**Descubriendo Colombia - Sistema de Reservas**

Este proyecto consiste en el desarrollo de un sistema de reservas de vuelos y hoteles, titulado "Descubriendo Colombia". El sistema permite gestionar paquetes turísticos con un enfoque en la promoción de destinos colombianos. El código implementa las siguientes funcionalidades clave:

1. **Gestión de Vuelos**:
   * Agregar nuevos vuelos con información de destino, fecha de salida, fecha de regreso y precio.
   * Editar vuelos existentes mediante un sistema de identificación por ID.
   * Eliminar vuelos con confirmación para evitar borrados accidentales.
   * Mostrar todos los vuelos disponibles con sus respectivos detalles.
2. **Gestión de Hoteles**:
   * Agregar nuevos hoteles con información de nombre, ubicación, precio por noche y disponibilidad.
   * Editar la información de hoteles existentes.
   * Eliminar hoteles con confirmación de acción.
   * Mostrar todos los hoteles disponibles.
3. **Validaciones y Manejo de Errores**:
   * Validación de entradas para asegurar que los datos ingresados (como fechas y precios) sean correctos.
   * Manejo de errores para prevenir datos inconsistentes y proporcionar retroalimentación clara al usuario.
4. **Interfaz de Usuario**:
   * La interfaz gráfica está diseñada con elementos claros y organizados para facilitar la interacción del usuario, con botones que permiten agregar, editar, eliminar y mostrar vuelos y hoteles.
   * Se incluye un sistema de scroll para visualizar todos los elementos de la interfaz.
5. **Confirmaciones de Acción**:
   * Al eliminar vuelos u hoteles, el sistema solicita confirmación al usuario para evitar eliminaciones accidentales.

El proyecto ha sido diseñado para ser escalable y fácil de mantener, con código modularizado y bien documentado. Es una solución eficaz para la gestión de reservas en el contexto de destinos turísticos colombianos; aunque también se pueden agregar diferentes puntos de destino a otros países y continentes, aún no se han implementado los vuelos hasta allá.

import tkinter as tk

from tkinter import messagebox

import sqlite3

from datetime import datetime

# Conexión a la base de datos

conn = sqlite3.connect('descubriendo\_colombia.db')

cursor = conn.cursor()

# Crear tablas para vuelos, hoteles, clientes y paquetes

def crear\_tablas():

cursor.execute('''CREATE TABLE IF NOT EXISTS vuelos (

id INTEGER PRIMARY KEY AUTOINCREMENT,

destino TEXT,

fecha\_salida TEXT,

fecha\_regreso TEXT,

precio REAL

)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS hoteles (

id INTEGER PRIMARY KEY AUTOINCREMENT,

nombre TEXT,

ubicacion TEXT,

precio\_por\_noche REAL,

disponibilidad INTEGER

)''')

conn.commit()

# Validar formato de fecha (YYYY-MM-DD)

def validar\_fecha(fecha\_texto):

try:

fecha = datetime.strptime(fecha\_texto, '%Y-%m-%d')

return fecha

except ValueError:

return None

# Función para mostrar vuelos

def mostrar\_vuelos():

cursor.execute("SELECT \* FROM vuelos")

vuelos = cursor.fetchall()

resultado = ""

for vuelo in vuelos:

resultado += f"ID: {vuelo[0]}, Destino: {vuelo[1]}, Salida: {vuelo[2]}, Regreso: {vuelo[3]}, Precio: ${vuelo[4]:.2f}\n"

messagebox.showinfo("Vuelos Disponibles", resultado if resultado else "No hay vuelos disponibles.")

# Función para mostrar hoteles

def mostrar\_hoteles():

cursor.execute("SELECT \* FROM hoteles")

hoteles = cursor.fetchall()

resultado = ""

for hotel in hoteles:

resultado += f"ID: {hotel[0]}, Nombre: {hotel[1]}, Ubicación: {hotel[2]}, Precio por noche: ${hotel[3]:.2f}, Disponibilidad: {hotel[4]}\n"

messagebox.showinfo("Hoteles Disponibles", resultado if resultado else "No hay hoteles disponibles.")

# Función para agregar un vuelo

def agregar\_vuelo():

destino = entry\_destino.get()

fecha\_salida = entry\_salida.get()

fecha\_regreso = entry\_regreso.get()

precio = entry\_precio\_vuelo.get()

if not destino or not fecha\_salida or not fecha\_regreso or not precio:

messagebox.showerror("Error", "Todos los campos son obligatorios.")

return

fecha\_salida\_valida = validar\_fecha(fecha\_salida)

fecha\_regreso\_valida = validar\_fecha(fecha\_regreso)

if not fecha\_salida\_valida or not fecha\_regreso\_valida:

messagebox.showerror("Error", "Formato de fecha incorrecto. Usa YYYY-MM-DD.")

return

if fecha\_regreso\_valida < fecha\_salida\_valida:

messagebox.showerror("Error", "La fecha de regreso no puede ser anterior a la fecha de salida.")

return

try:

precio = float(precio)

if precio <= 0:

raise ValueError("El precio debe ser positivo.")

except ValueError:

messagebox.showerror("Error", "El precio debe ser un número válido mayor que 0.")

return

try:

cursor.execute("INSERT INTO vuelos (destino, fecha\_salida, fecha\_regreso, precio) VALUES (?, ?, ?, ?)",

(destino, fecha\_salida, fecha\_regreso, precio))

conn.commit()

messagebox.showinfo("Éxito", "Vuelo agregado correctamente.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Función para editar un vuelo

def editar\_vuelo():

id\_vuelo = entry\_id\_vuelo.get()

destino = entry\_destino.get()

fecha\_salida = entry\_salida.get()

fecha\_regreso = entry\_regreso.get()

precio = entry\_precio\_vuelo.get()

if not id\_vuelo or not destino or not fecha\_salida or not fecha\_regreso or not precio:

messagebox.showerror("Error", "Todos los campos son obligatorios.")

return

fecha\_salida\_valida = validar\_fecha(fecha\_salida)

fecha\_regreso\_valida = validar\_fecha(fecha\_regreso)

if not fecha\_salida\_valida or not fecha\_regreso\_valida:

messagebox.showerror("Error", "Formato de fecha incorrecto. Usa YYYY-MM-DD.")

return

if fecha\_regreso\_valida < fecha\_salida\_valida:

messagebox.showerror("Error", "La fecha de regreso no puede ser anterior a la fecha de salida.")

return

try:

precio = float(precio)

if precio <= 0:

raise ValueError("El precio debe ser positivo.")

except ValueError:

messagebox.showerror("Error", "El precio debe ser un número válido mayor que 0.")

return

try:

cursor.execute('''UPDATE vuelos SET destino = ?, fecha\_salida = ?, fecha\_regreso = ?, precio = ?

WHERE id = ?''',

(destino, fecha\_salida, fecha\_regreso, precio, id\_vuelo))

conn.commit()

if cursor.rowcount > 0:

messagebox.showinfo("Éxito", "Vuelo editado correctamente.")

else:

messagebox.showerror("Error", "No se encontró un vuelo con ese ID.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Función para eliminar vuelos

def eliminar\_vuelo():

id\_vuelo = entry\_id\_vuelo.get()

if messagebox.askokcancel("Confirmación", f"¿Estás seguro de que deseas eliminar el vuelo con ID {id\_vuelo}?"):

try:

cursor.execute("DELETE FROM vuelos WHERE id = ?", (id\_vuelo,))

conn.commit()

if cursor.rowcount > 0:

messagebox.showinfo("Éxito", "Vuelo eliminado correctamente.")

else:

messagebox.showerror("Error", "No se encontró un vuelo con ese ID.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Función para agregar un hotel

def agregar\_hotel():

nombre = entry\_nombre\_hotel.get()

ubicacion = entry\_ubicacion.get()

precio = entry\_precio\_hotel.get()

disponibilidad = entry\_disponibilidad.get()

if not nombre or not ubicacion or not precio or not disponibilidad:

messagebox.showerror("Error", "Todos los campos son obligatorios.")

return

try:

precio = float(precio)

if precio <= 0:

raise ValueError("El precio debe ser positivo.")

except ValueError:

messagebox.showerror("Error", "El precio debe ser un número válido mayor que 0.")

return

try:

disponibilidad = int(disponibilidad)

if disponibilidad < 0:

raise ValueError("La disponibilidad debe ser un número entero positivo.")

except ValueError:

messagebox.showerror("Error", "La disponibilidad debe ser un número entero válido.")

return

try:

cursor.execute("INSERT INTO hoteles (nombre, ubicacion, precio\_por\_noche, disponibilidad) VALUES (?, ?, ?, ?)",

(nombre, ubicacion, precio, disponibilidad))

conn.commit()

messagebox.showinfo("Éxito", "Hotel agregado correctamente.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Función para editar un hotel

def editar\_hotel():

id\_hotel = entry\_id\_hotel.get()

nombre = entry\_nombre\_hotel.get()

ubicacion = entry\_ubicacion.get()

precio = entry\_precio\_hotel.get()

disponibilidad = entry\_disponibilidad.get()

if not id\_hotel or not nombre or not ubicacion or not precio or not disponibilidad:

messagebox.showerror("Error", "Todos los campos son obligatorios.")

return

try:

precio = float(precio)

if precio <= 0:

raise ValueError("El precio debe ser positivo.")

except ValueError:

messagebox.showerror("Error", "El precio debe ser un número válido mayor que 0.")

return

try:

disponibilidad = int(disponibilidad)

if disponibilidad < 0:

raise ValueError("La disponibilidad debe ser un número entero positivo.")

except ValueError:

messagebox.showerror("Error", "La disponibilidad debe ser un número entero válido.")

return

try:

cursor.execute('''UPDATE hoteles SET nombre = ?, ubicacion = ?, precio\_por\_noche = ?, disponibilidad = ?

WHERE id = ?''',

(nombre, ubicacion, precio, disponibilidad, id\_hotel))

conn.commit()

if cursor.rowcount > 0:

messagebox.showinfo("Éxito", "Hotel editado correctamente.")

else:

messagebox.showerror("Error", "No se encontró un hotel con ese ID.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Función para eliminar hoteles

def eliminar\_hotel():

id\_hotel = entry\_id\_hotel.get()

if messagebox.askokcancel("Confirmación", f"¿Estás seguro de que deseas eliminar el hotel con ID {id\_hotel}?"):

try:

cursor.execute("DELETE FROM hoteles WHERE id = ?", (id\_hotel,))

conn.commit()

if cursor.rowcount > 0:

messagebox.showinfo("Éxito", "Hotel eliminado correctamente.")

else:

messagebox.showerror("Error", "No se encontró un hotel con ese ID.")

except Exception as e:

messagebox.showerror("Error", f"Hubo un problema: {str(e)}")

# Crear la ventana principal

root = tk.Tk()

root.title("Descubriendo Colombia - Sistema de Reservas")

root.geometry("800x600") # Tamaño inicial de la ventana

# Crear el contenedor principal con scroll

frame\_principal = tk.Frame(root)

frame\_principal.pack(fill="both", expand=True)

# Crear el canvas donde estará el contenido y el scrollbar

canvas = tk.Canvas(frame\_principal)

canvas.pack(side="left", fill="both", expand=True)

scrollbar = tk.Scrollbar(frame\_principal, orient="vertical", command=canvas.yview)

scrollbar.pack(side="right", fill="y")

# Crear otro frame dentro del canvas para contener los widgets

frame\_contenido = tk.Frame(canvas)

canvas.create\_window((0, 0), window=frame\_contenido, anchor="n")

# Vincular el frame y el scrollbar

def configurar\_scroll(event):

canvas.configure(scrollregion=canvas.bbox("all"))

frame\_contenido.bind("<Configure>", configurar\_scroll)

# Crear un frame interno con grid layout para centrar los widgets

frame\_formulario = tk.Frame(frame\_contenido)

frame\_formulario.grid(row=0, column=0, padx=20, pady=20)

# Widgets de vuelo

label\_vuelo = tk.Label(frame\_formulario, text="Agregar/Editar un vuelo", font=("Arial", 14, "bold"))

label\_vuelo.grid(row=0, column=0, columnspan=2, pady=10)

label\_destino = tk.Label(frame\_formulario, text="Destino")

label\_destino.grid(row=1, column=0, pady=5)

entry\_destino = tk.Entry(frame\_formulario)

entry\_destino.grid(row=1, column=1, pady=5)

label\_salida = tk.Label(frame\_formulario, text="Fecha de salida (YYYY-MM-DD)")

label\_salida.grid(row=2, column=0, pady=5)

entry\_salida = tk.Entry(frame\_formulario)

entry\_salida.grid(row=2, column=1, pady=5)

label\_regreso = tk.Label(frame\_formulario, text="Fecha de regreso (YYYY-MM-DD)")

label\_regreso.grid(row=3, column=0, pady=5)

entry\_regreso = tk.Entry(frame\_formulario)

entry\_regreso.grid(row=3, column=1, pady=5)

label\_precio\_vuelo = tk.Label(frame\_formulario, text="Precio del vuelo")

label\_precio\_vuelo.grid(row=4, column=0, pady=5)

entry\_precio\_vuelo = tk.Entry(frame\_formulario)

entry\_precio\_vuelo.grid(row=4, column=1, pady=5)

button\_agregar\_vuelo = tk.Button(frame\_formulario, text="Agregar Vuelo", command=agregar\_vuelo)

button\_agregar\_vuelo.grid(row=5, column=0, columnspan=2, pady=10)

button\_editar\_vuelo = tk.Button(frame\_formulario, text="Editar Vuelo", command=editar\_vuelo)

button\_editar\_vuelo.grid(row=6, column=0, columnspan=2, pady=10)

button\_mostrar\_vuelos = tk.Button(frame\_formulario, text="Mostrar Vuelos", command=mostrar\_vuelos)

button\_mostrar\_vuelos.grid(row=7, column=0, columnspan=2, pady=10)

# Widgets de hotel

label\_hotel = tk.Label(frame\_formulario, text="Agregar/Editar un hotel", font=("Arial", 14, "bold"))

label\_hotel.grid(row=8, column=0, columnspan=2, pady=10)

label\_nombre\_hotel = tk.Label(frame\_formulario, text="Nombre del hotel")

label\_nombre\_hotel.grid(row=9, column=0, pady=5)

entry\_nombre\_hotel = tk.Entry(frame\_formulario)

entry\_nombre\_hotel.grid(row=9, column=1, pady=5)

label\_ubicacion = tk.Label(frame\_formulario, text="Ubicación")

label\_ubicacion.grid(row=10, column=0, pady=5)

entry\_ubicacion = tk.Entry(frame\_formulario)

entry\_ubicacion.grid(row=10, column=1, pady=5)

label\_precio\_hotel = tk.Label(frame\_formulario, text="Precio por noche")

label\_precio\_hotel.grid(row=11, column=0, pady=5)

entry\_precio\_hotel = tk.Entry(frame\_formulario)

entry\_precio\_hotel.grid(row=11, column=1, pady=5)

label\_disponibilidad = tk.Label(frame\_formulario, text="Disponibilidad")

label\_disponibilidad.grid(row=12, column=0, pady=5)

entry\_disponibilidad = tk.Entry(frame\_formulario)

entry\_disponibilidad.grid(row=12, column=1, pady=5)

button\_agregar\_hotel = tk.Button(frame\_formulario, text="Agregar Hotel", command=agregar\_hotel)

button\_agregar\_hotel.grid(row=13, column=0, columnspan=2, pady=10)

button\_editar\_hotel = tk.Button(frame\_formulario, text="Editar Hotel", command=editar\_hotel)

button\_editar\_hotel.grid(row=14, column=0, columnspan=2, pady=10)

button\_mostrar\_hoteles = tk.Button(frame\_formulario, text="Mostrar Hoteles", command=mostrar\_hoteles)

button\_mostrar\_hoteles.grid(row=15, column=0, columnspan=2, pady=10)

# Widgets para eliminar vuelos y hoteles

label\_eliminar\_vuelo = tk.Label(frame\_formulario, text="Eliminar vuelo (ID)")

label\_eliminar\_vuelo.grid(row=16, column=0, pady=5)

entry\_id\_vuelo = tk.Entry(frame\_formulario)

entry\_id\_vuelo.grid(row=16, column=1, pady=5)

button\_eliminar\_vuelo = tk.Button(frame\_formulario, text="Eliminar Vuelo", command=eliminar\_vuelo)

button\_eliminar\_vuelo.grid(row=17, column=0, columnspan=2, pady=10)

label\_eliminar\_hotel = tk.Label(frame\_formulario, text="Eliminar hotel (ID)")

label\_eliminar\_hotel.grid(row=18, column=0, pady=5)

entry\_id\_hotel = tk.Entry(frame\_formulario)

entry\_id\_hotel.grid(row=18, column=1, pady=5)

button\_eliminar\_hotel = tk.Button(frame\_formulario, text="Eliminar Hotel", command=eliminar\_hotel)

button\_eliminar\_hotel.grid(row=19, column=0, columnspan=2, pady=10)

# Ejecutar la interfaz

crear\_tablas()

root.mainloop()