iacc instituto profesional

EVALUACIÓN

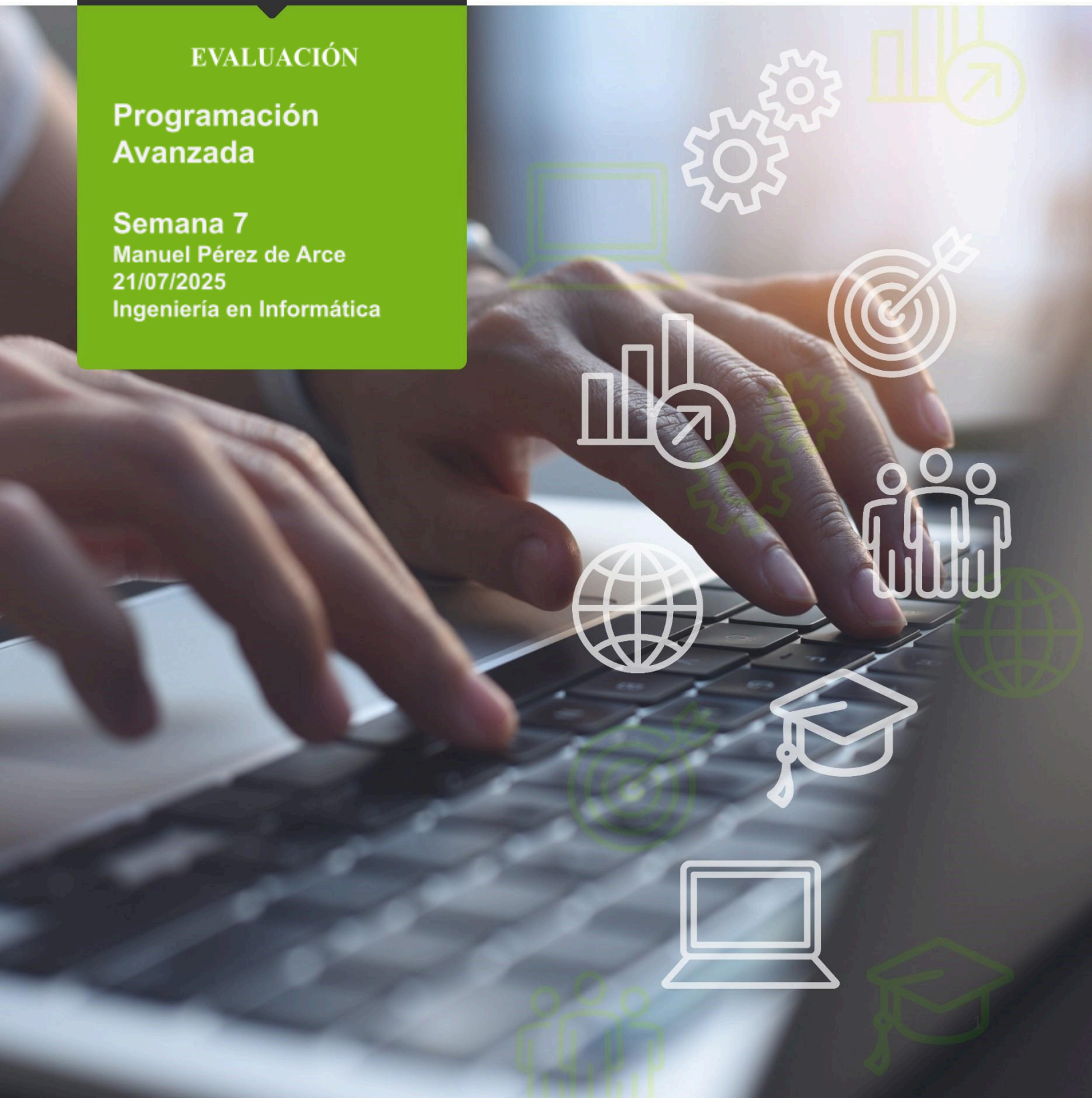
Programación Avanzada

Semana 7

Manuel Pérez de Arce

21/07/2025

Ingeniería en Informática



El código fuente completo del programa puede consultarse en el siguiente enlace, donde se encuentra disponible para su revisión y descarga: https://github.com/

1. ¿Cuáles son los principales elementos que deben examinarse al desarrollar e implementar una interfaz gráfica de usuario en el lenguaje de programación Python, tomando en consideración el caso de la biblioteca SaberX?

Al diseñar una interfaz gráfica de usuario (GUI) en Python para una aplicación como la biblioteca SaberX, es fundamental evaluar diversos elementos para asegurar que la experiencia del usuario sea funcional, intuitiva y eficiente.

En primer lugar, debe analizarse la estructura y organización visual. Esto implica planificar cómo estarán distribuidas las secciones, por ejemplo: un área para registrar libros, otra para visualizar los detalles de los autores y una para acciones como guardar o limpiar la información. Para lograrlo, es habitual usar contenedores como Frame o LabelFrame, que permiten agrupar elementos de forma lógica y ordenada.

Otro aspecto clave es la usabilidad y accesibilidad. Los widgets deben estar bien identificados, usando Label descriptivos que acompañen a los campos Entry o Text. Además, se debe procurar que las opciones sean fáciles de seleccionar mediante botones (Button), radiobuttons o checkbuttons, y que los mensajes de confirmación sean claros.

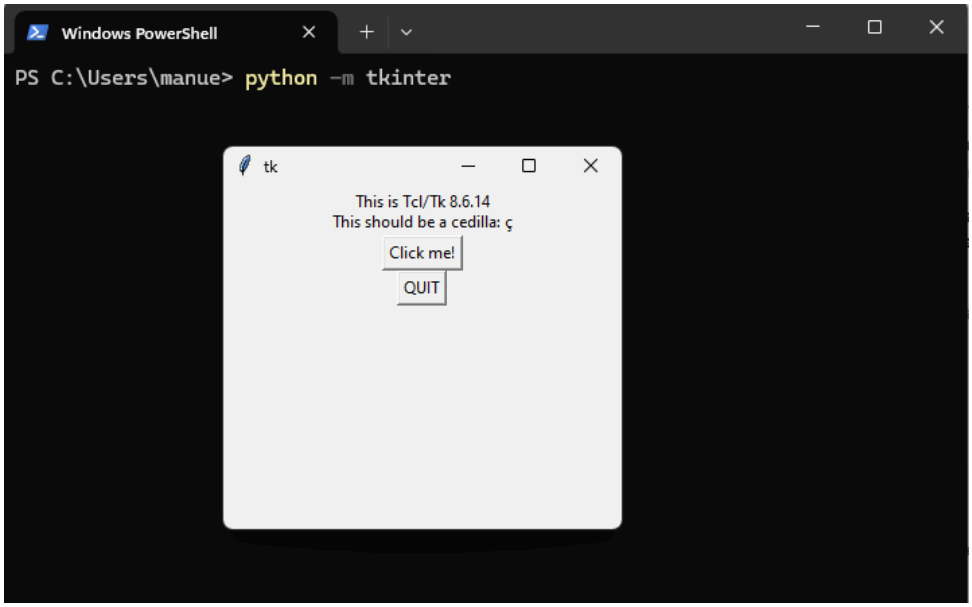
Finalmente, es importante considerar la persistencia y validación de datos, ya que, en el contexto de una biblioteca, los datos ingresados deben ser almacenados correctamente (por ejemplo, en archivos JSON o bases de datos) y validados para evitar errores como registros incompletos o duplicados.

Estos elementos son esenciales para que la GUI cumpla con su propósito: facilitar la gestión de libros y autores de forma sencilla para cualquier usuario.

2. ¿Cuál es el procedimiento de instalación de la librería Tkinter en un entorno de desarrollo Python para poder ejecutar programas que hagan uso de interfaces gráficas?

Tkinter es la librería estándar de Python para crear interfaces gráficas. En la mayoría de distribuciones de Python viene instalada por defecto, pero si no está disponible, el procedimiento varía según el sistema operativo:

Verificar instalación:
Ejecutar en terminal o consola: python -m tkinter. Si se abre una ventana de prueba, Tkinter está correctamente instalado.



Una vez disponible, se puede usar simplemente con:

```
import tkinter as tk
from tkinter import messagebox
```

Esto permite crear interfaces gráficas sin necesidad de instalar dependencias externas adicionales.

3. En el proyecto de la biblioteca, ¿cómo se pueden emplear los elementos de configuración de la librería Tkinter en Python para elaborar interfaces que faciliten la interacción del usuario, específicamente utilizando widgets como Frames, Labels, Entry, Text, Buttons, Radiobuttons, Checkbuttons y Menús? Además, agrega como evidencia el código fuente y la imagen de la interfaz generada con los datos de prueba que utilizaste del programa en Python, que cumpla con los requisitos mencionados en el problema.

En el proyecto de la biblioteca SaberX, la librería Tkinter permite construir una interfaz intuitiva y segmentada en secciones, donde cada elemento gráfico cumple un rol específico para facilitar la interacción con el usuario.

Por ejemplo, se utilizan Frames para dividir la ventana en áreas lógicas como “Detalles del libro”, “Estado de disponibilidad” o “Opciones de idioma”. Dentro de cada frame, los Labels sirven para identificar los campos, mientras que los Entry y Text permiten al usuario introducir datos como título, autor o resumen del libro.

Además, se incluyen Radiobuttons para seleccionar opciones excluyentes (por ejemplo, género: Ficción o No Ficción), y Checkbuttons para categorías múltiples (como Novela, Ciencia, Historia). Para determinar el idioma, se emplea un menú desplegable (OptionMenu), y finalmente, Buttons que permiten ejecutar acciones clave como “Registrar Libro” o “Limpiar Formulario”.

La interfaz facilita la interacción al guiar al usuario de forma visual y estructurada, reduciendo errores y mejorando la experiencia de uso.

index.py

```
import tkinter as tk

# Importar componentes
from components.header import create_header
from components.formulario import FormularioLibros


def centrar_ventana(ventana, ancho, alto):
    ventana.update_idletasks()
    screen_width = ventana.winfo_screenwidth()
    screen_height = ventana.winfo_screenheight()

    x = (screen_width // 2) - (ancho // 2)
    y = (screen_height // 2) - (alto // 2)

    ventana.geometry(f"{ancho}x{alto}+{x}+{y}")

# Crear ventana principal
ventana = tk.Tk()
ventana.title("Mi ventana Principal")
# Llamamos la función con las dimensiones deseadas
centrar_ventana(ventana, ancho=800, alto=800)

header=create_header(ventana)
formulario = FormularioLibros(ventana)
ventana.mainloop()
```

formulario.py

```
import tkinter as tk
from tkinter import messagebox

class FormularioLibros:
    def __init__(self, parent):
```

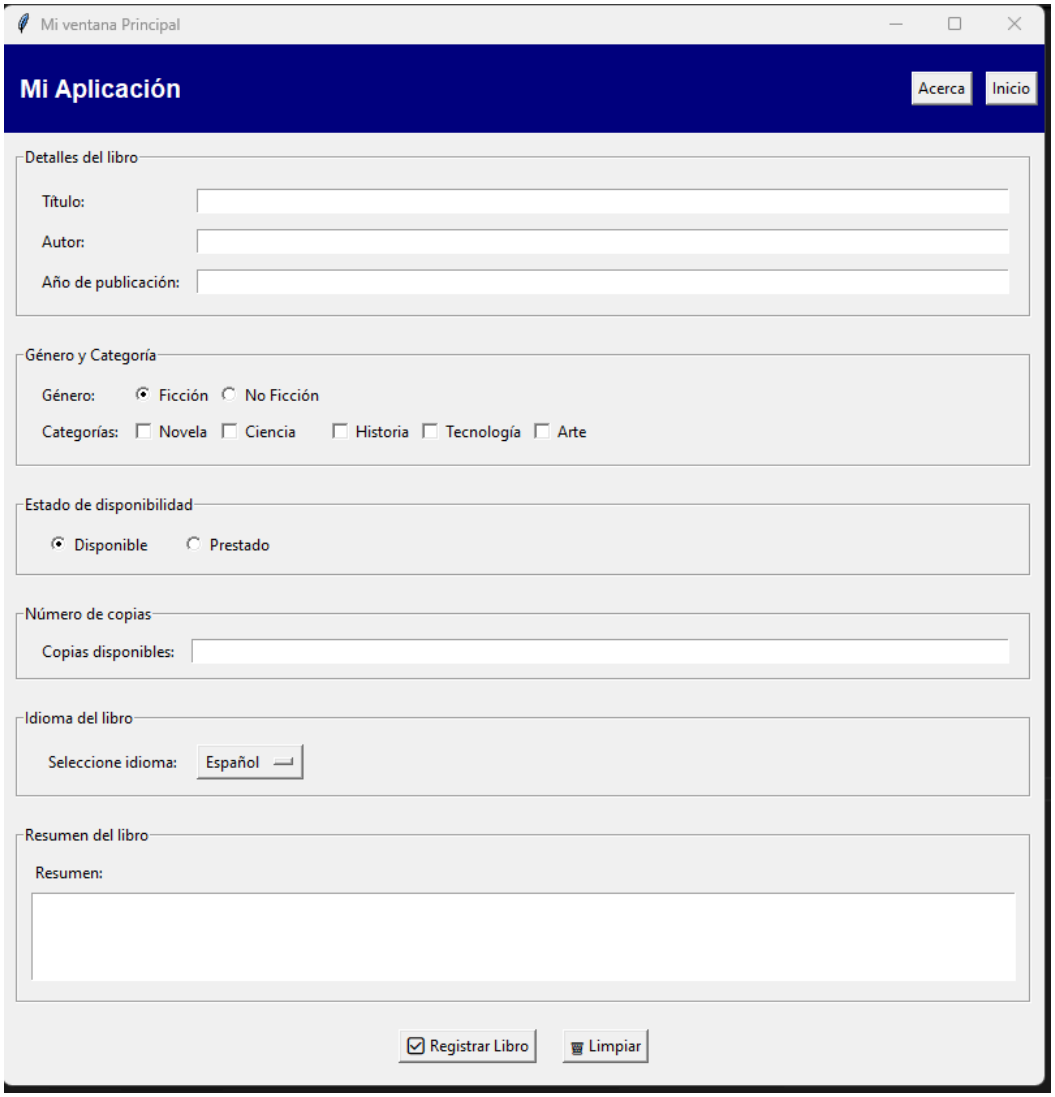
```
        self.parent = parent

        # ===== Frame Detalles del libro =====
        self.frame_detalles = tk.LabelFrame(self.parent, text="Detalles del
libro", padx=10, pady=10)
        self.frame_detalles.pack(fill="x", padx=10, pady=10)

        tk.Label(self.frame_detalles, text="Título:").grid(row=0, column=0,
sticky="w", padx=5, pady=5)
        self.entry_titulo = tk.Entry(self.frame_detalles)
        self.entry_titulo.grid(row=0, column=1, sticky="ew", padx=5,
pady=5)

        tk.Label(self.frame_detalles, text="Autor:").grid(row=1, column=0,
sticky="w", padx=5, pady=5)
        self.entry_autor = tk.Entry(self.frame_detalles)
        self.entry_autor.grid(row=1, column=1, sticky="ew", padx=5, pady=5)

        tk.Label(self.frame_detalles, text="Año de
publicación:").grid(row=2, column=0, sticky="w", padx=5, pady=5)
        self.entry_anio = tk.Entry(self.frame_detalles)
        self.entry_anio.grid(row=2, column=1, sticky="ew", padx=5, pady=5)

        self.frame_detalles.columnconfigure(1, weight=1)

        # ===== Frame Género y Categoría =====
        self.frame_genero_cat = tk.LabelFrame(self.parent, text="Género y
Categoría", padx=10, pady=10)
        self.frame_genero_cat.pack(fill="x", padx=10, pady=10)

        self.genero_var = tk.StringVar(value="Ficción")
        tk.Label(self.frame_genero_cat, text="Género:").grid(row=0,
column=0, sticky="w", padx=5)
        tk.Radiobutton(self.frame_genero_cat, text="Ficción",
variable=self.genero_var, value="Ficción").grid(row=0, column=1,
sticky="w")
        tk.Radiobutton(self.frame_genero_cat, text="No Ficción",
variable=self.genero_var, value="No Ficción").grid(row=0, column=2,
sticky="w")

        tk.Label(self.frame_genero_cat, text="Categorías:").grid(row=1,
```

```
column=0, sticky="nw", padx=5, pady=5)
        categorias = ["Novela", "Ciencia", "Historia", "Tecnología",
"Arte"]
        self.categorias_vars = {}
        for i, cat in enumerate(categorias):
            var = tk.IntVar()
            self.categorias_vars[cat] = var
            tk.Checkbutton(self.frame_genero_cat, text=cat,
variable=var).grid(row=1, column=i+1, sticky="w")

        # ===== Frame Estado de disponibilidad =====
        self.frame_estado = tk.LabelFrame(self.parent, text="Estado de
disponibilidad", padx=10, pady=10)
        self.frame_estado.pack(fill="x", padx=10, pady=10)

        self.estado_var = tk.StringVar(value="Disponible")
        tk.Radiobutton(self.frame_estado, text="Disponible",
variable=self.estado_var, value="Disponible").pack(side="left", padx=10)
        tk.Radiobutton(self.frame_estado, text="Prestado",
variable=self.estado_var, value="Prestado").pack(side="left", padx=10)

        # ===== Frame Número de copias =====
        self.frame_copias = tk.LabelFrame(self.parent, text="Número de
copias", padx=10, pady=10)
        self.frame_copias.pack(fill="x", padx=10, pady=10)

        tk.Label(self.frame_copias, text="Copias disponibles:").grid(row=0,
column=0, sticky="w", padx=5)
        self.entry_copias = tk.Entry(self.frame_copias)
        self.entry_copias.grid(row=0, column=1, sticky="ew", padx=5)
        self.frame_copias.columnconfigure(1, weight=1)

        # ===== Frame Idioma =====
        self.frame_idioma = tk.LabelFrame(self.parent, text="Idioma del
libro", padx=10, pady=10)
        self.frame_idioma.pack(fill="x", padx=10, pady=10)

        self.idioma_var = tk.StringVar(value="Español")
        idiomas_disponibles = ["Español", "Inglés", "Francés", "Alemán"]
        tk.Label(self.frame_idioma, text="Seleccione
idioma:").pack(side="left", padx=10)
```

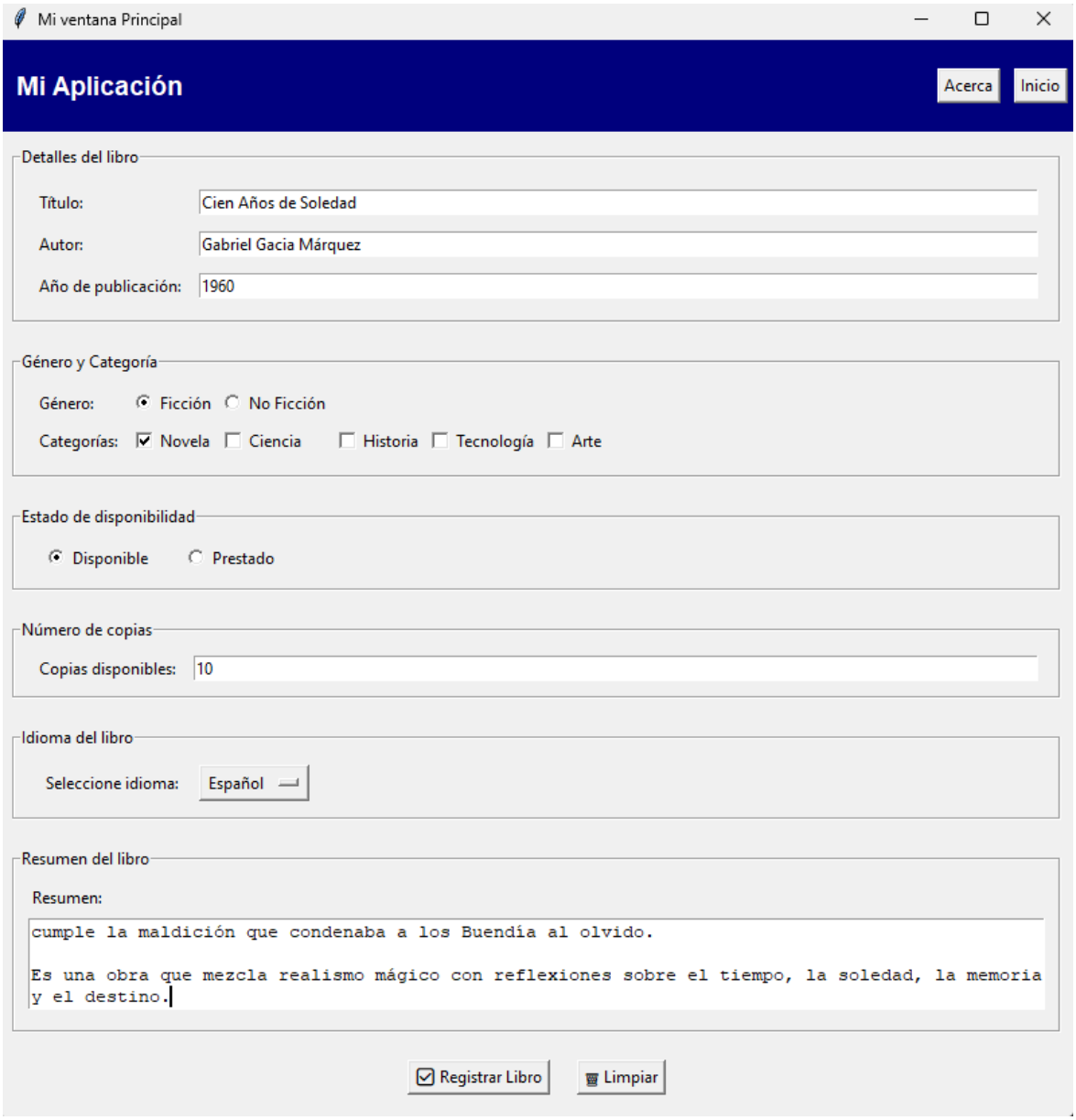
```
        tk.OptionMenu(self.frame_idioma, self.idioma_var,
*idiomas_disponibles).pack(side="left")

        # ===== Frame Resumen del libro =====
        self.frame_resumen = tk.LabelFrame(self.parent, text="Resumen del
libro", padx=10, pady=10)
        self.frame_resumen.pack(fill="both", expand=False, padx=10,
pady=10)

        tk.Label(self.frame_resumen, text="Resumen:").pack(anchor="w")
        self.text_resumen = tk.Text(self.frame_resumen, height=4,
wrap="word")
        self.text_resumen.pack(fill="both", expand=False, pady=5)

        # ===== Frame Botones de acción =====
        self.frame_botones = tk.Frame(self.parent, pady=10)
        self.frame_botones.pack()

        btn_registrar = tk.Button(self.frame_botones, text="✅ Registrar
Libro", command=self.registrar_libro)
        btn_registrar.grid(row=0, column=0, padx=10)

        btn_limpiar = tk.Button(self.frame_botones, text="🗑️ Limpiar",
command=self.limpiar_formulario)
        btn_limpiar.grid(row=0, column=1, padx=10)

    def registrar_libro(self):
        """Función para recolectar todos los datos y mostrarlos en
consola"""
        titulo = self.entry_titulo.get()
        autor = self.entry_autor.get()
        anio = self.entry_anio.get()
        genero = self.genero_var.get()
        estado = self.estado_var.get()
        copias = self.entry_copias.get()
        idioma = self.idioma_var.get()
        resumen = self.text_resumen.get("1.0", tk.END).strip()

        categorias_seleccionadas = [cat for cat, var in
self.categorias_vars.items() if var.get() == 1]
```

```
        print("\n=== Detalles del libro registrado ===")
        print(f"Título: {titulo}")
        print(f"Autor: {autor}")
        print(f"Año de publicación: {anio}")
        print(f"Género: {genero}")
        print(f"Categorías: {', '.join(categorias_seleccionadas)}")
        print(f"Estado: {estado}")
        print(f"Número de copias: {copias}")
        print(f"Idioma: {idioma}")
        print(f"Resumen:\n{resumen}")
        print("====================================\n")

        messagebox.showinfo("Registro", "📚 Libro registrado
correctamente")

    def limpiar_formulario(self):
        """Resetea todos los campos del formulario"""
        self.entry_titulo.delete(0, tk.END)
        self.entry_autor.delete(0, tk.END)
        self.entry_anio.delete(0, tk.END)
        self.genero_var.set("Ficción")
        self.estado_var.set("Disponible")
        self.entry_copias.delete(0, tk.END)
        self.idioma_var.set("Español")
        self.text_resumen.delete("1.0", tk.END)
        for var in self.categorias_vars.values():
            var.set(0)
```

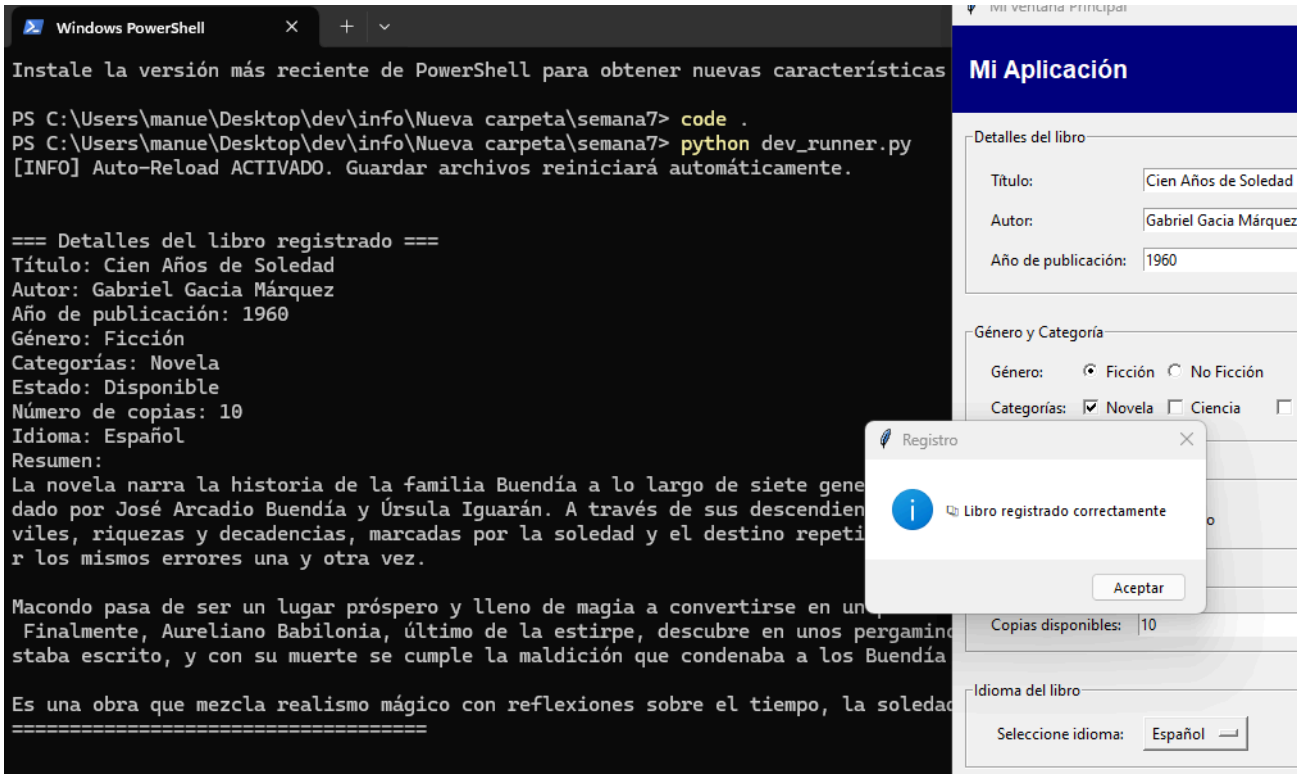
Formulario render



Ingresar registro


Mi ventana Principal
Mi Aplicación
Acerca
Inicio
Detalles del libro
Título:
Cien Años de Soledad
Autor:
Gabriel Gacia Márquez
Año de publicación:
1960
Género y Categoría
Género:
Ficción
No Ficción
Categorías:
Novela
Ciencia
Historia
Tecnología
Arte
Estado de disponibilidad
Disponible
Prestado
Número de copias
Copias disponibles:
10
Idioma del libro
Seleccione idioma:
Español
Resumen del libro
Resumen:
cumple la maldición que condenaba a los Buendía al olvido.
Es una obra que mezcla realismo mágico con reflexiones sobre el tiempo, la soledad, la memoria y el destino.
Registrar Libro
Limpiar

Envío de formulario



REFERENCIAS BIBLIOGRÁFICAS

- Algar Díaz, M. J. y Fernández de Sevilla Vellón, M. (2019). Introducción práctica a la programación con Python. Editorial Universidad de Alcalá.
- Cuevas Álvarez, A. (2018). Aplicaciones gráficas con Python 3. RA-MA Editorial.
- Moreno Muñoz, A. y Córcoles Córcoles, S. (2019). Python práctico: Herramientas, conceptos y técnicas. Paracuellos de Jarama, Madrid, RA-MA Editorial. Recuperado
- Muñoz Guerrero, L. E. (Il.) y Trejos Buriticá, O. I. (2021). Introducción a la programación con Python. RA-MA Editorial.