Actividad 4 grupal POO

Estudiantes

David Esteban Cartagena Mejia
C.C: 1001361568
Josué González Otálvaro
C.C: 1001236911
Juan Manuel Saldarriaga V.
C.C: 1001226798

Docente

Walter Hugo Arboleda Mazo

Asignatura

Programación Orientada a Objetos (POO)

Universidad Nacional de Colombia Sede Medellín
Mayo de 2025

Tabla de contenidos

Listado de Figuras

Punto 1

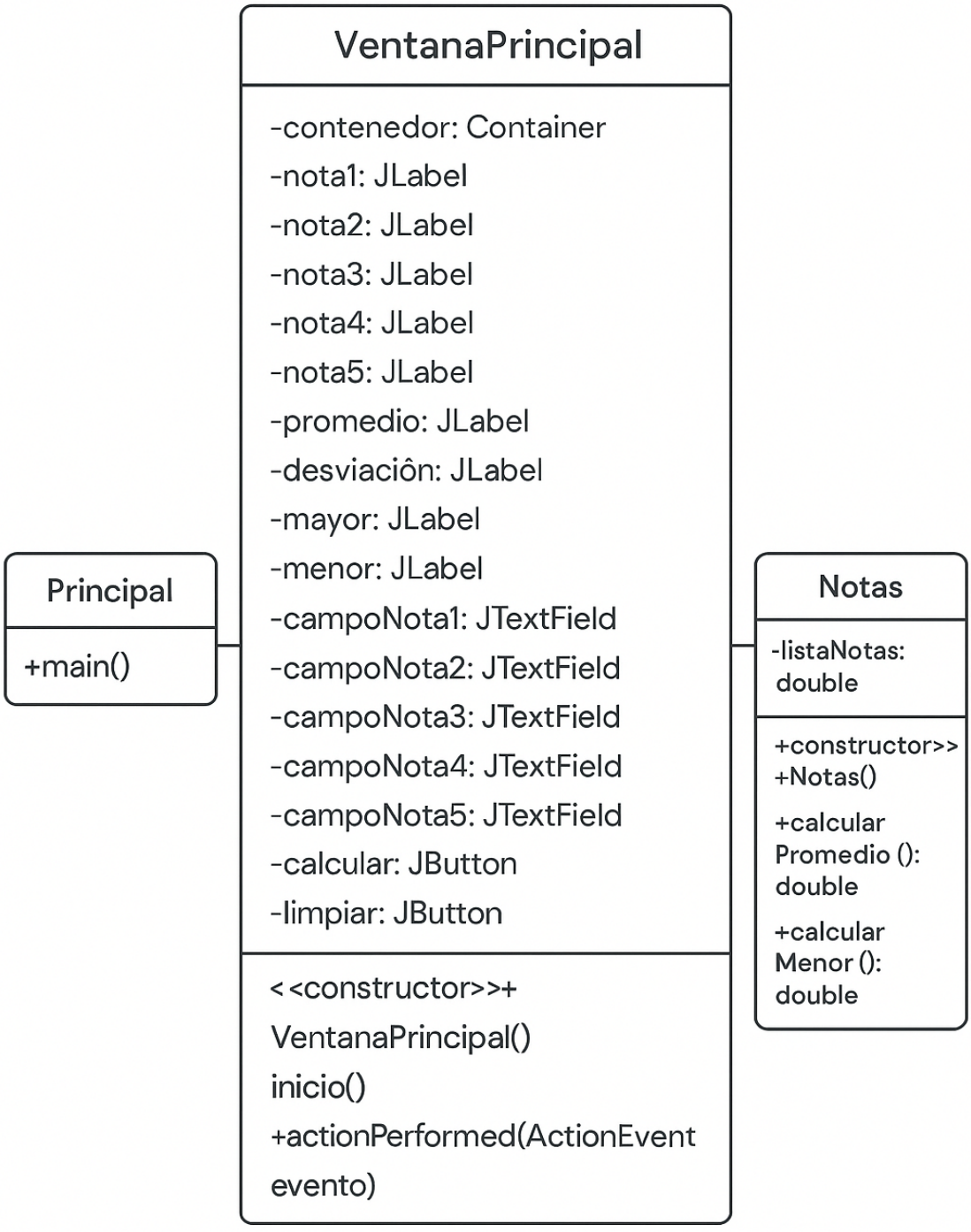

Figura 1: Diagrama de clases

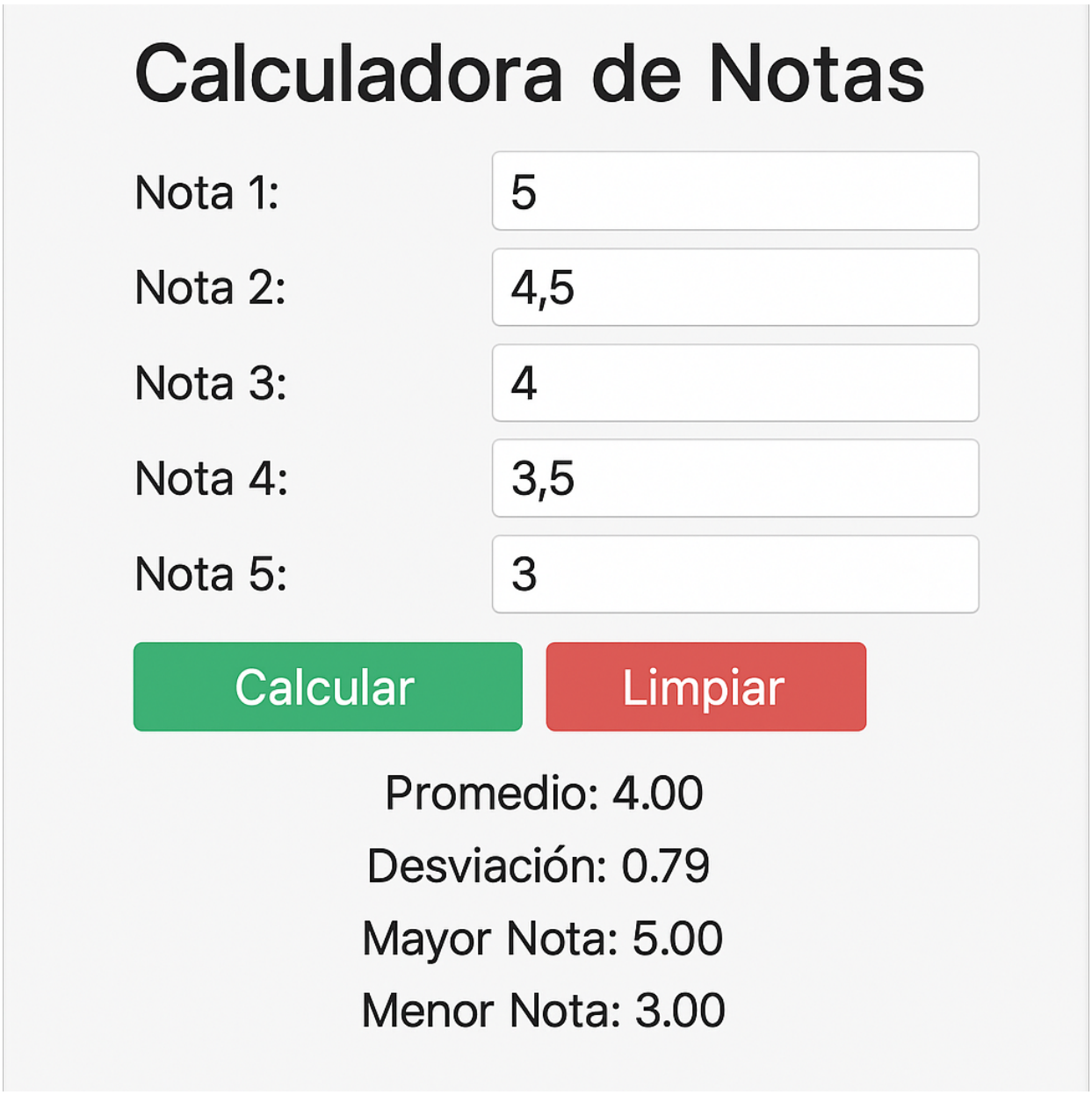

Figura 2: Interfaz de Usuario

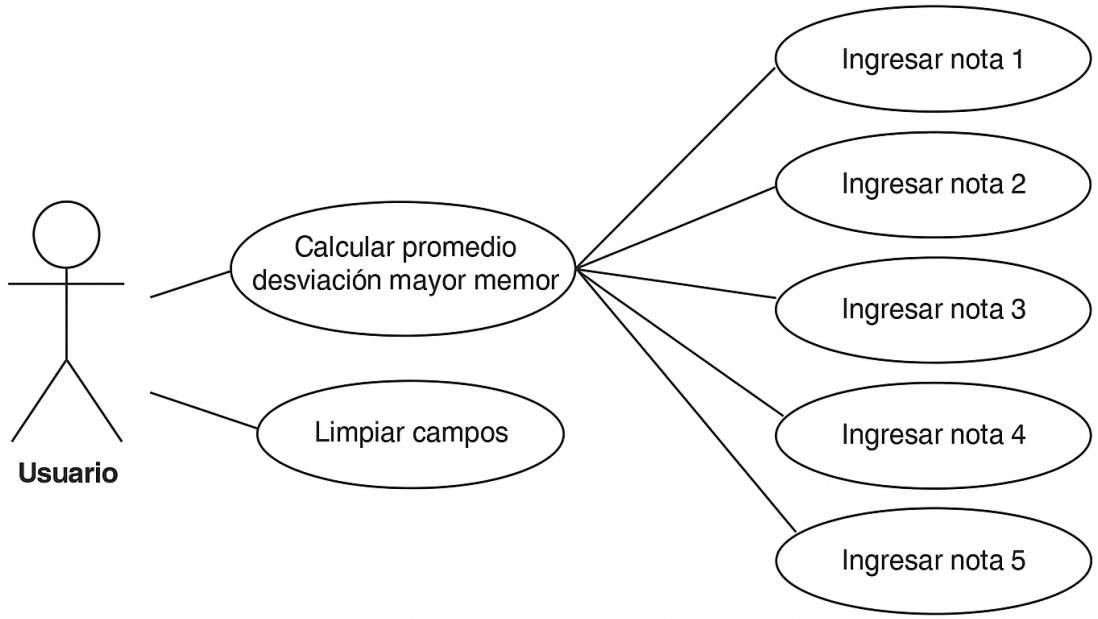

Figura 3: Diagrama de uso

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

class NotasApp:
    def __init__(self, master):
        self.master = master
        master.title("Notas del Estudiante")
        master.geometry("300x350")
        master.resizable(False, False)

        self.labels = []
        self.entries = []

        for i in range(5):
            label = tk.Label(master, text=f"Nota {i+1}:")
            label.place(x=20, y=20 + i*30)
```

```
            entry = tk.Entry(master)
            entry.place(x=100, y=20 + i*30)
            self.labels.append(label)
            self.entries.append(entry)

        self.btn_calcular = tk.Button(master, text="Calcular", command=self.calcular)
        self.btn_calcular.place(x=40, y=190)

        self.btn_limpiar = tk.Button(master, text="Limpiar", command=self.limpiar)
        self.btn_limpiar.place(x=160, y=190)

        self.lbl_promedio = tk.Label(master, text="Promedio = ")
        self.lbl_promedio.place(x=20, y=230)

        self.lbl_desviacion = tk.Label(master, text="Desviación estándar = ")
        self.lbl_desviacion.place(x=20, y=260)

        self.lbl_mayor = tk.Label(master, text="Nota mayor = ")
        self.lbl_mayor.place(x=20, y=290)

        self.lbl_menor = tk.Label(master, text="Nota menor = ")
        self.lbl_menor.place(x=20, y=320)

    def calcular(self):
        try:
            notas = [float(entry.get()) for entry in self.entries]

            if len(notas) != 5 or any(entry.get() == "" for entry in self.entries):
                raise ValueError("Debe ingresar las 5 notas.")

            promedio = sum(notas) / len(notas)
            desviacion = (sum((x - promedio)**2 for x in notas) / len(notas))**0.5  # Pobl
            mayor = max(notas)
            menor = min(notas)

            self.lbl_promedio.config(text=f"Promedio = {promedio:.2f}")
            self.lbl_desviacion.config(text=f"Desviación estándar = {desviacion:.2f}")
            self.lbl_mayor.config(text=f"Nota mayor = {mayor}")
            self.lbl_menor.config(text=f"Nota menor = {menor}")

        except ValueError:
            messagebox.showerror("Error", "Por favor ingrese solo números válidos en los 5
```

```
    def limpiar(self):
        for entry in self.entries:
            entry.delete(0, tk.END)
        self.lbl_promedio.config(text="Promedio = ")
        self.lbl_desviacion.config(text="Desviación estándar = ")
        self.lbl_mayor.config(text="Nota mayor = ")
        self.lbl_menor.config(text="Nota menor = ")

if __name__ == "__main__":
    root = tk.Tk()
    app = NotasApp(root)
    root.mainloop()
```

Punto 2

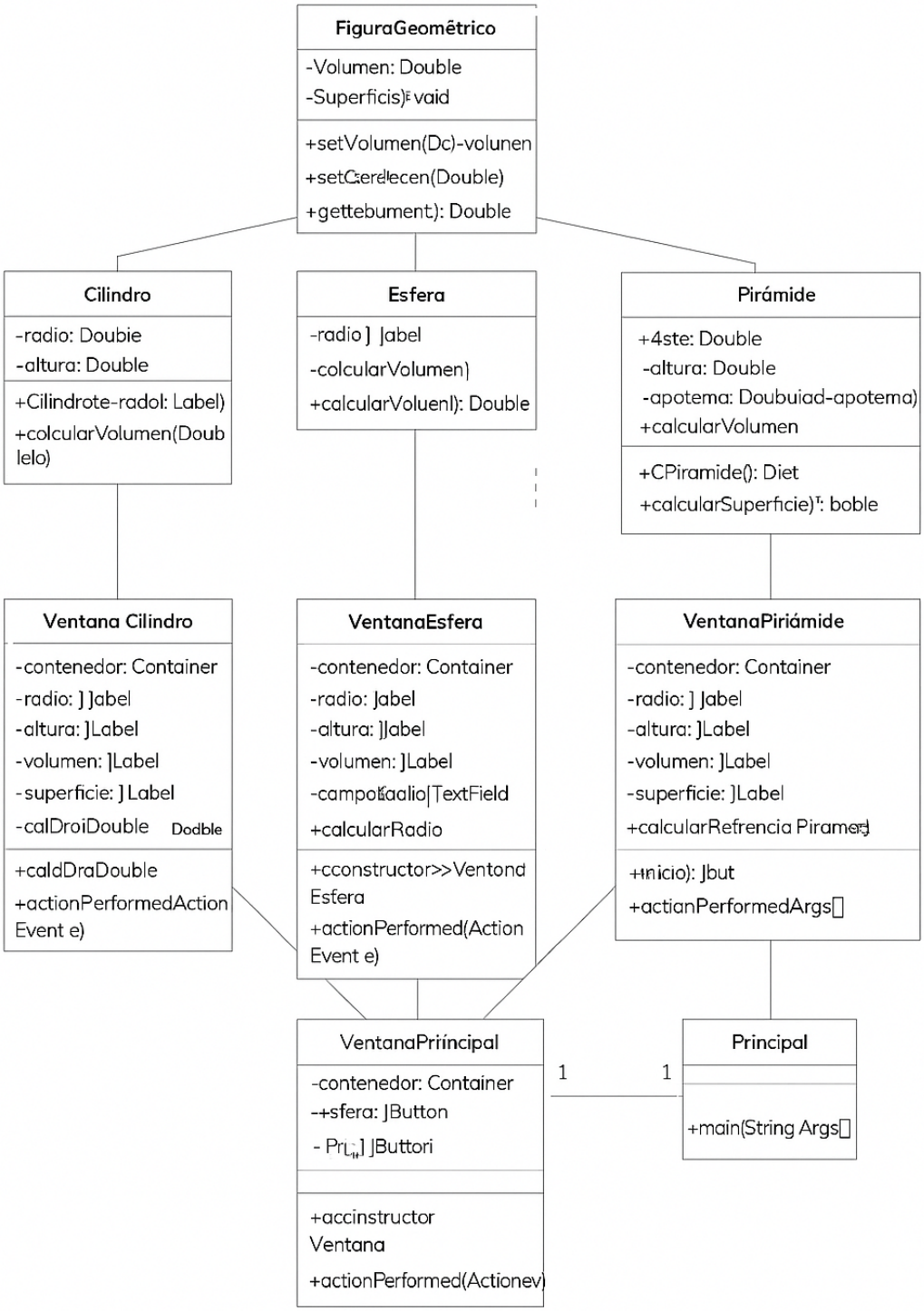

Figura 4: Diagrama de clases

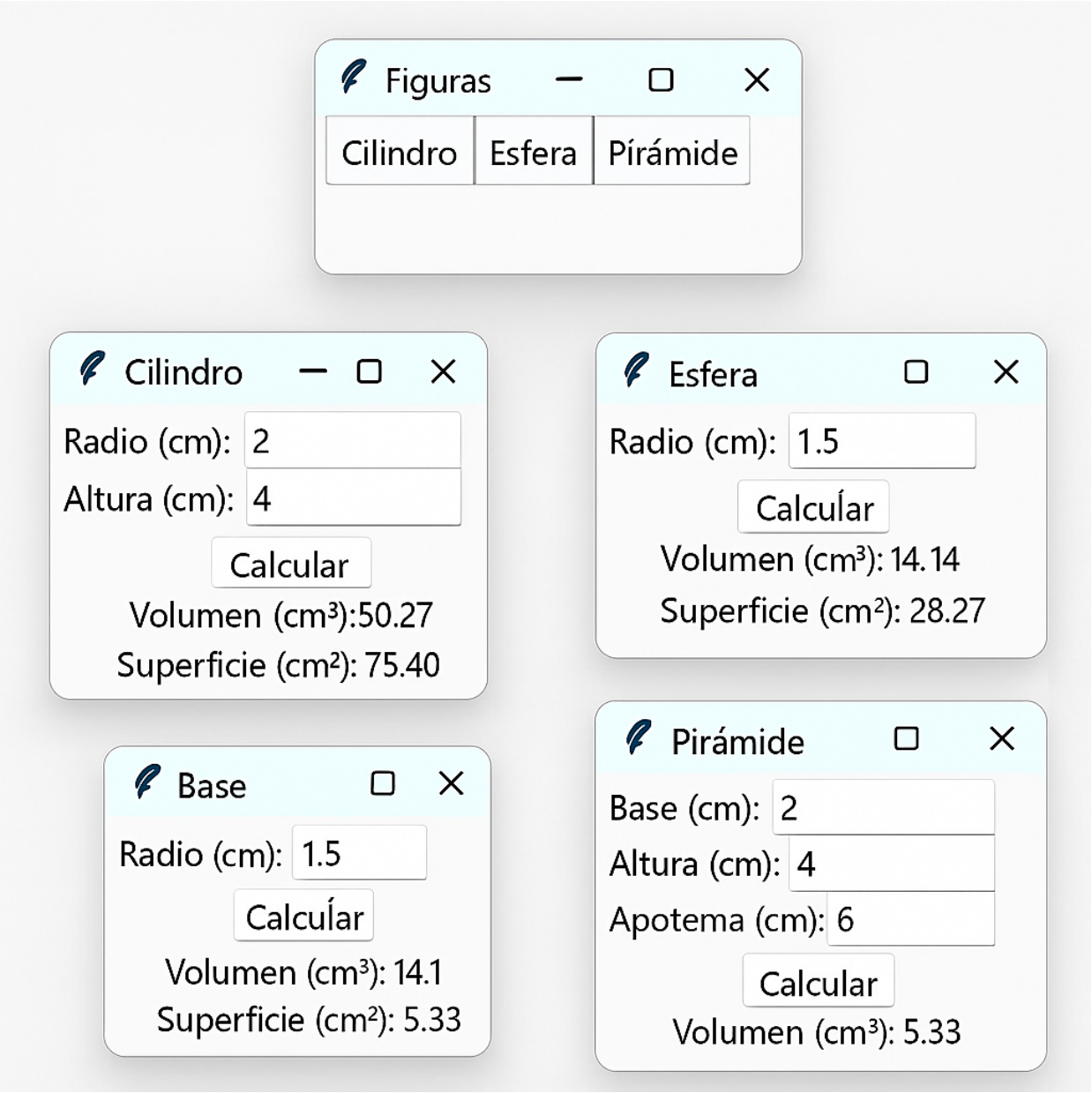

Figura 5: Interfaz de Usuario

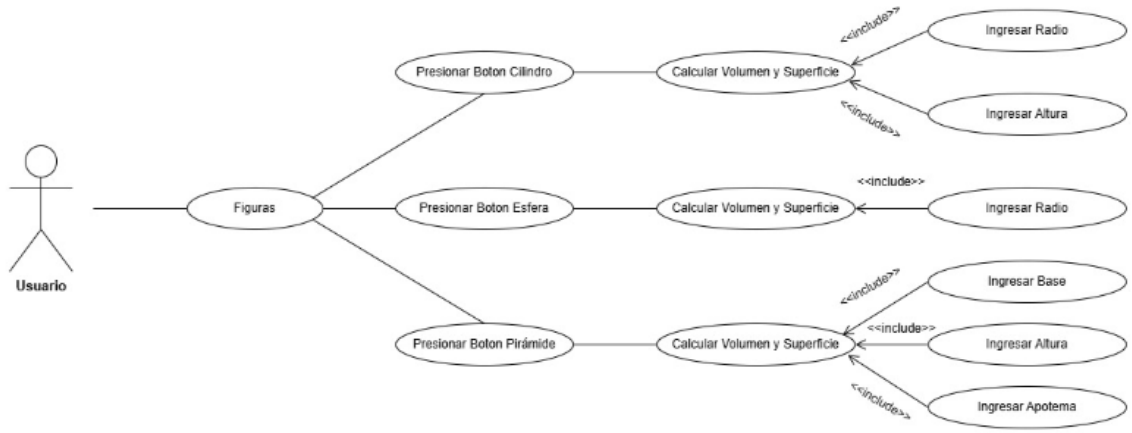

Figura 6: Diagrama de uso

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

class FiguraGeometrica:
    def __init__(self):
        self.volumen = 0
        self.superficie = 0

    def get_volumen(self):
        return self.volumen

    def get_superficie(self):
        return self.superficie

class Cilindro(FiguraGeometrica):
    def __init__(self, radio, altura):
        super().__init__()
        self.radio = radio
        self.altura = altura
        self.calcular_volumen()
        self.calcular_superficie()

    def calcular_volumen(self):
        self.volumen = math.pi * self.radio**2 * self.altura
```

```
    def calcular_superficie(self):
        self.superficie = 2 * math.pi * self.radio * (self.radio + self.altura)

class Esfera(FiguraGeometrica):
    def __init__(self, radio):
        super().__init__()
        self.radio = radio
        self.calcular_volumen()
        self.calcular_superficie()

    def calcular_volumen(self):
        self.volumen = (4/3) * math.pi * self.radio**3

    def calcular_superficie(self):
        self.superficie = 4 * math.pi * self.radio**2

class Piramide(FiguraGeometrica):
    def __init__(self, base, altura, apotema):
        super().__init__()
        self.base = base
        self.altura = altura
        self.apotema = apotema
        self.calcular_volumen()
        self.calcular_superficie()

    def calcular_volumen(self):
        self.volumen = (1/3) * self.base**2 * self.altura

    def calcular_superficie(self):
        self.superficie = self.base**2 + 2 * self.base * self.apotema

class VentanaFigura:
    def __init__(self, root, figura_tipo):
        self.root = root
        self.root.title(figura_tipo)

        self.figura_tipo = figura_tipo
        self.entries = {}

        if figura_tipo == "Cilindro":
            campos = ["Radio", "Altura"]
        elif figura_tipo == "Esfera":
```

```
            campos = ["Radio"]
        elif figura_tipo == "Piramide":
            campos = ["Base", "Altura", "Apotema"]

        for i, campo in enumerate(campos):
            label = tk.Label(root, text=f"{campo} (cm):")
            label.grid(row=i, column=0)
            entry = tk.Entry(root)
            entry.grid(row=i, column=1)
            self.entries[campo] = entry

        self.calcular_button = tk.Button(root, text="Calcular", command=self.calcular)
        self.calcular_button.grid(row=len(campos), column=0, columnspan=2)

        self.volumen_label = tk.Label(root, text="Volumen (cm³): ")
        self.volumen_label.grid(row=len(campos)+1, column=0, columnspan=2)

        self.superficie_label = tk.Label(root, text="Superficie (cm²): ")
        self.superficie_label.grid(row=len(campos)+2, column=0, columnspan=2)

    def calcular(self):
        try:
            valores = {campo: float(self.entries[campo].get()) for campo in self.entries i

            if self.figura_tipo == "Cilindro":
                obj = Cilindro(valores["Radio"], valores["Altura"])
            elif self.figura_tipo == "Esfera":
                obj = Esfera(valores["Radio"])
            elif self.figura_tipo == "Piramide":
                obj = Piramide(valores["Base"], valores["Altura"], valores["Apotema"])
            else:
                return

            self.volumen_label.config(text=f"Volumen (cm³): {obj.get_volumen():.2f}")
            self.superficie_label.config(text=f"Superficie (cm²): {obj.get_superficie():.2
        except ValueError:
            messagebox.showerror("Error", "Ingrese valores numéricos válidos")

class VentanaPrincipal:
    def __init__(self, root):
        self.root = root
        self.root.title("Figuras")
```

```
        tk.Button(root, text="Cilindro", command=lambda: self.abrir_ventana("Cilindro")).g
        tk.Button(root, text="Esfera", command=lambda: self.abrir_ventana("Esfera")).grid(
        tk.Button(root, text="Pirámide", command=lambda: self.abrir_ventana("Piramide")).g

    def abrir_ventana(self, figura_tipo):
        nueva_ventana = tk.Toplevel(self.root)
        VentanaFigura(nueva_ventana, figura_tipo)

if __name__ == "__main__":
    root = tk.Tk()
    app = VentanaPrincipal(root)
    root.mainloop()
```