

TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC
LẬP TRÌNH PYTHON NÂNG CAO

Chủ đề:

BÁO CÁO BÀI TẬP 2

SVTH: NGUYỄN PHƯỚC ĐẠI_2274802010143

LỚP: 241_71ITSE31003_02

TP. Hồ Chí Minh – 11/2024

I – Giao diện

Trong bài tập này, em đã xây dựng một giao diện làm việc với chức năng quản lý sinh viên với các chức năng thêm – xóa dữ liệu.

The screenshot shows a web application titled "Quản lý Sinh viên". It features three main panels on the left for database management and data entry, and a large table on the right for displaying student records.

- Kết nối Database:** Fields for "Tên Database" (test), "Người dùng" (postgres), "Mật khẩu" (***), "Host" (localhost), and "Port" (5432). A green "Kết nối" button is at the bottom.
- Truy vấn dữ liệu:** A field for "Tên bảng" (sinhvien) and a purple "Tải dữ liệu" button.
- Thêm dữ liệu:** Fields for "ID", "Họ", and "Tên". Green "Thêm dữ liệu" and red "Xóa" buttons are at the bottom.
- Table:** A table with columns "ID", "Họ", and "Tên". The table body is currently empty.

II – Chức năng

1. File UI cho ứng dụng

a. Cài đặt thư viện để sử dụng

```
import tkinter as tk
```

```
from tkinter import ttk, messagebox
```

```
import ttkbootstrap as ttkb
```

⇒ Sử dụng thư viện tkinter và ttkbootstrap để cài giao diện cho ứng dụng

b. Tạo lớp DatabaseUI để dễ quản lý

```

def __init__(self, root, db_operations):
    self.root = root
    self.root.title("Quản lý Sinh viên")
    self.root.geometry("1200x800")
    self.style = ttk.Style(theme="cosmo")
    self.db = db_operations

    # Cài đặt các Data để mặc định
    self.db_name = tk.StringVar(value='test')
    self.user = tk.StringVar(value='postgres')
    self.password = tk.StringVar(value='123')
    self.host = tk.StringVar(value='localhost')
    self.port = tk.StringVar(value='5432')
    self.table_name = tk.StringVar(value='sinhvien')

    # Lấy các giá trị
    self.id_var = tk.StringVar()
    self.column1 = tk.StringVar()
    self.column2 = tk.StringVar()
    self.selected_item = None

    self.create_widgets()

```

Hàm này định nghĩa cửa sổ và các giá trị mặc định khi mở chương trình.

```

def create_widgets(self):
    # Main container
    main_frame = ttk.Frame(self.root, padding="20 20 20 20")
    main_frame.pack(fill=tk.BOTH, expand=True)

    # Left panel
    left_panel = ttk.Frame(main_frame, width=300)
    left_panel.pack(side=tk.LEFT, fill=tk.Y, padx=(0, 10))
    left_panel.pack_propagate(False)

    # Create connection section
    self._create_connection_section(left_panel)

    # Create action section
    self._create_action_section(left_panel)

    # Create data display section
    self._create_data_display(main_frame)

```

Hàm này định nghĩa vị trí cho các panel trong cửa sổ chương trình

```

def _create_connection_section(self, parent):
    connection_frame = ttk.LabelFrame(parent, text="Kết nối Database", padding="10 10 10 10")
    connection_frame.pack(fill=tk.X, pady=(0, 20))

    fields = [
        ("Tên Database:", self.db_name),
        ("Người dùng:", self.user),
        ("Mật khẩu (parameter) self: Self@DatabaseUI", self.password),
        ("Host:", self.host),
        ("Port:", self.port)]

    for i, (label, var) in enumerate(fields):
        ttk.Label(connection_frame, text=label).grid(row=i, column=0, sticky="e", padx=(0, 10), pady=5)
        entry = ttk.Entry(connection_frame, textvariable=var)
        entry.grid(row=i, column=1, sticky="we", pady=5)
        if label == "Mật khẩu:":
            entry.config(show="*")

    ttk.Button(connection_frame, text="Kết nối", command=self.connect_db,
               style="success.TButton").grid(row=len(fields), columnspan=2, pady=(10, 0))

```

Hàm này định nghĩa về nhập Data vào và cửa sổ hiển thị.

```

def _create_action_section(self, parent):
    action_frame = ttk.Frame(parent)
    action_frame.pack(fill=tk.BOTH, expand=True)

    # Query section
    query_frame = ttk.LabelFrame(action_frame, text="Truy vấn dữ liệu", padding="10 10 10 10")
    query_frame.pack(fill=tk.X, pady=(0, 20))

    ttk.Label(query_frame, text="Tên bảng:").pack(anchor="w", pady=(0, 5))
    ttk.Entry(query_frame, textvariable=self.table_name).pack(fill=tk.X, pady=(0, 10))
    ttk.Button(query_frame, text="Tải dữ liệu", command=self.load_data,
               style="info.TButton").pack(fill=tk.X)

    # Insert section
    insert_frame = ttk.LabelFrame(action_frame, text="Thêm dữ liệu", padding="10 10 10 10")
    insert_frame.pack(fill=tk.X)

    fields = [
        ("ID:", self.id_var),
        ("Họ:", self.column1),
        ("Tên:", self.column2)]

    for label, var in fields:
        ttk.Label(insert_frame, text=label).pack(anchor="w", pady=(0, 5))
        ttk.Entry(insert_frame, textvariable=var).pack(fill=tk.X, pady=(0, 10))

    button_frame = ttk.Frame(insert_frame)
    button_frame.pack(fill=tk.X, pady=(10, 0))

    ttk.Button(button_frame, text="Thêm dữ liệu", command=self.insert_data,
               style="success.TButton").pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(0, 5))
    ttk.Button(button_frame, text="Xóa", command=self.delete_data,
               style="danger.TButton").pack(side=tk.LEFT, fill=tk.X, expand=True, padx=(5, 0))

```

Hàm này định nghĩa về các nút trong label dưới và hiển thị chúng

```

def _create_data_display(self, parent):
    right_panel = ttk.Frame(parent)
    right_panel.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)
    tree_frame = ttk.Frame(right_panel)
    tree_frame.pack(fill=tk.BOTH, expand=True)
    # Treeview
    self.tree = ttk.Treeview(tree_frame, columns=("ID", "Họ", "Tên"), show="headings")
    self.tree.heading("ID", text="ID")
    self.tree.heading("Họ", text="Họ")
    self.tree.heading("Tên", text="Tên")
    self.tree.column("ID", width=50, anchor="center")
    self.tree.column("Họ", width=200, anchor="w")
    self.tree.column("Tên", width=300, anchor="w")
    self.tree.bind('<<TreeviewSelect>>', self.on_select)
    # Styling
    self.style.configure("Treeview", background="#E1E1E1",
                        fieldbackground="#E1E1E1", foreground="black")
    self.style.map('Treeview', background=[('selected', '#347083')])
    # Scrollbars
    vsb = ttk.Scrollbar(tree_frame, orient="vertical", command=self.tree.yview)
    hsb = ttk.Scrollbar(right_panel, orient="horizontal", command=self.tree.xview)
    self.tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)

    # Pack everything
    self.tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
    vsb.pack(side=tk.RIGHT, fill=tk.Y)
    hsb.pack(fill=tk.X)

```

Hàm này định nghĩa màu sắc và định dạng hiển thị của các Data

```

def connect_db(self):
    success, message = self.db.connect(
        self.db_name.get(),
        self.user.get(),
        self.password.get(),
        self.host.get(),
        self.port.get()
    )
    if success:
        messagebox.showinfo("Thành công", message)
    else:
        messagebox.showerror("Lỗi", message)

def load_data(self):
    success, result = self.db.fetch_all_data(self.table_name.get())
    if success:
        # Clear existing data
        for i in self.tree.get_children():
            self.tree.delete(i)
        # Insert new data
        for row in result:
            self.tree.insert("", "end", values=row)
    else:
        messagebox.showerror("Lỗi", result)

```

Hàm hiển thị thông báo nếu kết nối thành công

```
def load_data(self):
    success, result = self.db.fetch_all_data(self.table_name.get())
    if success:
        # Clear existing data
        for i in self.tree.get_children():
            self.tree.delete(i)
        # Insert new data
        for row in result:
            self.tree.insert("", "end", values=row)
    else:
        messagebox.showerror("Lỗi", result)
```

Hàm này xử lý thông báo khi kết nối

```
def insert_data(self):
    success, message = self.db.insert_data(
        self.table_name.get(),
        self.id_var.get(),
        self.column1.get(),
        self.column2.get()
    )
    if success:
        self.tree.insert("", "end", values=(self.id_var.get(), self.column1.get(), self.column2.get()))
        self.clear_form()
        messagebox.showinfo("Thành công", message)
    else:
        messagebox.showerror("Lỗi", message)

def delete_data(self):
    if not self.selected_item:
        messagebox.showwarning("Cảnh báo", "Vui lòng chọn một dòng để xóa!")
        return

    if messagebox.askyesno("Xác nhận", "Bạn có chắc chắn muốn xóa dữ liệu này?"):
        selected_id = self.tree.item(self.selected_item)['values'][0]
        success, message = self.db.delete_data(self.table_name.get(), selected_id)

        if success:
            self.tree.delete(self.selected_item)
            self.clear_form()
            messagebox.showinfo("Thành công", message)
        else:
            messagebox.showerror("Lỗi", message)
```

Hàm này hiển thị thông báo khi thêm/xóa data

```

def on_select(self, event):
    selected_items = self.tree.selection()
    if selected_items:
        self.selected_item = selected_items[0]
        values = self.tree.item(self.selected_item)['values']
        if values:
            self.id_var.set(values[0])
            self.column1.set(values[1])
            self.column2.set(values[2])

def clear_form(self):
    self.id_var.set("")
    self.column1.set("")
    self.column2.set("")
    self.selected_item = None

```

2 Hàm này sẽ xử lý hiển thị chọn các giá trị và xóa chúng

2. File db_setting.py

Ở file này sẽ xử lý những gì làm việc với CSDL

```
def connect(self, dbname, user, password, host, port):
    """Establish database connection"""
    try:
        self.conn = psycopg2.connect(
            dbname=dbname,
            user=user,
            password=password,
            host=host,
            port=port
        )
        self.cur = self.conn.cursor()
        return True, "Kết nối đến cơ sở dữ liệu thành công!"
    except Exception as e:
        return False, f"Lỗi khi kết nối đến cơ sở dữ liệu: {e}"
```

Hàm xử lý kết nối

```
def fetch_all_data(self, table_name):
    """Fetch all data from specified table"""
    try:
        query = sql.SQL("SELECT * FROM {}").format(sql.Identifier(table_name))
        self.cur.execute(query)
        return True, self.cur.fetchall()
    except Exception as e:
        return False, f"Lỗi khi tải dữ liệu: {e}"
```

Xử lý lỗi tải dữ liệu

```
def insert_data(self, table_name, id_value, ho_value, ten_value):
    """Insert new record into database"""
    try:
        insert_query = sql.SQL("INSERT INTO {} (id, ho, ten) VALUES (%s, %s, %s)").format(
            sql.Identifier(table_name))
        self.cur.execute(insert_query, (id_value, ho_value, ten_value))
        self.conn.commit()
        return True, "Dữ liệu đã được thêm thành công!"
    except Exception as e:
        self.conn.rollback()
        return False, f"Lỗi khi thêm dữ liệu: {e}"
```

Xử lý khi thêm dữ liệu


```
def delete_data(self, table_name, id_value):
    """Delete record from database"""
    try:
        delete_query = sql.SQL("DELETE FROM {} WHERE id = %s").format(sql.Identifier(table_name))
        self.cur.execute(delete_query, (id_value,))
        self.conn.commit()
        return True, "Dữ liệu đã được xóa thành công!"
    except Exception as e:
        self.conn.rollback()
        return False, f"Lỗi khi xóa dữ liệu: {e}"
```

Xử lý khi xóa dữ liệu

3. File Main.py

```
import ttkbootstrap as ttkb
from db_setting import DatabaseConnection
from UI import DatabaseUI

def main():
    root = ttkb.Window(themename="cosmo")
    db = DatabaseConnection()
    app = DatabaseUI(root, db)
    root.mainloop()
    db.close()

if __name__ == "__main__":
    main()
```

Hàm main để chạy chương trình.

III – Code đầy đủ

1. File UI.py

app_ui.py

import tkinter as tk

from tkinter import ttk, messagebox

import ttkbootstrap as ttkb

class DatabaseUI:

def __init__(self, root, db_operations):

self.root = root

self.root.title("Quản lý Sinh viên")

```
self.root.geometry("1200x800")
self.style = ttk.Style(theme="cosmo")
self.db = db_operations
```

```
# Cài đặt các Data để mặc định
```

```
self.db_name = tk.StringVar(value='test')
self.user = tk.StringVar(value='postgres')
self.password = tk.StringVar(value='123')
self.host = tk.StringVar(value='localhost')
self.port = tk.StringVar(value='5432')
self.table_name = tk.StringVar(value='sinhvien')
```

```
# Lấy các giá trị
```

```
self.id_var = tk.StringVar()
self.column1 = tk.StringVar()
self.column2 = tk.StringVar()
self.selected_item = None
```

```
self.create_widgets()
```

```
def create_widgets(self):
```

```
    # Main container
```

```
    main_frame = ttk.Frame(self.root, padding="20 20 20 20")
    main_frame.pack(fill=tk.BOTH, expand=True)
```

```
    # Left panel
```

```
    left_panel = ttk.Frame(main_frame, width=300)
```

```

left_panel.pack(side=tk.LEFT, fill=tk.Y, padx=(0, 10))
left_panel.pack_propagate(False)

# Create connection section
self._create_connection_section(left_panel)

# Create action section
self._create_action_section(left_panel)

# Create data display section
self._create_data_display(main_frame)

def _create_connection_section(self, parent):
    connection_frame = ttk.LabelFrame(parent, text="Kết nối Database",
padding="10 10 10 10")
    connection_frame.pack(fill=tk.X, pady=(0, 20))

    fields = [
        ("Tên Database:", self.db_name),
        ("Người dùng:", self.user),
        ("Mật khẩu:", self.password),
        ("Host:", self.host),
        ("Port:", self.port)
    ]

    for i, (label, var) in enumerate(fields):

```

```
        ttk.Label(connection_frame, text=label).grid(row=i, column=0, sticky="e",
padx=(0, 10), pady=5)
        entry = ttk.Entry(connection_frame, textvariable=var)
        entry.grid(row=i, column=1, sticky="we", pady=5)
        if label == "Mật khẩu:":
            entry.config(show="*")
```

```
        ttk.Button(connection_frame, text="Kết nối", command=self.connect_db,
                    style="success.TButton").grid(row=len(fields), columnspan=2,
pady=(10, 0))
```

```
def _create_action_section(self, parent):
```

```
    action_frame = ttk.Frame(parent)
```

```
    action_frame.pack(fill=tk.BOTH, expand=True)
```

```
    # Query section
```

```
    query_frame = ttk.LabelFrame(action_frame, text="Truy vấn dữ liệu",
padding="10 10 10 10")
```

```
    query_frame.pack(fill=tk.X, pady=(0, 20))
```

```
    ttk.Label(query_frame, text="Tên bảng:").pack(anchor="w", pady=(0, 5))
```

```
    ttk.Entry(query_frame, textvariable=self.table_name).pack(fill=tk.X, pady=(0,
10))
```

```
    ttk.Button(query_frame, text="Tải dữ liệu", command=self.load_data,
                style="info.TButton").pack(fill=tk.X)
```

```
    # Insert section
```

```

insert_frame = ttk.LabelFrame(action_frame, text="Thêm dữ liệu",
padding="10 10 10 10")
insert_frame.pack(fill=tk.X)

fields = [
    ("ID:", self.id_var),
    ("Họ:", self.column1),
    ("Tên:", self.column2)
]

for label, var in fields:
    ttk.Label(insert_frame, text=label).pack(anchor="w", pady=(0, 5))
    ttk.Entry(insert_frame, textvariable=var).pack(fill=tk.X, pady=(0, 10))

button_frame = ttk.Frame(insert_frame)
button_frame.pack(fill=tk.X, pady=(10, 0))

ttk.Button(button_frame, text="Thêm dữ liệu", command=self.insert_data,
            style="success.TButton").pack(side=tk.LEFT, fill=tk.X, expand=True,
padx=(0, 5))

ttk.Button(button_frame, text="Xóa", command=self.delete_data,
            style="danger.TButton").pack(side=tk.LEFT, fill=tk.X, expand=True,
padx=(5, 0))

def _create_data_display(self, parent):
    right_panel = ttk.Frame(parent)
    right_panel.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

```

```
tree_frame = ttk.Frame(right_panel)
tree_frame.pack(fill=tk.BOTH, expand=True)

# Treeview

self.tree = ttk.Treeview(tree_frame, columns=("ID", "Họ", "Tên"),
show="headings")

self.tree.heading("ID", text="ID")
self.tree.heading("Họ", text="Họ")
self.tree.heading("Tên", text="Tên")

self.tree.column("ID", width=50, anchor="center")
self.tree.column("Họ", width=200, anchor="w")
self.tree.column("Tên", width=300, anchor="w")

self.tree.bind('<<TreeviewSelect>>', self.on_select)

# Styling

self.style.configure("Treeview", background="#E1E1E1",
                      fieldbackground="#E1E1E1", foreground="black")
self.style.map("Treeview", background=[('selected', '#347083')])

# Scrollbars

vsb = ttk.Scrollbar(tree_frame, orient="vertical", command=self.tree.yview)
hsb = ttk.Scrollbar(right_panel, orient="horizontal",
command=self.tree.xview)

self.tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
```

```
# Pack everything
self.tree.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
vsb.pack(side=tk.RIGHT, fill=tk.Y)
hsb.pack(fill=tk.X)
```

```
def connect_db(self):
    success, message = self.db.connect(
        self.db_name.get(),
        self.user.get(),
        self.password.get(),
        self.host.get(),
        self.port.get()
    )
    if success:
        messagebox.showinfo("Thành công", message)
    else:
        messagebox.showerror("Lỗi", message)
```

```
def load_data(self):
    success, result = self.db.fetch_all_data(self.table_name.get())
    if success:
        # Clear existing data
        for i in self.tree.get_children():
            self.tree.delete(i)
        # Insert new data
        for row in result:
```

```

        self.tree.insert("", "end", values=row)
    else:
        messagebox.showerror("Lỗi", result)

def insert_data(self):
    success, message = self.db.insert_data(
        self.table_name.get(),
        self.id_var.get(),
        self.column1.get(),
        self.column2.get()
    )
    if success:
        self.tree.insert("", "end", values=(self.id_var.get(), self.column1.get(),
self.column2.get()))
        self.clear_form()
        messagebox.showinfo("Thành công", message)
    else:
        messagebox.showerror("Lỗi", message)

def delete_data(self):
    if not self.selected_item:
        messagebox.showwarning("Cảnh báo", "Vui lòng chọn một dòng để xóa!")
        return

    if messagebox.askyesno("Xác nhận", "Bạn có chắc chắn muốn xóa dữ liệu
này?"):
        selected_id = self.tree.item(self.selected_item)['values'][0]

```



```

success, message = self.db.delete_data(self.table_name.get(), selected_id)

if success:
    self.tree.delete(self.selected_item)
    self.clear_form()
    messagebox.showinfo("Thành công", message)
else:
    messagebox.showerror("Lỗi", message)

def on_select(self, event):
    selected_items = self.tree.selection()
    if selected_items:
        self.selected_item = selected_items[0]
        values = self.tree.item(self.selected_item)['values']
        if values:
            self.id_var.set(values[0])
            self.column1.set(values[1])
            self.column2.set(values[2])

def clear_form(self):
    self.id_var.set("")
    self.column1.set("")
    self.column2.set("")
    self.selected_item = None

```

2. File db_setting.py

```
# db_operations.py
```

```
import psycopg2
```

```
from psycopg2 import sql
```

```
class DatabaseConnection:
```

```
    def __init__(self):
```

```
        self.conn = None
```

```
        self.cur = None
```

```
    def connect(self, dbname, user, password, host, port):
```

```
        """Establish database connection"""
```

```
        try:
```

```
            self.conn = psycopg2.connect(
```

```
                dbname=dbname,
```

```
                user=user,
```

```
                password=password,
```

```
                host=host,
```

```
                port=port
```

```
            )
```

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

```
            return True, "Kết nối đến cơ sở dữ liệu thành công!"
```

```
        except Exception as e:
```

```
            return False, f"Lỗi khi kết nối đến cơ sở dữ liệu: {e}"
```

```
    def fetch_all_data(self, table_name):
```

```
        """Fetch all data from specified table"""
```

```

try:
    query = sql.SQL("SELECT * FROM
{ }").format(sql.Identifier(table_name))
    self.cur.execute(query)
    return True, self.cur.fetchall()
except Exception as e:
    return False, f"Lỗi khi tải dữ liệu: {e}"

def insert_data(self, table_name, id_value, ho_value, ten_value):
    """Insert new record into database"""
    try:
        insert_query = sql.SQL("INSERT INTO { } (id, ho, ten) VALUES (%s, %s,
%s)").format(
            sql.Identifier(table_name))
        self.cur.execute(insert_query, (id_value, ho_value, ten_value))
        self.conn.commit()
        return True, "Dữ liệu đã được thêm thành công!"
    except Exception as e:
        self.conn.rollback()
        return False, f"Lỗi khi thêm dữ liệu: {e}"

def delete_data(self, table_name, id_value):
    """Delete record from database"""
    try:
        delete_query = sql.SQL("DELETE FROM { } WHERE id =
%s").format(sql.Identifier(table_name))
        self.cur.execute(delete_query, (id_value,))

```

```
        self.conn.commit()

        return True, "Dữ liệu đã được xóa thành công!"
    except Exception as e:
        self.conn.rollback()

        return False, f"Lỗi khi xóa dữ liệu: {e}"

def close(self):
    """Close database connection"""
    if self.cur:
        self.cur.close()
    if self.conn:
        self.conn.close()
```

3. Hàm Main.py

```
# main.py

import ttkbootstrap as ttkb
from db_setting import DatabaseConnection
from UI import DatabaseUI

def main():

    root = ttkb.Window(themename="cosmo")

    db = DatabaseConnection()

    app = DatabaseUI(root, db)

    root.mainloop()

    db.close()

if __name__ == "__main__":
    main()
```

IV – Link Github:

<https://github.com/SaikySu/PythonNC>