**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC VĂN LANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A red logo with white text

Description automatically generated\_\_\_\_\_\_\_\_\_\_\_\_\_**

**BÁO CÁO PROJECT 2**

**GIAO DIỆN QUẢN LÍ DATABASE ĐƠN GIẢN**

Giảng viên hướng dẫn: Huỳnh Thái Học

Sinh viên thực hiện: Trương Việt Vũ

Mã số sinh viên: 2274802011045

Học phần : Lập trình Python nâng cao

Tp Hồ Chí Minh ngày 20 tháng 10 năm 2024

**NỘI DUNG BÁO CÁO**

Giao diện GUI đơn giản viết bằng code python dùng để quản lí cơ sở dữ liệu liên quan đến sách

**1. Giao diện**

A screenshot of a computer

Description automatically generatedGiao diện đăng nhập

A screenshot of a computer

Description automatically generatedNếu nhập sai sẽ báo lỗi

Và nhập đúng sẽ hiện thông báo kết nối thành công và chuyển sang giao diện quản lí

A screenshot of a computer

Description automatically generated

Lúc này việc quản lí cơ sở dữ liệu sẽ gồm các chức năng:

A screenshot of a computer

Description automatically generatedChức năng tìm kiếm sách theo tên sách:

A screenshot of a computer

Description automatically generatedChức năng thêm vào sách mới, không cần nhập ID khi thêm sách vì ID sẽ được tạo tự động, sau khi thêm thành công dữ liệu sẽ lập tức được đưa xuống phần hiển thị dữ liệu bên dưới

A screenshot of a computer

Description automatically generatedChức năng cập nhật sách, người dung có thể nhập vào các entry hoặc chỉ cần nhấp vào dữ liệu bên dưới lập tức dữ liệu sẽ được đưa vào entry để người dung có thể sửa đổi, id sẽ không thay đổi được

A screenshot of a computer

Description automatically generatedChức năng xóa, người dung nhập vào id của sách muốn xóa hoặc nhấp vào hàng dữ liệu bên dưới id sẽ lập tức được đưa lên entry phía trên để người dùng xóa dễ dàng hơn

A screenshot of a computer

Description automatically generatedChức năng ngắt kết nối database, ấn vào button ngắt kết nối sẽ giúp bạn ngắt kết nối ra khỏi database và quay về trang login

Ngoài ra giao diện còn có menu File bao gồm việc save file và exit ra khỏi giao diện

A screenshot of a computer

Description automatically generated

**2. Chức năng**

Đây là 1 giao diện quản lí database liên quan đến sách đơn giản với các chức năng như là:

* Đăng nhập vào database
* Tìm kiếm sách
* Thêm vào sách mới
* Cập nhật thông tin của sách
* Xóa sách
* Ngắt kết nối database
* Các thao tác trên cơ sở dữ liệu sau khi thành công đều sẽ hiện thông báo ra màn hình cũng như hiện thông báo khi có lỗi xảy ra
* Sử dụng sql.Identifier để tránh xảy ra các xung đột trong câu lệnh sql
* Đã xử lí các trường hợp bị block transaction

**3. Mã nguồn**

File app.py : dung để kết nối vào database cùng với các giao diện chính

import tkinter as tk

from tkinter import filedialog, messagebox

from tkinter import ttk

import psycopg2

from functions import SearchBooks, AddBooks, UpdateBooks, DeleteBooks, DisconnectDB

class DatabaseApp:

    def \_\_init\_\_(*self*, *root*):

*self*.root = root

*self*.root.title("Database App")

        # Tabs

*self*.tab\_control = ttk.Notebook(*self*.root)

*self*.login\_tab = tk.Frame(*self*.tab\_control)

*self*.search\_tab = tk.Frame(*self*.tab\_control)

*self*.add\_tab = tk.Frame(*self*.tab\_control)

*self*.update\_tab = tk.Frame(*self*.tab\_control)

*self*.delete\_tab = tk.Frame(*self*.tab\_control)

*self*.file\_tab = tk.Frame(*self*.tab\_control)

*self*.tab\_control.add(*self*.login\_tab, *text*="Login")

*self*.tab\_control.add(*self*.search\_tab, *text*="Tìm kiếm sách")

*self*.tab\_control.add(*self*.add\_tab, *text*="Thêm sách")

*self*.tab\_control.add(*self*.update\_tab, *text*="Cập nhật sách")

*self*.tab\_control.add(*self*.delete\_tab, *text*="Xóa sách")

*self*.tab\_control.pack(*expand*=1, *fill*="both")

        # Hide tabs initially

*self*.tab\_control.hide(*self*.search\_tab)

*self*.tab\_control.hide(*self*.add\_tab)

*self*.tab\_control.hide(*self*.update\_tab)

*self*.tab\_control.hide(*self*.delete\_tab)

        # Database connection fields

*self*.db\_name = tk.StringVar(*value*='sach')

*self*.user = tk.StringVar(*value*='postgres')

*self*.password = tk.StringVar(*value*='1234567890')

*self*.host = tk.StringVar(*value*='localhost')

*self*.port = tk.StringVar(*value*='5432')

*self*.table\_name = tk.StringVar(*value*='sach')

        # Create the login interface

*self*.create\_login\_widgets()

        # Create the add, update, delete interfaces

*self*.search\_books = SearchBooks(*self*.search\_tab, *self*)

*self*.add\_books = AddBooks(*self*.add\_tab, *self*)

*self*.update\_books = UpdateBooks(*self*.update\_tab, *self*)

*self*.delete\_books = DeleteBooks(*self*.delete\_tab, *self*)

*self*.disconnect\_tab = tk.Frame(*self*.tab\_control)

*self*.tab\_control.add(*self*.disconnect\_tab, *text*="Ngắt kết nối")

*self*.disconnect\_db = DisconnectDB(*self*.disconnect\_tab, *self*)

*self*.create\_file\_menu()

    def create\_file\_menu(*self*):

        # Create a menu bar

        menubar = tk.Menu(*self*.root)

        # Create the File menu

        file\_menu = tk.Menu(menubar, *tearoff*=0)

        file\_menu.add\_command(*label*="Save", *command*=*self*.save\_data)

        file\_menu.add\_separator()  # Optional separator

        file\_menu.add\_command(*label*="Exit", *command*=*self*.exit\_app)

        # Add the File menu to the menubar

        menubar.add\_cascade(*label*="File", *menu*=file\_menu)

        # Configure the root window to use the menubar

*self*.root.config(*menu*=menubar)

    def save\_data(*self*):

        # Mở hộp thoại lưu file

        file\_path = filedialog.asksaveasfilename(

*defaultextension*=".txt",  # Định dạng mặc định

*filetypes*=[("Text files", "\*.txt"), ("All files", "\*.\*")],  # Các loại file hỗ trợ

*title*="Lưu file"

        )

        if file\_path:  # Nếu người dùng đã chọn file

            try:

                with open(file\_path, 'w', *encoding*='utf-8') as file:

                    # Ví dụ: Lưu dữ liệu từ một biến hoặc danh sách

                    data\_to\_save = "Dữ liệu ví dụ để lưu vào file"  # Thay thế bằng dữ liệu thực tế

                    file.write(data\_to\_save)

                messagebox.showinfo("Save", "Dữ liệu đã được lưu thành công!")

            except *Exception* as e:

                messagebox.showerror("Error", f"Không thể lưu file: {e}")

    def exit\_app(*self*):

        # Hàm thoát ứng dụng

*self*.root.quit()

    def create\_login\_widgets(*self*):

        # Connection section in login tab

        connection\_frame = tk.Frame(*self*.login\_tab)

        connection\_frame.pack(*pady*=10)

        tk.Label(connection\_frame, *text*="DB Name:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(connection\_frame, *textvariable*=*self*.db\_name).grid(*row*=0, *column*=1, *padx*=5, *pady*=5)

        tk.Label(connection\_frame, *text*="User:").grid(*row*=1, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(connection\_frame, *textvariable*=*self*.user).grid(*row*=1, *column*=1, *padx*=5, *pady*=5)

        tk.Label(connection\_frame, *text*="Password:").grid(*row*=2, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(connection\_frame, *textvariable*=*self*.password, *show*="\*").grid(*row*=2, *column*=1, *padx*=5, *pady*=5)

        tk.Label(connection\_frame, *text*="Host:").grid(*row*=3, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(connection\_frame, *textvariable*=*self*.host).grid(*row*=3, *column*=1, *padx*=5, *pady*=5)

        tk.Label(connection\_frame, *text*="Port:").grid(*row*=4, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(connection\_frame, *textvariable*=*self*.port).grid(*row*=4, *column*=1, *padx*=5, *pady*=5)

        tk.Button(connection\_frame, *text*="Connect", *command*=*self*.connect\_db).grid(*row*=5, *columnspan*=2, *pady*=10)

    def connect\_db(*self*):

        try:

*self*.conn = psycopg2.connect(

*dbname*=*self*.db\_name.get(),

*user*=*self*.user.get(),

*password*=*self*.password.get(),

*host*=*self*.host.get(),

*port*=*self*.port.get()

            )

*self*.cur = *self*.conn.cursor()

            messagebox.showinfo("Success", "Kết nối cơ sở dữ liệu thành công!")

            # Show search, add, update, delete book tabs

*self*.tab\_control.add(*self*.search\_tab, *text*="Tìm kiếm sách")

*self*.tab\_control.add(*self*.add\_tab, *text*="Thêm sách")

*self*.tab\_control.add(*self*.update\_tab, *text*="Cập nhật sách")

*self*.tab\_control.add(*self*.delete\_tab, *text*="Xóa sách")

*self*.tab\_control.select(*self*.search\_tab)

            # Load data in all Treeviews

*self*.search\_books.load\_data()

*self*.add\_books.load\_data()

*self*.update\_books.load\_data()

*self*.delete\_books.load\_data()

            # Bind tab change event to load data

*self*.tab\_control.bind("<<NotebookTabChanged>>", *self*.on\_tab\_change)

        except *Exception* as e:

            messagebox.showerror("Error", f"Lỗi xảy ra trong quá trình kết nối cơ sở dữ liệu: {e}")

    def on\_tab\_change(*self*, *event*):

        current\_tab = event.widget.tab(event.widget.select(), "text")

        if current\_tab == "Tìm kiếm sách":

*self*.search\_books.load\_data()

        elif current\_tab == "Thêm sách":

*self*.add\_books.load\_data()

        elif current\_tab == "Cập nhật sách":

*self*.update\_books.load\_data()

        elif current\_tab == "Xóa sách":

*self*.delete\_books.load\_data()

File functions.py : dung để chứa các chức năng tìm kiếm sách, thêm sách, cập nhật sách, xóa sách và ngắt kết nối database

import tkinter as tk

from tkinter import messagebox

from tkinter import ttk

from psycopg2 import sql

class SearchBooks:

    def \_\_init\_\_(*self*, *tab*, *app*):

*self*.tab = *tab*

*self*.app = *app*

        search\_frame = tk.Frame(*self*.tab)

        search\_frame.pack(*pady*=10)

*self*.search\_var = tk.StringVar()  # Tìm kiếm theo tên sách

        tk.Label(search\_frame, *text*="Tìm kiếm theo tên sách:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(search\_frame, *textvariable*=*self*.search\_var).grid(*row*=0, *column*=1, *padx*=5, *pady*=5)

        tk.Button(search\_frame, *text*="Tìm kiếm", *command*=*self*.search\_data).grid(*row*=0, *column*=2, *padx*=5, *pady*=5)

        tk.Button(search\_frame, *text*="Tải lại dữ liệu", *command*=*self*.load\_data).grid(*row*=0, *column*=3, *padx*=5, *pady*=5)

        # Treeview for displaying search results

*self*.data\_tree = ttk.Treeview(*self*.tab, *columns*=("ID", "Tên sách", "Tác giả", "Năm xuất bản", "Thể loại"), *show*="headings", *height*=10)

*self*.data\_tree.column("ID", *width*=100, *anchor*="center")

*self*.data\_tree.column("Tên sách", *width*=200, *anchor*="center")

*self*.data\_tree.column("Tác giả", *width*=200, *anchor*="center")

*self*.data\_tree.column("Năm xuất bản", *width*=150, *anchor*="center")

*self*.data\_tree.column("Thể loại", *width*=150, *anchor*="center")

*self*.data\_tree.heading("ID", *text*="ID")

*self*.data\_tree.heading("Tên sách", *text*="Tên sách")

*self*.data\_tree.heading("Tác giả", *text*="Tác giả")

*self*.data\_tree.heading("Năm xuất bản", *text*="Năm xuất bản")

*self*.data\_tree.heading("Thể loại", *text*="Thể loại")

*self*.data\_tree.pack(*side*="bottom", *fill*="both", *expand*=True)

    def load\_data(*self*):

        try:

            query = sql.SQL("SELECT \* FROM {}").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(query)

            rows = *self*.app.cur.fetchall()

            # Clear previous data

            for item in *self*.data\_tree.get\_children():

*self*.data\_tree.delete(item)

            # Insert new data into Treeview

            for row in rows:

*self*.data\_tree.insert("", "end", *values*=row)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi tải dữ liệu: {e}")

    def search\_data(*self*):

        try:

            search\_query = sql.SQL("SELECT \* FROM {} WHERE ten\_sach ILIKE %s").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(search\_query, ('%' + *self*.search\_var.get() + '%',))

            rows = *self*.app.cur.fetchall()

            # Clear previous data

            for item in *self*.data\_tree.get\_children():

*self*.data\_tree.delete(item)

            # Insert search results into Treeview

            for row in rows:

*self*.data\_tree.insert("", "end", *values*=row)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi tìm kiếm: {e}")

class AddBooks:

    def \_\_init\_\_(*self*, *tab*, *app*):

*self*.tab = *tab*

*self*.app = *app*

        # Insert section

        insert\_frame = tk.Frame(*self*.tab)

        insert\_frame.pack(*pady*=10)

*self*.column1 = tk.StringVar()  # Tên sách

*self*.column2 = tk.StringVar()  # Tác giả

*self*.column3 = tk.StringVar()  # NXB

*self*.column4 = tk.StringVar()  # Thể loại

        tk.Label(insert\_frame, *text*="Tên sách:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(insert\_frame, *textvariable*=*self*.column1).grid(*row*=0, *column*=1, *padx*=5, *pady*=5)

        tk.Label(insert\_frame, *text*="Tác giả:").grid(*row*=1, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(insert\_frame, *textvariable*=*self*.column2).grid(*row*=1, *column*=1, *padx*=5, *pady*=5)

        tk.Label(insert\_frame, *text*="Năm xuất bản:").grid(*row*=2, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(insert\_frame, *textvariable*=*self*.column3).grid(*row*=2, *column*=1, *padx*=5, *pady*=5)

        tk.Label(insert\_frame, *text*="Thể loại:").grid(*row*=3, *column*=0, *padx*=5, *pady*=5)

        tk.Entry(insert\_frame, *textvariable*=*self*.column4).grid(*row*=3, *column*=1, *padx*=5, *pady*=5)

        tk.Button(insert\_frame, *text*="Thêm sách", *command*=*self*.insert\_data).grid(*row*=4, *columnspan*=2, *pady*=10)

        tk.Button(insert\_frame, *text*="Tải lại dữ liệu", *command*=*self*.load\_data).grid(*row*=5, *columnspan*=2, *pady*=10)

        # Treeview for displaying data

*self*.data\_tree = ttk.Treeview(*self*.tab, *columns*=("ID", "Tên sách", "Tác giả", "Năm xuất bản", "Thể loại"), *show*="headings", *height*=10)

*self*.data\_tree.column("ID", *width*=100, *anchor*="center")

*self*.data\_tree.column("Tên sách", *width*=200, *anchor*="center")

*self*.data\_tree.column("Tác giả", *width*=200, *anchor*="center")

*self*.data\_tree.column("Năm xuất bản", *width*=150, *anchor*="center")

*self*.data\_tree.column("Thể loại", *width*=150, *anchor*="center")

*self*.data\_tree.heading("ID", *text*="ID")

*self*.data\_tree.heading("Tên sách", *text*="Tên sách")

*self*.data\_tree.heading("Tác giả", *text*="Tác giả")

*self*.data\_tree.heading("Năm xuất bản", *text*="Năm xuất bản")

*self*.data\_tree.heading("Thể loại", *text*="Thể loại")

*self*.data\_tree.pack(*side*="bottom", *fill*="both", *expand*=True)

    def load\_data(*self*):

        try:

            query = sql.SQL("SELECT \* FROM {}").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(query)

            rows = *self*.app.cur.fetchall()

            # Clear previous data

            for item in *self*.data\_tree.get\_children():

*self*.data\_tree.delete(item)

            # Insert new data into Treeview

            for row in rows:

*self*.data\_tree.insert("", "end", *values*=row)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi tải dữ liệu: {e}")

    def insert\_data(*self*):

        try:

            if not *self*.column1.get() or not *self*.column2.get():

                messagebox.showwarning("Cảnh báo", "Vui lòng điền đầy đủ thông tin sách!")

                return

            add\_query = sql.SQL("INSERT INTO {} (ten\_sach, tac\_gia, nxb, the\_loai) VALUES (%s, %s, %s, %s)").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(add\_query, (*self*.column1.get(), *self*.column2.get(), *self*.column3.get(), *self*.column4.get()))

*self*.app.conn.commit()

            messagebox.showinfo("Thành công", "Sách đã được thêm thành công!")

*self*.load\_data()  # Refresh data after addition

        except Exception as e:

*self*.app.conn.rollback()  # Rollback the transaction on error

            messagebox.showerror("Lỗi", f"Lỗi thêm sách: {e}")

class UpdateBooks:

    def \_\_init\_\_(*self*, *tab*, *app*):

*self*.tab = *tab*

*self*.app = *app*

        # Insert section

        update\_frame = tk.Frame(*self*.tab)

        update\_frame.pack(*pady*=10)

*self*.column1 = tk.StringVar()  # ID (disabled)

*self*.column2 = tk.StringVar()  # Tên sách

*self*.column3 = tk.StringVar()  # Tác giả

*self*.column4 = tk.StringVar()  # NXB

*self*.column5 = tk.StringVar()  # Thể loại

        tk.Label(update\_frame, *text*="ID:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

*self*.id\_entry = tk.Entry(update\_frame, *textvariable*=*self*.column1, *state*='disabled')  # ID disabled

*self*.id\_entry.grid(*row*=0, *column*=1, *padx*=5, *pady*=5)

        tk.Label(update\_frame, *text*="Tên sách:").grid(*row*=1, *column*=0, *padx*=5, *pady*=5)

*self*.name\_entry = tk.Entry(update\_frame, *textvariable*=*self*.column2)

*self*.name\_entry.grid(*row*=1, *column*=1, *padx*=5, *pady*=5)

        tk.Label(update\_frame, *text*="Tác giả:").grid(*row*=2, *column*=0, *padx*=5, *pady*=5)

*self*.author\_entry = tk.Entry(update\_frame, *textvariable*=*self*.column3)

*self*.author\_entry.grid(*row*=2, *column*=1, *padx*=5, *pady*=5)

        tk.Label(update\_frame, *text*="Năm xuất bản:").grid(*row*=3, *column*=0, *padx*=5, *pady*=5)

*self*.nxb\_entry = tk.Entry(update\_frame, *textvariable*=*self*.column4)

*self*.nxb\_entry.grid(*row*=3, *column*=1, *padx*=5, *pady*=5)

        tk.Label(update\_frame, *text*="Thể loại:").grid(*row*=4, *column*=0, *padx*=5, *pady*=5)

*self*.the\_loai\_entry = tk.Entry(update\_frame, *textvariable*=*self*.column5)

*self*.the\_loai\_entry.grid(*row*=4, *column*=1, *padx*=5, *pady*=5)

        tk.Button(update\_frame, *text*="Cập nhật dữ liệu", *command*=*self*.update\_data).grid(*row*=5, *columnspan*=2, *pady*=10)

        tk.Button(update\_frame, *text*="Tải lại dữ liệu", *command*=*self*.load\_data).grid(*row*=6, *columnspan*=2, *pady*=10)

        # Treeview for displaying data

*self*.data\_tree = ttk.Treeview(*self*.tab, *columns*=("ID", "Tên sách", "Tác giả", "Năm xuất bản", "Thể loại"), *show*="headings", *height*=10)

*self*.data\_tree.column("ID", *width*=100, *anchor*="center")

*self*.data\_tree.column("Tên sách", *width*=200, *anchor*="center")

*self*.data\_tree.column("Tác giả", *width*=200, *anchor*="center")

*self*.data\_tree.column("Năm xuất bản", *width*=150, *anchor*="center")

*self*.data\_tree.column("Thể loại", *width*=150, *anchor*="center")

*self*.data\_tree.heading("ID", *text*="ID")

*self*.data\_tree.heading("Tên sách", *text*="Tên sách")

*self*.data\_tree.heading("Tác giả", *text*="Tác giả")

*self*.data\_tree.heading("Năm xuất bản", *text*="Năm xuất bản")

*self*.data\_tree.heading("Thể loại", *text*="Thể loại")

*self*.data\_tree.pack(*side*="bottom", *fill*="both", *expand*=True)

        # Bind Treeview click event

*self*.data\_tree.bind("<ButtonRelease-1>", *self*.select\_item)

    def load\_data(*self*):

        try:

            query = sql.SQL("SELECT \* FROM {}").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(query)

            rows = *self*.app.cur.fetchall()

            # Clear previous data

            for item in *self*.data\_tree.get\_children():

*self*.data\_tree.delete(item)

            # Insert new data into Treeview

            for row in rows:

*self*.data\_tree.insert("", "end", *values*=row)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi tải dữ liệu: {e}")

    def select\_item(*self*, *event*):

        # Get selected item from Treeview

        selected\_item = *self*.data\_tree.selection()[0]

        item\_values = *self*.data\_tree.item(selected\_item, 'values')

        # Set the selected values into the Entry widgets

*self*.column1.set(item\_values[0])  # ID

*self*.column2.set(item\_values[1])  # Tên sách

*self*.column3.set(item\_values[2])  # Tác giả

*self*.column4.set(item\_values[3])  # NXB

*self*.column5.set(item\_values[4])  # Thể loại

    def update\_data(*self*):

        try:

            if not *self*.column1.get():

                messagebox.showwarning("Cảnh báo", "Vui lòng nhập ID của sách cần sửa!")

                return

            update\_query = sql.SQL("UPDATE {} SET ten\_sach=%s, tac\_gia=%s, nxb=%s, the\_loai=%s WHERE id=%s").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(update\_query, (*self*.column2.get(), *self*.column3.get(), *self*.column4.get(), *self*.column5.get(), *self*.column1.get()))

            if *self*.app.cur.rowcount == 0:  # Kiểm tra số bản ghi bị ảnh hưởng

                messagebox.showwarning("Cảnh báo", "Không có ID nào trùng khớp để sửa!")

            else:

*self*.app.conn.commit()

                messagebox.showinfo("Thành công", "Sách đã được sửa thành công!")

*self*.load\_data()  # Refresh data after update

        except Exception as e:

*self*.app.conn.rollback()  # Rollback the transaction on error

            messagebox.showerror("Lỗi", f"Lỗi sửa sách: {e}")

class DeleteBooks:

    def \_\_init\_\_(*self*, *tab*, *app*):

*self*.tab = *tab*

*self*.app = *app*

        delete\_frame = tk.Frame(*self*.tab)

        delete\_frame.pack(*pady*=10)

*self*.id\_var = tk.StringVar()  # ID

        tk.Label(delete\_frame, *text*="ID sách để xóa:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

*self*.id\_entry = tk.Entry(delete\_frame, *textvariable*=*self*.id\_var)

*self*.id\_entry.grid(*row*=0, *column*=1, *padx*=5, *pady*=5)

        tk.Button(delete\_frame, *text*="Xóa sách", *command*=*self*.delete\_data).grid(*row*=1, *columnspan*=2, *pady*=10)

        tk.Button(delete\_frame, *text*="Tải lại dữ liệu", *command*=*self*.load\_data).grid(*row*=2, *columnspan*=2, *pady*=10)

        # Treeview for displaying data

*self*.data\_tree = ttk.Treeview(*self*.tab, *columns*=("ID", "Tên sách", "Tác giả", "Năm xuất bản", "Thể loại"), *show*="headings", *height*=10)

*self*.data\_tree.column("ID", *width*=100, *anchor*="center")

*self*.data\_tree.column("Tên sách", *width*=200, *anchor*="center")

*self*.data\_tree.column("Tác giả", *width*=200, *anchor*="center")

*self*.data\_tree.column("Năm xuất bản", *width*=150, *anchor*="center")

*self*.data\_tree.column("Thể loại", *width*=150, *anchor*="center")

*self*.data\_tree.heading("ID", *text*="ID")

*self*.data\_tree.heading("Tên sách", *text*="Tên sách")

*self*.data\_tree.heading("Tác giả", *text*="Tác giả")

*self*.data\_tree.heading("Năm xuất bản", *text*="Năm xuất bản")

*self*.data\_tree.heading("Thể loại", *text*="Thể loại")

*self*.data\_tree.pack(*side*="bottom", *fill*="both", *expand*=True)

        # Bind double-click event

*self*.data\_tree.bind("<ButtonRelease-1>", *self*.on\_treeview\_select)

    def on\_treeview\_select(*self*, *event*):

        # Get selected item

        selected\_item = *self*.data\_tree.selection()

        if selected\_item:

            item\_values = *self*.data\_tree.item(selected\_item, "values")

            if item\_values:

*self*.id\_var.set(item\_values[0])  # Set ID vào entry

    def load\_data(*self*):

        try:

            query = sql.SQL("SELECT \* FROM {}").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(query)

            rows = *self*.app.cur.fetchall()

            # Clear previous data

            for item in *self*.data\_tree.get\_children():

*self*.data\_tree.delete(item)

            # Insert new data into Treeview

            for row in rows:

*self*.data\_tree.insert("", "end", *values*=row)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi tải dữ liệu: {e}")

    def delete\_data(*self*):

        try:

            delete\_query = sql.SQL("DELETE FROM {} WHERE id=%s").format(sql.Identifier(*self*.app.table\_name.get()))

*self*.app.cur.execute(delete\_query, (*self*.id\_var.get(),))

            if *self*.app.cur.rowcount == 0:  # Kiểm tra số bản ghi bị ảnh hưởng

                messagebox.showwarning("Cảnh báo", "Không có ID nào trùng khớp!")

            else:

*self*.app.conn.commit()

                messagebox.showinfo("Thành công", "Sách đã được xóa thành công!")

*self*.load\_data()  # Refresh data after deletion

        except Exception as e:

*self*.app.conn.rollback()  # Rollback the transaction on error

            messagebox.showerror("Lỗi", f"Lỗi xóa sách: {e}")

class DisconnectDB:

    def \_\_init\_\_(*self*, *tab*, *app*):

*self*.tab = *tab*

*self*.app = *app*

        disconnect\_frame = tk.Frame(*self*.tab)

        disconnect\_frame.pack(*pady*=10)

        tk.Label(disconnect\_frame, *text*="Ngắt kết nối cơ sở dữ liệu:").grid(*row*=0, *column*=0, *padx*=5, *pady*=5)

        tk.Button(disconnect\_frame, *text*="Ngắt kết nối", *command*=*self*.disconnect\_data).grid(*row*=1, *columnspan*=2, *pady*=10)

    def disconnect\_data(*self*):

        try:

            if hasattr(*self*.app, 'conn') and *self*.app.conn is not None:

*self*.app.cur.close()

*self*.app.conn.close()

                messagebox.showinfo("Thành công", "Đã ngắt kết nối cơ sở dữ liệu!")

                # Quay về tab đăng nhập

*self*.app.tab\_control.select(*self*.app.login\_tab)

                # Ẩn các tab khác sau khi ngắt kết nối

*self*.app.tab\_control.hide(*self*.app.search\_tab)

*self*.app.tab\_control.hide(*self*.app.add\_tab)

*self*.app.tab\_control.hide(*self*.app.update\_tab)

*self*.app.tab\_control.hide(*self*.app.delete\_tab)

        except Exception as e:

            messagebox.showerror("Lỗi", f"Lỗi ngắt kết nối: {e}")

File main.py : dung để xử lí toàn bộ code và chạy giao diện

import tkinter as tk

from tkinter import ttk

from app import DatabaseApp

def main():

    root = tk.Tk()

    app = DatabaseApp(root)

    root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**4. Git-hub**

Link: [vuviet1207/LapTrinhPythonNangCao\_BaiTap2](https://github.com/vuviet1207/LapTrinhPythonNangCao_BaiTap2)