**A black and white border

AI-generated content may be incorrect.**

**TRƯỜNG ĐẠI HỌC AN GIANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A yellow circle with blue and green text

Description automatically generated

A blue and black striped logo

AI-generated content may be incorrect.

**BÁO CÁO ĐỒ ÁN**

**Chuyên đề Python (COS525)**

**QUẢN LÝ BÁN HÀNG**

**(Sử dụng thư viện Tkinter)**

Giảng Viên: Nguyễn Ngọc Minh

**SINH VIÊN THỰC HIỆN:**

**DTH235607 – Lê Tuấn Anh**

**DTH235633 – TrầnVũ Duy**

**AN GIANG, 11-2025**

**MỤC LỤC**

[1. YÊU CẦU: 5](#_Toc213947730)

[1.1. MỤC TIÊU CỦA ĐỒ ÁN QUẢN LÝ BÁN HÀNG: 5](#_Toc213947731)

[**a) Quản lý Thông tin Hàng hóa (Product Management) 5**](#_Toc213947732)

[**b) Quản lý Khách hàng (Customer Management) 5**](#_Toc213947733)

[**c) Quản lý Nhân viên (Employee Management) 5**](#_Toc213947734)

[**d) Quản lý Tài khoản & Phân quyền (Account Management & Authentication) 6**](#_Toc213947735)

[**e) Quản lý Nghiệp vụ Bán hàng (Sales Invoice) 6**](#_Toc213947736)

[**f) Quản lý Báo cáo & Thống kê (Reporting & Dashboard) 6**](#_Toc213947737)

[1.2. CÁC TÍNH NĂNG CHÍNH CỦA PHẦN MỀM: 7](#_Toc213947738)

[**1.2.1. Thiết lập Giao diện Người dùng (GUI) 7**](#_Toc213947739)

[**1.2.2. Thao tác Sử dụng Đơn giản và Trực quan 7**](#_Toc213947740)

[**1.2.3. Thao tác CRUD Trực tiếp và Kiểm tra Dữ liệu 8**](#_Toc213947741)

[**1.2.4. Quản lý Hóa đơn Bán hàng (Sales Invoice) 9**](#_Toc213947742)

[2. THIẾT KẾ CƠ SỞ DỮ LIỆU (DATABASE DESIGN) 9](#_Toc213947743)

[2.1. Mô tả tổng quan về Cơ sở dữ liệu (CSDL) 9](#_Toc213947744)

[2.2 Thiết kế Cơ sở dữ liệu 9](#_Toc213947745)

[3. SOURCE CODE 13](#_Toc213947746)

[3.1. Cấu trúc thư mục và các file source code 13](#_Toc213947747)

[3.2. Quá trình xây dựng chi tiết và thực thi đồ án 15](#_Toc213947748)

[**1. Kiến trúc Nền tảng và Kết nối DB (auth.py, connect\_db.py) 15**](#_Toc213947749)

[**2. Quản lý Giao diện và Luồng Ứng dụng (splash\_form.py, login\_form.py, main\_form.py) 18**](#_Toc213947750)

[**3. Các Module Quản lý Dữ liệu (CRUD Forms) 40**](#_Toc213947751)

[**4. Module Nghiệp vụ Chính: Lập Hóa đơn (sales\_invoice\_form.py) 89**](#_Toc213947752)

[**5. Module Báo cáo (revenue\_report\_form.py) 90**](#_Toc213947753)

[3.3. Những phần đạt được và chưa đạt được của đồ án 91](#_Toc213947754)

[**a) Những phần đạt được: 91**](#_Toc213947755)

[**b) Những phần chưa đạt được: 92**](#_Toc213947756)

[**c) Đề xuất cải thiện: 92**](#_Toc213947757)

[4. TÀI LIỆU KHAM KHẢO: 92](#_Toc213947758)

**LỜI NÓI ĐẦU**

Trong bối cảnh kinh doanh hiện đại, việc quản lý các nghiệp vụ bán hàng một cách hiệu quả, từ kiểm soát tồn kho, theo dõi thông tin khách hàng, đến xử lý giao dịch và lập báo cáo doanh thu, là yếu tố then chốt quyết định sự thành công của mọi cửa hàng bán lẻ và doanh nghiệp thương mại nhỏ. Tuy nhiên, nhiều cơ sở kinh doanh vẫn đang gặp khó khăn với các phương pháp quản lý thủ công, tốn thời gian, và dễ xảy ra sai sót dữ liệu.

Xuất phát từ nhu cầu thực tiễn đó, đồ án chuyên đề **Quản lý Bán hàng** (thuộc chuyên đề Python) này được thực hiện nhằm mục tiêu xây dựng một hệ thống phần mềm ứng dụng các nguyên tắc lập trình hướng đối tượng (OOP) và sử dụng thư viện **Tkinter** của Python để thiết lập một giao diện người dùng đồ họa (GUI) thân thiện, trực quan.

Mục tiêu chính của đồ án là tự động hóa các quy trình nghiệp vụ cốt lõi, bao gồm: Quản lý chi tiết danh mục Hàng hóa, Khách hàng, và Nhân viên; Đảm bảo tính bảo mật thông qua cơ chế Quản lý Tài khoản & Phân quyền truy cập; Đặc biệt là thực hiện nghiệp vụ Bán hàng (Lập hóa đơn) với tính năng tính toán tự động và cập nhật tồn kho tức thì. Đồng thời, hệ thống còn cung cấp các Báo cáo và Thống kê tổng quan để hỗ trợ việc ra quyết định kinh doanh.

Báo cáo này sẽ trình bày chi tiết về các yêu cầu chức năng, quy trình thiết kế cơ sở dữ liệu (CSDL) với tên QuanLyBanHang, và các tính năng chính đã được triển khai, từ thiết lập giao diện Tkinter đến xử lý các thao tác CRUD và ràng buộc dữ liệu.

**Chủ đề: Xây dựng phần mềm quản lí sách**

**Bảng: Phân công công việc, tỷ lệ tham gia làm bài của từng thành viên**

|  |  |  |  |
| --- | --- | --- | --- |
| **HỌ VÀ TÊN** | **MSSV** | **Công việc thực hiện** | **Phần trăm tham gia** |
| Lê Tuấn Anh | DTH235607 | Phân tích, Thiết kế Word, Backend | 50% |
| Trần Vũ Duy | DTH235633 | Phân tích, Tester Word | 50% |

**NHẬN XÉT CỦA GIẢNG VIÊN**

....................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

|  |
| --- |
| **An Giang, Ngày 18 tháng 11 năm 2025** |
| ***Giảng viên*** |
| ***(Kí tên ghi rõ họ tên)*** |

# **1. YÊU CẦU:**

## **1.1. MỤC TIÊU CỦA ĐỒ ÁN QUẢN LÝ BÁN HÀNG:**

Đồ án này tập trung xây dựng một hệ thống phần mềm quản lý các nghiệp vụ cơ bản và nâng cao trong một cửa hàng bán lẻ hoặc doanh nghiệp thương mại nhỏ, với các mục tiêu chính sau:

### **a) Quản lý Thông tin Hàng hóa (Product Management)**

Mục tiêu là quản lý chi tiết danh mục hàng hóa (sản phẩm) để kiểm soát tồn kho và giá bán.

* **Mã hàng/Tên hàng:** Định danh và tên duy nhất của sản phẩm.
* **Chất liệu (tblChatLieu):** Phân loại sản phẩm theo chất liệu để dễ dàng quản lý và tìm kiếm.
* **Tồn kho (Số lượng):** Theo dõi số lượng hàng hóa hiện có trong kho.
* **Đơn giá nhập:** Giá vốn của sản phẩm, dùng cho tính toán lợi nhuận.
* **Đơn vị tính:** Đơn vị quy ước cho sản phẩm (Cái, Chiếc, Mét,...)
* **Ghi chú:** Thông tin bổ sung về sản phẩm.
* **Chức năng:** Hỗ trợ Thêm, Sửa, Xóa thông tin hàng hóa, và chức năng tìm kiếm theo Tên hàng.

### **b) Quản lý Khách hàng (Customer Management)**

Mục tiêu là theo dõi thông tin chi tiết của khách hàng để phục vụ cho nghiệp vụ bán hàng (lập hóa đơn) và các chương trình chăm sóc khách hàng.

* **Mã Khách/Tên Khách:** Định danh duy nhất và tên đầy đủ của khách hàng.
* **Địa chỉ/Điện thoại:** Thông tin liên hệ cần thiết.
* **Chức năng:** Hỗ trợ Thêm, Sửa, Xóa thông tin khách hàng, và chức năng tìm kiếm theo Tên khách hàng.

### **c) Quản lý Nhân viên (Employee Management)**

Mục tiêu là quản lý thông tin hồ sơ của đội ngũ nhân viên làm việc tại cửa hàng.

* **Mã NV/Tên NV:** Định danh và tên đầy đủ của nhân viên.
* **Thông tin cá nhân:** Giới tính, Ngày sinh, Địa chỉ, Điện thoại.
* **Chức năng:** Hỗ trợ Thêm, Sửa, Xóa thông tin nhân viên, và chức năng tìm kiếm theo Tên hoặc Số điện thoại.

### **d) Quản lý Tài khoản & Phân quyền (Account Management & Authentication)**

Mục tiêu là đảm bảo tính bảo mật và phân quyền truy cập chức năng cho từng nhân viên, liên kết tài khoản với hồ sơ nhân viên.

* **Đăng nhập & Xác thực:** Cho phép người dùng đăng nhập bằng Tên đăng nhập/Mật khẩu và kiểm tra trạng thái hoạt động/khóa tài khoản.
* **Phân quyền:** Gán quyền hạn cụ thể (ADMIN: 0, Quản lý: 1, Nhân viên: 2) để giới hạn truy cập vào các module nhạy cảm (ví dụ: Quản lý Nhân viên/Tài khoản, Báo cáo Doanh thu).
* **Quản lý Tài khoản:** Hỗ trợ Thêm, Sửa, Xóa tài khoản, gán/thay đổi Nhân viên liên kết, và phân quyền/khóa tài khoản.

### **e) Quản lý Nghiệp vụ Bán hàng (Sales Invoice)**

Mục tiêu là thực hiện nghiệp vụ bán hàng cốt lõi: lập hóa đơn và tính toán chi phí, đồng thời tự động cập nhật kho.

* **Lập Hóa đơn:** Cho phép tạo hóa đơn bán hàng.
* **Chi tiết Hóa đơn:** Ghi nhận các mặt hàng bán ra, số lượng, đơn giá và giảm giá.
* **Tính toán tự động:** Tự động tính Thành tiền của từng mặt hàng và Tổng tiền của toàn bộ hóa đơn.
* **Cập nhật kho:** Sau khi bán, hệ thống phải tự động giảm số lượng tồn kho của hàng hóa.

### **f) Quản lý Báo cáo & Thống kê (Reporting & Dashboard)**

Mục tiêu là cung cấp cái nhìn tổng quan và chi tiết về hiệu suất kinh doanh để hỗ trợ ra quyết định.

* **Bảng Điều khiển (Dashboard):** Hiển thị các chỉ số hiệu suất chính (KPI) như Tổng doanh thu, Tổng đơn hàng, Tổng sản phẩm, Top sản phẩm bán chạy nhất (Chỉ hiển thị cho ADMIN/Quản lý).
* **Báo cáo Chi tiết Giao dịch:** Cho phép lọc giao dịch theo ngày, theo nhân viên và xuất ra chi tiết từng mặt hàng đã bán, bao gồm tính năng Xuất File CSV.

## **1.2. CÁC TÍNH NĂNG CHÍNH CỦA PHẦN MỀM:**

Trong quá trình phát triển đồ án Quản lý Bán hàng, các tính năng chính được triển khai dựa trên nguyên tắc thân thiện với người dùng (GUI), tự động hóa nghiệp vụ và đảm bảo tính toàn vẹn dữ liệu.

### **1.2.1. Thiết lập Giao diện Người dùng (GUI)**

Hệ thống sử dụng thư viện tkinter của Python để xây dựng giao diện người dùng đồ họa (GUI) cho toàn bộ ứng dụng.

* **Công nghệ sử dụng:** Thư viện Tkinter kết hợp với ttk (Themed Tkinter) để tạo ra các widget hiện đại và thống nhất về mặt thẩm mỹ.
* **Cấu trúc giao diện:**

**+ Màn hình khởi động (Splash Screen):** Hiển thị trước khi vào màn hình đăng nhập.

**+ Form Đăng nhập (login\_form.py):** Giao diện đơn giản, trực quan, hỗ trợ hiển thị/ẩn mật khẩu và kiểm tra trạng thái tài khoản.

**+ Form Chính (Main Form):** Sử dụng thiết kế Sidebar Menu để điều hướng và một Content Frame để hiển thị các Module chức năng, đảm bảo giao diện đẹp, dễ nhìn, dễ sử dụng.

**+ Phân quyền Menu:** Thanh menu (Sidebar) tự động điều chỉnh các mục hiển thị tùy theo quyền hạn của người dùng đã đăng nhập (ADMIN, Quản lý, Nhân viên).

### **1.2.2. Thao tác Sử dụng Đơn giản và Trực quan**

* **Thiết bị đầu vào:** Khách hàng/Nhân viên sử dụng các đầu vào tiêu chuẩn bao gồm Bàn phím và Chuột.
* **Cấu trúc chức năng:**

+ Mỗi cửa sổ chức năng (Quản lý Hàng hóa, Quản lý Khách hàng,...) được xây dựng độc lập trong một Module (\*.py) để dễ dàng bảo trì và quản lý.

+ Sử dụng Treeview với các cột dữ liệu rõ ràng, hỗ trợ cuộn dọc và hiển thị dữ liệu lớn.

**+ Thao tác nhanh:** Hỗ trợ tính năng Double-click vào một dòng dữ liệu trên Treeview để tự động chuyển sang chế độ Sửa (EDIT), giúp tăng tốc độ nhập liệu.

**+ Phản hồi trạng thái:** Có thanh trạng thái (status\_bar) ở cuối mỗi form để thông báo về quá trình tải dữ liệu, tìm kiếm, hoặc kết quả CRUD (Thêm/Sửa/Xóa).

### **1.2.3. Thao tác CRUD Trực tiếp và Kiểm tra Dữ liệu**

Tính năng Thêm mới, Xóa, Sửa (CRUD) dữ liệu được triển khai cho các danh mục cốt lõi như Hàng hóa, Khách hàng, Nhân viên và Tài khoản.

* **Thêm mới dữ liệu trực tiếp:** Cho phép nhập dữ liệu mới (ví dụ: thông tin khách hàng, thông tin hàng hóa, nhân viên) vào hệ thống thông qua Form nhập liệu chi tiết.

**+ Quản lý Mã tự động:** Tự động tính toán và gán Mã ID tiếp theo (ví dụ: MaHang, MaKhach, MaNhanVien), bao gồm logic tìm và lấp các khoảng trống (gap ID) do việc xóa trước đó.

* **Xóa Sửa trực tiếp:** Khách hàng (người dùng) có thể chọn một bản ghi trên bảng và thực hiện các thao tác Sửa (EDIT) hoặc Xóa (DELETE) trực tiếp.

**+ Quản lý Trạng thái:** Giao diện Form được quản lý theo ba trạng thái chính (VIEW, ADD, EDIT), tự động bật/tắt (enable/disable) các trường nhập liệu và các nút chức năng tương ứng để tránh lỗi nhập liệu.

* **Xử lý Lỗi và Ràng buộc:**

**+ Lỗi thiếu thông tin:** Hệ thống hiển thị cảnh báo (messagebox.showwarning) nếu người dùng không nhập đầy đủ các trường thông tin bắt buộc.

**+ Lỗi trùng lặp:** Xử lý lỗi trùng lặp Tên Đăng nhập (pyodbc.IntegrityError) khi Thêm/Sửa Tài khoản.

**+ Lỗi khóa ngoại:** Ngăn chặn các thao tác xóa dữ liệu (Nhân viên, Khách hàng) nếu có dữ liệu giao dịch liên quan (Hóa đơn), đồng thời cung cấp thông báo lỗi rõ ràng.

### **1.2.4. Quản lý Hóa đơn Bán hàng (Sales Invoice)**

* **Tính toán và Lưu trữ Hóa đơn:** Hóa đơn sẽ được tính toán tự động dựa trên các thông tin về Mặt hàng, Số lượng, Đơn giá và Giảm giá (%) mà nhân viên cung cấp.

**+ Tự động tính Thành tiền:** Trường Thành tiền của từng mặt hàng được tính toán và lưu vào hệ thống.

**+ Tự động cập nhật Tổng tiền:** Tổng tiền của toàn bộ hóa đơn được tính toán và lưu vào bảng tblHDBan.

* **Cập nhật tồn kho:**

**+ Cập nhật tự động:** Sau khi hóa đơn được lập và lưu thành công, hệ thống phải tự động giảm số lượng tồn kho (tblHang.SoLuong) của các mặt hàng đã bán.

* **Tải dữ liệu sau khi xử lý:**

*+* **Lưu ý (Cải chính yêu cầu):** Dữ liệu hóa đơn, sau khi xuất và lưu vào DB, sẽ không bị xóa khỏi hệ thống mà sẽ được duy trì trong các bảng tblHDBan và tblChiTietHDBan để phục vụ cho các nghiệp vụ Báo cáo và Truy vấn lịch sử sau này.

# **2. THIẾT KẾ CƠ SỞ DỮ LIỆU (DATABASE DESIGN)**

## **2.1. Mô tả tổng quan về Cơ sở dữ liệu (CSDL)**

Cơ sở dữ liệu được thiết kế với tên **QuanLyBanHang** nhằm mục đích lưu trữ và quản lý toàn bộ dữ liệu nghiệp vụ chính của một cửa hàng bán lẻ:

* **Danh mục:** Thông tin về Hàng hóa, Chất liệu, Khách hàng, và Hồ sơ Nhân viên.
* **Hệ thống:** Thông tin về Tài khoản và Phân quyền người dùng.
* **Giao dịch:** Thông tin chi tiết về các Hóa đơn Bán hàng và các Mặt hàng đã bán.

## **2.2 Thiết kế Cơ sở dữ liệu**

Cơ sở dữ liệu được tạo là **QuanLyBanHang:**

* Bảng “Chất Liệu”:

MaChatLieu INT IDENTITY(1,1) PRIMARY KEY, -- Khóa chính, tự động tăng (1, 2, 3...)

TenChatLieu NVARCHAR(100) NOT NULL -- Tên chất liệu (Ví dụ: Vải, Da, Gỗ)

* Bảng “Hàng”:

MaHang INT IDENTITY(1,1) PRIMARY KEY, -- Khóa chính, tự động tăng

    TenHang NVARCHAR(200) NOT NULL, -- Tên mặt hàng

MaChatLieu INT NULL, -- Khóa ngoại liên kết tới tblChatLieu (NULLable:có thể không rõ chất liệu)

    SoLuong INT NOT NULL DEFAULT 0, -- Số lượng tồn kho (mặc định 0)

    DonGiaNhap DECIMAL(18,2) NULL, -- Giá nhập vào

    DonVi NVARCHAR(50) NULL, -- Đơn vị tính (Cái, Kg...)

    Anh VARBINARY(MAX) NULL, -- Lưu trữ hình ảnh sản phẩm

    GhiChu NVARCHAR(400) NULL, -- Ghi chú thêm

CONSTRAINT FK\_Hang\_ChatLieu FOREIGN KEY (MaChatLieu) -- Ràng buộc Khóa ngoại

        REFERENCES dbo.tblChatLieu(MaChatLieu)

ON UPDATE CASCADE -- Cập nhật MaChatLieu bên tblHang nếu tblChatLieu thay đổi

        ON DELETE SET NULL -- Đặt MaChatLieu về NULL nếu chất liệu bị xóa

* Bảng “Nhân Viên”:

MaNhanVien INT IDENTITY(1,1) PRIMARY KEY, -- Khóa chính, tự động tăng

    TenNhanVien NVARCHAR(150) NOT NULL,

    GioiTinh NVARCHAR(10) NULL,

    DiaChi NVARCHAR(300) NULL,

    DienThoai NVARCHAR(50) NULL,

    NgaySinh DATE NULL

* Bảng “Khách”:

MaKhach INT IDENTITY(1,1) PRIMARY KEY, -- Khóa chính, tự động tăng

TenKhach NVARCHAR(200) NOT NULL,

DiaChi NVARCHAR(300) NULL,

DienThoaiNVARCHAR(50)NULL

* Bảng “Hóa Đơn Bán Hàng”:

MaHDBan INT IDENTITY(1,1) PRIMARY KEY, -- Khóa chính, tự động tăng

MaNhanVien INT NOT NULL, -- Khóa ngoại (Bắt buộc phải có nhân viên lập)

MaKhach INT NULL, -- Khóa ngoại (Có thể không cần khách hàng cụ thể)

NgayBan DATE NOT NULL DEFAULT GETDATE(), -- Ngày tạo Hóa đơn (mặc định là ngày hiện tại)

TongTien DECIMAL(18,2) NOT NULL DEFAULT 0, -- Tổng tiền của Hóa đơn (sẽ được cập nhật sau)

    CONSTRAINT FK\_HDBan\_NhanVien FOREIGN KEY (MaNhanVien)

        REFERENCES dbo.tblNhanVien(MaNhanVien)

        ON UPDATE CASCADE

        ON DELETE NO ACTION,

    CONSTRAINT FK\_HDBan\_Khach FOREIGN KEY (MaKhach)

      REFERENCES dbo.tblKhach(MaKhach)

        ON UPDATE CASCADE

        ON DELETE SET NULL

* Bảng “Chi Tiết Hóa Đơn”:

MaHDBan INT NOT NULL, -- Khóa ngoại (liên kết với tblHDBan)

MaHang INT NOT NULL, -- Khóa ngoại (liên kết với tblHang)

SoLuong INT NOT NULL DEFAULT 1,

DonGia DECIMAL(18,2) NOT NULL, -- Đơn giá bán tại thời điểm lập chi tiết

GiamGia DECIMAL(5,2) NULL DEFAULT 0, -- Giảm giá (dưới dạng %)

-- Cột tính toán tự động: Số lượng \* Đơn giá \* (1 - Giảm giá/100)

ThanhTien AS (SoLuong \* DonGia \* (1 - ISNULL(GiamGia,0)/100.0)) PERSISTED,

PRIMARY KEY (MaHDBan, MaHang), -- Khóa chính là tổ hợp của Mã HĐ và Mã Hàng

CONSTRAINT FK\_CTHDBan\_HDBan FOREIGN KEY (MaHDBan)

REFERENCES dbo.tblHDBan(MaHDBan)

ON UPDATE CASCADE

ON DELETE CASCADE, -- Xóa Chi tiết HĐ khi Hóa đơn chính bị xóa

CONSTRAINT FK\_CTHDBan\_Hang FOREIGN KEY (MaHang)

REFERENCES dbo.tblHang(MaHang)

ON UPDATE CASCADE

ON DELETE NO ACTION -- Cấm xóa Hàng hóa nếu còn tồn tại trong Chi tiết HĐ

* Bảng “Tài Khoản”:

MaTK INT IDENTITY(1,1) PRIMARY KEY,

MaNhanVien INT NULL, -- Khóa ngoại liên kết với Nhân viên (Tài khoản thuộc về NV nào?)

TenDangNhap NVARCHAR(100) NOT NULL UNIQUE, -- Tên đăng nhập (Bắt buộc và không trùng lặp)

    MatKhau NVARCHAR(100) NOT NULL,

    PhanQuyen INT NOT NULL DEFAULT 0, -- Phân quyền (0: Admin, 1: Nhân viên)

    TrangThai BIT NOT NULL DEFAULT 1, -- Trạng thái tài khoản (1: Hoạt động)

    CONSTRAINT FK\_TaiKhoan\_NhanVien FOREIGN KEY (MaNhanVien)

        REFERENCES dbo.tblNhanVien(MaNhanVien)

        ON UPDATE CASCADE

      ON DELETE SET NULL -- Đặt MaNhanVien về NULL nếu Nhân viên bị xóa (TK vẫn tồn tại)

* Relationship:

A computer screen shot of a computer

AI-generated content may be incorrect.

* Tham Chiếu:
* tblHang(MaChatLieu) => tblChatLieu(MaChatLieu)
* tblChiTietHDBan(MaHDBan) => tblHDBan(MaHDBan)
* tblChiTietHDBan(MaHang) => tblHang(MaHang)
* tblHDBan(MaNhanVien) => tblNhanVien(MaNhanVien)
* tblHDBan(MaKhach) => tblKhach(MaKhach)
* tblTaiKhoan(MaNhanVien) => tblNhanVien(MaNhanVien)

# **3. SOURCE CODE**

## **3.1. Cấu trúc thư mục và các file source code**

* **Thư mục:** DoAn\_QuanLyBanHang
* **Thư mục Database**: Chứa các file chứa cơ sở dữ liệu sales management.sql
* **Thư mục BaoCao\_Word**: Chứa file WORD\_DOAN\_QLBH.docx
* **Thư mục SRC chứa các file.py**:

+ **connect\_db.py:**

Kiểm tra Kết nối Hệ thống: Đảm bảo hệ thống có thể kết nối được với máy chủ dữ liệu.

+ **auth.py:**

Xác thực Truy cập: Cung cấp các hàm để quản lý kết nối và kiểm tra tính hợp lệ của thông tin đăng nhập.

+ **login\_form.py:**

Giao diện Khởi động: Thu thập thông tin định danh và cấp quyền truy cập vào hệ thống.

+ **main\_form.py:**

Trung tâm Điều hành & Phân quyền: Cung cấp giao diện chính, điều hướng người dùng và kiểm soát quyền truy cập tính năng dựa trên vai trò.

+ **Dashboard (trong main\_form.py):**

Tổng quan Hiệu suất: Trình bày các chỉ số kinh doanh quan trọng và tổng kết hoạt động.

**+ product\_management\_form.py:**

Quản lý Hàng tồn kho: Thêm, sửa, xóa và theo dõi thông tin chi tiết về các mặt hàng được mua bán.

+ **customer\_management\_form.py:**

Quản lý Thông tin Khách hàng: Cập nhật, tìm kiếm và lưu trữ hồ sơ của khách hàng giao dịch.

+ **employee\_management\_form.py:**

Quản lý Hồ sơ Nhân sự: Lưu trữ và cập nhật thông tin cá nhân của đội ngũ nhân viên.

+ **account\_management\_form.py:**

Quản lý Tài khoản Người dùng: Tạo, điều chỉnh quyền hạn và theo dõi trạng thái sử dụng của tài khoản đăng nhập.

+ **sales\_invoice\_form.py:**

Thực hiện Giao dịch Bán hàng: Lập phiếu tính tiền cho khách, quản lý giỏ hàng, tính tổng tiền và tự động cập nhật lại số lượng tồn kho.

+ **revenue\_report\_form.py:**

Báo cáo và Phân tích Doanh thu: Thống kê chi tiết các giao dịch trong một khoảng thời gian, hỗ trợ lọc và xuất dữ liệu để phục vụ công tác báo cáo.

## **3.2. Quá trình xây dựng chi tiết và thực thi đồ án**

Đồ án được xây dựng theo mô hình module hóa, trong đó mỗi chức năng lớn được đặt trong một file Python riêng biệt, cho phép quản lý mã nguồn dễ dàng và tái sử dụng logic (ví dụ: kết nối DB).

### **1. Kiến trúc Nền tảng và Kết nối DB (auth.py, connect\_db.py)**

* **Kết nối CSDL (auth.py/get\_connection):**
* Sử dụng thư viện pyodbc để thiết lập kết nối đến SQL Server.
* Thông tin kết nối (Server, Database, UID, PWD) được định nghĩa cố định trong hàm get\_connection().
* Kết nối sử dụng tài khoản sa và cần thiết lập TrustServerCertificate=yes để tránh lỗi chứng chỉ.
* **Xác thực Đăng nhập (auth.py/verify\_login):**
* Truy vấn bảng tblTaiKhoan để kiểm tra TenDangNhap và MatKhau.
* Thực hiện LEFT JOIN với tblNhanVien để lấy TenNhanVien và MaNhanVien phục vụ cho việc hiển thị thông tin và lập hóa đơn.
* Kiểm tra TrangThai của tài khoản (0: bị khóa, 1: hoạt động).
* **Code auth.py**:
* # -\*- coding: utf-8 -\*-
* import pyodbc
* # ==========================
* # KẾT NỐI SQL SERVER
* # ==========================
* def get\_connection():
* try:
* conn = pyodbc.connect(
* 'DRIVER={SQL Server};'
* 'SERVER=LAPTOP-TUANANH;'      # 👈 Tên máy của bạn
* 'DATABASE=QuanLyBanHang;'     # 👈 Tên CSDL
* 'UID=sa;'                     # 👈 Tài khoản sa
* 'PWD=123;'                    # 👈 Mật khẩu
* 'TrustServerCertificate=yes;' # Tránh lỗi SSL
* )
* # Để pyodbc tự động xử lý encoding (không force UTF-8)
* return conn
* except Exception as e:
* # Giữ lại print để bạn thấy lỗi kết nối trong console
* print("❌ Lỗi kết nối SQL Server:", e)
* return None
* # ==========================
* # HÀM KIỂM TRA ĐĂNG NHẬP
* # ==========================
* def verify\_login(username, password):
* conn = get\_connection()
* if conn is None:
* # Đã cập nhật thông báo lỗi rõ ràng hơn
* return False, None, "Không thể kết nối đến cơ sở dữ liệu. Vui lòng kiểm tra lại cấu hình kết nối."
* try:
* cursor = conn.cursor()
* cursor.execute("""
* SELECT tk.TenDangNhap, tk.MatKhau, tk.PhanQuyen, tk.TrangThai, nv.TenNhanVien, nv.MaNhanVien
* FROM tblTaiKhoan tk
* LEFT JOIN tblNhanVien nv ON tk.MaNhanVien = nv.MaNhanVien
* WHERE tk.TenDangNhap = ? AND tk.MatKhau = ?
* """, (username, password))
* row = cursor.fetchone()
* if not row:
* return False, None, "Sai tên đăng nhập hoặc mật khẩu."
* ten\_dang\_nhap, mat\_khau, phan\_quyen, trang\_thai, ten\_nhan\_vien, ma\_nhan\_vien = row
* # Kiểm tra trạng thái tài khoản
* if trang\_thai == 0:
* return False, None, "Tài khoản đã bị khóa."
* # ✅ Thông tin người dùng
* user\_info = {
* "TenDangNhap": ten\_dang\_nhap,
* "PhanQuyen": phan\_quyen,
* "HoTen": ten\_nhan\_vien if ten\_nhan\_vien else "Không rõ",
* "MaNhanVien": ma\_nhan\_vien
* }
* return True, user\_info, "Đăng nhập thành công."
* except Exception as e:
* return False, None, f"Lỗi truy vấn: {e}"
* finally:
* if conn:
* conn.close()
* **Code connect\_db.py:**
* import pyodbc
* def connect\_db():
* try:
* conn = pyodbc.connect(
* 'DRIVER={ODBC Driver 18 for SQL Server};'
* 'SERVER=LAPTOP-TUANANH;'
* 'DATABASE=QuanLyBanHang;'
* 'UID=sa;'
* 'PWD=123;'
* 'Encrypt=no;'
* 'TrustServerCertificate=yes;'
* )
* print("✅ Kết nối SQL Server thành công!")
* return conn
* except Exception as e:
* print("❌ Lỗi kết nối SQL Server:", e)
* return None
* if \_\_name\_\_ == "\_\_main\_\_":
* connect\_db()

### **2. Quản lý Giao diện và Luồng Ứng dụng (splash\_form.py, login\_form.py, main\_form.py**)

* **Luồng Khởi động (main\_form.py if \_\_name\_\_ == "\_\_main\_\_":):**
* Hiển thị SplashScreen (splash\_form).
* Chuyển sang LoginForm (login\_form.py).
* Sau khi đăng nhập thành công (verify\_login trả về True), chuyển sang MainApp (main\_form.py).

***Splash***

A green box with blue text

AI-generated content may be incorrect.

* **Code Splash\_form**
* # -\*- coding: utf-8 -\*-
* # splash\_form.py - Màn hình chào (Splash Screen)
* import tkinter as tk
* from tkinter import ttk
* def center\_window(win, width, height):
* """Căn giữa cửa sổ trên màn hình"""
* win.update\_idletasks()
* screen\_width = win.winfo\_screenwidth()
* screen\_height = win.winfo\_screenheight()
* x = (screen\_width // 2) - (width // 2)
* y = (screen\_height // 2) - (height // 2)
* win.geometry(f"{width}x{height}+{x}+{y}")
* class SplashScreen:
* def \_\_init\_\_(self, root, on\_done=None):
* self.root = root
* self.on\_done = on\_done
* self.root.title("Đang khởi động...")
* center\_window(self.root, 480, 300)
* self.root.configure(bg="#f5f5f5")
* self.root.overrideredirect(True)  # Ẩn thanh tiêu đề
* frame = tk.Frame(self.root, bg="#f5f5f5")
* frame.pack(expand=True, fill="both")
* tk.Label(frame, text="🛒 PHẦN MỀM QUẢN LÝ BÁN HÀNG",
* font=("Arial", 16, "bold"), bg="#f5f5f5", fg="#1976D2").pack(pady=(80, 10))
* self.progress = ttk.Progressbar(frame, orient="horizontal", mode="determinate", length=350)
* self.progress.pack(pady=20)
* self.label\_status = tk.Label(frame, text="Đang tải dữ liệu...",
* bg="#f5f5f5", fg="#555", font=("Arial", 10))
* self.label\_status.pack()
* self.progress\_value = 0
* self.update\_progress()
* def update\_progress(self):
* if self.progress\_value < 100:
* self.progress\_value += 2
* self.progress["value"] = self.progress\_value
* self.label\_status.config(text=f"Đang khởi động... {self.progress\_value}%")
* self.root.after(60, self.update\_progress)
* else:
* self.finish()
* def finish(self):
* self.root.destroy()
* if self.on\_done:
* self.on\_done()
* # ==== Test độc lập ====
* if \_\_name\_\_ == "\_\_main\_\_":
* def open\_login():
* print("✅ Splash kết thúc — mở LoginForm...")
* root = tk.Tk()
* app = SplashScreen(root, on\_done=open\_login)
* root.mainloop()

***Login***

A screenshot of a computer

AI-generated content may be incorrect.

* **Code login\_form**
* # -\*- coding: utf-8 -\*-
* # login\_form.py - Form đăng nhập cho hệ thống quản lý bán hàng (Phiên bản nâng cao)
* import tkinter as tk
* from tkinter import messagebox
* from auth import verify\_login
* from splash\_form import center\_window
* # from PIL import Image, ImageTk # Không cần nếu không dùng hình ảnh
* class LoginForm:
* def \_\_init\_\_(self, root, on\_success\_callback):
* self.root = root
* self.on\_success = on\_success\_callback
* self.show\_pass\_var = tk.BooleanVar() # Biến để lưu trạng thái checkbox
* self.setup\_ui()
* def setup\_ui(self):
* # 1. Cấu hình cơ bản
* self.root.title("Đăng nhập - Quản lý Bán hàng")
* center\_window(self.root, 480, 480) # Tăng nhẹ chiều cao để chứa checkbox
* self.root.resizable(False, False)
* self.root.configure(bg="#E0E0E0")
* # 2. Frame chính với padding
* main\_frame = tk.Frame(self.root, bg="white", padx=40, pady=40, bd=1, relief=tk.SOLID)
* main\_frame.pack(expand=True, fill="both", padx=30, pady=30)
* # 3. Tiêu đề
* tk.Label(main\_frame, text="🔑 ĐĂNG NHẬP HỆ THỐNG",
* font=("Arial", 18, "bold"), bg="white", fg="#0288D1").pack(pady=(0, 30))
* # 4. Input Tài khoản
* tk.Label(main\_frame, text="Tên đăng nhập:", font=("Arial", 11, "bold"), bg="white", fg="#424242").pack(anchor="w", pady=(5, 0))
* self.entry\_user = tk.Entry(main\_frame, font=("Arial", 12), bd=1, relief=tk.FLAT, highlightthickness=1, highlightbackground="#BDBDBD", highlightcolor="#03A9F4", insertbackground="#03A9F4")
* self.entry\_user.pack(fill="x", ipady=5, pady=(2, 20))
* self.entry\_user.focus()
* # 5. Input Mật khẩu
* tk.Label(main\_frame, text="Mật khẩu:", font=("Arial", 11, "bold"), bg="white", fg="#424242").pack(anchor="w", pady=(5, 0))
* self.entry\_pass = tk.Entry(main\_frame, font=("Arial", 12), show="●", bd=1, relief=tk.FLAT, highlightthickness=1, highlightbackground="#BDBDBD", highlightcolor="#03A9F4", insertbackground="#03A9F4")
* self.entry\_pass.pack(fill="x", ipady=5, pady=(2, 10))
* # 6. Checkbox "Hiện mật khẩu"
* tk.Checkbutton(main\_frame, text="Hiện mật khẩu", font=("Arial", 10), bg="white", fg="#424242",
* variable=self.show\_pass\_var, command=self.toggle\_password\_visibility).pack(anchor="w", pady=(0, 30))
* # 7. Khung chứa nút (tạo một frame riêng để căn chỉnh)
* button\_frame = tk.Frame(main\_frame, bg="white")
* button\_frame.pack(fill="x", pady=(10, 0))
* # 8. Nút Đăng nhập (bên trái)
* self.btn\_login = tk.Button(button\_frame, text="ĐĂNG NHẬP", font=("Arial", 12, "bold"),
* bg="#4CAF50", fg="white", relief=tk.FLAT, cursor="hand2", # SỬ DỤNG tk.FLAT
* command=self.login, activebackground="#43A047", activeforeground="white")
* self.btn\_login.pack(side=tk.LEFT, ipadx=10, ipady=8, expand=True, fill="x", padx=(0, 5)) # Căn trái
* # 9. Nút Thoát (bên phải)
* self.btn\_exit = tk.Button(button\_frame, text="THOÁT", font=("Arial", 12, "bold"),
* bg="#E53935", fg="white", relief=tk.FLAT, cursor="hand2", # SỬ DỤNG tk.FLAT
* command=self.exit\_app, activebackground="#D32F2F", activeforeground="white")
* self.btn\_exit.pack(side=tk.RIGHT, ipadx=10, ipady=8, expand=True, fill="x", padx=(5, 0)) # Căn phải
* # 10. Enter key & Gợi ý
* self.entry\_user.bind("<Return>", lambda e: self.entry\_pass.focus())
* self.entry\_pass.bind("<Return>", lambda e: self.login())
* # Cập nhật Demo để khớp với mật khẩu SQL mới
* tk.Label(main\_frame, text="💡 Demo: admin / admin123 hoặc nhanvien / nhanvien123",
* font=("Arial", 9, "italic"), bg="white", fg="#757575").pack(pady=(25, 0))
* # ==== Hàm xử lý Hiện/Ẩn mật khẩu ====
* def toggle\_password\_visibility(self):
* if self.show\_pass\_var.get():
* # Hiện mật khẩu
* self.entry\_pass.config(show="")
* else:
* # Ẩn mật khẩu bằng ký tự ●
* self.entry\_pass.config(show="●")
* # ==== Xử lý đăng nhập (Giữ nguyên logic) ====
* def login(self):
* username = self.entry\_user.get().strip()
* password = self.entry\_pass.get().strip()
* if not username or not password:
* messagebox.showwarning("Thiếu thông tin", "Vui lòng nhập tài khoản và mật khẩu!")
* return
* self.btn\_login.config(state=tk.DISABLED, text="ĐANG KIỂM TRA...")
* self.root.update()
* success, user\_info, msg = verify\_login(username, password)
* self.btn\_login.config(state=tk.NORMAL, text="ĐĂNG NHẬP")
* if success:
* self.root.destroy()
* if self.on\_success:
* self.on\_success(user\_info)
* else:
* messagebox.showerror("Đăng nhập thất bại", msg)
* self.entry\_pass.delete(0, tk.END)
* self.entry\_pass.focus()
* # ==== Xử lý thoát (Giữ nguyên logic) ====
* def exit\_app(self):
* if messagebox.askyesno("Thoát chương trình", "Bạn có chắc chắn muốn thoát không?"):
* self.root.destroy()
* # ==== Test riêng ====
* if \_\_name\_\_ == "\_\_main\_\_":
* def after\_login(user):
* print("✅ Đăng nhập thành công:", user["HoTen"])
* root = tk.Tk()
* app = LoginForm(root, after\_login)
* root.mainloop()
* **Giao diện Chính (main\_form.py/MainApp):**
* Sử dụng kiến trúc Sidebar (Menu) và Content Frame.
* Phân quyền (Role-based access control - RBAC): Các mục menu được hiển thị dựa trên PhanQuyen (user\_role):

+ ***ADMIN (0)***: Xem tất cả.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.+ ***QUẢN LÝ (1***): Xem hầu hết, ngoại trừ Quản lý Nhân viên và Quản lý Tài khoản.

+ ***NHÂN VIÊN (2)***: Chỉ xem Bảng điều khiển, Quản lý Khách hàng, Lập Hóa đơn.

A screenshot of a computer

AI-generated content may be incorrect.

* Dashboard: Hiển thị các chỉ số KPI cơ bản (Doanh thu, Đơn hàng, Sản phẩm, Khách hàng) bằng cách truy vấn trực tiếp vào DB, chỉ hiển thị thông tin tài chính cho ADMIN/QUẢN LÝ.
* **Chuyển đổi Form (open\_module):** Dựa vào lựa chọn trên menu, form module tương ứng được khởi tạo bên trong Content Frame.
* **Code main\_form**
* # -\*- coding: utf-8 -\*-
* # main\_form.py - Form Chính (Main Form) với giao diện Sidebar và logic Phân quyền
* import tkinter as tk
* from tkinter import messagebox, ttk
* import sys
* from splash\_form import center\_window, SplashScreen
* from auth import verify\_login
* import pyodbc # Cần thiết cho các hàm DB mô phỏng
* # Đảm bảo console (nếu có log) dùng UTF-8 để không lỗi Unicode khi in tiếng Việt
* try:
* sys.stdout.reconfigure(encoding='utf-8')
* sys.stderr.reconfigure(encoding='utf-8')
* except Exception:
* pass
* # 1. IMPORT MODULES (Giữ nguyên)
* try:
* from product\_management\_form import ProductManagementForm
* from customer\_management\_form import CustomerManagementForm
* from employee\_management\_form import EmployeeManagementForm
* from account\_management\_form import AccountManagementForm
* from revenue\_report\_form import DetailedInvoiceReportForm
* from sales\_invoice\_form import SalesInvoiceForm
* except ImportError as e:
* print(f"❌ Lỗi Import Modules: Vui lòng đảm bảo các file form có trong thư mục. Chi tiết: {e}")
* ProductManagementForm = None
* CustomerManagementForm = None
* EmployeeManagementForm = None
* AccountManagementForm = None
* RevenueReportForm = None
* SalesInvoiceForm = None
* DetailedInvoiceReportForm = None
* # ==========================================================
* # LỚP BẢNG ĐIỀU KHIỂN (DASHBOARD) MỚI
* # ==========================================================
* class Dashboard:
* def \_\_init\_\_(self, master, user\_info):
* self.master = master
* self.user\_info = user\_info
* self.user\_role = user\_info['PhanQuyen']
* self.setup\_ui()
* def get\_conn(self):
* # Dùng lại hàm get\_conn từ auth.py (Giả định)
* try:
* from auth import get\_connection
* return get\_connection()
* except Exception:
* return None
* def fetch\_stats(self):
* """Mô phỏng/Thực hiện tải các số liệu thống kê từ DB."""
* stats = {
* "TotalRevenue": 0, "TotalOrders": 0, "TotalProducts": 0, "TotalCustomers": 0, "TopProduct": "N/A"
* }
* # Nếu là ADMIN/QUẢN LÝ (role 0, 1) thì mới hiển thị doanh thu
* if self.user\_role in [0, 1]:
* conn = self.get\_conn()
* if conn:
* try:
* cursor = conn.cursor()
* # 1. Tổng doanh thu (Tổng TongTien từ tblHDBan)
* cursor.execute("SELECT ISNULL(SUM(TongTien), 0) FROM tblHDBan")
* stats["TotalRevenue"] = cursor.fetchone()[0]
* # 2. Tổng số đơn hàng
* cursor.execute("SELECT COUNT(MaHDBan) FROM tblHDBan")
* stats["TotalOrders"] = cursor.fetchone()[0]
* # 3. Tổng số sản phẩm
* cursor.execute("SELECT COUNT(MaHang) FROM tblHang")
* stats["TotalProducts"] = cursor.fetchone()[0]
* # 4. Tổng số khách hàng
* cursor.execute("SELECT COUNT(MaKhach) FROM tblKhach")
* stats["TotalCustomers"] = cursor.fetchone()[0]
* # 5. Top sản phẩm bán chạy nhất (ví dụ: theo số lượng)
* cursor.execute("""
* SELECT TOP 1 h.TenHang, SUM(ct.SoLuong) AS TotalSL
* FROM tblChiTietHDBan ct
* JOIN tblHang h ON ct.MaHang = h.MaHang
* GROUP BY h.TenHang
* ORDER BY TotalSL DESC
* """)
* top\_row = cursor.fetchone()
* if top\_row:
* stats["TopProduct"] = f"{top\_row[0]} ({top\_row[1]} SP)"
* except Exception as e:
* print(f"Lỗi tải thống kê DB: {e}")
* finally:
* conn.close()
* else:
* # Dữ liệu cho Nhân viên (chỉ được xem những thứ không liên quan đến tiền)
* conn = self.get\_conn()
* if conn:
* try:
* cursor = conn.cursor()
* cursor.execute("SELECT COUNT(MaHang) FROM tblHang")
* stats["TotalProducts"] = cursor.fetchone()[0]
* cursor.execute("SELECT COUNT(MaKhach) FROM tblKhach")
* stats["TotalCustomers"] = cursor.fetchone()[0]
* except Exception:
* pass
* finally:
* conn.close()
* return stats
* def setup\_ui(self):
* # 1. Header
* tk.Label(self.master, text="📊 BẢNG ĐIỀU KHIỂN CHÍNH",
* font=("Arial", 20, "bold"), fg="#00796B", bg="#ECEFF1").pack(pady=(20, 15))
* # 2. Khung thống kê chính (KPI Cards)
* kpi\_frame = tk.Frame(self.master, bg="#ECEFF1", padx=10, pady=10)
* kpi\_frame.pack(fill="x", pady=10)
* stats = self.fetch\_stats()
* # Danh sách các thẻ KPI
* kpi\_cards = [
* {"title": "Tổng Doanh thu (VND)", "value": f"{stats['TotalRevenue']:,.0f}", "color": "#4CAF50", "visible": self.user\_role in [0, 1], "icon": "💵"},
* {"title": "Tổng Đơn hàng", "value": f"{stats['TotalOrders']:,.0f}", "color": "#03A9F4", "visible": self.user\_role in [0, 1], "icon": "🧾"},
* {"title": "Tổng Sản phẩm", "value": f"{stats['TotalProducts']:,.0f}", "color": "#FFC107", "visible": True, "icon": "📦"},
* {"title": "Tổng Khách hàng", "value": f"{stats['TotalCustomers']:,.0f}", "color": "#795548", "visible": True, "icon": "👤"},
* ]
* # Tạo các thẻ
* col = 0
* for card in kpi\_cards:
* if card["visible"]:
* self.create\_kpi\_card(kpi\_frame, card["title"], card["value"], card["color"], col, card["icon"])
* col += 1
* kpi\_frame.grid\_columnconfigure(0, weight=1)
* kpi\_frame.grid\_columnconfigure(1, weight=1)
* kpi\_frame.grid\_columnconfigure(2, weight=1)
* kpi\_frame.grid\_columnconfigure(3, weight=1)
* # 3. Khung chi tiết (Thông tin người dùng & Bán chạy)
* detail\_frame = tk.Frame(self.master, bg="#ECEFF1", padx=10, pady=10)
* detail\_frame.pack(fill="both", expand=True, pady=10)
* # Khung Thông tin người dùng
* user\_info\_frame = tk.LabelFrame(detail\_frame, text="Thông tin phiên làm việc", bg="#FFFFFF", padx=15, pady=15)
* user\_info\_frame.pack(side="left", fill="y", padx=(0, 20))
* user\_role\_text = MainApp.ROLE\_MAP.get(self.user\_role, 'Không rõ')
* tk.Label(user\_info\_frame, text=f"Tài khoản: ", font=("Arial", 11, "bold"), bg="#FFFFFF", fg="#2196F3").grid(row=0, column=0, sticky="w", pady=2)
* tk.Label(user\_info\_frame, text=self.user\_info['HoTen'], font=("Arial", 11), bg="#FFFFFF").grid(row=0, column=1, sticky="w", padx=10, pady=2)
* tk.Label(user\_info\_frame, text=f"Tên ĐN: ", font=("Arial", 11, "bold"), bg="#FFFFFF", fg="#2196F3").grid(row=1, column=0, sticky="w", pady=2)
* tk.Label(user\_info\_frame, text=self.user\_info['TenDangNhap'], font=("Arial", 11), bg="#FFFFFF").grid(row=1, column=1, sticky="w", padx=10, pady=2)
* tk.Label(user\_info\_frame, text=f"Quyền hạn: ", font=("Arial", 11, "bold"), bg="#FFFFFF", fg="#2196F3").grid(row=2, column=0, sticky="w", pady=2)
* tk.Label(user\_info\_frame, text=user\_role\_text, font=("Arial", 11, "bold"), bg="#FFFFFF", fg="#E65100").grid(row=2, column=1, sticky="w", padx=10, pady=2)
* # Khung Bán chạy nhất
* top\_selling\_frame = tk.LabelFrame(detail\_frame, text="Sản phẩm Bán chạy nhất (theo SL)", bg="#FFFFFF", padx=15, pady=15)
* top\_selling\_frame.pack(side="left", fill="both", expand=True)
* tk.Label(top\_selling\_frame, text=f"Tên Hàng: ", font=("Arial", 11, "bold"), bg="#FFFFFF", fg="#00796B").grid(row=0, column=0, sticky="w", pady=5)
* tk.Label(top\_selling\_frame, text=stats['TopProduct'], font=("Arial", 12, "bold"), bg="#FFFFFF", fg="#00796B").grid(row=0, column=1, sticky="w", padx=10, pady=5)
* tk.Label(top\_selling\_frame, text="💡 Dữ liệu này được cập nhật từ DB.", font=("Arial", 10, "italic"), bg="#FFFFFF", fg="#757575").grid(row=1, column=0, columnspan=2, sticky="w", pady=(10, 0))
* def create\_kpi\_card(self, parent, title, value, color, col, icon):
* """Tạo một thẻ KPI đơn giản."""
* card = tk.Frame(parent, bg="#FFFFFF", padx=15, pady=15, bd=1, relief=tk.SOLID)
* card.grid(row=0, column=col, padx=10, sticky="nsew")
* # Icon
* tk.Label(card, text=icon, font=("Arial", 24), bg="#FFFFFF", fg=color).pack(pady=(0, 5))
* # Title
* tk.Label(card, text=title, font=("Arial", 10), bg="#FFFFFF", fg="#757575").pack(pady=(0, 2))
* # Value
* tk.Label(card, text=value, font=("Arial", 18, "bold"), bg="#FFFFFF", fg=color).pack()
* # ------------------------------------------------
* # CÁC HÀM XỬ LÝ TRONG MAIN APP
* # ------------------------------------------------
* def set\_active\_menu(button):
* """Đặt button được chọn làm active và reset các button khác"""
* # Reset tất cả các button về màu mặc định
* for btn in MainApp.menu\_buttons:
* btn.config(bg="#263238")
* # Đặt button hiện tại thành active (màu nổi bật)
* button.config(bg="#00796B")
* MainApp.active\_button = button
* def open\_module(title, button=None):
* """Xử lý hiển thị form module tương ứng trong Content Frame"""
* try:
* print(f"--- Dang mo module: {title}")
* except Exception:
* # Fallback tránh crash nếu console không hỗ trợ Unicode
* print("--- Dang mo module")
* if not MainApp.content\_frame:
* print("Lỗi: Content Frame chưa được khởi tạo.")
* return
* # Highlight button được chọn
* if button:
* set\_active\_menu(button)
* # Xóa nội dung cũ trong Content Frame
* for widget in MainApp.content\_frame.winfo\_children():
* widget.destroy()
* # LOGIC TẢI FORM THEO TITLE
* if title == "Bảng điều khiển":
* FormClass = Dashboard
* elif title == "Quản lý Hàng hóa":
* FormClass = ProductManagementForm
* elif title == "Quản lý Khách hàng":
* FormClass = CustomerManagementForm
* elif title == "Quản lý Nhân viên":
* FormClass = EmployeeManagementForm
* elif title == "Quản lý Tài khoản":
* FormClass = AccountManagementForm
* elif title == "Báo cáo Doanh thu":
* FormClass = DetailedInvoiceReportForm
* elif title == "Lập Hóa đơn Bán hàng":
* FormClass = SalesInvoiceForm
* else:
* FormClass = None
* # Xử lý tải Form
* if FormClass:
* try:
* # Truyền user\_info nếu là Dashboard
* if title == "Bảng điều khiển" or title == "Lập Hóa đơn Bán hàng":
* FormClass(MainApp.content\_frame, MainApp.user\_info)
* else:
* FormClass(MainApp.content\_frame)
* except Exception as e:
* error\_msg = f"LỖI KHỞI TẠO FORM {title.upper()}: {e}"
* tk.Label(MainApp.content\_frame, text=error\_msg,
* font=("Arial", 16, "bold"), fg="#D32F2F", bg="#ECEFF1", wraplength=700).pack(expand=True, pady=50)
* print(f"❌ Lỗi khởi tạo {title}: {e}")
* else:
* # Placeholder cho các module chưa code
* tk.Label(MainApp.content\_frame, text=f"MODULE: {title} (Đang phát triển)",
* font=("Arial", 20, "bold"), fg="#E64A19", bg="#ECEFF1").pack(expand=True, pady=50)
* # ------------------------------------------------
* # LỚP ỨNG DỤNG CHÍNH (Giữ nguyên)
* # ------------------------------------------------
* class MainApp:
* ROLE\_MAP = { 0: 'ADMIN', 1: 'QUẢN LÝ', 2: 'NHÂN VIÊN' }
* MENU\_ITEMS = [
* {"text": "📊 Bảng điều khiển", "command": lambda: open\_module("Bảng điều khiển"), "roles": [0, 1, 2]},
* {"text": "🛒 Quản lý Hàng hóa", "command": lambda: open\_module("Quản lý Hàng hóa"), "roles": [0, 1]},
* {"text": "👤 Quản lý Khách hàng", "command": lambda: open\_module("Quản lý Khách hàng"), "roles": [0, 1, 2]},
* {"text": "👨‍💼 Quản lý Nhân viên", "command": lambda: open\_module("Quản lý Nhân viên"), "roles": [0]},
* {"text": "💰 Lập Hóa đơn Bán hàng", "command": lambda: open\_module("Lập Hóa đơn Bán hàng"), "roles": [0, 1, 2]},
* {"text": "📜 Báo cáo Doanh thu", "command": lambda: open\_module("Báo cáo Doanh thu"), "roles": [0, 1]},
* {"text": "⚙️ Quản lý Tài khoản", "command": lambda: open\_module("Quản lý Tài khoản"), "roles": [0]},
* ]
* user\_info = None
* content\_frame = None
* menu\_buttons = []  # Danh sách các button menu để quản lý highlight
* active\_button = None  # Button đang được chọn
* def \_\_init\_\_(self, root, user\_info, start\_login\_callback):
* self.root = root
* MainApp.user\_info = user\_info
* self.start\_login = start\_login\_callback
* self.user\_role = user\_info['PhanQuyen']
* self.setup\_ui()
* def setup\_ui(self):
* self.root.title(f"🏪 PHẦN MỀM QUẢN LÝ BÁN HÀNG - Quyền: {MainApp.ROLE\_MAP.get(self.user\_role, 'Không rõ')}")
* self.root.state('zoomed')
* self.root.configure(bg="#F4F4F4")
* # 1. HEADER FRAME
* header\_frame = tk.Frame(self.root, bg="#00796B", height=60)
* header\_frame.pack(fill="x")
* # Logo/Tiêu đề
* tk.Label(header\_frame, text="🏪 QUẢN LÝ BÁN HÀNG",
* font=("Arial", 18, "bold"), fg="white", bg="#00796B").pack(side="left", padx=20)
* # Thông tin người dùng
* user\_text = f"Xin chào {self.user\_info['HoTen']} | Quyền: {MainApp.ROLE\_MAP.get(self.user\_role, 'Không rõ')}"
* tk.Label(header\_frame, text=user\_text,
* font=("Arial", 12), fg="#E0F2F1", bg="#00796B").pack(side="right", padx=10)
* # Nút Đăng xuất
* tk.Button(header\_frame, text="Đăng xuất",
* font=("Arial", 10, "bold"), bg="#D32F2F", fg="white", relief=tk.FLAT,
* command=self.logout).pack(side="right", padx=10, pady=10)
* # 2. MAIN CONTAINER
* container = tk.Frame(self.root, bg="#F4F4F4")
* container.pack(expand=True, fill="both")
* # 3. SIDEBAR FRAME
* sidebar\_frame = tk.Frame(container, bg="#263238", width=250)
* sidebar\_frame.pack(side="left", fill="y")
* # Xóa danh sách button cũ (nếu có)
* MainApp.menu\_buttons = []
* for item in MainApp.MENU\_ITEMS:
* if self.user\_role in item["roles"]:
* # Tạo button với command mới có truyền button
* btn = tk.Button(sidebar\_frame, text=item["text"],
* font=("Arial", 11), fg="white", bg="#263238",
* activebackground="#37474F", activeforeground="white",
* anchor="w", bd=0, padx=20, pady=10, relief=tk.FLAT)
* btn.pack(fill="x", pady=1)
* # Lưu button vào danh sách
* MainApp.menu\_buttons.append(btn)
* # Cấu hình command với lambda để truyền cả title và button
* title = item["text"].split(" ", 1)[1]  # Lấy text không có icon
* btn.config(command=lambda t=title, b=btn: open\_module(t, b))
* # Giữ nguyên hover effect
* btn.bind("<Enter>", lambda e, b=btn: b.config(bg="#37474F") if b != MainApp.active\_button else None)
* btn.bind("<Leave>", lambda e, b=btn: b.config(bg="#263238") if b != MainApp.active\_button else b.config(bg="#00796B"))
* # 4. CONTENT FRAME
* MainApp.content\_frame = tk.Frame(container, bg="#ECEFF1")
* MainApp.content\_frame.pack(side="right", expand=True, fill="both")
* # Hiển thị mặc định (Dashboard) và set active cho button đầu tiên
* if MainApp.menu\_buttons:
* open\_module("Bảng điều khiển", MainApp.menu\_buttons[0])
* def logout(self):
* """Xử lý đăng xuất"""
* if messagebox.askyesno("Xác nhận", "Bạn có muốn đăng xuất không?"):
* self.root.destroy()
* self.start\_login()
* # ------------------------------------------------
* # LUỒNG CHẠY CHÍNH (Giữ nguyên)
* # ------------------------------------------------
* def start\_login():
* """Mở form đăng nhập sau khi splash xong"""
* import login\_form
* login\_root = tk.Tk()
* app = login\_form.LoginForm(login\_root, on\_success\_callback=start\_main\_form)
* login\_root.mainloop()
* def start\_main\_form(user\_info):
* """Form chính sau khi đăng nhập thành công"""
* main = tk.Tk()
* app = MainApp(main, user\_info, start\_login)
* main.mainloop()
* if \_\_name\_\_ == "\_\_main\_\_":
* # --- LUỒNG CHẠY CHÍNH (Splash -> Login -> Main) ---
* root = tk.Tk()
* # Khởi động bằng SplashScreen, rồi gọi start\_login khi xong
* app = SplashScreen(root, on\_done=start\_login)
* root.mainloop()

### **3. Các Module Quản lý Dữ liệu (CRUD Forms)**

Các form quản lý **(product\_management\_form.py, customer\_management\_form.py, employee\_management\_form.py, account\_management\_form.py)** đều tuân theo một kiến trúc đồng nhất:

***Product\_management***

*A screenshot of a computer

AI-generated content may be incorrect.*

* **Code product\_management**
* # -\*- coding: utf-8 -\*-
* import tkinter as tk
* from tkinter import ttk, messagebox
* import pyodbc
* import threading
* from auth import get\_connection
* class ProductManagementForm:
* def \_\_init\_\_(self, master):
* self.master = master
* if isinstance(self.master, tk.Tk):
* self.master.title("Quản lý Hàng hóa")
* self.master.config(bg="#ECEFF1")
* self.current\_state = 'VIEW'
* self.selected\_item = None
* self.chatlieu\_dict = {}
* main\_frame = tk.Frame(master, bg="#ECEFF1", padx=10, pady=10)
* main\_frame.pack(fill=tk.BOTH, expand=True)
* tk.Label(main\_frame, text="📦 QUẢN LÝ HÀNG HÓA",
* font=("Arial", 20, "bold"), fg="#00796B", bg="#ECEFF1").pack(pady=(0, 15))
* # --- Khung tìm kiếm ---
* search\_frame = tk.Frame(main\_frame, bg="#E0E0E0", padx=10, pady=5)
* search\_frame.pack(fill="x", pady=(0, 10))
* tk.Label(search\_frame, text="Tìm kiếm (Tên hàng):", bg="#E0E0E0").pack(side="left")
* self.search\_var = tk.StringVar()
* self.search\_entry = tk.Entry(search\_frame, textvariable=self.search\_var, width=40)
* self.search\_entry.pack(side="left", padx=(5, 10))
* ttk.Button(search\_frame, text="🔍 Tìm", command=self.search\_products).pack(side="left", padx=5)
* ttk.Button(search\_frame, text="🔄 Đặt lại", command=self.reset\_search).pack(side="left", padx=5)
* # --- Form nhập liệu ---
* form\_frame = tk.LabelFrame(main\_frame, text="Thông tin hàng hóa", bg="#ECEFF1", padx=10, pady=10)
* form\_frame.pack(fill=tk.X, pady=5)
* self.entries = {}
* labels = [
* ("Mã hàng:", "ma\_hang"),
* ("Tên hàng:", "ten\_hang"),
* ("Chất liệu:", "ma\_chatlieu"),
* ("Số lượng:", "so\_luong"),
* ("Đơn giá nhập:", "don\_gia\_nhap"),
* ("Đơn vị:", "don\_vi"),
* ("Ghi chú:", "ghi\_chu")
* ]
* for i, (label, field) in enumerate(labels):
* tk.Label(form\_frame, text=label, bg="#ECEFF1").grid(row=i, column=0, sticky="w", pady=3)
* if field == "ma\_chatlieu":
* combo = ttk.Combobox(form\_frame, state="normal")
* combo.grid(row=i, column=1, sticky="ew", pady=3)
* self.entries[field] = combo
* else:
* entry = tk.Entry(form\_frame)
* entry.grid(row=i, column=1, sticky="ew", pady=3)
* self.entries[field] = entry
* self.entries["ma\_hang"].config(state="readonly")
* form\_frame.columnconfigure(1, weight=1)
* # --- Nút chức năng ---
* button\_frame = tk.Frame(main\_frame, bg="#ECEFF1")
* button\_frame.pack(fill=tk.X, pady=5)
* self.btn\_add = tk.Button(button\_frame, text="➕ Thêm", command=self.set\_add\_state, bg="#AED581", width=10)
* self.btn\_save = tk.Button(button\_frame, text="💾 Lưu", command=self.save\_data, bg="#64B5F6", width=10, state=tk.DISABLED)
* self.btn\_edit = tk.Button(button\_frame, text="📝 Sửa", command=self.set\_edit\_state, bg="#FFB74D", width=10)
* self.btn\_delete = tk.Button(button\_frame, text="❌ Xóa", command=self.delete\_item, bg="#E57373", width=10)
* self.btn\_cancel = tk.Button(button\_frame, text="🗑️ Hủy", command=self.cancel\_action, bg="#90A4AE", width=10, state=tk.DISABLED)
* for b in [self.btn\_add, self.btn\_save, self.btn\_edit, self.btn\_delete, self.btn\_cancel]:
* b.pack(side=tk.LEFT, padx=5)
* # --- TreeView ---
* columns = ("MaHang", "TenHang", "TenChatLieu", "SoLuong", "DonGiaNhap", "DonVi", "GhiChu")
* self.tree = ttk.Treeview(main\_frame, columns=columns, show="headings", height=15)
* widths = [80, 200, 120, 80, 120, 80, 200]
* for col, text, width in zip(columns, ["Mã hàng", "Tên hàng", "Chất liệu", "SL", "Đơn giá", "Đơn vị", "Ghi chú"], widths):
* self.tree.heading(col, text=text, anchor="center")
* self.tree.column(col, width=width, anchor="center")
* self.tree.pack(fill=tk.BOTH, expand=True)
* self.tree.bind("<<TreeviewSelect>>", self.on\_tree\_select)
* self.tree.bind("<Double-Button-1>", self.on\_tree\_double\_click)
* # --- Thanh trạng thái ---
* self.status\_bar = tk.Label(main\_frame, text="Sẵn sàng.", bd=1, relief=tk.SUNKEN, anchor="w")
* self.status\_bar.pack(side="bottom", fill="x")
* self.set\_form\_state("VIEW")
* self.master.after(200, self.load\_reference\_data)
* self.master.after(300, self.load\_data)
* # =======================================================
* # KẾT NỐI & LOAD
* # =======================================================
* def get\_conn(self):
* try:
* return get\_connection()
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Kết nối SQL thất bại:\n{e}"))
* return None
* def load\_reference\_data(self):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* cursor.execute("SELECT MaChatLieu, TenChatLieu FROM tblChatLieu ORDER BY TenChatLieu")
* data = cursor.fetchall()
* self.chatlieu\_dict = {r.TenChatLieu: r.MaChatLieu for r in data}
* self.entries["ma\_chatlieu"]["values"] = list(self.chatlieu\_dict.keys())
* except Exception as e:
* messagebox.showerror("Lỗi", f"Lỗi tải chất liệu: {e}")
* finally:
* conn.close()
* def load\_data(self, search\_term=""):
* def \_task():
* conn = self.get\_conn()
* if not conn:
* self.master.after(0, lambda: self.status\_bar.config(text="❌ Lỗi kết nối"))
* return
* try:
* cursor = conn.cursor()
* query = """
* SELECT h.MaHang, h.TenHang, c.TenChatLieu, h.SoLuong, h.DonGiaNhap, h.DonVi, h.GhiChu
* FROM tblHang h
* LEFT JOIN tblChatLieu c ON h.MaChatLieu = c.MaChatLieu
* """
* params = []
* if search\_term:
* query += " WHERE h.TenHang COLLATE Vietnamese\_CI\_AI LIKE ?"
* params.append(f"%{search\_term}%")
* query += " ORDER BY h.MaHang"
* cursor.execute(query, params)
* rows = cursor.fetchall()
* self.master.after(0, lambda: self.\_update\_treeview(rows))
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Tải dữ liệu thất bại:\n{e}"))
* finally:
* conn.close()
* self.status\_bar.config(text="🔄 Đang tải dữ liệu...")
* threading.Thread(target=\_task, daemon=True).start()
* def \_update\_treeview(self, rows):
* self.tree.delete(\*self.tree.get\_children())
* for row in rows:
* self.tree.insert('', tk.END, values=(
* row.MaHang, row.TenHang, row.TenChatLieu,
* row.SoLuong, f"{row.DonGiaNhap:,.0f}", row.DonVi, row.GhiChu
* ))
* self.status\_bar.config(text=f"✅ Đã tải {len(rows)} bản ghi.")
* # =======================================================
* # STATE
* # =======================================================
* def set\_form\_state(self, state):
* self.current\_state = state
* editable = state in ("ADD", "EDIT")
* for field, entry in self.entries.items():
* if field == "ma\_hang":
* # Mã hàng luôn readonly
* entry.config(state="readonly")
* elif field == "ma\_chatlieu":
* # Chất liệu: editable khi ADD/EDIT, disabled khi VIEW
* entry.config(state="normal" if editable else "disabled")
* else:
* # Các trường khác: editable khi ADD/EDIT, disabled khi VIEW
* entry.config(state="normal" if editable else "disabled")
* has\_selection = self.selected\_item is not None
* self.btn\_add.config(state=tk.NORMAL if state == "VIEW" else tk.DISABLED)
* self.btn\_edit.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_delete.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_save.config(state=tk.NORMAL if editable else tk.DISABLED)
* self.btn\_cancel.config(state=tk.NORMAL if editable else tk.DISABLED)
* def set\_add\_state(self):
* self.clear\_entries()
* # Lấy mã hàng tiếp theo
* next\_id = self.get\_next\_mahang()
* if next\_id:
* self.entries["ma\_hang"].config(state="normal")
* self.entries["ma\_hang"].delete(0, tk.END)
* self.entries["ma\_hang"].insert(0, next\_id)
* self.entries["ma\_hang"].config(state="readonly")
* self.set\_form\_state("ADD")
* self.entries["ten\_hang"].focus()
* def get\_next\_mahang(self):
* """Lấy mã hàng tiếp theo, ưu tiên lấp khoảng trống (gap) nếu có."""
* conn = self.get\_conn()
* if not conn:
* return None
* try:
* cursor = conn.cursor()
* # Tìm khoảng trống (gap) trong dãy mã
* cursor.execute("""
* SELECT MIN(t1.MaHang + 1) AS NextID
* FROM tblHang t1
* WHERE NOT EXISTS (
* SELECT 1 FROM tblHang t2
* WHERE t2.MaHang = t1.MaHang + 1
* )
* AND t1.MaHang + 1 <= (SELECT MAX(MaHang) FROM tblHang)
* """)
* result = cursor.fetchone()
* gap\_id = result[0] if result and result[0] else None
* if gap\_id:
* # Có khoảng trống → dùng mã đó
* return gap\_id
* else:
* # Không có khoảng trống → lấy MAX + 1
* cursor.execute("SELECT MAX(MaHang) FROM tblHang")
* result = cursor.fetchone()
* max\_id = result[0] if result and result[0] else 0
* return max\_id + 1
* except Exception as e:
* messagebox.showerror("Lỗi", f"Không thể lấy mã hàng tiếp theo:\n{e}")
* return None
* finally:
* conn.close()
* def set\_edit\_state(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn hàng để sửa!")
* return
* # Đảm bảo dữ liệu đã được load lên form
* values = self.tree.item(self.selected\_item)['values']
* if len(values) >= 7:
* self.entries["ma\_hang"].config(state="normal")
* self.entries["ma\_hang"].delete(0, tk.END)
* self.entries["ma\_hang"].insert(0, values[0])
* self.entries["ma\_hang"].config(state="readonly")
* self.entries["ten\_hang"].delete(0, tk.END)
* self.entries["ten\_hang"].insert(0, values[1])
* self.entries["ma\_chatlieu"].set(values[2])
* self.entries["so\_luong"].delete(0, tk.END)
* self.entries["so\_luong"].insert(0, values[3])
* self.entries["don\_gia\_nhap"].delete(0, tk.END)
* gia\_nhap = str(values[4]).replace(",", "").replace(".", "")
* self.entries["don\_gia\_nhap"].insert(0, gia\_nhap)
* self.entries["don\_vi"].delete(0, tk.END)
* self.entries["don\_vi"].insert(0, values[5])
* self.entries["ghi\_chu"].delete(0, tk.END)
* self.entries["ghi\_chu"].insert(0, values[6])
* self.set\_form\_state("EDIT")
* self.entries["ten\_hang"].focus()
* def cancel\_action(self):
* self.clear\_entries()
* self.set\_form\_state("VIEW")
* self.load\_data()
* def clear\_entries(self):
* for entry in self.entries.values():
* entry.config(state="normal")
* if isinstance(entry, ttk.Combobox):
* entry.set("")
* else:
* entry.delete(0, tk.END)
* self.selected\_item = None
* # =======================================================
* # CRUD (DÙNG THREAD)
* # =======================================================
* def save\_data(self):
* ma\_hang, data = self.get\_validated\_data(self.current\_state == 'ADD')
* if data is None: return
* if self.current\_state == "ADD":
* self.status\_bar.config(text="🔄 Đang thêm hàng hóa...")
* threading.Thread(target=self.\_execute\_add\_item, args=(data,), daemon=True).start()
* elif self.current\_state == "EDIT":
* self.status\_bar.config(text=f"🔄 Đang cập nhật Mã {ma\_hang}...")
* threading.Thread(target=self.\_execute\_update\_item, args=(ma\_hang, data), daemon=True).start()
* def \_execute\_add\_item(self, data):
* try:
* conn = self.get\_conn()
* if not conn: return
* cursor = conn.cursor()
* ma\_chatlieu = self.process\_chatlieu(cursor, data["TenChatLieu"], conn)
* # Lấy mã hàng từ form (đã tính toán gap)
* ma\_hang\_moi = int(self.entries["ma\_hang"].get())
* # Bật IDENTITY\_INSERT để chèn mã tùy chỉnh
* cursor.execute("SET IDENTITY\_INSERT tblHang ON")
* cursor.execute("""
* INSERT INTO tblHang (MaHang, TenHang, MaChatLieu, SoLuong, DonGiaNhap, DonVi, GhiChu)
* VALUES (?, ?, ?, ?, ?, ?, ?)
* """, (ma\_hang\_moi, data["TenHang"], ma\_chatlieu, data["SoLuong"], data["DonGiaNhap"], data["DonVi"], data["GhiChu"]))
* cursor.execute("SET IDENTITY\_INSERT tblHang OFF")
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã thêm hàng mới với Mã {ma\_hang\_moi}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã thêm Mã {ma\_hang\_moi}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Thêm thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* def \_execute\_update\_item(self, ma\_hang, data):
* try:
* conn = self.get\_conn()
* if not conn: return
* cursor = conn.cursor()
* ma\_chatlieu = self.process\_chatlieu(cursor, data["TenChatLieu"], conn)
* cursor.execute("""
* UPDATE tblHang
* SET TenHang=?, MaChatLieu=?, SoLuong=?, DonGiaNhap=?, DonVi=?, GhiChu=?
* WHERE MaHang=?
* """, (data["TenHang"], ma\_chatlieu, data["SoLuong"], data["DonGiaNhap"], data["DonVi"], data["GhiChu"], ma\_hang))
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", "Đã cập nhật hàng hóa!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã cập nhật Mã {ma\_hang}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Cập nhật thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* def delete\_item(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn hàng để xóa!")
* return
* ma\_hang = self.entries["ma\_hang"].get()
* if not messagebox.askyesno("Xác nhận", f"Xóa hàng hóa Mã {ma\_hang}?"):
* return
* self.status\_bar.config(text=f"🔄 Đang xóa Mã {ma\_hang}...")
* threading.Thread(target=self.\_execute\_delete\_item, args=(ma\_hang,), daemon=True).start()
* def \_execute\_delete\_item(self, ma\_hang):
* try:
* conn = self.get\_conn()
* if not conn: return
* cursor = conn.cursor()
* cursor.execute("DELETE FROM tblHang WHERE MaHang=?", ma\_hang)
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã xóa Mã {ma\_hang}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã xóa Mã {ma\_hang}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Xóa thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* # =======================================================
* # TIỆN ÍCH
* # =======================================================
* def get\_validated\_data(self, is\_add):
* ma\_hang = self.entries['ma\_hang'].get().strip() if not is\_add else None
* ten\_hang = self.entries['ten\_hang'].get().strip()
* ten\_chatlieu = self.entries['ma\_chatlieu'].get().strip()
* so\_luong = self.entries['so\_luong'].get().strip() or "0"
* don\_gia = self.entries['don\_gia\_nhap'].get().strip() or "0"
* don\_vi = self.entries['don\_vi'].get().strip()
* ghi\_chu = self.entries['ghi\_chu'].get().strip()
* try:
* so\_luong = int(so\_luong)
* don\_gia = float(don\_gia.replace(",", ""))
* except:
* messagebox.showwarning("Lỗi", "Số lượng hoặc đơn giá không hợp lệ.")
* return None, None
* if not ten\_hang or not ten\_chatlieu:
* messagebox.showwarning("Thiếu thông tin", "Tên hàng và Chất liệu không được để trống.")
* return None, None
* return ma\_hang, {
* "TenHang": ten\_hang,
* "TenChatLieu": ten\_chatlieu,
* "SoLuong": so\_luong,
* "DonGiaNhap": don\_gia,
* "DonVi": don\_vi,
* "GhiChu": ghi\_chu
* }
* def process\_chatlieu(self, cursor, ten\_chatlieu, conn):
* if ten\_chatlieu in self.chatlieu\_dict:
* return self.chatlieu\_dict[ten\_chatlieu]
* cursor.execute("INSERT INTO tblChatLieu (TenChatLieu) VALUES (?)", ten\_chatlieu)
* conn.commit()
* cursor.execute("SELECT @@IDENTITY")
* new\_id = int(cursor.fetchone()[0])
* self.chatlieu\_dict[ten\_chatlieu] = new\_id
* return new\_id
* def search\_products(self):
* """Hàm tìm kiếm sản phẩm theo tên hoặc chất liệu."""
* keyword = self.search\_var.get().strip().lower()
* if not keyword:
* self.status\_bar.config(text="🔍 Vui lòng nhập từ khóa tìm kiếm!")
* return
* # Xóa dữ liệu cũ trên TreeView
* for item in self.tree.get\_children():
* self.tree.delete(item)
* # Mở kết nối DB và tìm kiếm
* try:
* conn = get\_connection()
* cursor = conn.cursor()
* query = """
* SELECT h.MaHang, h.TenHang, c.TenChatLieu, h.SoLuong, h.DonGiaNhap, h.DonVi, h.GhiChu
* FROM tblHang AS h
* JOIN tblChatLieu AS c ON h.MaChatLieu = c.MaChatLieu
* WHERE LOWER(h.TenHang) LIKE ? OR LOWER(c.TenChatLieu) LIKE ?
* """
* cursor.execute(query, ('%' + keyword + '%', '%' + keyword + '%'))
* rows = cursor.fetchall()
* for row in rows:
* self.tree.insert("", "end", values=row)
* self.status\_bar.config(text=f"🔍 Tìm thấy {len(rows)} sản phẩm khớp '{keyword}'")
* except Exception as e:
* messagebox.showerror("Lỗi", f"Lỗi khi tìm kiếm: {e}")
* finally:
* if conn:
* conn.close()
* def reset\_search(self):
* """Đặt lại ô tìm kiếm và tải lại toàn bộ dữ liệu."""
* self.search\_var.set("")
* self.load\_data()
* self.status\_bar.config(text="🔄 Đã tải lại toàn bộ dữ liệu.")
* def on\_tree\_select(self, event):
* """Xử lý khi chọn một dòng trong Treeview."""
* # Chỉ xử lý khi đang ở chế độ VIEW
* if self.current\_state != "VIEW":
* return
* selected = self.tree.selection()
* if not selected:
* self.selected\_item = None
* self.set\_form\_state("VIEW")
* return
* self.selected\_item = selected[0]
* values = self.tree.item(self.selected\_item)['values']
* if len(values) >= 7:
* # Clear tất cả trước
* for key in self.entries:
* if key == "ma\_chatlieu":
* self.entries[key].set("")
* else:
* self.entries[key].config(state="normal")
* self.entries[key].delete(0, tk.END)
* # Điền dữ liệu mới
* self.entries["ma\_hang"].insert(0, str(values[0]))
* self.entries["ma\_hang"].config(state="readonly")
* self.entries["ten\_hang"].insert(0, str(values[1]) if values[1] else "")
* self.entries["ma\_chatlieu"].set(str(values[2]) if values[2] else "")
* self.entries["so\_luong"].insert(0, str(values[3]) if values[3] else "0")
* # Xóa dấu phẩy trong giá trước khi hiển thị
* gia\_nhap = str(values[4]).replace(",", "").replace(".", "") if values[4] else "0"
* self.entries["don\_gia\_nhap"].insert(0, gia\_nhap)
* self.entries["don\_vi"].insert(0, str(values[5]) if values[5] else "")
* self.entries["ghi\_chu"].insert(0, str(values[6]) if values[6] else "")
* # Giữ nguyên state VIEW, cập nhật lại các nút
* self.set\_form\_state("VIEW")
* def on\_tree\_double\_click(self, event):
* """Double-click vào dòng → tự động chuyển sang chế độ Sửa"""
* if self.selected\_item and self.current\_state == "VIEW":
* self.set\_edit\_state()
* if \_\_name\_\_ == "\_\_main\_\_":
* root = tk.Tk()
* root.state('zoomed')
* app = ProductManagementForm(root)
* root.mainloop()

***Customer\_management***

*A screenshot of a computer

AI-generated content may be incorrect.*

* **Code customer\_management**
* # -\*- coding: utf-8 -\*-
* import tkinter as tk
* from tkinter import ttk, messagebox
* import pyodbc
* import threading
* from auth import get\_connection # Giả định module auth.py tồn tại và cung cấp get\_connection()
* class CustomerManagementForm:
* def \_\_init\_\_(self, master):
* self.master = master
* if isinstance(self.master, tk.Tk):
* self.master.title("Quản lý Khách hàng")
* self.master.config(bg="#ECEFF1")
* self.current\_state = 'VIEW'
* self.selected\_item = None
* main\_frame = tk.Frame(master, bg="#ECEFF1", padx=10, pady=10)
* main\_frame.pack(fill=tk.BOTH, expand=True)
* tk.Label(main\_frame, text="👤 QUẢN LÝ KHÁCH HÀNG",
* font=("Arial", 20, "bold"), fg="#00796B", bg="#ECEFF1").pack(pady=(0, 15))
* # --- Khung tìm kiếm (Giống ProductManagementForm) ---
* search\_frame = tk.Frame(main\_frame, bg="#E0E0E0", padx=10, pady=5)
* search\_frame.pack(fill="x", pady=(0, 10))
* tk.Label(search\_frame, text="Tìm kiếm (Tên khách):", bg="#E0E0E0").pack(side="left")
* self.search\_var = tk.StringVar()
* self.search\_entry = tk.Entry(search\_frame, textvariable=self.search\_var, width=40)
* self.search\_entry.pack(side="left", padx=(5, 10))
* ttk.Button(search\_frame, text="🔍 Tìm", command=self.search\_customers).pack(side="left", padx=5)
* ttk.Button(search\_frame, text="🔄 Đặt lại", command=self.reset\_search).pack(side="left", padx=5)
* # --- Form nhập liệu (Sử dụng LabelFrame và bố cục 2 cột) ---
* form\_frame = tk.LabelFrame(main\_frame, text="Thông tin khách hàng", bg="#ECEFF1", padx=10, pady=10)
* form\_frame.pack(fill=tk.X, pady=5)
* self.entries = {}
* labels = [
* ("Mã Khách:", "ma\_khach"),
* ("Tên Khách:", "ten\_khach"),
* ("Điện thoại:", "dien\_thoai"),
* ("Địa chỉ:", "dia\_chi"), # Đưa Địa chỉ xuống dòng 2 (cột trái)
* ]
* for i, (label, field) in enumerate(labels):
* row = i // 2
* col = (i % 2) \* 2
* tk.Label(form\_frame, text=label, bg="#ECEFF1").grid(row=row, column=col, sticky="w", padx=(5, 0), pady=3)
* entry = tk.Entry(form\_frame)
* entry.grid(row=row, column=col + 1, sticky="ew", padx=5, pady=3)
* self.entries[field] = entry
* self.entries["ma\_khach"].config(state="readonly")
* form\_frame.columnconfigure(1, weight=1)
* form\_frame.columnconfigure(3, weight=1)
* # --- Nút chức năng (Sử dụng tk.Button với màu sắc) ---
* button\_frame = tk.Frame(main\_frame, bg="#ECEFF1")
* button\_frame.pack(fill=tk.X, pady=5)
* self.btn\_add = tk.Button(button\_frame, text="➕ Thêm", command=self.set\_add\_state, bg="#AED581", width=10)
* self.btn\_save = tk.Button(button\_frame, text="💾 Lưu", command=self.save\_data, bg="#64B5F6", width=10, state=tk.DISABLED)
* self.btn\_edit = tk.Button(button\_frame, text="📝 Sửa", command=self.set\_edit\_state, bg="#FFB74D", width=10)
* self.btn\_delete = tk.Button(button\_frame, text="❌ Xóa", command=self.delete\_customer, bg="#E57373", width=10)
* self.btn\_cancel = tk.Button(button\_frame, text="🗑️ Hủy", command=self.cancel\_action, bg="#90A4AE", width=10, state=tk.DISABLED)
* for b in [self.btn\_add, self.btn\_save, self.btn\_edit, self.btn\_delete, self.btn\_cancel]:
* b.pack(side=tk.LEFT, padx=5)
* # --- TreeView ---
* columns = ("MaKhach", "TenKhach", "DiaChi", "DienThoai")
* self.tree = ttk.Treeview(main\_frame, columns=columns, show="headings", height=15)
* widths = [100, 220, 300, 130]
* for col, text, width in zip(columns, ["Mã Khách", "Tên Khách", "Địa chỉ", "Điện thoại"], widths):
* self.tree.heading(col, text=text, anchor="center")
* self.tree.column(col, width=width, anchor="center")
* self.tree.pack(fill=tk.BOTH, expand=True)
* # Thanh cuộn (giữ nguyên)
* scrollbar = ttk.Scrollbar(self.tree, orient="vertical", command=self.tree.yview)
* self.tree.configure(yscrollcommand=scrollbar.set)
* scrollbar.pack(side="right", fill="y")
* # Style xen kẽ (giữ nguyên)
* self.tree.tag\_configure('oddrow', background="#F5F5F5")
* self.tree.tag\_configure('evenrow', background="#FFFFFF")
* # --- Bắt sự kiện chọn ---
* self.tree.bind("<<TreeviewSelect>>", self.on\_tree\_select) # Đổi tên hàm
* self.tree.bind("<Double-Button-1>", self.on\_tree\_double\_click)
* # --- Thanh trạng thái ---
* self.status\_bar = tk.Label(main\_frame, text="Sẵn sàng.", bd=1, relief=tk.SUNKEN, anchor="w")
* self.status\_bar.pack(side="bottom", fill="x")
* self.set\_form\_state("VIEW")
* self.master.after(300, self.load\_data) # Bắt đầu tải dữ liệu bằng Threading
* # =======================================================
* # KẾT NỐI & LOAD (Dùng Threading, giống ProductForm)
* # =======================================================
* def get\_conn(self):
* try:
* return get\_connection()
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Kết nối SQL thất bại:\n{e}"))
* return None
* def load\_data(self, search\_term=""):
* def \_task():
* conn = self.get\_conn()
* if not conn:
* self.master.after(0, lambda: self.status\_bar.config(text="❌ Lỗi kết nối"))
* return
* try:
* cursor = conn.cursor()
* query = "SELECT MaKhach, TenKhach, DiaChi, DienThoai FROM tblKhach"
* params = []
* if search\_term:
* # Sửa lại query để chỉ tìm kiếm theo TenKhach (theo yêu cầu ban đầu)
* query += " WHERE TenKhach COLLATE Vietnamese\_CI\_AI LIKE ?"
* params.append(f"%{search\_term}%")
* query += " ORDER BY MaKhach ASC"
* cursor.execute(query, params)
* rows = cursor.fetchall()
* self.master.after(0, lambda: self.\_update\_treeview(rows))
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Tải dữ liệu thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* self.status\_bar.config(text="🔄 Đang tải dữ liệu...")
* threading.Thread(target=\_task, daemon=True).start()
* def \_update\_treeview(self, rows):
* self.tree.delete(\*self.tree.get\_children())
* for i, row in enumerate(rows):
* formatted\_row = (
* row.MaKhach, row.TenKhach, row.DiaChi or '', row.DienThoai or ''
* )
* tag = 'evenrow' if i % 2 == 0 else 'oddrow'
* self.tree.insert('', tk.END, values=formatted\_row, tags=(tag,))
* self.status\_bar.config(text=f"✅ Đã tải {len(rows)} bản ghi.")
* def search\_customers(self):
* keyword = self.search\_var.get().strip()
* if not keyword:
* self.status\_bar.config(text="🔍 Vui lòng nhập từ khóa tìm kiếm!")
* return
* self.load\_data(keyword)
* def reset\_search(self):
* self.search\_var.set("")
* self.load\_data()
* self.status\_bar.config(text="🔄 Đã tải lại toàn bộ dữ liệu.")
* # =======================================================
* # STATE (Đã đồng bộ với ProductForm)
* # =======================================================
* def set\_form\_state(self, state):
* self.current\_state = state
* editable = state in ("ADD", "EDIT")
* for field, entry in self.entries.items():
* if field == "ma\_khach":
* entry.config(state="readonly")
* else:
* entry.config(state=tk.NORMAL if editable else tk.DISABLED)
* has\_selection = self.selected\_item is not None
* self.btn\_add.config(state=tk.NORMAL if state == "VIEW" else tk.DISABLED)
* self.btn\_edit.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_delete.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_save.config(state=tk.NORMAL if editable else tk.DISABLED)
* self.btn\_cancel.config(state=tk.NORMAL if editable else tk.DISABLED)
* def set\_add\_state(self):
* self.clear\_entries()
* next\_id = self.get\_next\_makhach()
* if next\_id is not None:
* self.entries["ma\_khach"].config(state="normal")
* self.entries["ma\_khach"].delete(0, tk.END)
* self.entries["ma\_khach"].insert(0, next\_id)
* self.entries["ma\_khach"].config(state="readonly")
* self.set\_form\_state("ADD")
* self.entries["ten\_khach"].focus()
* def get\_next\_makhach(self):
* """Lấy mã khách tiếp theo (tìm gap ID), đã đồng bộ với ProductForm."""
* conn = self.get\_conn()
* if not conn: return None
* try:
* cursor = conn.cursor()
* # Tìm khoảng trống (gap)
* cursor.execute("""
* SELECT MIN(t1.MaKhach + 1) AS NextID
* FROM tblKhach t1
* WHERE NOT EXISTS (
* SELECT 1 FROM tblKhach t2
* WHERE t2.MaKhach = t1.MaKhach + 1
* )
* AND t1.MaKhach + 1 <= (SELECT MAX(MaKhach) FROM tblKhach)
* """)
* result = cursor.fetchone()
* gap\_id = result[0] if result and result[0] else None
* if gap\_id:
* return gap\_id
* else:
* cursor.execute("SELECT MAX(MaKhach) FROM tblKhach")
* result = cursor.fetchone()
* max\_id = result[0] if result and result[0] else 0
* return max\_id + 1
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Không thể lấy mã khách tiếp theo:\n{e}"))
* return None
* finally:
* conn.close()
* def set\_edit\_state(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn khách hàng để sửa!")
* return
* # Dữ liệu đã được điền trong on\_tree\_select
* self.set\_form\_state("EDIT")
* self.entries["ten\_khach"].focus()
* def cancel\_action(self):
* self.clear\_entries()
* self.set\_form\_state("VIEW")
* self.load\_data()
* def clear\_entries(self):
* self.entries['ma\_khach'].config(state='normal')
* for entry in self.entries.values():
* entry.config(state="normal")
* entry.delete(0, tk.END)
* self.entries['ma\_khach'].config(state='readonly')
* self.selected\_item = None
* self.tree.selection\_remove(self.tree.selection())
* # =======================================================
* # CRUD (Dùng Threading, giống ProductForm)
* # =======================================================
* def save\_data(self):
* ma\_khach, data = self.get\_validated\_data()
* if data is None: return
* if self.current\_state == "ADD":
* next\_id = int(self.entries["ma\_khach"].get()) # Lấy ID đã tính gap
* self.status\_bar.config(text="🔄 Đang thêm khách hàng...")
* threading.Thread(target=self.\_execute\_add\_item, args=(next\_id, data,), daemon=True).start()
* elif self.current\_state == "EDIT":
* self.status\_bar.config(text=f"🔄 Đang cập nhật Mã {ma\_khach}...")
* threading.Thread(target=self.\_execute\_update\_item, args=(ma\_khach, data), daemon=True).start()
* def \_execute\_add\_item(self, next\_id, data):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* # Bật IDENTITY\_INSERT để chèn mã tùy chỉnh
* cursor.execute("SET IDENTITY\_INSERT tblKhach ON")
* cursor.execute("""
* INSERT INTO tblKhach (MaKhach, TenKhach, DiaChi, DienThoai)
* VALUES (?, ?, ?, ?)
* """, (next\_id, data['TenKhach'], data['DiaChi'], data['DienThoai']))
* cursor.execute("SET IDENTITY\_INSERT tblKhach OFF")
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã thêm khách hàng với Mã {next\_id}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã thêm Mã {next\_id}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Thêm thất bại:\n{e}"))
* finally:
* if conn:
* try: cursor.execute("SET IDENTITY\_INSERT tblKhach OFF")
* except: pass
* conn.close()
* def \_execute\_update\_item(self, ma\_khach, data):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* cursor.execute("""
* UPDATE tblKhach
* SET TenKhach=?, DiaChi=?, DienThoai=?
* WHERE MaKhach=?
* """, (data['TenKhach'], data['DiaChi'], data['DienThoai'], ma\_khach))
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã cập nhật Mã {ma\_khach}!"),
* self.load\_data(), # Tải lại để cập nhật TreeView
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã cập nhật Mã {ma\_khach}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Cập nhật thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* def delete\_customer(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn khách hàng để xóa!")
* return
* ma\_khach = self.entries["ma\_khach"].get()
* if not messagebox.askyesno("Xác nhận", f"Xóa khách hàng Mã {ma\_khach}?"):
* return
* self.status\_bar.config(text=f"🔄 Đang xóa Mã {ma\_khach}...")
* threading.Thread(target=self.\_execute\_delete\_item, args=(ma\_khach,), daemon=True).start()
* def \_execute\_delete\_item(self, ma\_khach):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* cursor.execute("DELETE FROM tblKhach WHERE MaKhach=?", (ma\_khach,))
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã xóa Mã {ma\_khach}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã xóa Mã {ma\_khach}")
* ])
* except pyodbc.IntegrityError:
* self.master.after(0, lambda: messagebox.showerror("Lỗi DB", f"Không thể xóa khách hàng Mã {ma\_khach} vì có dữ liệu liên quan (Hóa đơn, ...)"))
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Xóa thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* # =======================================================
* # TIỆN ÍCH
* # =======================================================
* def get\_validated\_data(self):
* ma\_khach = self.entries['ma\_khach'].get().strip()
* ten\_khach = self.entries['ten\_khach'].get().strip()
* dia\_chi = self.entries['dia\_chi'].get().strip()
* dien\_thoai = self.entries['dien\_thoai'].get().strip()
* if not ten\_khach:
* messagebox.showwarning("Thiếu thông tin", "Tên khách hàng không được để trống.")
* return None, None
* return ma\_khach, {
* "TenKhach": ten\_khach,
* "DiaChi": dia\_chi,
* "DienThoai": dien\_thoai,
* }
* def on\_tree\_select(self, event):
* """Xử lý khi chọn một dòng trong Treeview."""
* if self.current\_state != "VIEW":
* return
* selected = self.tree.selection()
* if not selected:
* self.selected\_item = None
* self.set\_form\_state("VIEW")
* return
* self.selected\_item = selected[0]
* values = self.tree.item(self.selected\_item)['values']
* if values:
* # Tương tự ProductForm, clear entries và điền dữ liệu
* self.entries['ma\_khach'].config(state='normal')
* for entry in self.entries.values():
* entry.config(state='normal')
* entry.delete(0, tk.END)
* self.entries["ma\_khach"].insert(0, str(values[0]))
* self.entries["ma\_khach"].config(state="readonly")
* self.entries["ten\_khach"].insert(0, str(values[1]) if values[1] else "")
* self.entries["dia\_chi"].insert(0, str(values[2]) if values[2] else "")
* self.entries["dien\_thoai"].insert(0, str(values[3]) if values[3] else "")
* self.set\_form\_state("VIEW")
* def on\_tree\_double\_click(self, event):
* """Double-click vào dòng → tự động chuyển sang chế độ Sửa"""
* if self.selected\_item and self.current\_state == "VIEW":
* self.set\_edit\_state()
* if \_\_name\_\_ == "\_\_main\_\_":
* root = tk.Tk()
* root.title("QUẢN LÝ KHÁCH HÀNG")
* root.state('zoomed')
* app = CustomerManagementForm(root)
* root.mainloop()

***Employee\_management***

*A screenshot of a computer

AI-generated content may be incorrect.*

* **Code employee\_management**
* # -\*- coding: utf-8 -\*-
* # employee\_management\_form.py - Module Quản lý Nhân viên (Dùng Date Picker)
* import tkinter as tk
* from tkinter import ttk, messagebox
* import pyodbc
* import threading
* from datetime import datetime
* from tkcalendar import DateEntry
* from auth import get\_connection # Giả định module auth.py tồn tại và cung cấp get\_connection()
* class EmployeeManagementForm:
* def \_\_init\_\_(self, master):
* self.master = master
* if isinstance(self.master, tk.Tk):
* self.master.title("Quản lý Nhân viên")
* self.master.config(bg="#ECEFF1")
* self.current\_state = 'VIEW'
* self.selected\_item = None
* main\_frame = tk.Frame(master, bg="#ECEFF1", padx=10, pady=10)
* main\_frame.pack(fill=tk.BOTH, expand=True)
* tk.Label(main\_frame, text="👨‍💼 QUẢN LÝ NHÂN VIÊN",
* font=("Arial", 20, "bold"), fg="#00796B", bg="#ECEFF1").pack(pady=(0, 15))
* # --- Khung tìm kiếm (Giống ProductManagementForm) ---
* search\_frame = tk.Frame(main\_frame, bg="#E0E0E0", padx=10, pady=5)
* search\_frame.pack(fill="x", pady=(0, 10))
* tk.Label(search\_frame, text="Tìm kiếm (Tên/SĐT):", bg="#E0E0E0").pack(side="left")
* self.search\_var = tk.StringVar()
* self.search\_entry = tk.Entry(search\_frame, textvariable=self.search\_var, width=40)
* self.search\_entry.pack(side="left", padx=(5, 10))
* ttk.Button(search\_frame, text="🔍 Tìm", command=self.search\_employees).pack(side="left", padx=5)
* ttk.Button(search\_frame, text="🔄 Đặt lại", command=self.reset\_search).pack(side="left", padx=5)
* # --- Form nhập liệu (Giống ProductManagementForm) ---
* form\_frame = tk.LabelFrame(main\_frame, text="Thông tin nhân viên", bg="#ECEFF1", padx=10, pady=10)
* form\_frame.pack(fill=tk.X, pady=5)
* self.entries = {}
* # Danh sách các trường
* labels = [
* ("Mã NV:", "ma\_nhanvien"),
* ("Tên NV:", "ten\_nhanvien"),
* ("Giới tính:", "gioi\_tinh"),
* ("Ngày sinh:", "ngay\_sinh"),
* ("Điện thoại:", "dien\_thoai"),
* ("Địa chỉ:", "dia\_chi"),
* ]
* for i, (label, field) in enumerate(labels):
* row = i // 2
* col = (i % 2) \* 2
* tk.Label(form\_frame, text=label, bg="#ECEFF1").grid(row=row, column=col, sticky="w", padx=(5, 0), pady=3)
* if field == "gioi\_tinh":
* combo = ttk.Combobox(form\_frame, state="readonly", values=["Nam", "Nữ", "Khác"])
* combo.grid(row=row, column=col + 1, sticky="ew", padx=5, pady=3)
* self.entries[field] = combo
* self.entries[field].set("Nam")
* elif field == "ngay\_sinh":
* date\_entry = DateEntry(
* form\_frame,
* date\_pattern='yyyy-mm-dd',
* locale='vi\_VN',
* font=("Arial", 10),
* selectbackground="#00796B",
* selectforeground="white",
* headersbackground="#00796B",
* headersforeground="white"
* )
* date\_entry.grid(row=row, column=col + 1, sticky="ew", padx=5, pady=3)
* self.entries[field] = date\_entry
* self.entries[field].set\_date(datetime.now().date())
* else:
* entry = tk.Entry(form\_frame)
* entry.grid(row=row, column=col + 1, sticky="ew", padx=5, pady=3)
* self.entries[field] = entry
* self.entries["ma\_nhanvien"].config(state="readonly")
* form\_frame.columnconfigure(1, weight=1)
* form\_frame.columnconfigure(3, weight=1) # Cột cho trường bên phải
* # --- Nút chức năng (Giống ProductManagementForm) ---
* button\_frame = tk.Frame(main\_frame, bg="#ECEFF1")
* button\_frame.pack(fill=tk.X, pady=5)
* self.btn\_add = tk.Button(button\_frame, text="➕ Thêm", command=self.set\_add\_state, bg="#AED581", width=10)
* self.btn\_save = tk.Button(button\_frame, text="💾 Lưu", command=self.save\_data, bg="#64B5F6", width=10, state=tk.DISABLED)
* self.btn\_edit = tk.Button(button\_frame, text="📝 Sửa", command=self.set\_edit\_state, bg="#FFB74D", width=10)
* self.btn\_delete = tk.Button(button\_frame, text="❌ Xóa", command=self.delete\_employee, bg="#E57373", width=10)
* self.btn\_cancel = tk.Button(button\_frame, text="🗑️ Hủy", command=self.cancel\_action, bg="#90A4AE", width=10, state=tk.DISABLED)
* for b in [self.btn\_add, self.btn\_save, self.btn\_edit, self.btn\_delete, self.btn\_cancel]:
* b.pack(side=tk.LEFT, padx=5)
* # --- TreeView ---
* columns = ("MaNV", "TenNV", "GioiTinh", "NgaySinh", "DienThoai", "DiaChi")
* self.tree = ttk.Treeview(main\_frame, columns=columns, show="headings", height=15)
* widths = [80, 200, 70, 120, 130, 250]
* for col, text, width in zip(columns, ["Mã NV", "Tên Nhân Viên", "GT", "Ngày Sinh", "Điện thoại", "Địa chỉ"], widths):
* self.tree.heading(col, text=text, anchor="center")
* self.tree.column(col, width=width, anchor="center")
* self.tree.pack(fill=tk.BOTH, expand=True)
* self.tree.bind("<<TreeviewSelect>>", self.on\_tree\_select)
* self.tree.bind("<Double-Button-1>", self.on\_tree\_double\_click)
* # Thêm Scrollbar
* vscrollbar = ttk.Scrollbar(self.tree, orient="vertical", command=self.tree.yview)
* self.tree.configure(yscrollcommand=vscrollbar.set)
* vscrollbar.pack(side="right", fill="y")
* # --- Thanh trạng thái ---
* self.status\_bar = tk.Label(main\_frame, text="Sẵn sàng.", bd=1, relief=tk.SUNKEN, anchor="w")
* self.status\_bar.pack(side="bottom", fill="x")
* self.set\_form\_state("VIEW")
* self.master.after(300, self.load\_data)
* # =======================================================
* # KẾT NỐI & LOAD (Dựa trên ProductManagementForm, dùng threading)
* # =======================================================
* def get\_conn(self):
* try:
* return get\_connection()
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Kết nối SQL thất bại:\n{e}"))
* return None
* def load\_data(self, search\_term=""):
* def \_task():
* conn = self.get\_conn()
* if not conn:
* self.master.after(0, lambda: self.status\_bar.config(text="❌ Lỗi kết nối"))
* return
* try:
* cursor = conn.cursor()
* query = "SELECT MaNhanVien, TenNhanVien, GioiTinh, NgaySinh, DienThoai, DiaChi FROM tblNhanVien"
* params = []
* if search\_term:
* query += " WHERE TenNhanVien COLLATE Vietnamese\_CI\_AI LIKE ? OR DienThoai LIKE ?"
* params.extend([f"%{search\_term}%", f"%{search\_term}%"])
* query += " ORDER BY MaNhanVien ASC"
* cursor.execute(query, params)
* rows = cursor.fetchall()
* self.master.after(0, lambda: self.\_update\_treeview(rows))
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Tải dữ liệu thất bại:\n{e}"))
* finally:
* conn.close()
* self.status\_bar.config(text="🔄 Đang tải dữ liệu...")
* threading.Thread(target=\_task, daemon=True).start()
* def \_update\_treeview(self, rows):
* self.tree.delete(\*self.tree.get\_children())
* for i, row in enumerate(rows):
* # Format NgaySinh
* ngay\_sinh\_str = row.NgaySinh.strftime("%Y-%m-%d") if isinstance(row.NgaySinh, datetime) else str(row.NgaySinh or '')
* formatted\_row = (
* row.MaNhanVien, row.TenNhanVien, row.GioiTinh or '',
* ngay\_sinh\_str, row.DienThoai or '', row.DiaChi or ''
* )
* tag = 'evenrow' if i % 2 == 0 else 'oddrow' # Giữ lại màu xen kẽ
* self.tree.insert('', tk.END, values=formatted\_row, tags=(tag,))
* self.status\_bar.config(text=f"✅ Đã tải {len(rows)} bản ghi.")
* def search\_employees(self):
* keyword = self.search\_var.get().strip()
* if not keyword:
* self.load\_data()
* self.status\_bar.config(text="🔍 Đặt lại tìm kiếm.")
* return
* self.load\_data(keyword)
* def reset\_search(self):
* self.search\_var.set("")
* self.load\_data()
* self.status\_bar.config(text="🔄 Đã tải lại toàn bộ dữ liệu.")
* # =======================================================
* # STATE (Điều chỉnh cho form NV)
* # =======================================================
* def set\_form\_state(self, state):
* self.current\_state = state
* editable = state in ("ADD", "EDIT")
* for field, entry in self.entries.items():
* if field == "ma\_nhanvien":
* # Mã nhân viên luôn readonly
* entry.config(state="readonly")
* elif field == "gioi\_tinh":
* # Combobox: normal khi ADD/EDIT, disabled khi VIEW
* entry.config(state="normal" if editable else "disabled")
* elif field == "ngay\_sinh":
* # DateEntry: normal khi ADD/EDIT, disabled khi VIEW
* entry.config(state=tk.NORMAL if editable else tk.DISABLED)
* else:
* # Các trường khác: normal khi ADD/EDIT, disabled khi VIEW
* entry.config(state=tk.NORMAL if editable else tk.DISABLED)
* has\_selection = self.selected\_item is not None
* self.btn\_add.config(state=tk.NORMAL if state == "VIEW" else tk.DISABLED)
* self.btn\_edit.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_delete.config(state=tk.NORMAL if state == "VIEW" and has\_selection else tk.DISABLED)
* self.btn\_save.config(state=tk.NORMAL if editable else tk.DISABLED)
* self.btn\_cancel.config(state=tk.NORMAL if editable else tk.DISABLED)
* def set\_add\_state(self):
* self.clear\_entries()
* # Lấy mã NV tiếp theo
* next\_id = self.get\_next\_manhanvien()
* if next\_id is not None:
* self.entries["ma\_nhanvien"].config(state="normal")
* self.entries["ma\_nhanvien"].delete(0, tk.END)
* self.entries["ma\_nhanvien"].insert(0, next\_id)
* self.entries["ma\_nhanvien"].config(state="readonly")
* self.set\_form\_state("ADD")
* self.entries["ten\_nhanvien"].focus()
* def get\_next\_manhanvien(self):
* """Lấy mã NV tiếp theo, ưu tiên lấp khoảng trống (gap) nếu có."""
* conn = self.get\_conn()
* if not conn: return None
* try:
* cursor = conn.cursor()
* # Tương tự logic get\_next\_mahang của file Product
* cursor.execute("""
* SELECT MIN(t1.MaNhanVien + 1) AS NextID
* FROM tblNhanVien t1
* WHERE NOT EXISTS (
* SELECT 1 FROM tblNhanVien t2
* WHERE t2.MaNhanVien = t1.MaNhanVien + 1
* )
* AND t1.MaNhanVien + 1 <= (SELECT MAX(MaNhanVien) FROM tblNhanVien)
* """)
* result = cursor.fetchone()
* gap\_id = result[0] if result and result[0] else None
* if gap\_id:
* return gap\_id
* else:
* cursor.execute("SELECT MAX(MaNhanVien) FROM tblNhanVien")
* result = cursor.fetchone()
* max\_id = result[0] if result and result[0] else 0
* return max\_id + 1
* except Exception as e:
* messagebox.showerror("Lỗi", f"Không thể lấy mã NV tiếp theo:\n{e}")
* return None
* finally:
* conn.close()
* def set\_edit\_state(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn nhân viên để sửa!")
* return
* # Dữ liệu đã được load lên form trong on\_tree\_select, chỉ cần chuyển trạng thái
* self.set\_form\_state("EDIT")
* self.entries["ten\_nhanvien"].focus()
* def cancel\_action(self):
* self.clear\_entries()
* self.set\_form\_state("VIEW")
* self.load\_data()
* def clear\_entries(self):
* self.selected\_item = None
* self.entries['ma\_nhanvien'].config(state='normal')
* for key, entry in self.entries.items():
* entry.config(state="normal")
* if key in ['gioi\_tinh']:
* entry.set("Nam")
* elif key == 'ngay\_sinh':
* entry.set\_date(datetime.now().date())
* else:
* entry.delete(0, tk.END)
* self.entries['ma\_nhanvien'].config(state='readonly')
* self.tree.selection\_remove(self.tree.selection())
* # =======================================================
* # CRUD (DÙNG THREADING)
* # =======================================================
* def save\_data(self):
* ma\_nv, data = self.get\_validated\_data(self.current\_state == 'ADD')
* if data is None: return
* if self.current\_state == "ADD":
* self.status\_bar.config(text="🔄 Đang thêm nhân viên...")
* # Truyền next\_id đã tính toán vào thread
* next\_id = int(self.entries["ma\_nhanvien"].get())
* threading.Thread(target=self.\_execute\_add\_item, args=(next\_id, data,), daemon=True).start()
* elif self.current\_state == "EDIT":
* self.status\_bar.config(text=f"🔄 Đang cập nhật Mã {ma\_nv}...")
* threading.Thread(target=self.\_execute\_update\_item, args=(ma\_nv, data), daemon=True).start()
* def \_execute\_add\_item(self, next\_id, data):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* # Bật IDENTITY\_INSERT để chèn mã tùy chỉnh
* cursor.execute("SET IDENTITY\_INSERT tblNhanVien ON")
* cursor.execute("""
* INSERT INTO tblNhanVien (MaNhanVien, TenNhanVien, GioiTinh, DiaChi, DienThoai, NgaySinh)
* VALUES (?, ?, ?, ?, ?, ?)
* """, (next\_id, data['TenNhanVien'], data['GioiTinh'], data['DiaChi'], data['DienThoai'], data['NgaySinh']))
* cursor.execute("SET IDENTITY\_INSERT tblNhanVien OFF")
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã thêm nhân viên mới với Mã {next\_id}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã thêm Mã {next\_id}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Thêm thất bại:\n{e}"))
* finally:
* if conn:
* try: cursor.execute("SET IDENTITY\_INSERT tblNhanVien OFF")
* except: pass
* conn.close()
* def \_execute\_update\_item(self, ma\_nv, data):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* cursor.execute("""
* UPDATE tblNhanVien
* SET TenNhanVien=?, GioiTinh=?, DiaChi=?, DienThoai=?, NgaySinh=?
* WHERE MaNhanVien=?
* """, (data['TenNhanVien'], data['GioiTinh'], data['DiaChi'], data['DienThoai'], data['NgaySinh'], ma\_nv))
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã cập nhật Mã {ma\_nv}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã cập nhật Mã {ma\_nv}")
* ])
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Cập nhật thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* def delete\_employee(self):
* if not self.selected\_item:
* messagebox.showwarning("Chưa chọn", "Vui lòng chọn nhân viên để xóa!")
* return
* ma\_nv = self.entries["ma\_nhanvien"].get()
* if not messagebox.askyesno("Xác nhận", f"Xóa nhân viên Mã {ma\_nv}?"):
* return
* self.status\_bar.config(text=f"🔄 Đang xóa Mã {ma\_nv}...")
* threading.Thread(target=self.\_execute\_delete\_item, args=(ma\_nv,), daemon=True).start()
* def \_execute\_delete\_item(self, ma\_nv):
* conn = self.get\_conn()
* if not conn: return
* try:
* cursor = conn.cursor()
* # Xử lý khóa ngoại (Giống như logic cũ)
* cursor.execute("UPDATE tblTaiKhoan SET MaNhanVien = NULL WHERE MaNhanVien = ?", (ma\_nv,))
* # Xóa nhân viên
* cursor.execute("DELETE FROM tblNhanVien WHERE MaNhanVien=?", (ma\_nv,))
* conn.commit()
* self.master.after(0, lambda: [
* messagebox.showinfo("Thành công", f"Đã xóa Mã {ma\_nv}!"),
* self.load\_data(),
* self.clear\_entries(),
* self.set\_form\_state("VIEW"),
* self.status\_bar.config(text=f"✅ Đã xóa Mã {ma\_nv}")
* ])
* except pyodbc.IntegrityError:
* self.master.after(0, lambda: messagebox.showerror("Lỗi DB", "Không thể xóa nhân viên này vì đã có giao dịch (Hóa đơn bán hàng) liên quan."))
* except Exception as e:
* self.master.after(0, lambda: messagebox.showerror("Lỗi", f"Xóa thất bại:\n{e}"))
* finally:
* if conn: conn.close()
* # =======================================================
* # TIỆN ÍCH
* # =======================================================
* def get\_validated\_data(self, is\_add):
* ma\_nv = self.entries['ma\_nhanvien'].get().strip()
* ten\_nv = self.entries['ten\_nhanvien'].get().strip()
* gioi\_tinh = self.entries['gioi\_tinh'].get().strip()
* dien\_thoai = self.entries['dien\_thoai'].get().strip()
* dia\_chi = self.entries['dia\_chi'].get().strip()
* # DateEntry trả về ngày theo format string đã định
* ngay\_sinh = self.entries['ngay\_sinh'].get()
* if not ten\_nv:
* messagebox.showwarning("Thiếu thông tin", "Tên nhân viên không được để trống.")
* return None, None
* return ma\_nv, {
* "TenNhanVien": ten\_nv,
* "GioiTinh": gioi\_tinh,
* "DiaChi": dia\_chi,
* "DienThoai": dien\_thoai,
* "NgaySinh": ngay\_sinh,
* }
* def on\_tree\_select(self, event):
* """Xử lý khi chọn một dòng trong Treeview - Lấy đầy đủ thông tin lên textbox."""
* # Chỉ xử lý khi đang ở chế độ VIEW
* if self.current\_state != "VIEW":
* return
* selected = self.tree.selection()
* if not selected:
* self.selected\_item = None
* self.set\_form\_state("VIEW")
* return
* self.selected\_item = selected[0]
* values = self.tree.item(self.selected\_item)['values']
* if len(values) >= 6:
* # Clear tất cả trước - PHẢI enable trước khi clear/insert
* for key, widget in self.entries.items():
* if key == "ma\_nhanvien":
* widget.config(state="normal")
* widget.delete(0, tk.END)
* elif key == "gioi\_tinh":
* # Tạm chuyển về readonly để hiển thị giá trị
* widget.config(state="readonly")
* widget.set("")
* elif key == "ngay\_sinh":
* # Cho phép set\_date
* widget.config(state=tk.NORMAL)
* else:
* widget.config(state=tk.NORMAL)
* widget.delete(0, tk.END)
* # Điền đầy đủ dữ liệu từ treeview lên textbox
* # Index: 0=MaNV, 1=TenNV, 2=GioiTinh, 3=NgaySinh, 4=DienThoai, 5=DiaChi
* self.entries["ma\_nhanvien"].insert(0, str(values[0]))
* self.entries["ma\_nhanvien"].config(state="readonly")
* self.entries["ten\_nhanvien"].insert(0, str(values[1]) if values[1] else "")
* # Ghi giá trị combobox giới tính khi đang readonly để đảm bảo hiển thị
* self.entries["gioi\_tinh"].config(state="readonly")
* self.entries["gioi\_tinh"].set(str(values[2]) if values[2] else "Nam")
* # Cập nhật DateEntry cho Ngày sinh
* ngay\_sinh\_str = str(values[3]) if values[3] else ""
* try:
* date\_obj = datetime.strptime(ngay\_sinh\_str, '%Y-%m-%d').date()
* self.entries['ngay\_sinh'].set\_date(date\_obj)
* except (ValueError, AttributeError):
* self.entries['ngay\_sinh'].set\_date(datetime.now().date())
* self.entries["dien\_thoai"].insert(0, str(values[4]) if values[4] else "")
* self.entries["dia\_chi"].insert(0, str(values[5]) if values[5] else "")
* # Giữ nguyên state VIEW (disable lại tất cả), cập nhật lại các nút
* self.set\_form\_state("VIEW")
* def on\_tree\_double\_click(self, event):
* """Double-click vào dòng → tự động chuyển sang chế độ Sửa"""
* if self.selected\_item and self.current\_state == "VIEW":
* self.set\_edit\_state()
* if \_\_name\_\_ == "\_\_main\_\_":
* root = tk.Tk()
* root.state('zoomed')
* app = EmployeeManagementForm(root)
* root.mainloop()

***Account\_management***

*A screenshot of a computer

AI-generated content may be incorrect.*

* **Giao diện:** Gồm Khung nhập liệu, Khung nút chức năng (Thêm/Sửa/Xóa/Lưu/Hủy) và TreeView để hiển thị danh sách.
* **Quản lý Trạng thái (set\_form\_state):** Chuyển đổi giữa các trạng thái VIEW, ADD, EDIT để kích hoạt/vô hiệu hóa các trường nhập liệu và nút bấm tương ứng.
* **Xử lý Bất đồng bộ (Threading):**
* Các thao tác tải dữ liệu (load\_data) và CRUD (\_execute\_add\_item, \_execute\_update\_item, \_execute\_delete\_item) được thực thi trong một luồng nền (daemon thread).
* Sau khi thao tác DB hoàn tất, hàm self.master.after(0, lambda: [...]) được sử dụng để quay lại luồng chính, cập nhật giao diện (TreeView, StatusBar, MessageBox) một cách an toàn.
* **Xử lý ID Tự động/Lấp khoảng trống (get\_next\_mahang, get\_next\_makhach):**
* Thay vì chỉ lấy MAX(ID) + 1, các module (Hàng hóa, Khách hàng, Nhân viên) ưu tiên tìm kiếm và lấp các khoảng trống ID (gap) đã bị xóa trước đó.
* Khi ADD, mã ID được tính toán trước, và lệnh SET IDENTITY\_INSERT <TableName> ON được sử dụng để chèn mã ID tùy chỉnh này.

### **4. Module Nghiệp vụ Chính: Lập Hóa đơn (sales\_invoice\_form.py)**

Đây là module phức tạp nhất, xử lý quy trình bán hàng:

* **Tải dữ liệu Tham chiếu (load\_reference\_data):** Tải danh sách Khách hàng và Mặt hàng vào các Combobox (dùng product\_map, customer\_map) để người dùng chọn.
* **Quản lý Giỏ hàng (cart\_items, add\_to\_cart, refresh\_cart\_view):**
  + Sử dụng một dictionary (self.cart\_items) để mô phỏng giỏ hàng.
  + Khi add\_to\_cart, nếu hàng đã có thì cộng dồn số lượng.
  + refresh\_cart\_view cập nhật TreeView Giỏ hàng và tính toán tổng tiền.
* **Lưu Hóa đơn (save\_invoice):**
  1. Thực hiện 2 thao tác DB trong 1 giao dịch (Transaction):
     + INSERT vào bảng **tblHDBan** (lấy ra MaHDBan vừa tạo).
     + Lặp qua cart\_items: INSERT vào bảng **tblChiTietHDBan** và **UPDATE** số lượng tồn kho (tblHang).
  2. Sử dụng conn.commit() để hoàn tất giao dịch. Nếu có lỗi (ví dụ: số lượng tồn < số lượng bán), conn.rollback() sẽ được thực hiện để hủy toàn bộ giao dịch.

***Sales\_invoice***

***A screenshot of a computer

AI-generated content may be incorrect.***

### **5. Module Báo cáo (revenue\_report\_form.py)**

* **Báo cáo Chi tiết Giao dịch:**
  + Cho phép lọc theo **Khoảng thời gian** (sử dụng tkcalendar.DateEntry) và **Nhân viên**.
  + Truy vấn sử dụng nhiều JOIN (Hóa đơn, Nhân viên, Khách hàng, Chi tiết HĐ, Hàng hóa) để lấy toàn bộ thông tin chi tiết từng mặt hàng trong từng hóa đơn.
* **Tính toán Tổng kết:**
  + Tính tổng doanh thu và tổng số hóa đơn duy nhất (unique\_hds).
  + Cập nhật kết quả vào khung Tổng hợp.
* **Xuất File (Export):** Cho phép người dùng xuất dữ liệu báo cáo hiện tại ra file **CSV** thông qua thư viện csv và hộp thoại filedialog.

***Revenue\_report***

*A screenshot of a computer

AI-generated content may be incorrect.*

## **3.3. Những phần đạt được và chưa đạt được của đồ án**

### **a) Những phần đạt được:**

* **Giao diện thân thiện và dễ sử dụng:**
* Phần mềm được thiết kế với giao diện đơn giản, trực quan, phù hợp cho cả những người không quen sử dụng công nghệ.
* Các chức năng được sắp xếp hợp lý, dễ dàng điều hướng và thực hiện các thao tác.
* **Thao tác nhanh chóng:**
* Các chức năng chính hổ trợ mua bán , và quản lý thông tin khách hàng được thực hiện một cách nhanh gọn.
* **Thân thiện với người dùng:**
* Cung cấp các công cụ hỗ trợ như combobox, bảng chi tiết, và nút xác nhận rõ ràng.
* Giao diện sử dụng biểu tượng minh họa (icon) giúp tăng tính trực quan.
* **Phần mềm chạy Mượt mà (Responsive):** Tôi đã sử dụng kỹ thuật lập trình đa luồng (**Threading**) nên dù xử lý dữ liệu lớn (tìm kiếm, tải bảng), giao diện vẫn **không bị treo**.
* **Quản lý Bán hàng An toàn (Transaction):** Việc lập hóa đơn được thực hiện theo nguyên tắc giao dịch (**Transaction**), đảm bảo nếu có lỗi xảy ra trong quá trình trừ tồn kho, hóa đơn sẽ **tự động bị hủy (Rollback)** để giữ nguyên vẹn dữ liệu.
* **Thiết kế Hợp lý:** Các chức năng được phân chia thành các module riêng biệt, dễ dàng quản lý và nâng cấp.
* **Tính năng Báo cáo Tốt:** Có thể lọc báo cáo chi tiết theo thời gian, nhân viên và xuất ra file CSV.

### **b) Những phần chưa đạt được:**

* **Cơ sở dữ liệu chưa hoàn toàn tối ưu:**
* Hệ thống cơ sở dữ liệu còn đơn giản, thiếu ràng buộc chặt chẽ giữa các bảng.
* Một số mối quan hệ dữ liệu chưa được tối ưu, có thể dẫn đến lỗi hoặc mất tính toàn vẹn dữ liệu.
* **Bố cục mã nguồn phức tạp:**
* Mã nguồn chưa được tổ chức gọn gàng, dẫn đến khó khăn trong việc bảo trì hoặc nâng cấp.
* Thiếu áp dụng các nguyên tắc lập trình tốt (clean code, modularization).
* **Bảo mật Yếu:** Mật khẩu người dùng đang được lưu trữ dưới dạng **văn bản thường (Plain Text)** trong cơ sở dữ liệu, chưa được mã hóa. Cần phải Hash mật khẩu ngay.
* **Chưa Quản lý Hết Nghiệp vụ:** Phần mềm **chưa có tính năng Giảm giá** cho hóa đơn. Đồng thời, giá bán đang lấy từ giá nhập, chưa có quản lý **giá bán** riêng biệt.
* **Chưa Kiểm tra Tồn kho Sớm:** Hệ thống chỉ phát hiện thiếu tồn kho **sau khi nhấn Lưu hóa đơn**, cần phải kiểm tra sớm hơn (ngay khi thêm vào giỏ hàng) để báo lỗi cho người dùng.

### **c) Đề xuất cải thiện:**

* **Nâng cao Tính năng:** Thêm tính năng Quản lý nhập hàng (Phiếu Nhập) để quản lý tồn kho và giá nhập/xuất một cách chuyên nghiệp hơn.
* **Trực quan hóa Dữ liệu:** Tích hợp đồ thị, biểu đồ vào Bảng điều khiển (Dashboard) để dễ dàng theo dõi doanh thu hàng tháng.

# **4. TÀI LIỆU KHAM KHẢO:**

* Nguồn tài liệu kham khảo:

<https://timoday.edu.vn/xay-dung-chuong-trinh-quan-ly-ban-hang-bang-c/>

<https://viettuts.vn/python-tkinter>

<https://www.geeksforgeeks.org/python/python-projects-beginner-to-advanced/>