        if tecla == "C":
            self.variable_pantalla.set("0")
        elif tecla == "←":
            self.variable_pantalla.set(actual[:-1] or "0")
        elif tecla == "=":
            try:
                resultado = eval(actual)
                self.variable_pantalla.set(str(resultado))
            except:
                self.variable_pantalla.set("Error")
        else:
            if actual == "0" or actual == "Error":
                self.variable_pantalla.set(tecla)
            else:
                self.variable_pantalla.set(actual + tecla)
```

3. Bloc de notas funcional

Este apartado documenta el funcionamiento del Bloc de Notas, que permite crear nuevos documentos, abrir archivos de texto existentes y guardar los cambios realizados. Incluye el uso de cuadros de diálogo para seleccionar archivos.

```python
def nuevo_archivo(self):
    self.texto.delete("1.0", tk.END)
    self.archivo_actual = None
    self.ventana.title("Bloc de Notas - Nuevo")

def abrir_archivo(self):
    archivo = filedialog.askopenfilename(filetypes=[("Text files", "*.txt")])
    if archivo:
        with open(archivo, "r", encoding="utf-8") as f:
            self.texto.delete("1.0", tk.END)
            self.texto.insert("1.0", f.read())
        self.archivo_actual = archivo
        self.ventana.title(f"Bloc de Notas - {os.path.basename(archivo)}")

def guardar_archivo(self):
    if not self.archivo_actual:
        self.archivo_actual = filedialog.asksaveasfilename(
            defaultextension=".txt", filetypes=[("Text files", "*.txt")]
        )
```

4. Funciones para cambiar el fondo de pantalla

En esta sección se explican las funciones que permiten personalizar el fondo del escritorio. Incluye la selección de colores sólidos, la elección de imágenes de fondo (wallpapers) desde una carpeta o desde el sistema de archivos, y la función que aplica el fondo ya sea como color o imagen redimensionada a toda la pantalla

```python
def elegir_color(self):
    color = colorchooser.askcolor(title="Elegir color")[1]
    if color:
        self.escritorio.establecer_fondo(color)

def cargar_imagen(self):
    archivo = filedialog.askopenfilename(
        filetypes=[("Images", "*.png *.jpg *.jpeg *.bmp")]
    )
    if archivo:
        self.escritorio.establecer_fondo(archivo)
```

5. Actualización del reloj y apertura del menú

Este apartado describe el reloj que se muestra tanto en la barra de tareas como en la ventana de reloj digital. Se usa el método after() para actualizar la hora cada segundo. También se documenta la función que abre el menú de inicio, desde el cual se pueden lanzar las distintas aplicaciones (calculadora, bloc de notas, configuración, reloj)

```
        # Reloj del sistema
        self.etiqueta_reloj = tk.Label(
            self.barra_tareas, text="", bg="#1f1f1f", fg="white", font=("Arial", 10)
        )
        self.etiqueta_reloj.pack(side=tk.RIGHT, padx=15)

    def actualizar_reloj_sistema(self):
#  Actualiza reloj del sistema en barra de tareas
        ahora = datetime.now()
        self.etiqueta_reloj.config(text=ahora.strftime("%H:%M:%S\n%d/%m/%Y"))
        self.raiz.after(1000, self.actualizar_reloj_sistema)

    def mostrar_menu(self):
#  Boton del task bar para mostrar el menu
        menu = tk.Menu(self.raiz, tearoff=0)
        menu.add_command(label="🖩 Calculadora", command=lambda: Calculadora(self.raiz))
        menu.add_command(label="📝 Bloc de Notas", command=lambda: BlocNotas(self.raiz))
        menu.add_command(
            label="⚙ Configuración", command=lambda: Configuracion(self.raiz, self)
        )
```

6. Índice de funcionalidades y autor

Código Propio

- Imports y constantes globales
- Configuración básica de ventanas
- Variables de instancia sencillas
- Botones y labels simples
- Lista de colores predefinidos

Código IA Editado

- Calculadora: botones dinámicos + lógica de operaciones
- Barra de tareas y estructura general
- Iconos del escritorio y posicionamiento
- Reloj del sistema y actualización

Código IA (sin editar)

- Bloc de Notas completo
- Configuración: carga de wallpapers, miniaturas, tooltips
- establecer_fondo(): gestión de imágenes, capas y colores
- Reloj digital independiente

7. Introducción y objetivos del proyecto

Este proyecto simula un escritorio tipo Windows utilizando Python y Tkinter. Se incluyen aplicaciones básicas como calculadora, bloc de notas, configuración de fondo de pantalla y un reloj digital. Los objetivos principales son: aprendizaje de integración de múltiples ventanas y widgets, experimentar con la generación de código mediante IA, y gestionar código modular y limpio.

8. Flujo de trabajo con IA

IAs utilizadas: OpenAI GPT-4 / Claude Sonet 4.5
Prompts clave: "Genera un sistema tipo escritorio con Tkinter: calculadora, bloc de notas, fondo personalizable, reloj, organizado en clases, código modular y limpio."

Evaluación crítica de las respuestas: Se adaptó el código generado para mejorar la estética, gestión de capas, uso de scrollbars y tooltips. Algunas partes fueron utilizadas tal cual y otras fueron optimizadas.

9. Proceso de desarrollo

Diseño del escritorio y las ventanas: Se creó un contenedor principal con Frame, barra de tareas, iconos posicionados con place y enumeración, colores predefinidos

Implementación de cada aplicación: Calculadora con botones dinámicos y evaluación de operaciones, Bloc de notas funcional, Configuración de fondos con scroll y miniaturas, Reloj digital actualizado mediante after().

Problemas encontrados y soluciones: Capas de iconos y fondos , resize de imágenes para fondos, overflow de widgets solucionado con canvas y scrollbar.

10. Conclusiones técnicas y aprendizajes

El proyecto permitio experimentar un desarrollo con Tkinter y la integracion de distintas funcionalidades como aplicacion de calculadora, bloc de notas, reloj o ajustes de fondo de pantalla. Modularidad del codigo, gestionando eventos y aplicaciones en distintas capas (clases) junto a implementacion de UX en python.