TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**LẬP TRÌNH PYTHON NÂNG CAO**

NGÀNH: CÔNG NGHỆ THÔNG TIN

***Đồ án:***

**QUẢN LÝ SINH VIÊN**

**SVTH: Nguyễn Thành Nhân**

**MSSV: 2274802010600**

**GVHD: Huỳnh Thái Học**

Tp. Hồ Chí Minh – năm 2024

LỜI CẢM ƠN

Viết một báo cáo đồ án môn học là một trong những việc khó nhất mà chúng em phải  
hoàn thành trong quá trình học một môn học. Trong quá trình thực hiện đề tài chúng  
em đã gặp rất nhiều khó khăn và bỡ ngỡ. Nếu không có những sự giúp đỡ và lời động  
viên chân thành của nhiều người có lẽ chúng em khó có thể hoàn thành tốt tiểu luận  
này. Đầu tiên chúng em xin gửi lời biết ơn chân thành đến thầy NV A,  
người trực tiếp hướng dẫn chúng em hoàn thành tiểu luận này.  
Những ý kiến đóng góp của thầy là vô cùng hữu ích, nó giúp chúng em nhận ra các  
khuyết điểm của đồ án. Cảm ơn thầy và các bạn trường Đại học Văn Lang là những  
người đã cùng nhóm em sát cánh và trải nghiệm để hoàn thành đồ án môn học.  
Nhóm thực hiện báo cáo

MỤC LỤC

LỜI CẢM ƠN **....................................................................................................... 8**CHƯƠNG 1: MỞ ĐẦU**............................................................................................... 10**1.1 Lý do chọn đề tài **.................................................................................................... 11**1.2 Mục tiêu của đề tài **.......................................................................................... 12**

### 1. **Giới Thiệu**

Bài toán đặt ra là xây dựng một hệ thống quản lý sinh viên đơn giản, sử dụng Python và thư viện Tkinter để tạo giao diện người dùng (GUI). Hệ thống cho phép quản lý thông tin sinh viên như mã số sinh viên (MSSV), họ tên, lớp, và ngày sinh. Dữ liệu được lưu trữ trong cơ sở dữ liệu PostgreSQL để đảm bảo tính bền vững và khả năng truy xuất dữ liệu.

### 2. **Mục Tiêu**

* Tạo ra một ứng dụng giúp quản lý thông tin sinh viên một cách thuận tiện và dễ dàng.
* Sử dụng Tkinter để xây dựng giao diện thân thiện với người dùng.
* Sử dụng PostgreSQL để quản lý cơ sở dữ liệu, đảm bảo tính ổn định và bảo mật.

### 3. **Phân Tích Yêu Cầu**

#### 3.1 **Yêu Cầu Chức Năng**

* **Đăng nhập**: Người dùng cần đăng nhập với tên tài khoản và mật khẩu để truy cập hệ thống.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động**

* **Thêm sinh viên**: Người dùng có thể thêm mới thông tin sinh viên vào cơ sở dữ liệu.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

* **Xem danh sách sinh viên**: Hiển thị danh sách sinh viên dưới dạng bảng.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* **Tìm kiếm sinh viên**: Cho phép tìm kiếm sinh viên theo MSSV hoặc tên.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* **Cập nhật sinh viên:**

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* **Xóa sinh viên**: Xóa thông tin sinh viên khỏi hệ thống.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

#### 

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

#### Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình Mô tả được tạo tự động

#### 3.2 **Yêu Cầu Phi Chức Năng**

* **Bảo mật**: Đảm bảo an toàn cho thông tin người dùng và dữ liệu sinh viên.
* **Tính trực quan**: Giao diện dễ sử dụng, thân thiện với người dùng.
* **Tính hiệu quả**: Hệ thống phải phản hồi nhanh, truy xuất dữ liệu ổn định.

### 4. **Thiết Kế Hệ Thống**

#### 4.1 **Công Nghệ Sử Dụng**

* **Python**: Ngôn ngữ lập trình chính.
* **Tkinter**: Thư viện để xây dựng giao diện người dùng.
* **psycopg2**: Thư viện kết nối Python với PostgreSQL.
* **PostgreSQL**: Hệ quản trị cơ sở dữ liệu.

#### 4.2 **Thiết Kế Giao Diện**

* **Màn hình đăng nhập**: Người dùng nhập tên đăng nhập và mật khẩu để truy cập vào hệ thống.
* **Tab "Thêm Sinh Viên"**: Gồm các trường nhập liệu để thêm thông tin sinh viên.
* **Tab "Danh Sách Sinh Viên"**: Hiển thị danh sách sinh viên dưới dạng bảng và cho phép tìm kiếm, xóa sinh viên.

### 5. **Cấu Trúc Mã Nguồn**

#### 5.1 **Lớp** StudentApp

Lớp StudentApp là lớp chính của ứng dụng, chứa toàn bộ logic và giao diện người dùng.

##### **Thuộc tính chính**

* self.conn: Kết nối tới cơ sở dữ liệu PostgreSQL.
* self.cur: Con trỏ để thực hiện các truy vấn SQL.
* self.tab\_control: Điều hướng giữa các tab "Thêm Sinh Viên" và "Danh Sách Sinh Viên".

##### **Các hàm chính**

* connect\_db(): Kết nối tới cơ sở dữ liệu và xác thực người dùng.
* create\_login\_ui(): Tạo giao diện đăng nhập.
* create\_main\_ui(): Tạo giao diện chính gồm các tab.
* create\_add\_tab(): Tạo tab "Thêm Sinh Viên".
* insert\_data(): Thêm sinh viên mới vào cơ sở dữ liệu.
* create\_view\_tab(): Tạo tab "Danh Sách Sinh Viên".
* load\_data(): Tải danh sách sinh viên từ cơ sở dữ liệu.
* search\_student(): Tìm kiếm sinh viên theo từ khóa.
* delete\_student(): Xóa sinh viên khỏi cơ sở dữ liệu.

### 6. **Chi Tiết Mã Nguồn**

import tkinter as tk

from tkinter import ttk, messagebox

import psycopg2

class StudentApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Hệ Thống Quản Lý Sinh Viên")

        self.root.geometry("1000x700")

        self.conn = None

        self.cur = None

        # Tạo style cho giao diện

        self.style = ttk.Style()

        self.style.configure("TLabel", font=("Helvetica", 12), background="#f5f5f5")

        self.style.configure("TButton", font=("Helvetica", 12, 'bold'), padding=5)

        self.style.configure("TEntry", font=("Helvetica", 12))

        self.create\_login\_ui()

    def create\_login\_ui(self):

        """Giao diện đăng nhập"""

        self.login\_frame = tk.Frame(self.root, bg="#f5f5f5")

        self.login\_frame.place(relx=0.5, rely=0.5, anchor='center')

        tk.Label(self.login\_frame, text="Đăng Nhập", font=("Helvetica", 24, 'bold'), bg="#f5f5f5").pack(pady=20)

        ttk.Label(self.login\_frame, text="Tên đăng nhập:").pack(pady=5)

        self.username\_entry = ttk.Entry(self.login\_frame, width=30)

        self.username\_entry.pack()

        ttk.Label(self.login\_frame, text="Mật khẩu:").pack(pady=5)

        self.password\_entry = ttk.Entry(self.login\_frame, show="\*", width=30)

        self.password\_entry.pack()

        login\_btn = ttk.Button(self.login\_frame, text="Đăng nhập", command=self.connect\_db)

        login\_btn.pack(pady=20)

    def connect\_db(self):

        """Kết nối tới cơ sở dữ liệu"""

        username = self.username\_entry.get().strip()

        password = self.password\_entry.get().strip()

        try:

            self.conn = psycopg2.connect(

                dbname='dbtest',

                user=username,

                password=password,

                host='localhost',

                port='5432'

            )

            self.cur = self.conn.cursor()

            messagebox.showinfo("Thành công", "Kết nối thành công!")

            self.login\_frame.destroy()

            self.create\_main\_ui()

        except Exception as e:

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

    def create\_main\_ui(self):

        """Tạo giao diện chính"""

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

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

        self.add\_tab = ttk.Frame(self.tab\_control, padding=20)

        self.view\_tab = ttk.Frame(self.tab\_control, padding=20)

        self.tab\_control.add(self.add\_tab, text='Thêm Sinh Viên')

        self.tab\_control.add(self.view\_tab, text='Danh Sách Sinh Viên')

        self.create\_add\_tab()

        self.create\_view\_tab()

    def create\_add\_tab(self):

        """Tab thêm sinh viên"""

        form\_frame = tk.Frame(self.add\_tab)

        form\_frame.place(relx=0.5, rely=0.3, anchor='center')

        ttk.Label(form\_frame, text="MSSV:").grid(row=0, column=0, padx=10, pady=10, sticky='e')

        self.mssv\_entry = ttk.Entry(form\_frame, width=30)

        self.mssv\_entry.grid(row=0, column=1)

        ttk.Label(form\_frame, text="Họ và Tên:").grid(row=1, column=0, padx=10, pady=10, sticky='e')

        self.hoten\_entry = ttk.Entry(form\_frame, width=30)

        self.hoten\_entry.grid(row=1, column=1)

        ttk.Label(form\_frame, text="Ngày sinh (YYYY-MM-DD):").grid(row=2, column=0, padx=10, pady=10, sticky='e')

        self.ngaysinh\_entry = ttk.Entry(form\_frame, width=30)

        self.ngaysinh\_entry.grid(row=2, column=1)

        ttk.Label(form\_frame, text="Lớp:").grid(row=3, column=0, padx=10, pady=10, sticky='e')

        self.lop\_entry = ttk.Entry(form\_frame, width=30)

        self.lop\_entry.grid(row=3, column=1)

        ttk.Button(form\_frame, text="Thêm Sinh Viên", command=self.insert\_data).grid(row=4, columnspan=2, pady=20)

    def insert\_data(self):

        """Thêm sinh viên vào cơ sở dữ liệu"""

        mssv = self.mssv\_entry.get().strip()

        hoten = self.hoten\_entry.get().strip()

        ngaysinh = self.ngaysinh\_entry.get().strip()

        lop = self.lop\_entry.get().strip()

        if not mssv or not hoten or not ngaysinh or not lop:

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

            return

        try:

            self.cur.execute(

                "INSERT INTO sinhvien (mssv, hoten, ngaysinh, lop) VALUES (%s, %s, %s, %s)",

                (mssv, hoten, ngaysinh, lop)

            )

            self.conn.commit()

            messagebox.showinfo("Thành công", "Thêm sinh viên thành công!")

            self.load\_data()

        except Exception as e:

            self.conn.rollback()

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

    def create\_view\_tab(self):

        """Tab xem danh sách sinh viên"""

        search\_frame = tk.Frame(self.view\_tab)

        search\_frame.pack(anchor='w', pady=10)

        self.search\_entry = ttk.Entry(search\_frame, width=30)

        self.search\_entry.grid(row=0, column=0, padx=10)

        ttk.Button(search\_frame, text="Tìm kiếm", command=self.search\_student).grid(row=0, column=1)

        self.tree = ttk.Treeview(self.view\_tab, columns=('STT', 'MSSV', 'Họ và Tên', 'Ngày sinh', 'Lớp'), show='headings')

        self.tree.heading('STT', text='STT')

        self.tree.heading('MSSV', text='MSSV')

        self.tree.heading('Họ và Tên', text='Họ và Tên')

        self.tree.heading('Ngày sinh', text='Ngày sinh')

        self.tree.heading('Lớp', text='Lớp')

        self.tree.pack(fill='both', expand=True, pady=20)

        # Nút chức năng

        button\_frame = tk.Frame(self.view\_tab)

        button\_frame.pack(pady=10)

        ttk.Button(button\_frame, text="Load danh sách", command=self.load\_data).pack(side='left', padx=5)

        ttk.Button(button\_frame, text="Xóa Sinh Viên", command=self.delete\_student).pack(side='left', padx=5)

        ttk.Button(button\_frame, text="Chỉnh sửa", command=self.edit\_student).pack(side='left', padx=5)

        ttk.Button(button\_frame, text="Sắp xếp theo Tên", command=self.sort\_by\_name).pack(side='left', padx=5)

    def load\_data(self):

        """Load danh sách sinh viên"""

        self.tree.delete(\*self.tree.get\_children())

        self.cur.execute("SELECT mssv, hoten, ngaysinh, lop FROM sinhvien ORDER BY mssv")

        rows = self.cur.fetchall()

        for index, row in enumerate(rows, start=1):

            self.tree.insert("", "end", values=(index, \*row))

    def search\_student(self):

        """Tìm kiếm sinh viên"""

        search\_term = self.search\_entry.get().strip()

        self.cur.execute("SELECT \* FROM sinhvien WHERE mssv LIKE %s OR hoten LIKE %s ORDER BY mssv", (f"%{search\_term}%", f"%{search\_term}%"))

        rows = self.cur.fetchall()

        self.tree.delete(\*self.tree.get\_children())

        for index, row in enumerate(rows, start=1):

            self.tree.insert("", "end", values=(index, \*row))

    def delete\_student(self):

        """Xóa sinh viên"""

        selected\_item = self.tree.selection()

        if not selected\_item:

            messagebox.showwarning("Cảnh báo", "Vui lòng chọn sinh viên để xóa!")

            return

        mssv = str(self.tree.item(selected\_item)['values'][1])

        confirm = messagebox.askyesno("Xác nhận", f"Bạn có chắc chắn muốn xóa sinh viên với MSSV: {mssv}?")

        if not confirm:

            return

        try:

            self.cur.execute("DELETE FROM sinhvien WHERE mssv = %s", (mssv,))

            self.conn.commit()

            messagebox.showinfo("Thành công", "Xóa sinh viên thành công!")

            self.load\_data()

        except Exception as e:

            self.conn.rollback()

            messagebox.showerror("Lỗi", f"Lỗi khi xóa sinh viên: {e}")

    def edit\_student(self):

        selected\_item = self.tree.selection()

        if not selected\_item:

            messagebox.showwarning("Cảnh báo", "Vui lòng chọn sinh viên để chỉnh sửa!")

            return

        # Lấy thông tin sinh viên từ bảng

        mssv = str(self.tree.item(selected\_item)['values'][1])  # MSSV

        hoten = str(self.tree.item(selected\_item)['values'][2])

        ngaysinh = str(self.tree.item(selected\_item)['values'][3])

        lop = str(self.tree.item(selected\_item)['values'][4])

        # Tạo cửa sổ chỉnh sửa

        edit\_window = tk.Toplevel(self.root)

        edit\_window.title("Chỉnh sửa thông tin sinh viên")

        edit\_window.geometry("400x300")

        form\_frame = tk.Frame(edit\_window)

        form\_frame.place(relx=0.5, rely=0.3, anchor='center')

        # Thêm trường MSSV cho phép chỉnh sửa

        ttk.Label(form\_frame, text="MSSV:").grid(row=0, column=0, padx=10, pady=10, sticky='e')

        self.mssv\_entry = ttk.Entry(form\_frame, width=30)

        self.mssv\_entry.grid(row=0, column=1)

        self.mssv\_entry.insert(0, mssv)  # Hiển thị MSSV để chỉnh sửa

        # Các trường khác

        ttk.Label(form\_frame, text="Họ và Tên:").grid(row=1, column=0, padx=10, pady=10, sticky='e')

        self.hoten\_entry = ttk.Entry(form\_frame, width=30)

        self.hoten\_entry.grid(row=1, column=1)

        self.hoten\_entry.insert(0, hoten)

        ttk.Label(form\_frame, text="Ngày sinh (YYYY-MM-DD):").grid(row=2, column=0, padx=10, pady=10, sticky='e')

        self.ngaysinh\_entry = ttk.Entry(form\_frame, width=30)

        self.ngaysinh\_entry.grid(row=2, column=1)

        self.ngaysinh\_entry.insert(0, ngaysinh)

        ttk.Label(form\_frame, text="Lớp:").grid(row=3, column=0, padx=10, pady=10, sticky='e')

        self.lop\_entry = ttk.Entry(form\_frame, width=30)

        self.lop\_entry.grid(row=3, column=1)

        self.lop\_entry.insert(0, lop)

        # Nút để lưu thay đổi

        ttk.Button(form\_frame, text="Lưu Thay Đổi", command=lambda: self.update\_student(edit\_window)).grid(row=4, columnspan=2, pady=20)

    def update\_student(self, edit\_window):

        mssv = self.mssv\_entry.get().strip()  # Lấy MSSV từ trường nhập liệu

        hoten = self.hoten\_entry.get().strip()

        ngaysinh = self.ngaysinh\_entry.get().strip()

        lop = self.lop\_entry.get().strip()

        if not mssv or not hoten or not ngaysinh or not lop:

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

            return

        try:

            self.cur.execute(

                "UPDATE sinhvien SET mssv = %s, hoten = %s, ngaysinh = %s, lop = %s WHERE mssv = %s",

                (mssv, hoten, ngaysinh, lop, mssv)  # Cập nhật MSSV

            )

            self.conn.commit()

            messagebox.showinfo("Thành công", "Cập nhật thông tin sinh viên thành công!")

            self.load\_data()

            edit\_window.destroy()

        except Exception as e:

            self.conn.rollback()

            messagebox.showerror("Lỗi", f"Lỗi khi cập nhật thông tin sinh viên: {e}")

    def sort\_by\_name(self):

        """Sắp xếp sinh viên theo họ và tên"""

        self.tree.delete(\*self.tree.get\_children())

        self.cur.execute("SELECT mssv, hoten, ngaysinh, lop FROM sinhvien ORDER BY hoten")

        rows = self.cur.fetchall()

        for index, row in enumerate(rows, start=1):

            self.tree.insert("", "end", values=(index, \*row))

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

    root = tk.Tk()

    app = StudentApp(root)

    root.mainloop()

### 7. **Kết Quả Đạt Được**

* **Giao diện**: Ứng dụng có giao diện trực quan, thân thiện với người dùng.
* **Chức năng**: Đáp ứng đầy đủ các yêu cầu đã đề ra như đăng nhập, thêm, tìm kiếm, và xóa sinh viên.
* **Tích hợp cơ sở dữ liệu**: Ứng dụng kết nối thành công với PostgreSQL, đảm bảo tính bền vững của dữ liệu.

### 8. **Kết Luận**

Bài toán đã được giải quyết thành công với một ứng dụng quản lý sinh viên cơ bản. Hệ thống có khả năng mở rộng thêm các chức năng như cập nhật thông tin sinh viên, phân quyền truy cập, và cải thiện giao diện người dùng.

### 9. **Hướng Phát Triển Trong Tương Lai**

* **Cập nhật thông tin sinh viên**: Thêm chức năng cho phép người dùng chỉnh sửa thông tin sinh viên.
* **Bảo mật**: Áp dụng các kỹ thuật bảo mật như mã hóa mật khẩu.
* **Báo cáo**: Tạo các báo cáo thống kê về sinh viên theo lớp hoặc độ tuổi.

### 10. **Tài Liệu Tham Khảo**

* [Tkinter Documentation](https://docs.python.org/3/library/tkinter.html)
* [PostgreSQL Documentation](https://www.postgresql.org/docs/)
* Psycopg2 Documentation

**Github Repository**: [NGUYENTHANHNHAN/baitap2 at main · NhanDet/NGUYENTHANHNHAN](https://github.com/NhanDet/NGUYENTHANHNHAN/tree/main/baitap2)