**TRƯỜNG ĐẠI HỌC AN GIANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**Chuyên đề Python (COS525)**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ CỦA HÀNG VẬT LIỆU XÂY DỰNG VỚI PYTHON, TKINTER VÀ MYSQL**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | Ths. Nguyễn Ngọc Minh |
| Sinh viên thực hiện: | DTH235700. Quách Thành Minh. Lớp DH24TH2- Nhóm 2- Tổ TH2  DTH235737. Lê Ngọc Minh Phúc. Lớp DH24TH2- Nhóm 2- Tổ TH2 |

**AN GIANG 11/2025**

**Mục Lục**

[**1.** **Đặt vấn đề** 1](#_Toc214984476)

[**1.1.** **Tính cần thiết của đề tài** 1](#_Toc214984477)

[**1.2.** **Mục tiêu nghiên cứu của đề tài** 1](#_Toc214984478)

[**2.** **Tổng quan và cơ sở lí thuyết** 1](#_Toc214984479)

[**2.1.** **Đặt vấn đề** 1](#_Toc214984480)

[**2.2.** **Lịch sử giải quyết vấn đề và tồn tại** 1](#_Toc214984481)

[**2.3.** **Phạm vi của đề tài** 2](#_Toc214984482)

[**3.** **Kết quả đạt được và demo code** 3](#_Toc214984483)

[**A.** **THIẾT KẾ GIAO DIỆN QUẢN LÝ CỦA HÀNG VẬT LIỆU XÂY DỰNG** 3](#_Toc214984484)

[**B.** **CÁC BƯỚC XÂY DỰNG ỨNG DỤNG** 3](#_Toc214984485)

[**1.** **Cài đặt các thư viện cần thiết** 3](#_Toc214984486)

[**2.** **Thiết kế cơ sở dữ liệu MYSQL** 4](#_Toc214984487)

[**3.** **Xây dựng giao diện người dùng** 4](#_Toc214984488)

[**C.** **CODE ứng dụng** 4](#_Toc214984489)

[**1.** **Cài đặt môi trường** 4](#_Toc214984490)

[**2.** **Chuẩn bị CSDL MYSQL** 5](#_Toc214984491)

[**2.1. Mở MySQL Workbench và đăng nhập** 5](#_Toc214984492)

[**3.** **Viết code với ứng dụng MYSQL + Tkinter** 10](#_Toc214984493)

[**3.1.** **Màn hình đăng nhập và phân quyền truy cập** 10](#_Toc214984494)

[**3.2.** **Quản lý nhân viên** 12](#_Toc214984495)

[**3.3.** **Quản lý khách hàng** 19](#_Toc214984496)

[**3.4.** **Quản lý nhà cung cấp** 24](#_Toc214984497)

[**3.5.** **Quản lý vật liệu/ sản phẩm** 30](#_Toc214984498)

[**3.6.** **Quản lý hóa đơn** 37](#_Toc214984499)

[**3.7.** **Quản lý ct\_hóa đơn** 45](#_Toc214984500)

[**3.8.** **Quản lý phiếu nhập** 54](#_Toc214984501)

[**3.9.** **Quản lý ct\_phiếu nhập** 61](#_Toc214984502)

[**4.** **Các module liên kết khác** 69](#_Toc214984503)

[**5.** **Liên kết giao diện Tkinter với các hàm xử lý cơ sở dữ liệu đã xây dựng ở module db\_functions.py** 75](#_Toc214984504)

**Danh mục hình ảnh**

Hình 1: Giao diện chính của hệ thống. 3

Hình 2: Sơ đồ cơ sở dữ liệu. 4

Hình 3: Màn hình đăng nhập vào MySQL. 5

Hình 5: Màn hình đăng nhập. 10

Hình 6: Màn hình đăng nhập với vai trò là quản lý (admin) 11

Hình 7: Màn hình đăng nhập với vai trò là nhân viên. 11

Hình 8: Giao diện quản lý nhân viên với vai trò là quản lý 12

Hình 9: Giao diện quản lý nhân viên với vai trò là nhân viên. 12

Hình 10: Giao diện quản lý khách hàng 19

Hình 11: Giao diện quản lý nhà cung cấp với vai trò quản lý 24

Hình 12: Giao diện quản lý nhà cung cấp với vai trò là nhân viên. 25

Hình 13: Giao diện quản lý vật liệu 30

Hình 14: Giao diện quản lý hóa đơn 37

Hình 15: Giao diện quản lý chi tiết hóa đơn 45

Hình 16: Giao diện quản lý phiếu nhập 54

Hình 17: Giao diện quản lý chi tiết phiếu nhập 61

Hình 18: Giao diện quản lý nhân viên khi đã thêm nhân viên thành công 75

Hình 19: Giao diện quản lý nhân viên khi quản lý không nhập đầy đủ thông tin 75

Hình 20: Giao diện quản lý nhân viên thông báo yêu cầu chọn nhân viên để sửa 76

Hình 21: Giao diện quản lý nhân viên thông báo đã sửa và cập nhật thành công 76

Hình 22: Giao diện quản lý nhân viên, các ô nhập liệu khi chưa dùng nút Hủy 77

Hình 23: Giao diện quản lý nhân viên, các ô nhập liệu sau khi đã dùng nút Hủy 77

Hình 24: Giao diện quản lý nhân viên hiện câu hỏi xác nhận muốn Xóa 78

Hình 25: Giao diện quản lý nhân viên sau khi xác nhận Xóa 78

Hình 26: Giao diện quản lý nhân viên khi tìm kiếm nhân viên 79

Hình 27: Giao diện quản lý nhân viên hiển thị hộp thoại yêu cầu xác nhận Thoát 79

Hình 28: Giao diện của quản lý quay lại màn hình menu chính 80

Hình 29: Giao diện quay lại phần đăng nhập 80

## **Đặt vấn đề**

### **Tính cần thiết của đề tài**

* Tính cần thiết: Trong lĩnh vực kinh doanh vật liệu xây dựng, việc quản lý số lượng tồn kho của hàng ngàn mặt hàng (gạch, cát, xi măng, sắt thép...), quản lý giá bán thay đổi liên tục, và quản lý các giao dịch xuất nhập kho là rất phức tạp. Các phương pháp thủ công hoặc file Excel đơn giản dẫn đến sai sót trong tính toán lãi lỗ, thấtthoát hàng hóa, và không kiểm soát được hàng tồn kho tối thiểu.
* Ý nghĩa thực tiễn: đề tài này giải quyết trực tiếp những hạn chế trên bằng cách xây dựng một ứng dụng quản lý gọn nhẹ, chi phí thấp, tập trung vào các nghiệp vụ cốt lõi của cửa hàng vật liệu xây dựng.

### **Mục tiêu nghiên cứu của đề tài**

* Mục tiêu: xây dựng hệ thông quản lý cho phép lưu trữ, cập nhât, chỉnh sửa thông tin nhanh chóng, chính xác. Giúp thao tác nhanh, chính xác với dữ liệu, đạt độ tin cậy cao trong các nghiệp vụ giao dịch thực tế.

## **Tổng quan và cơ sở lí thuyết**

### **Đặt vấn đề**

* Bối cảnh: Hiện nay các cửa hàng vật liệu xây dựng quy mô vừa và nhỏ chủ yếu quản lý bằng phương pháp thủ công (sổ sách, Excel), dẫn đến các vấn đề nghiêm trọng về sai sót dữ liệu, thất thoát hàng tồn kho, và chậm trễ trong việc tra cứu thông tin công nợ/giao dịch.

### **Lịch sử giải quyết vấn đề và tồn tại**

* Trên thị trường hiện nay, có hai loại hệ thống giải quyết bài toán quản lý tồn kho và bán hàng vật liệu xây dựng (VLXD) chính: Hệ thống phần mềm thương mại và phương pháp Thủ công.
* Hệ thống phần mềm thương mại (POS/ERP): Trên thị trường hiện nay có các hệ thống Phần mềm Bán hàng (POS) như KiotViet hoặc Sapo POS, mặc dù chúng cung cấp tính năng cốt lõi là xử lý giao dịch bán hàng nhanh chóng, quản lý tồn kho đa điểm và tổng hợp báo cáo doanh thu. Tuy nhiên, chúng có hạn chế lớn là chi phí vận hành cao (dưới hình thức phí thuê bao định kỳ) và độ phức tạp về tính năng do được thiết kế cho nhiều ngành nghề, gây khó khăn cho việc triển khai và đào tạo người dùng tại các cửa hàng VLXD nhỏ lẻ. Các tính năng mở rộng không cần thiết làm hệ thống trở nên cồng kềnh, không phù hợp với ngân sách hẹp và nhu cầu tinh gọn của hộ kinh doanh cá thể.
* Phương pháp thủ công (Bảng tính): Các cửa hàng nhỏ thường dựa vào hệ thống bảng tính như Microsoft Excel hoặc sổ sách truyền thống, mặc dù giải pháp này có tính năng cốt lõi là miễn phí và dễ dàng làm quen. Tuy nhiên, chúng có hạn chế nghiêm trọng là thiếu tính toàn vẹn dữ liệu và hiệu suất kém. Excel không có cơ chế ràng buộc cơ sở dữ liệu (ví dụ: kiểm tra mã sản phẩm trùng lặp, kiểm tra số lượng tồn kho âm), dẫn đến rủi ro sai sót trong tính toán cao, làm mất tính tin cậy của thông tin quản lý.
* Chính vì những tồn tại trên, dự án này khẳng định lý do chọn dự án là phát triển một hệ thống quản lý thông tin (MIS) tùy chỉnh bằng Python, Tkinter và MySQ**L**. Giải pháp này sẽ tận dụng lợi thế mã nguồn mở (loại bỏ chi phí bản quyền) và tập trung tuyệt đối vào các chức năng CRUDcơ bản để đảm bảo sự chính xác, tin cậy và hiệu quả trong quản lý tồn kho, đồng thời giữ chi phí triển khai ở mức thấp nhất, lý tưởng cho các cửa hàng vật liệu xây dựng quy mô nhỏ.

### **Phạm vi của đề tài**

* Nội dung đề tài tập trung vào thiết kế giao diện desktop với ngôn ngữ Python và thư viên Tkinter để quản lý thông tin khách hàng, nhân viên, hóa đơn, vật liệu xây dựng, chi tiết hóa đơn, phiếu nhập và chi tiết phiếu nhập cùng, nhà cung cấp với hệ quản trị dữ liệu MySQL để lưu trữ dữ liệu.

+ Quản lý khách hàng: bao gồm quản lý thông tin khách hàng, số lượng sản phẩm khách mua, sản phẩm khách mua và lịch sử giao dịch (thông qua bảng hóa đơn).

+ Quản lý nhân viên: bao gồm quản lý thông tin và chức vụ của nhân viên.

+ Quản lý hóa đơn: bao gồm quản lý sản phẩm đã bán cho khách hang và tạo lịch sử giao dịch.

+ Quản lý vật liệu xây dựng: bao gồm kiểm tra và ghi lại các sản phẩm đã bán, chưa bán và hàng tồn.

+ Quản lý chi tiết hóa đơn: bao gồm lưu trữ thông tin chi tiết các sản phẩm (mã sản phẩm, số lượng) đã bán cho khách hàng.

+ Quản lý phiếu nhập: bao gồm ghi nhận chi tiết các lần nhập hàng để kiểm soát giá vốn và cập nhật số lượng tồn kho.

+ Quản lýchi tiết phiếu nhập: bao gồm lưu trữ số lượng và giá nhập của từng loại vật liệu trong phiếu nhập cụ thể.

+ Quản lý nhà cung cấp: bao gồm quản lý thông tin hồ sơ của đối tác (tên, SĐT, địa chỉ) và theo dõi lịch sử nhập hàng (thông qua bảng phiếu nhập).

* Hệ thống được xây dựng bao gồm các chức năng:

+ Thêm: Dùng để thêm thông tin khách hàng, nhân viên, vật liệu xây dựng, hóa đơn, chi tiết hóa đơn, phiếu nhập và chi tiết phiếu nhập của cửa hàng.

+ Sửa: dùng để sửa thông tin khách hàng, nhân viên, vật liệu xây dựng, hóa đơn, chi tiết hóa đơn, phiếu nhập và chi tiết phiếu nhập của cửa hàng.

+ Lưu: dùng để lưu thông tin khách hàng, nhân viên, vật liệu xây dựng, hóa đơn, chi tiết hóa đơn, phiếu nhập và chi tiết phiếu nhập của cửa hàng.

+ Hủy: dùng để hủy hành động muốn làm phía trước.

+ Xóa: dùng để xóa thông tin khách hàng, nhân viên, vật liệu xây dựng, hóa đơn, chi tiết hóa đơn, phiếu nhập và chi tiết phiếu nhập của cửa hàng.

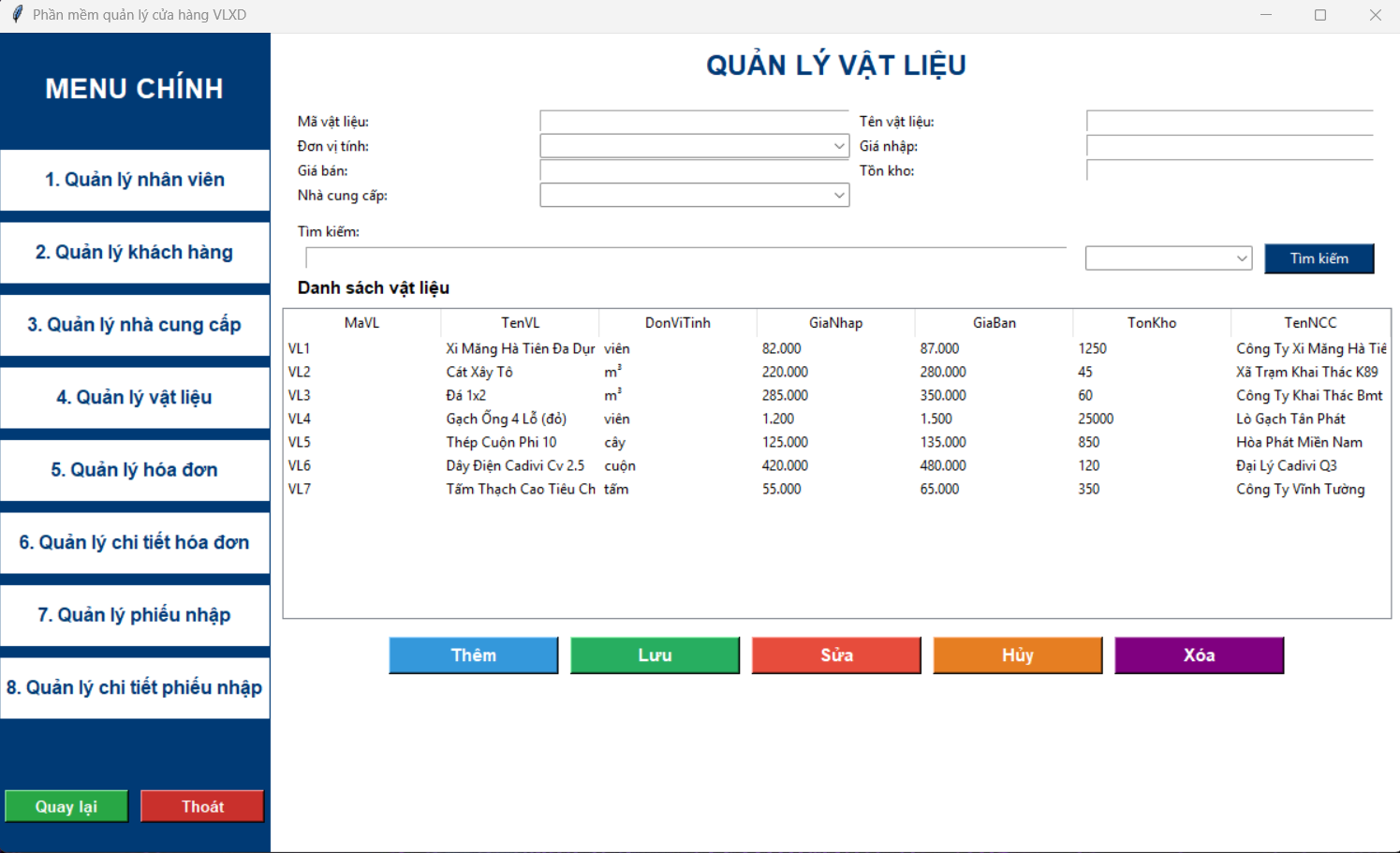
+ Thoát: dùng để thoát khỏi giao diện quản lý.

+ Quay lại: chức năng này được dùng khi muốn quay lại giao diện phía trước.

+ Tìm kiếm: chức năng quan trọng cho phép người dùng lọc hoặc tìm kiếm bản ghi theo các tiêu chí đã xác định.

## **Kết quả đạt được và demo code**

# **THIẾT KẾ GIAO DIỆN QUẢN LÝ CỦA HÀNG VẬT LIỆU XÂY DỰNG**

Hình 1: Giao diện chính của hệ thống.

# **CÁC BƯỚC XÂY DỰNG ỨNG DỤNG**

Để xây dựng ứng dụng quản lý sản phẩm bằng Python, Tkinter và MySQL, cần cài đặt các thư viện cần thiết, thiết kế cơ sở dữ liệu MySQL để lưu trữ thông tin sản phẩm, sử dụng Tkinter để tạo giao diện người dùng, và kết hợp Python với thư viện mysql.connector để thực hiện các thao tác như thêm, xóa, sửa, và hiển thị danh sách quản lý.

## **Cài đặt các thư viện cần thiết**

* Python
* MySQL
* mysql-connector-python: Cài đặt thư viện này để kết nối Python với cơ sở dữ liệu MySQL(**pip install mysql-connector-python**)

## **Thiết kế cơ sở dữ liệu MYSQL**

A computer screen shot of a computer

AI-generated content may be incorrect.

Hình 2: Sơ đồ cơ sở dữ liệu.

## **Xây dựng giao diện người dùng**

* Sử dụng thư viện Tkinter để tạo các cửa sổ, trường nhập liệu, nút bấm và bảng hiển thị thông tin.
* Thiết kế giao diện cho phép người dùng thực hiện các chức năng sau:

+ Thêm thông tin mới.

+ Lưu thông tin

+ Sửa thông tin

+ Hủy thông tin

+ Xóa thông tin

+ Tìm kiếm thông tin

+ Thoát cửa sổ

+ Quay lại nếu muốn

# **CODE ứng dụng**

## **Cài đặt môi trường**

* mysql-connector-python: mysql-connector-python là thư viện (library) do chính MySQL phát triển, dùng để cho Python kết nối và làm việc với MySQL Database**.**
* tkcalendar: là một thư viện Python dành cho Tkinter, giúp bạn tạo lịch (calendar) và chọn ngày (date picker) trong giao diện ứng dụng.

## **Chuẩn bị CSDL MYSQL**

### **2.1. Mở MySQL Workbench và đăng nhập**

- Để đăng nhập vào MySQL Workbench, người dùng cần cung cấp các thông tin sau:

* **Host:** localhost
* **Port:** 3307
* **User:** root
* **Password:** Minh@120705

Sau khi nhập đầy đủ các thông tin trên và nhấn **OK**, hệ thống sẽ xác thực và đưa người dùng vào giao diện quản lý cơ sở dữ liệu.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3: Màn hình đăng nhập vào MySQL.

**2.2 Tạo mới CSDL cho chương trình**

-- Xóa database nếu đã tồn tại

DROP DATABASE IF EXISTS qlch\_vlxd;

-- Tạo database mới với chuẩn UTF-8 đầy đủ

CREATE DATABASE qlch\_vlxd

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

-- Sử dụng database

USE qlch\_vlxd;

-----------------------------------------------------

2.3 **Tạo các bảng dữ liệu**

-- BẢNG NHÂN VIÊN

-----------------------------------------------------

CREATE TABLE nhanvien (

MaNV VARCHAR(20) PRIMARY KEY, -- Mã nhân viên

HoTenNV VARCHAR(100) NOT NULL, -- Họ tên nhân viên

Sdt VARCHAR(15) NOT NULL, -- Số điện thoại

Phai VARCHAR(5) NOT NULL, -- Giới tính

NgaySinh DATE NOT NULL, -- Ngày sinh

ChucVu VARCHAR(50) NOT NULL, -- Chức vụ

DiaChi VARCHAR(200), -- Địa chỉ

Luong INT NOT NULL, -- Lương cơ bản

TinhTrang VARCHAR(20) NOT NULL -- Tình trạng làm việc

);

-----------------------------------------------------

-- BẢNG KHÁCH HÀNG

-----------------------------------------------------

CREATE TABLE khachhang (

MaKH VARCHAR(20) PRIMARY KEY, -- Mã khách hàng

HoTen VARCHAR(100) NOT NULL, -- Tên khách hàng

SDT VARCHAR(15) NOT NULL, -- Số điện thoại

DiaChi VARCHAR(255) NOT NULL, -- Địa chỉ

CCCD VARCHAR(20) NOT NULL, -- Căn cước công dân

GhiChu VARCHAR(255) -- Ghi chú thêm

);

-----------------------------------------------------

-- BẢNG NHÀ CUNG CẤP

-----------------------------------------------------

CREATE TABLE nhacungcap (

MaNCC VARCHAR(20) PRIMARY KEY, -- Mã nhà cung cấp

TenNCC VARCHAR(100) NOT NULL, -- Tên nhà cung cấp

SDT VARCHAR(15) NOT NULL, -- Số điện thoại

DiaChi VARCHAR(200), -- Địa chỉ

GhiChu VARCHAR(255) -- Ghi chú thêm

);

-----------------------------------------------------

-- BẢNG VẬT LIỆU

-----------------------------------------------------

CREATE TABLE vatlieu (

MaVL VARCHAR(20) PRIMARY KEY, -- Mã vật liệu

TenVL VARCHAR(100) NOT NULL, -- Tên vật liệu

DonViTinh VARCHAR(50), -- Đơn vị tính

GiaNhap DECIMAL(15,2) NOT NULL, -- Giá nhập

GiaBan DECIMAL(15,2) NOT NULL, -- Giá bán

TonKho INT DEFAULT 0, -- Số lượng tồn kho

MaNCC VARCHAR(20), -- Mã nhà cung cấp

FOREIGN KEY (MaNCC) REFERENCES nhacungcap(MaNCC)

ON UPDATE CASCADE -- Nếu mã NCC đổi → cập nhật theo

ON DELETE SET NULL -- Nếu NCC bị xóa → để NULL

);

-----------------------------------------------------

-- BẢNG HÓA ĐƠN BÁN HÀNG

-----------------------------------------------------

CREATE TABLE hoadon (

MaHD VARCHAR(20) PRIMARY KEY, -- Mã hóa đơn

MaKH VARCHAR(20), -- Mã khách hàng

MaNV VARCHAR(20), -- Mã nhân viên bán

NgayBan DATE, -- Ngày bán hàng

HinhThucTT VARCHAR(50), -- Hình thức thanh toán

GhiChu VARCHAR(255), -- Ghi chú

TongTien DECIMAL(15,2) DEFAULT 0, -- Tổng tiền hóa đơn

FOREIGN KEY (MaKH) REFERENCES khachhang(MaKH),

FOREIGN KEY (MaNV) REFERENCES nhanvien(MaNV)

);

-----------------------------------------------------

-- BẢNG CHI TIẾT HÓA ĐƠN

-----------------------------------------------------

CREATE TABLE ct\_hoadon (

MaHD VARCHAR(20) NOT NULL, -- Mã hóa đơn

MaVL VARCHAR(20) NOT NULL, -- Mã vật liệu

SoLuong INT NOT NULL, -- Số lượng bán ra

DonGia DECIMAL(15,2) NOT NULL, -- Đơn giá bán

ThanhTien DECIMAL(15,2) AS (SoLuong \* DonGia) STORED, -- Tự tính

PRIMARY KEY (MaHD, MaVL), -- 1 vật liệu chỉ xuất hiện 1 lần trong hóa đơn

FOREIGN KEY (MaHD) REFERENCES hoadon(MaHD),

FOREIGN KEY (MaVL) REFERENCES vatlieu(MaVL)

);

-----------------------------------------------------

-- BẢNG PHIẾU NHẬP KHO

-----------------------------------------------------

CREATE TABLE phieunhap (

MaPN VARCHAR(20) PRIMARY KEY, -- Mã phiếu nhập

NgayNhap DATE NOT NULL, -- Ngày nhập kho

MaNCC VARCHAR(20) NOT NULL, -- Mã nhà cung cấp

MaNV VARCHAR(20) NOT NULL, -- Mã nhân viên nhập kho

GhiChu VARCHAR(255), -- Ghi chú

TongTien DECIMAL(18,2) DEFAULT 0, -- Tổng tiền phiếu nhập

FOREIGN KEY (MaNCC) REFERENCES nhacungcap(MaNCC),

FOREIGN KEY (MaNV) REFERENCES nhanvien(MaNV)

);

-----------------------------------------------------

-- BẢNG CHI TIẾT PHIẾU NHẬP

-----------------------------------------------------

CREATE TABLE ct\_phieunhap (

MaPN VARCHAR(20) NOT NULL, -- Mã phiếu nhập

MaVL VARCHAR(20) NOT NULL, -- Mã vật liệu

SoLuongNhap INT NOT NULL, -- Số lượng nhập

DonGiaNhap DECIMAL(15,2) NOT NULL, -- Giá nhập

ThanhTien DECIMAL(15,2) AS (SoLuongNhap \* DonGiaNhap) STORED, -- Tự tính

PRIMARY KEY (MaPN, MaVL),

FOREIGN KEY (MaPN) REFERENCES phieunhap(MaPN),

FOREIGN KEY (MaVL) REFERENCES vatlieu(MaVL)

);

**A screenshot of a computer

AI-generated content may be incorrect.**

**Hình 4: Màn hình sau khi đăng nhập, tạo csdl và tạo bảng thành công.**

## **Viết code với ứng dụng MYSQL + Tkinter**

### **Màn hình đăng nhập và phân quyền truy cập**

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 5: Màn hình đăng nhập.

A white background with red and blue text

AI-generated content may be incorrect.

Hình 6: Màn hình đăng nhập với vai trò là quản lý (admin)

A white background with red and blue text

AI-generated content may be incorrect.

Hình 7: Màn hình đăng nhập với vai trò là nhân viên.

### **Quản lý nhân viên**

A screenshot of a computer

AI-generated content may be incorrect.Giúp doanh nghiệp quản lý đội ngũ nhân sự, theo dõi thông tin nhân viên để phục vụ phân công công việc, phân quyền hệ thống và đảm bảo hoạt động bán hàng/nhập kho diễn ra chính xác (mỗi hóa đơn, phiếu nhập đều gắn với một nhân viên).

Hình 8: Giao diện quản lý nhân viên với vai trò là quản lý

A screenshot of a computer

AI-generated content may be incorrect.

Hình 9: Giao diện quản lý nhân viên với vai trò là nhân viên.

**Chi tiết code dùng để quản lý nhân viên:**

class NhanVienFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
 self.create\_ui()  
 self.load\_nv()  
  
 # ==============================================================================  
 # VALIDATION & FORMAT  
 # ==============================================================================  
 def valid\_phone(self, phone):  
 return phone.isdigit() and len(phone) == 10  
  
 def valid\_date(self, date\_text):  
 try:  
 datetime.datetime.strptime(date\_text, "%Y-%m-%d")  
 return True  
 except:  
 return False  
  
 def valid\_salary(self, salary):  
 cleaned = salary.replace(".", "").replace(",", "")  
 return cleaned.isdigit()  
  
 def format\_name(self, text):  
 return " ".join(word.capitalize() for word in text.split())  
  
 def format\_salary\_display(self, s):  
 *"""Hiện lương dạng 1.000.000"""* try:  
 return "{:,}".format(int(s)).replace(",", ".")  
 except:  
 return s  
  
 # ====================================================================================  
 # TẠO GIAO DIỆN  
 # ====================================================================================  
 def create\_ui(self):  
  
 tk.Label(  
 self, text="QUẢN LÝ NHÂN VIÊN",  
 font=("Arial", 18, "bold"),  
 bg="white", fg="#003A75"  
 ).pack(pady=10)  
  
 frame\_nv\_info = tk.Frame(self, bg="white")  
 frame\_nv\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_nv\_info.columnconfigure(i, weight=1)  
  
 self.entry\_manv = self.\_make\_entry(frame\_nv\_info, "Mã nhân viên:", 0, 0)  
 self.cbb\_chucvu = self.\_make\_combo(frame\_nv\_info, "Chức vụ:", 0, 2,  
 [" ", "Quản lý", "Kế toán", "Lái xe", "Nhân viên bán hàng", "Nhân viên kho"])  
  
 self.entry\_hotennv = self.\_make\_entry(frame\_nv\_info, "Họ tên:", 1, 0)  
 self.entry\_sdt = self.\_make\_entry(frame\_nv\_info, "Số điện thoại:", 1, 2)  
  
 tk.Label(frame\_nv\_info, text="Phái:", bg="white").grid(row=2, column=0, sticky="w", pady=5)  
 frame\_gt = tk.Frame(frame\_nv\_info, bg="white")  
 frame\_gt.grid(row=2, column=1, sticky="we", padx=5)  
  
 self.var\_phai = tk.StringVar(value="Nam")  
 tk.Radiobutton(frame\_gt, text="Nam", variable=self.var\_phai, value="Nam", bg="white").pack(side="left")  
 tk.Radiobutton(frame\_gt, text="Nữ", variable=self.var\_phai, value="Nữ", bg="white").pack(side="left")  
  
 self.entry\_ngaysinh = self.\_make\_entry(frame\_nv\_info, "Ngày sinh (yyyy-mm-dd):", 2, 2)  
  
 self.entry\_diachi = self.\_make\_entry(frame\_nv\_info, "Địa chỉ:", 3, 0)  
 self.entry\_luong = self.\_make\_entry(frame\_nv\_info, "Lương:", 3, 2)  
  
 self.cbb\_tinhtrang = self.\_make\_combo(frame\_nv\_info, "Tình trạng:", 4, 0, [" ", "Đang làm", "Nghỉ việc", "Tạm nghỉ"])  
  
 # ================================= SEARCH ==============================  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, pady=10, fill="x")  
  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(frame\_search, values=[" ", "MaNV", "HoTenNV", "Sdt", "ChucVu", "TinhTrang"],  
 state="readonly")  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
  
 tk.Button(frame\_search, text="Tìm", bg="#003A75", fg="white", width=12, command=self.search\_nv).grid(row=1, column=2, padx=5)  
  
 # ================================ TABLE ========================================  
 tk.Label(self, text="Danh sách nhân viên", bg="white", font=("Arial", 11, "bold")).pack(anchor="w", padx=20)  
  
 columns = ("MaNV", "HoTenNV", "Sdt", "Phai", "NgaySinh", "ChucVu", "DiaChi", "Luong", "TinhTrang")  
  
 self.tree = ttk.Treeview(self, columns=columns, show="headings", height=11)  
 for col in columns:  
 self.tree.heading(col, text=col)  
 self.tree.column(col, width=120)  
  
 self.tree.pack(padx=10, pady=5, fill="both")  
  
 # ============================== BUTTONS ======================================  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_nv, \*\*btn).grid(row=0, column=0, padx=5)  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
 tk.Button(frame\_btn, text="Sửa", bg="#E74C3C",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
 tk.Button(frame\_btn, text="Hủy", bg="#E67E22",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=3, padx=5)  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_nv, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ========================================================================================  
 # ENTRY / COMBO TẠO TỰ ĐỘNG  
 # ========================================================================================  
 def \_make\_entry(self, parent, label, row, col):  
 tk.Label(parent, text=label, bg="white").grid(row=row, column=col, sticky="w", pady=5)  
 e = tk.Entry(parent)  
 e.grid(row=row, column=col+1, sticky="we", padx=5)  
 return e  
  
 def \_make\_combo(self, parent, label, row, col, values):  
 tk.Label(parent, text=label, bg="white").grid(row=row, column=col, sticky="w", pady=5)  
 cb = ttk.Combobox(parent, values=values, state="readonly")  
 cb.grid(row=row, column=col+1, sticky="we", padx=5)  
 return cb  
  
 # ======================================================================================  
 # CHỨC NĂNG  
 # ======================================================================================  
 def get\_values(self):  
 return [  
 self.entry\_manv.get(),  
 self.format\_name(self.entry\_hotennv.get()),  
 self.entry\_sdt.get(),  
 self.var\_phai.get(),  
 self.entry\_ngaysinh.get(),  
 self.cbb\_chucvu.get(),  
 self.format\_name(self.entry\_diachi.get()), # <== viết hoa địa chỉ  
 self.entry\_luong.get(),  
 self.cbb\_tinhtrang.get(),]  
  
 # =================================== LOAD TABLE =============================================  
 def load\_nv(self):  
 self.tree.delete(\*self.tree.get\_children())  
 for r in load\_table("nhanvien"):  
 r = list(r)  
  
 # Format lương để HIỂN THỊ  
 try:  
 r[7] = self.format\_salary\_display(r[7])  
 except:  
 pass  
  
 self.tree.insert("", "end", values=r)  
  
 # ====================================== ADD ============================================  
 def add\_nv(self):  
 values = self.get\_values()  
 manv, hoten, sdt, phai, ngaysinh, chucvu, diachi, luong, tinhtrang = values  
  
 if "" in values:  
 return messagebox.showwarning("Thiếu dữ liệu", "Vui lòng nhập đầy đủ thông tin!")  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "Số điện thoại phải đúng 10 số!")  
 if not self.valid\_date(ngaysinh):  
 return messagebox.showerror("Lỗi", "Ngày phải dạng YYYY-MM-DD!")  
 if not self.valid\_salary(luong):  
 return messagebox.showerror("Lỗi", "Lương phải là số hợp lệ!")  
  
 # CHUYỂN LƯƠNG VỀ SỐ INT  
 luong\_num = int(luong.replace(".", "").replace(",", ""))  
 values[7] = luong\_num  
  
 fields = ["MaNV", "HoTenNV", "Sdt", "Phai", "NgaySinh",  
 "ChucVu", "DiaChi", "Luong", "TinhTrang"]  
 add\_record("nhanvien", fields, values)  
 messagebox.showinfo("OK", "Thêm nhân viên thành công!")  
 self.load\_nv()  
 self.clear\_input()  
  
 # ==================================== DELETE ==============================================  
 def delete\_nv(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn nhân viên để xóa!")  
  
 manv = self.tree.item(sel)["values"][0]  
 if not messagebox.askyesno("Xác nhận", f"Bạn có chắc muốn xóa nhân viên {manv}?"):  
 return  
  
 delete\_record("nhanvien", "MaNV", manv)  
 messagebox.showinfo("OK", "Đã xóa!")  
 self.load\_nv()  
 self.clear\_input()  
  
 # ========================================== LOAD TO FORM ==========================================  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn nhân viên để sửa!")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_manv.delete(0, tk.END)  
 self.entry\_manv.insert(0, data[0])  
  
 self.entry\_hotennv.delete(0, tk.END)  
 self.entry\_hotennv.insert(0, data[1])  
  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_sdt.insert(0, data[2])  
  
 self.var\_phai.set(data[3])  
  
 self.entry\_ngaysinh.delete(0, tk.END)  
 self.entry\_ngaysinh.insert(0, data[4])  
  
 self.cbb\_chucvu.set(data[5])  
  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_diachi.insert(0, data[6])  
  
 self.entry\_luong.delete(0, tk.END)  
 self.entry\_luong.insert(0, data[7])  
  
 self.cbb\_tinhtrang.set(data[8])  
  
 # ================================= SAVE EDIT =======================================  
 def save\_edit(self):  
 manv = self.entry\_manv.get()  
 values = self.get\_values()  
 (\_, hoten, sdt, phai, ngaysinh, chucvu, diachi, luong, tinhtrang) = values  
  
 if "" in values:  
 return messagebox.showwarning("Thiếu dữ liệu", "Vui lòng nhập đầy đủ thông tin!")  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "SĐT phải đúng 10 số!")  
 if not self.valid\_date(ngaysinh):  
 return messagebox.showerror("Lỗi", "Ngày sinh phải dạng YYYY-MM-DD!")  
 if not self.valid\_salary(luong):  
 return messagebox.showerror("Lỗi", "Lương phải là số hợp lệ!")  
  
 # CHUYỂN LƯƠNG VỀ INT  
 luong\_num = int(luong.replace(".", "").replace(",", ""))  
 fields = ["HoTenNV", "Sdt", "Phai", "NgaySinh",  
 "ChucVu", "DiaChi", "Luong", "TinhTrang"]  
 new\_values = [hoten, sdt, phai, ngaysinh, chucvu, diachi, luong\_num, tinhtrang]  
 update\_record("nhanvien", "MaNV", manv, fields, new\_values)  
  
 messagebox.showinfo("OK", "Cập nhật thành công!")  
 self.load\_nv()  
 self.clear\_input()  
  
 # ================================ SEARCH ===========================================  
 def search\_nv(self):  
 keyword = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
 if field == " ":  
 return self.load\_nv()  
   
 rows = search\_record("nhanvien", field, keyword)  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 # Format salary display  
 r = list(r)  
 try:  
 r[7] = self.format\_salary\_display(r[7])  
 except:  
 pass  
 self.tree.insert("", "end", values=r)  
  
 # =================================== CLEAR ======================================  
 def clear\_input(self):  
 self.entry\_manv.delete(0, tk.END)  
 self.entry\_hotennv.delete(0, tk.END)  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_ngaysinh.delete(0, tk.END)  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_luong.delete(0, tk.END)  
 self.var\_phai.set("Nam")  
 self.cbb\_chucvu.set("")  
 self.cbb\_tinhtrang.set("")

### **Quản lý khách hàng**

A screenshot of a computer

AI-generated content may be incorrect.Hỗ trợ lưu trữ và tra cứu thông tin khách hàng, phục vụ chăm sóc khách hàng, bán hàng và thống kê lịch sử giao dịch để cải thiện dịch vụ và tăng doanh số.

Hình 10: Giao diện quản lý khách hàng

**Chi tiết code dùng để quản lý khách hàng:**

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
if BASE\_DIR not in sys.path:  
 sys.path.append(BASE\_DIR)  
  
import tkinter as tk  
from tkinter import ttk, messagebox  
  
from db\_functions import (  
 load\_table, add\_record, delete\_record,  
 update\_record, search\_record  
)  
  
  
class KhachHangFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
 self.create\_ui()  
 self.load\_kh()  
  
 # ======================================================  
 # VALIDATION & FORMAT  
 # ======================================================  
 def valid\_phone(self, phone):  
 return phone.isdigit() and len(phone) == 10  
  
 def valid\_cccd(self, cccd):  
 return cccd.isdigit() and len(cccd) == 12  
  
 def format\_name(self, text):  
 return " ".join(word.capitalize() for word in text.split())  
  
 # ======================================================  
 # UI  
 # ======================================================  
 def create\_ui(self):  
  
 tk.Label(  
 self, text="QUẢN LÝ KHÁCH HÀNG",  
 font=("Arial", 18, "bold"),  
 bg="white", fg="#003A75"  
 ).pack(pady=10)  
  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1, uniform="col")  
  
 # Hàng 1  
 self.entry\_makh = self.\_make\_entry(frame\_info, "Mã KH:", 0, 0)  
 self.entry\_tenkh = self.\_make\_entry(frame\_info, "Họ tên KH:", 0, 2)  
  
 # Hàng 2  
 self.entry\_sdt = self.\_make\_entry(frame\_info, "Số điện thoại:", 1, 0)  
 self.entry\_cccd = self.\_make\_entry(frame\_info, "CCCD:", 1, 2)  
  
 # Hàng 3  
 self.entry\_diachi = self.\_make\_entry(frame\_info, "Địa chỉ:", 2, 0)  
  
 # ================== SEARCH ==================  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, pady=10, fill="x")  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(  
 frame\_search,  
 values=[" ", "MaKH", "HoTen", "SDT", "CCCD", "DiaChi"],  
 state="readonly"  
 )  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
 self.cbb\_search\_type.set(" ")  
  
 tk.Button(  
 frame\_search, text="Tìm", width=12,  
 bg="#003A75", fg="white", command=self.search\_kh  
 ).grid(row=1, column=2, padx=5)  
  
 # ================== TABLE ==================  
 tk.Label(self, text="Danh sách khách hàng", bg="white",  
 font=("Arial", 11, "bold")).pack(anchor="w", padx=20)  
  
 self.columns = ("MaKH", "HoTen", "SDT", "DiaChi", "CCCD")  
  
 self.tree = ttk.Treeview(self, columns=self.columns, show="headings", height=11)  
 for col in self.columns:  
 self.tree.heading(col, text=col)  
 self.tree.column(col, width=150)  
 self.tree.pack(padx=10, pady=5, fill="both")  
  
 # ================== BUTTONS ==================  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_kh, \*\*btn).grid(row=0, column=0, padx=5)  
  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
  
 tk.Button(frame\_btn, text="Sửa", bg="#E74C3C",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
  
 tk.Button(frame\_btn, text="Hủy", bg="#E67E22",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=3, padx=5)  
  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_kh, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ======================================================  
 # ENTRY HELPER  
 # ======================================================  
 def \_make\_entry(self, parent, text, row, col):  
 tk.Label(parent, text=text, bg="white").grid(row=row, column=col, sticky="w", pady=5)  
 e = tk.Entry(parent)  
 e.grid(row=row, column=col+1, sticky="we", padx=5)  
 return e  
  
 # ======================================================  
 # GET VALUES  
 # ======================================================  
 def get\_values(self):  
 return [  
 self.entry\_makh.get().strip(),  
 self.format\_name(self.entry\_tenkh.get()),  
 self.entry\_sdt.get().strip(),  
 self.format\_name(self.entry\_diachi.get()),  
 self.entry\_cccd.get().strip()  
 ]  
  
 # ======================================================  
 # LOAD TABLE  
 # ======================================================  
 def load\_kh(self):  
 self.tree.delete(\*self.tree.get\_children())  
 for r in load\_table("khachhang"):  
 self.tree.insert("", "end", values=r)  
  
 # ======================================================  
 # ADD  
 # ======================================================  
 def add\_kh(self):  
 makh, hoten, sdt, diachi, cccd = self.get\_values()  
  
 if "" in [makh, hoten, sdt, diachi, cccd]:  
 return messagebox.showwarning("Thiếu dữ liệu", "Vui lòng nhập đầy đủ thông tin!")  
  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "Số điện thoại phải đúng 10 số!")  
  
 if not self.valid\_cccd(cccd):  
 return messagebox.showerror("Lỗi", "CCCD phải đúng 12 số!")  
  
 fields = ["MaKH", "HoTen", "SDT", "DiaChi", "CCCD"]  
 add\_record("khachhang", fields, [makh, hoten, sdt, diachi, cccd])  
  
 messagebox.showinfo("OK", "Thêm khách hàng thành công!")  
 self.load\_kh()  
 self.clear\_input()  
  
 # ======================================================  
 # DELETE  
 # ======================================================  
 def delete\_kh(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn khách hàng để xóa!")  
  
 makh = self.tree.item(sel)["values"][0]  
  
 if not messagebox.askyesno("Xác nhận", f"Bạn có chắc muốn xóa khách hàng {makh}?"):  
 return  
  
 delete\_record("khachhang", "MaKH", makh)  
  
 messagebox.showinfo("OK", "Đã xóa!")  
 self.load\_kh()  
 self.clear\_input()  
  
 # ======================================================  
 # LOAD TO EDIT FORM  
 # ======================================================  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn khách hàng để sửa!")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_makh.delete(0, tk.END)  
 self.entry\_makh.insert(0, data[0])  
  
 self.entry\_tenkh.delete(0, tk.END)  
 self.entry\_tenkh.insert(0, data[1])  
  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_sdt.insert(0, data[2])  
  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_diachi.insert(0, data[3])  
  
 self.entry\_cccd.delete(0, tk.END)  
 self.entry\_cccd.insert(0, data[4])  
  
 # ======================================================  
 # SAVE EDIT  
 # ======================================================  
 def save\_edit(self):  
 makh = self.entry\_makh.get().strip()  
 \_, hoten, sdt, diachi, cccd = self.get\_values()  
  
 if "" in [makh, hoten, sdt, diachi, cccd]:  
 return messagebox.showwarning("Lỗi", "Không được để trống!")  
  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "SĐT phải đúng 10 số!")  
  
 if not self.valid\_cccd(cccd):  
 return messagebox.showerror("Lỗi", "CCCD phải đúng 12 số!")  
  
 fields = ["HoTen", "SDT", "DiaChi", "CCCD"]  
 new\_data = [hoten, sdt, diachi, cccd]  
  
 update\_record("khachhang", "MaKH", makh, fields, new\_data)  
  
 messagebox.showinfo("OK", "Cập nhật thành công!")  
 self.load\_kh()  
 self.clear\_input()  
  
 # ======================================================  
 # SEARCH  
 # ======================================================  
 def search\_kh(self):  
 keyword = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
  
 if field == " ":  
 return self.load\_kh()  
  
 rows = search\_record("khachhang", field, keyword)  
  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 self.tree.insert("", "end", values=r)  
  
 # ======================================================  
 # CLEAR INPUT  
 # ======================================================  
 def clear\_input(self):  
 self.entry\_makh.delete(0, tk.END)  
 self.entry\_tenkh.delete(0, tk.END)  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_cccd.delete(0, tk.END)

### **Quản lý nhà cung cấp**

A screenshot of a computer

AI-generated content may be incorrect.Quản lý thông tin các nhà cung cấp để thuận tiện cho việc nhập hàng, theo dõi nguồn cung, kiểm soát giá nhập, và đảm bảo hoạt động mua – bán liên tục, ổn định.

Hình 11: Giao diện quản lý nhà cung cấp với vai trò quản lý

A screenshot of a computer

AI-generated content may be incorrect.Hình 12: Giao diện quản lý nhà cung cấp với vai trò là nhân viên.

**Chi tiết code dùng để quản lý nhà cung cấp:**

class NhaCungCapFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
 self.create\_ui()  
 self.load\_ncc()  
  
 # ============================================================  
 # VALIDATION & FORMAT  
 # ============================================================  
 def valid\_phone(self, phone: str):  
 return phone.isdigit() and len(phone) == 10  
  
 def format\_name(self, text):  
 *"""Viết hoa mỗi chữ cái đầu và loại bỏ khoảng trắng thừa"""* return " ".join(word.capitalize() for word in text.split())  
  
 # ============================================================  
 # TẠO GIAO DIỆN  
 # ============================================================  
 def create\_ui(self):  
  
 tk.Label(  
 self, text="QUẢN LÝ NHÀ CUNG CẤP",  
 font=("Arial", 18, "bold"),  
 bg="white", fg="#003A75"  
 ).pack(pady=10)  
  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1, uniform="col")  
  
 # Hàng 1  
 self.entry\_mancc = self.\_make\_entry(frame\_info, "Mã NCC:", 0, 0)  
 self.entry\_tenncc = self.\_make\_entry(frame\_info, "Tên NCC:", 0, 2)  
  
 # Hàng 2  
 self.entry\_sdt = self.\_make\_entry(frame\_info, "Số điện thoại:", 1, 0)  
 self.entry\_diachi = self.\_make\_entry(frame\_info, "Địa chỉ:", 1, 2)  
  
 # Hàng 3  
 self.entry\_ghichu = self.\_make\_entry(frame\_info, "Ghi chú:", 2, 0)  
  
 # ================== SEARCH ==================  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, pady=10, fill="x")  
  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(  
 frame\_search,  
 values=[" ", "MaNCC", "TenNCC", "SDT"],  
 state="readonly"  
 )  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
 self.cbb\_search\_type.set(" ")  
  
 tk.Button(  
 frame\_search, text="Tìm kiếm", bg="#003A75", fg="white",  
 width=12, command=self.search\_ncc  
 ).grid(row=1, column=2, padx=5)  
  
 # ================= TABLE ====================  
 tk.Label(self, text="Danh sách nhà cung cấp",  
 bg="white",  
 font=("Arial", 11, "bold")  
 ).pack(anchor="w", padx=20)  
  
 columns = ("MaNCC", "TenNCC", "SDT", "DiaChi", "GhiChu")  
 self.tree = ttk.Treeview(self, columns=columns, show="headings", height=12)  
  
 for col in columns:  
 self.tree.heading(col, text=col)  
 self.tree.column(col, width=150)  
  
 self.tree.pack(padx=10, pady=5, fill="both")  
  
 # ================= BUTTONS ==================  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_ncc, \*\*btn).grid(row=0, column=0, padx=5)  
  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
  
 tk.Button(frame\_btn, text="Sửa", bg="#E74C3C",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
  
 tk.Button(frame\_btn, text="Hủy", bg="#E67E22",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=3, padx=5)  
  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_ncc, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ============================================================  
 # HÀM TẠO ENTRY CHUẨN  
 # ============================================================  
 def \_make\_entry(self, parent, text, row, col):  
 tk.Label(parent, text=text, bg="white").grid(row=row, column=col, sticky="w", pady=5)  
 e = tk.Entry(parent)  
 e.grid(row=row, column=col + 1, sticky="we", padx=5)  
 return e  
  
 # ============================================================  
 # CRUD  
 # ============================================================  
 def get\_values(self):  
 return [  
 self.entry\_mancc.get().strip(),  
 self.format\_name(self.entry\_tenncc.get()),  
 self.entry\_sdt.get().strip(),  
 self.format\_name(self.entry\_diachi.get()),  
 self.entry\_ghichu.get().strip()  
 ]  
  
 # ---------------- LOAD ----------------  
 def load\_ncc(self):  
 self.tree.delete(\*self.tree.get\_children())  
 for r in load\_table("nhacungcap"):  
 self.tree.insert("", "end", values=r)  
  
 # ---------------- ADD -----------------  
 def add\_ncc(self):  
 values = self.get\_values()  
  
 mancc, tenncc, sdt, diachi, ghichu = values  
  
 if "" in [mancc, tenncc, sdt, diachi]:  
 return messagebox.showwarning("Thiếu dữ liệu", "Vui lòng nhập đầy đủ thông tin!")  
  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "SĐT phải đúng 10 số!")  
  
 fields = ["MaNCC", "TenNCC", "SDT", "DiaChi", "GhiChu"]  
  
 add\_record("nhacungcap", fields, values)  
 messagebox.showinfo("OK", "Thêm nhà cung cấp thành công!")  
 self.load\_ncc()  
 self.clear\_input()  
  
 # ---------------- DELETE -----------------  
 def delete\_ncc(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn NCC để xóa!")  
  
 mancc = self.tree.item(sel)["values"][0]  
  
 if not messagebox.askyesno("Xác nhận", f"Xóa nhà cung cấp {mancc}?"):  
 return  
  
 delete\_record("nhacungcap", "MaNCC", mancc)  
 messagebox.showinfo("OK", "Đã xóa!")  
 self.load\_ncc()  
 self.clear\_input()  
  
 # ---------------- LOAD EDIT -----------------  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn NCC để sửa!")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_mancc.delete(0, tk.END)  
 self.entry\_mancc.insert(0, data[0])  
  
 self.entry\_tenncc.delete(0, tk.END)  
 self.entry\_tenncc.insert(0, data[1])  
  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_sdt.insert(0, data[2])  
  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_diachi.insert(0, data[3])  
  
 self.entry\_ghichu.delete(0, tk.END)  
 self.entry\_ghichu.insert(0, data[4])  
  
 # ---------------- SAVE EDIT -----------------  
 def save\_edit(self):  
 mancc = self.entry\_mancc.get().strip()  
 \_, tenncc, sdt, diachi, ghichu = self.get\_values()  
  
 if "" in [mancc, tenncc, sdt, diachi]:  
 return messagebox.showwarning("Lỗi", "Không được để trống!")  
  
 if not self.valid\_phone(sdt):  
 return messagebox.showerror("Lỗi", "SĐT phải đúng 10 số!")  
  
 fields = ["TenNCC", "SDT", "DiaChi", "GhiChu"]  
 new\_values = [tenncc, sdt, diachi, ghichu]  
  
 update\_record("nhacungcap", "MaNCC", mancc, fields, new\_values)  
  
 messagebox.showinfo("OK", "Cập nhật thành công!")  
 self.load\_ncc()  
 self.clear\_input()  
  
 # ---------------- SEARCH -----------------  
 def search\_ncc(self):  
 keyword = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
  
 if field == " ":  
 return self.load\_ncc()  
  
 rows = search\_record("nhacungcap", field, keyword)  
  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 self.tree.insert("", "end", values=r)  
  
 # ---------------- CLEAR -----------------  
 def clear\_input(self):  
 self.entry\_mancc.delete(0, tk.END)  
 self.entry\_tenncc.delete(0, tk.END)  
 self.entry\_sdt.delete(0, tk.END)  
 self.entry\_diachi.delete(0, tk.END)  
 self.entry\_ghichu.delete(0, tk.END)

### **Quản lý vật liệu/ sản phẩm**

**A screenshot of a computer

AI-generated content may be incorrect.**Giúp hệ thống theo dõi đầy đủ thông tin về sản phẩm hoặc vật liệu (giá, tồn kho, đơn vị tính…), hỗ trợ bán hàng, nhập kho và kiểm soát hàng tồn, tránh thiếu hàng hoặc dư thừa.

Hình 13: Giao diện quản lý vật liệu

**Chi tiết code dùng để quản lý vật liệu:**

class VatLieuFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
  
 self.ncc\_map = {}  
 self.reload\_ncc\_map()  
  
 self.create\_ui()  
 self.load\_vl()  
  
 # ===================================================================  
 # FORMAT & VALIDATION (THÊM GIỐNG NHÂN VIÊN)  
 # ===================================================================  
  
 def format\_name(self, text):  
 *"""Viết hoa từng chữ đầu giống Nhân Viên"""* return " ".join(word.capitalize() for word in text.split())  
  
 def valid\_price(self, s):  
 *"""Kiểm tra giá hợp lệ"""* return s.replace(".", "").replace(",", "").isdigit()  
  
 def format\_price\_display(self, s):  
 *"""Hiển thị giá thành 1.000.000"""* try:  
 return "{:,}".format(int(s)).replace(",", ".")  
 except:  
 return s  
  
 # ===================================================================  
 # LOAD MAP NHÀ CUNG CẤP  
 # ===================================================================  
 def reload\_ncc\_map(self):  
 self.ncc\_map = {}  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("SELECT MaNCC, TenNCC FROM nhacungcap")  
 for ma, ten in cur.fetchall():  
 self.ncc\_map[ten] = ma  
 conn.close()  
 except:  
 self.ncc\_map = {}  
  
 # ===================================================================  
 # UI  
 # ===================================================================  
 def create\_ui(self):  
 tk.Label(self, text="QUẢN LÝ VẬT LIỆU", font=("Arial", 18, "bold"),  
 bg="white", fg="#003A75").pack(pady=10)  
  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1)  
  
 self.entry\_mavl = self.\_make\_entry(frame\_info, "Mã vật liệu:", 0, 0)  
 self.entry\_tenvl = self.\_make\_entry(frame\_info, "Tên vật liệu:", 0, 2)  
  
 tk.Label(frame\_info, text="Đơn vị tính:", bg="white").grid(row=1, column=0, sticky="w")  
 self.cbb\_donvi = ttk.Combobox(  
 frame\_info, state="readonly",  
 values=[" ", "cây", "bao", "viên", "thùng", "m", "m²", "m³", "cuộn", "tấm", "thùng"]  
 )  
 self.cbb\_donvi.grid(row=1, column=1, sticky="we", padx=5)  
 self.cbb\_donvi.set(" ")  
  
 self.entry\_gianhap = self.\_make\_entry(frame\_info, "Giá nhập:", 1, 2)  
 self.entry\_giaban = self.\_make\_entry(frame\_info, "Giá bán:", 2, 0)  
 self.entry\_tonkho = self.\_make\_entry(frame\_info, "Tồn kho:", 2, 2)  
  
 tk.Label(frame\_info, text="Nhà cung cấp:", bg="white").grid(row=3, column=0, sticky="w")  
 self.cbb\_ncc = ttk.Combobox(frame\_info, state="readonly", values=list(self.ncc\_map.keys()))  
 self.cbb\_ncc.grid(row=3, column=1, sticky="we", padx=5)  
  
 # ------------------------------------------------------------------  
 # TÌM KIẾM  
 # ------------------------------------------------------------------  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(fill="x", padx=20)  
  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(  
 frame\_search,  
 values=[" ", "MaVL", "TenVL", "DonViTinh", "TenNCC"],  
 state="readonly"  
 )  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
 self.cbb\_search\_type.set(" ")  
  
 tk.Button(frame\_search, text="Tìm kiếm",  
 bg="#003A75", fg="white",  
 width=12, command=self.search\_vl).grid(row=1, column=2, padx=5)  
  
 # ------------------------------------------------------------------  
 # TABLE  
 # ------------------------------------------------------------------  
 tk.Label(self, text="Danh sách vật liệu", bg="white",  
 font=("Arial", 11, "bold")).pack(anchor="w", padx=20)  
  
 columns = ("MaVL", "TenVL", "DonViTinh", "GiaNhap", "GiaBan", "TonKho", "TenNCC")  
  
 self.tree = ttk.Treeview(self, columns=columns, show="headings", height=12)  
 for col in columns:  
 self.tree.heading(col, text=col)  
 self.tree.column(col, width=120)  
  
 self.tree.pack(fill="both", padx=10, pady=5)  
  
 # ------------------------------------------------------------------  
 # BUTTONS  
 # ------------------------------------------------------------------  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_vl, \*\*btn).grid(row=0, column=0, padx=5)  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
 tk.Button(frame\_btn, text="Sửa", bg="#E74C3C",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
 tk.Button(frame\_btn, text="Hủy", bg="#E67E22",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=3, padx=5)  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_vl, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ===================================================================  
 # ENTRY TẠO NHANH  
 # ===================================================================  
 def \_make\_entry(self, parent, text, row, col):  
 tk.Label(parent, text=text, bg="white").grid(row=row, column=col, sticky="w")  
 e = tk.Entry(parent)  
 e.grid(row=row, column=col+1, sticky="we", padx=5)  
 return e  
  
 # ===================================================================  
 # LOAD VẬT LIỆU  
 # ===================================================================  
 def load\_vl(self):  
 self.reload\_ncc\_map()  
 self.cbb\_ncc["values"] = list(self.ncc\_map.keys())  
  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 SELECT v.MaVL, v.TenVL, v.DonViTinh, v.GiaNhap,  
 v.GiaBan, v.TonKho, n.TenNCC  
 FROM vatlieu v  
 LEFT JOIN nhacungcap n ON v.MaNCC = n.MaNCC  
 """)  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
  
 # format giá giống Nhân Viên  
 for r in rows:  
 r = list(r)  
 r[3] = self.format\_price\_display(r[3]) # giá nhập  
 r[4] = self.format\_price\_display(r[4]) # giá bán  
 self.tree.insert("", "end", values=r)  
  
 # ===================================================================  
 # THÊM  
 # ===================================================================  
 def add\_vl(self):  
 mavl = self.entry\_mavl.get()  
 tenvl = self.format\_name(self.entry\_tenvl.get())  
 dv = self.cbb\_donvi.get()  
 gianhap = self.entry\_gianhap.get()  
 giaban = self.entry\_giaban.get()  
 tonkho = self.entry\_tonkho.get()  
 tenncc = self.cbb\_ncc.get()  
  
 if not all([mavl, tenvl, dv, gianhap, giaban, tonkho, tenncc]):  
 return messagebox.showwarning("Thiếu", "Vui lòng nhập đủ thông tin!")  
  
 if not self.valid\_price(gianhap) or not self.valid\_price(giaban):  
 return messagebox.showerror("Lỗi", "Giá phải là số!")  
  
 mancc = self.ncc\_map.get(tenncc)  
  
 gianhap\_num = int(gianhap.replace(".", "").replace(",", ""))  
 giaban\_num = int(giaban.replace(".", "").replace(",", ""))  
  
 conn = connect\_db()  
 cur = conn.cursor()  
 try:  
 cur.execute("""  
 INSERT INTO vatlieu (MaVL, TenVL, DonViTinh, GiaNhap, GiaBan, TonKho, MaNCC)  
 VALUES (%s,%s,%s,%s,%s,%s,%s)  
 """, (mavl, tenvl, dv, gianhap\_num, giaban\_num, tonkho, mancc))  
 conn.commit()  
 messagebox.showinfo("OK", "Thêm thành công!")  
 self.load\_vl()  
 self.clear\_input()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
 conn.close()  
  
 # ===================================================================  
 # XÓA  
 # ===================================================================  
 def delete\_vl(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn!", "Chọn vật liệu để xóa!")  
  
 mavl = self.tree.item(sel)["values"][0]  
 if not messagebox.askyesno("Xác nhận", f"Bạn có chắc muốn xóa nhân viên {mavl}?"):  
 return  
  
 conn = connect\_db()  
 cur = conn.cursor()  
 try:  
 cur.execute("DELETE FROM vatlieu WHERE MaVL = %s", (mavl,))  
 conn.commit()  
 messagebox.showinfo("OK", "Xóa thành công!")  
 self.load\_vl()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
 conn.close()  
  
 # ===================================================================  
 # LOAD ĐỂ SỬA  
 # ===================================================================  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn!", "Chọn vật liệu để sửa!")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_mavl.delete(0, tk.END)  
 self.entry\_mavl.insert(0, data[0])  
  
 self.entry\_tenvl.delete(0, tk.END)  
 self.entry\_tenvl.insert(0, data[1])  
  
 self.cbb\_donvi.set(data[2])  
  
 self.entry\_gianhap.delete(0, tk.END)  
 self.entry\_gianhap.insert(0, data[3])  
  
 self.entry\_giaban.delete(0, tk.END)  
 self.entry\_giaban.insert(0, data[4])  
  
 self.entry\_tonkho.delete(0, tk.END)  
 self.entry\_tonkho.insert(0, data[5])  
  
 self.cbb\_ncc.set(data[6])  
  
 # ===================================================================  
 # LƯU SỬA  
 # ===================================================================  
 def save\_edit(self):  
 mavl = self.entry\_mavl.get()  
 tenvl = self.format\_name(self.entry\_tenvl.get())  
 dv = self.cbb\_donvi.get()  
 gianhap = self.entry\_gianhap.get()  
 giaban = self.entry\_giaban.get()  
 tonkho = self.entry\_tonkho.get()  
 tenncc = self.cbb\_ncc.get()  
  
 if not self.valid\_price(gianhap) or not self.valid\_price(giaban):  
 return messagebox.showerror("Lỗi", "Giá phải là số hợp lệ!")  
  
 gianhap\_num = int(gianhap.replace(".", "").replace(",", ""))  
 giaban\_num = int(giaban.replace(".", "").replace(",", ""))  
  
 mancc = self.ncc\_map.get(tenncc)  
  
 conn = connect\_db()  
 cur = conn.cursor()  
 try:  
 cur.execute("""  
 UPDATE vatlieu SET  
 TenVL=%s, DonViTinh=%s, GiaNhap=%s, GiaBan=%s, TonKho=%s, MaNCC=%s  
 WHERE MaVL=%s  
 """, (tenvl, dv, gianhap\_num, giaban\_num, tonkho, mancc, mavl))  
  
 conn.commit()  
 messagebox.showinfo("OK", "Lưu thành công!")  
 self.load\_vl()  
 self.clear\_input()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
 conn.close()  
  
 # ===================================================================  
 # TÌM KIẾM  
 # ===================================================================  
 def search\_vl(self):  
 keyword = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
  
 if keyword == "":  
 return self.load\_vl()  
  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 if field == "TenNCC":  
 cur.execute("""  
 SELECT v.MaVL, v.TenVL, v.DonViTinh, v.GiaNhap,  
 v.GiaBan, v.TonKho, n.TenNCC  
 FROM vatlieu v  
 LEFT JOIN nhacungcap n ON v.MaNCC = n.MaNCC  
 WHERE n.TenNCC LIKE %s  
 """, ('%' + keyword + '%',))  
 else:  
 cur.execute(f"""  
 SELECT v.MaVL, v.TenVL, v.DonViTinh, v.GiaNhap,  
 v.GiaBan, v.TonKho, n.TenNCC  
 FROM vatlieu v  
 LEFT JOIN nhacungcap n ON v.MaNCC = n.MaNCC  
 WHERE {field} LIKE %s  
 """, ('%' + keyword + '%',))  
  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
  
 # format giá lại  
 for r in rows:  
 r = list(r)  
 r[3] = self.format\_price\_display(r[3])  
 r[4] = self.format\_price\_display(r[4])  
 self.tree.insert("", "end", values=r)  
  
 # ===================================================================  
 # CLEAR INPUT  
 # ===================================================================  
 def clear\_input(self):  
 self.entry\_mavl.delete(0, tk.END)  
 self.entry\_tenvl.delete(0, tk.END)  
 self.cbb\_donvi.set(" ")  
 self.entry\_gianhap.delete(0, tk.END)  
 self.entry\_giaban.delete(0, tk.END)  
 self.entry\_tonkho.delete(0, tk.END)  
 self.cbb\_ncc.set("")

### **Quản lý hóa đơn**

**A screenshot of a computer

AI-generated content may be incorrect.**Quản lý toàn bộ hoạt động bán hàng của doanh nghiệp. Hỗ trợ ghi nhận giao dịch, tính tiền, phục vụ cho việc thống kê doanh thu, công nợ và quản lý hiệu quả kinh doanh.

Hình 14: Giao diện quản lý hóa đơn

**Chi tiết code dùng để quản lý hóa đơn:**

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
if BASE\_DIR not in sys.path:  
 sys.path.append(BASE\_DIR)  
  
import tkinter as tk  
from tkinter import ttk, messagebox  
from lienket import connect\_db  
import datetime  
  
  
class HoaDonFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
  
 self.kh\_map = {} # HoTen → MaKH  
 self.nv\_map = {} # HoTenNV → MaNV  
  
 self.load\_map\_kh\_nv()  
 self.create\_ui()  
 self.load\_hd()  
  
 # =====================================================  
 # VALIDATION   
 # =====================================================  
 def valid\_date(self, date\_text):  
 try:  
 datetime.datetime.strptime(date\_text, "%Y-%m-%d")  
 return True  
 except:  
 return False  
  
 def format\_name(self, text):  
 return " ".join(word.capitalize() for word in text.split())  
  
 # =====================================================  
 # LOAD MAP KH & NV  
 # =====================================================  
 def load\_map\_kh\_nv(self):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("SELECT MaKH, HoTen FROM khachhang ORDER BY HoTen")  
 self.kh\_map = {ten: ma for ma, ten in cur.fetchall()}  
  
 cur.execute("SELECT MaNV, HoTenNV FROM nhanvien ORDER BY HoTenNV")  
 self.nv\_map = {ten: ma for ma, ten in cur.fetchall()}  
  
 conn.close()  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # UI  
 # =====================================================  
 def create\_ui(self):  
 tk.Label(self, text="QUẢN LÝ HÓA ĐƠN",  
 font=("Arial", 18, "bold"), bg="white", fg="#003A75").pack(pady=10)  
  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1)  
  
 # Mã hóa đơn  
 self.entry\_mahd = self.\_make\_entry(frame\_info, "Mã hóa đơn:", 0, 0)  
  
 # KH Combobox  
 tk.Label(frame\_info, text="Khách hàng:", bg="white").grid(row=0, column=2, sticky="w")  
 self.cbb\_kh = ttk.Combobox(frame\_info, state="readonly", values=list(self.kh\_map.keys()))  
 self.cbb\_kh.grid(row=0, column=3, sticky="we", padx=5)  
  
 # NV Combobox  
 tk.Label(frame\_info, text="Nhân viên lập:", bg="white").grid(row=1, column=0, sticky="w")  
 self.cbb\_nv = ttk.Combobox(frame\_info, state="readonly", values=list(self.nv\_map.keys()))  
 self.cbb\_nv.grid(row=1, column=1, sticky="we", padx=5)  
  
 # Ngày bán  
 self.entry\_ngayban = self.\_make\_entry(frame\_info, "Ngày bán (yyyy-mm-dd):", 1, 2)  
  
 # Hình thức thanh toán  
 tk.Label(frame\_info, text="Hình thức TT:", bg="white").grid(row=2, column=0, sticky="w")  
 self.cbb\_hinhthuc = ttk.Combobox(frame\_info, state="readonly",  
 values=["Tiền mặt", "Chuyển khoản", "Quẹt thẻ"])  
 self.cbb\_hinhthuc.grid(row=2, column=1, sticky="we", padx=5)  
  
 # Ghi chú  
 self.entry\_ghichu = self.\_make\_entry(frame\_info, "Ghi chú:", 2, 2)  
  
 # =====================================================  
 # SEARCH  
 # =====================================================  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, pady=10, fill="x")  
  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(  
 frame\_search,  
 values=[" ", "MaHD", "HoTen", "HoTenNV", "NgayBan", "HinhThucTT"],  
 state="readonly"  
 )  
 self.cbb\_search\_type.set(" ")  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
  
 tk.Button(frame\_search, text="Tìm kiếm", bg="#003A75", fg="white",  
 width=15, command=self.search\_hd).grid(row=1, column=2, padx=5)  
  
 # =====================================================  
 # TABLE  
 # =====================================================  
 cols = ("MaHD", "HoTen", "HoTenNV", "NgayBan", "HinhThucTT", "GhiChu", "TongTien")  
 self.tree = ttk.Treeview(self, columns=cols, show="headings", height=12)  
  
 for col in cols:  
 self.tree.heading(col, text=col)  
 self.tree.column(col, width=150)  
  
 tk.Label(self, text="Danh sách hóa đơn",  
 bg="white", font=("Arial", 11, "bold")).pack(anchor="w", padx=20)  
  
 self.tree.pack(fill="both", padx=20, pady=10)  
  
 # =====================================================  
 # BUTTONS  
 # =====================================================  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_hd, \*\*btn).grid(row=0, column=0, padx=5)  
  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
  
 tk.Button(frame\_btn, text="Sửa", bg="#E74C3C",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
  
 tk.Button(frame\_btn, text="Hủy", bg="#E67E22",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=3, padx=5)  
  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_hd, \*\*btn).grid(row=0, column=4, padx=5)  
  
 def \_make\_entry(self, parent, text, row, col):  
 tk.Label(parent, text=text, bg="white").grid(row=row, column=col, sticky="w")  
 e = tk.Entry(parent)  
 e.grid(row=row, column=col+1, sticky="we", padx=5)  
 return e  
  
 # =====================================================  
 # LOAD TABLE  
 # =====================================================  
 def load\_hd(self):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 SELECT h.MaHD, k.HoTen, n.HoTenNV, h.NgayBan,  
 h.HinhThucTT, h.GhiChu, h.TongTien  
 FROM hoadon h  
 LEFT JOIN khachhang k ON h.MaKH = k.MaKH  
 LEFT JOIN nhanvien n ON h.MaNV = n.MaNV  
 """)  
  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
  
 for r in rows:  
 # Format tổng tiền  
 tong = "{:,}".format(int(r[6])).replace(",", ".")  
 new\_r = list(r)  
 new\_r[6] = tong  
 self.tree.insert("", "end", values=new\_r)  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # ADD  
 # =====================================================  
 def add\_hd(self):  
 mahd = self.entry\_mahd.get().strip()  
 tenkh = self.cbb\_kh.get()  
 tennv = self.cbb\_nv.get()  
 ngayban = self.entry\_ngayban.get().strip()  
 hinhthuc = self.cbb\_hinhthuc.get()  
  
 if not mahd or not tenkh or not tennv:  
 return messagebox.showwarning("Thiếu", "Vui lòng nhập đầy đủ thông tin!")  
  
 if tenkh not in self.kh\_map:  
 return messagebox.showerror("Lỗi", "Khách hàng không tồn tại!")  
  
 if tennv not in self.nv\_map:  
 return messagebox.showerror("Lỗi", "Nhân viên không tồn tại!")  
  
 if not self.valid\_date(ngayban):  
 return messagebox.showerror("Lỗi ngày", "Ngày bán phải dạng yyyy-mm-dd!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 INSERT INTO hoadon (MaHD, MaKH, MaNV, NgayBan, HinhThucTT, GhiChu, TongTien)  
 VALUES (%s,%s,%s,%s,%s,%s, 0)  
 """, (  
 mahd,  
 self.kh\_map[tenkh],  
 self.nv\_map[tennv],  
 ngayban,  
 hinhthuc,  
 self.entry\_ghichu.get().strip()  
 ))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_hd()  
 self.clear\_input()  
 messagebox.showinfo("OK", "Thêm thành công!")  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # SEARCH  
 # =====================================================  
 def search\_hd(self):  
 keyword = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 if field == "HoTen":  
 sql = """  
 SELECT h.MaHD, k.HoTen, n.HoTenNV, h.NgayBan, h.HinhThucTT, h.GhiChu, h.TongTien  
 FROM hoadon h  
 JOIN khachhang k ON h.MaKH = k.MaKH  
 JOIN nhanvien n ON h.MaNV = n.MaNV  
 WHERE k.HoTen LIKE %s  
 """  
 elif field == "HoTenNV":  
 sql = """  
 SELECT h.MaHD, k.HoTen, n.HoTenNV, h.NgayBan, h.HinhThucTT, h.GhiChu, h.TongTien  
 FROM hoadon h  
 JOIN khachhang k ON h.MaKH = k.MaKH  
 JOIN nhanvien n ON h.MaNV = n.MaNV  
 WHERE n.HoTenNV LIKE %s  
 """  
 else:  
 sql = f"""  
 SELECT h.MaHD, k.HoTen, n.HoTenNV, h.NgayBan, h.HinhThucTT, h.GhiChu, h.TongTien  
 FROM hoadon h  
 JOIN khachhang k ON h.MaKH = k.MaKH  
 JOIN nhanvien n ON h.MaNV = n.MaNV  
 WHERE h.{field} LIKE %s  
 """  
  
 cur.execute(sql, ('%' + keyword + '%',))  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
  
 for r in rows:  
 tong = "{:,}".format(int(r[6])).replace(",", ".")  
 new\_r = list(r)  
 new\_r[6] = tong  
 self.tree.insert("", "end", values=new\_r)  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # LOAD SELECTED  
 # =====================================================  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn", "Chọn hóa đơn để sửa")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_mahd.delete(0, tk.END)  
 self.entry\_mahd.insert(0, data[0])  
  
 self.cbb\_kh.set(data[1])  
 self.cbb\_nv.set(data[2])  
  
 self.entry\_ngayban.delete(0, tk.END)  
 self.entry\_ngayban.insert(0, data[3])  
  
 self.cbb\_hinhthuc.set(data[4])  
  
 self.entry\_ghichu.delete(0, tk.END)  
 self.entry\_ghichu.insert(0, data[5])  
  
 # =====================================================  
 # SAVE EDIT  
 # =====================================================  
 def save\_edit(self):  
 mahd = self.entry\_mahd.get().strip()  
 tenkh = self.cbb\_kh.get()  
 tennv = self.cbb\_nv.get()  
 ngayban = self.entry\_ngayban.get().strip()  
 hinhthuc = self.cbb\_hinhthuc.get()  
  
 if tenkh not in self.kh\_map:  
 return messagebox.showerror("Sai KH", "Khách hàng không tồn tại!")  
  
 if tennv not in self.nv\_map:  
 return messagebox.showerror("Sai NV", "Nhân viên không tồn tại!")  
  
 if not self.valid\_date(ngayban):  
 return messagebox.showerror("Sai ngày", "Ngày bán phải dạng yyyy-mm-dd!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 UPDATE hoadon SET  
 MaKH=%s, MaNV=%s, NgayBan=%s,  
 HinhThucTT=%s, GhiChu=%s  
 WHERE MaHD=%s  
 """, (  
 self.kh\_map[tenkh],  
 self.nv\_map[tennv],  
 ngayban,  
 hinhthuc,  
 self.entry\_ghichu.get().strip(),  
 mahd  
 ))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_hd()  
 self.clear\_input()  
 messagebox.showinfo("OK", "Sửa thành công!")  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # DELETE  
 # =====================================================  
 def delete\_hd(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn", "Hãy chọn hóa đơn để xóa!")  
  
 mahd = self.tree.item(sel[0])["values"][0]  
  
 if not messagebox.askyesno("Xác nhận", f"Bạn có chắc muốn xóa hóa đơn {mahd}?"):  
 return  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("DELETE FROM ct\_hoadon WHERE MaHD=%s", (mahd,))  
 cur.execute("DELETE FROM hoadon WHERE MaHD=%s", (mahd,))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_hd()  
 messagebox.showinfo("OK", "Đã xóa hóa đơn!")  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # =====================================================  
 # CLEAR  
 # =====================================================  
 def clear\_input(self):  
 self.entry\_mahd.delete(0, tk.END)  
 self.cbb\_kh.set("")  
 self.cbb\_nv.set("")  
 self.entry\_ngayban.delete(0, tk.END)  
 self.cbb\_hinhthuc.set("")  
 self.entry\_ghichu.delete(0, tk.END)

### **Quản lý ct\_hóa đơn**

A screenshot of a computer

AI-generated content may be incorrect.Quản lý danh sách sản phẩm đã bán trong từng hóa đơn. Giúp đảm bảo tính chính xác của giá bán, số lượng và tổng tiền, đồng thời hỗ trợ kiểm tra lại giao dịch khi cần đối chiếu.

Hình 15: Giao diện quản lý chi tiết hóa đơn

**Chi tiết code đùng để quản lý ct\_hóa đơn:**

class CTHoaDonFrame(tk.Frame):  
  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
  
 self.vl\_name\_to\_code = {}  
 self.list\_mahd = []  
 self.\_editing\_original = None  
  
 # load danh sách Mã hóa đơn và Vật liệu (TenVL->MaVL)  
 self.load\_maps()  
  
 self.create\_ui()  
 self.load\_table()  
  
 # ----------------------  
 # Load maps from DB  
 # ----------------------  
 def load\_maps(self):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 # danh sách hóa đơn (MaHD)  
 cur.execute("SELECT MaHD FROM hoadon ORDER BY MaHD")  
 self.list\_mahd = [r[0] for r in cur.fetchall()]  
  
 # tên vật liệu -> mã vật liệu  
 cur.execute("SELECT MaVL, TRIM(TenVL) FROM vatlieu ORDER BY TenVL")  
 mats = cur.fetchall()  
 conn.close()  
 self.vl\_name\_to\_code = {ten: ma for ma, ten in mats}  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
 self.list\_mahd = []  
 self.vl\_name\_to\_code = {}  
  
 # ----------------------  
 # UI  
 # ----------------------  
 def create\_ui(self):  
 tk.Label(self, text="CHI TIẾT HÓA ĐƠN",  
 font=("Arial", 18, "bold"),  
 bg="white", fg="#003A75").pack(pady=10)  
  
 # ------------------- FORM INPUT -------------------  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, pady=10, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1)  
  
 # ---- Row 1: MaHD (combobox) and TenVL (combobox) ----  
 tk.Label(frame\_info, text="Mã hóa đơn:", bg="white").grid(row=0, column=0, sticky="w", pady=5)  
 self.cbb\_mahd = ttk.Combobox(frame\_info, state="readonly", values=self.list\_mahd)  
 self.cbb\_mahd.grid(row=0, column=1, sticky="we", padx=5)  
 # allow refresh of mahd list from DB  
 tk.Button(frame\_info, text="↻", width=3, command=self.refresh\_mahd\_list).grid(row=0, column=2, sticky="w")  
  
 tk.Label(frame\_info, text="Tên vật liệu:", bg="white").grid(row=0, column=2, sticky="w", pady=5)  
 self.cbb\_tenvl = ttk.Combobox(frame\_info, state="readonly", values=list(self.vl\_name\_to\_code.keys()))  
 self.cbb\_tenvl.grid(row=0, column=3, sticky="we", padx=5)  
 self.cbb\_tenvl.bind("<<ComboboxSelected>>", lambda e: self.on\_tenvl\_selected())  
  
 # ---- Row 2: SoLuong & DonGia ----  
 tk.Label(frame\_info, text="Số lượng:", bg="white").grid(row=1, column=0, sticky="w", pady=5)  
 self.entry\_soluong = tk.Entry(frame\_info)  
 self.entry\_soluong.grid(row=1, column=1, sticky="we", padx=5)  
 self.entry\_soluong.bind("<KeyRelease>", lambda e: self.compute\_thanhtien())  
  
 tk.Label(frame\_info, text="Đơn giá:", bg="white").grid(row=1, column=2, sticky="w", pady=5)  
 self.entry\_dongia = tk.Entry(frame\_info)  
 self.entry\_dongia.grid(row=1, column=3, sticky="we", padx=5)  
 self.entry\_dongia.bind("<KeyRelease>", lambda e: self.compute\_thanhtien())  
  
 # ---- Row 3: ThanhTien (readonly) ----  
 tk.Label(frame\_info, text="Thành tiền:", bg="white").grid(row=2, column=0, sticky="w", pady=5)  
 self.var\_thanhtien = tk.StringVar(value="0")  
 tk.Entry(frame\_info, textvariable=self.var\_thanhtien, state="readonly").grid(row=2, column=1, sticky="we", padx=5)  
  
 # ------------------- SEARCH -------------------  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, pady=10, fill="x")  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(frame\_search,  
 values=[" ", "MaHD", "TenVL", "SoLuong"], state="readonly")  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
 self.cbb\_search\_type.set(" ")  
  
 tk.Button(frame\_search, text="Tìm", width=12,  
 bg="#003A75", fg="white",  
 command=self.search).grid(row=1, column=2, padx=5)  
  
 # ------------------- TABLE -------------------  
 tk.Label(self, text="Danh sách chi tiết hóa đơn",  
 bg="white", font=("Arial", 11, "bold")).pack(anchor="w", padx=20)  
  
 cols = ("MaHD", "TenVL", "SoLuong", "DonGia", "ThanhTien", "MaVL")  
 self.tree = ttk.Treeview(self, columns=cols, show="headings", height=12)  
  
 for c in cols:  
 self.tree.heading(c, text=c)  
 self.tree.column(c, anchor="center", width=140)  
  
 self.tree.pack(padx=10, pady=6, fill="both")  
 self.tree.bind("<Double-1>", lambda e: self.load\_selected\_edit())  
  
 # ------------------- BUTTONS -------------------  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB",  
 command=self.add\_ct, \*\*btn).grid(row=0, column=0, padx=5)  
 tk.Button(frame\_btn, text="Sửa", bg="#E67E22",  
 command=self.load\_selected\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
 tk.Button(frame\_btn, text="Xóa", bg="#800080",  
 command=self.delete\_ct, \*\*btn).grid(row=0, column=3, padx=5)  
 tk.Button(frame\_btn, text="Hủy", bg="#E74C3C",  
 command=self.clear\_inputs, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ----------------------  
 # helper format: format money for display 1.000.000  
 # ----------------------  
 def format\_money\_display(self, value):  
 try:  
 # allow floats or ints; display without decimals if whole  
 if value is None:  
 return "0"  
 # if it's a decimal.Decimal or str, convert to int if no fraction  
 v = float(value)  
 if v.is\_integer():  
 return "{:,}".format(int(v)).replace(",", ".")  
 else:  
 # show with 2 decimals, replace comma with dot thousands  
 s = "{:,.2f}".format(v)  
 return s.replace(",", ".")  
 except:  
 return str(value)  
  
 # ----------------------  
 # compute thanhtien and show formatted  
 # ----------------------  
 def compute\_thanhtien(self):  
 try:  
 sl\_text = self.entry\_soluong.get().strip() or "0"  
 dg\_text = self.entry\_dongia.get().strip() or "0"  
  
 # accept numbers with thousand separators either '.' or ','  
 sl\_clean = sl\_text.replace(".", "").replace(",", "")  
 dg\_clean = dg\_text.replace(".", "").replace(",", "")  
  
 sl = decimal.Decimal(sl\_clean)  
 dg = decimal.Decimal(dg\_clean)  
 tt = sl \* dg  
 # display formatted  
 self.var\_thanhtien.set(self.format\_money\_display(tt))  
 except:  
 self.var\_thanhtien.set("0")  
  
 # ----------------------  
 # load table  
 # ----------------------  
 def load\_table(self):  
 self.reload\_maps\_and\_comboboxes()  
 self.tree.delete(\*self.tree.get\_children())  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 SELECT c.MaHD, v.TenVL, c.SoLuong, c.DonGia, c.ThanhTien, c.MaVL  
 FROM ct\_hoadon c  
 LEFT JOIN vatlieu v ON c.MaVL = v.MaVL  
 ORDER BY c.MaHD  
 """)  
 rows = cur.fetchall()  
 conn.close()  
  
 for r in rows:  
 r = list(r)  
 # format DonGia and ThanhTien for display  
 try:  
 r[3] = self.format\_money\_display(r[3])  
 except:  
 pass  
 try:  
 r[4] = self.format\_money\_display(r[4])  
 except:  
 pass  
 self.tree.insert("", "end", values=r)  
  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # ----------------------  
 # reload maps & combobox values  
 # ----------------------  
 def reload\_maps\_and\_comboboxes(self):  
 self.load\_maps()  
 # refresh combobox lists  
 self.cbb\_mahd['values'] = self.list\_mahd  
 self.cbb\_tenvl['values'] = list(self.vl\_name\_to\_code.keys())  
  
 def refresh\_mahd\_list(self):  
 self.reload\_maps\_and\_comboboxes()  
 messagebox.showinfo("OK", "Đã refresh danh sách Mã hóa đơn và Vật liệu")  
  
 def on\_tenvl\_selected(self):  
 # when a TenVL selected, optionally you could auto-fill DonGia from vatlieu table  
 # but to keep "thêm không sửa" spirit, we just ensure mapping exists  
 sel = self.cbb\_tenvl.get()  
 if sel and sel not in self.vl\_name\_to\_code:  
 messagebox.showwarning("Lỗi", "Tên vật liệu không tồn tại trong DB!")  
  
 # ----------------------  
 # ADD  
 # ----------------------  
 def add\_ct(self):  
 mahd = self.cbb\_mahd.get().strip()  
 tenvl = self.cbb\_tenvl.get().strip()  
  
 if not mahd:  
 return messagebox.showwarning("Thiếu dữ liệu", "Chưa chọn Mã hóa đơn!")  
 if not tenvl:  
 return messagebox.showwarning("Thiếu dữ liệu", "Chưa chọn Tên vật liệu!")  
  
 if tenvl not in self.vl\_name\_to\_code:  
 return messagebox.showerror("Lỗi", "Tên vật liệu không tồn tại!")  
  
 mavl = self.vl\_name\_to\_code[tenvl]  
  
 # parse numbers, accept thousand separators  
 try:  
 sl\_text = self.entry\_soluong.get().strip()  
 dg\_text = self.entry\_dongia.get().strip()  
 sl = decimal.Decimal(sl\_text.replace(".", "").replace(",", ""))  
 dg = decimal.Decimal(dg\_text.replace(".", "").replace(",", ""))  
 except Exception:  
 return messagebox.showwarning("Sai dữ liệu", "Số lượng hoặc đơn giá không hợp lệ!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 # insert  
 cur.execute("""  
 INSERT INTO ct\_hoadon (MaHD, MaVL, SoLuong, DonGia)  
 VALUES (%s, %s, %s, %s)  
 """, (mahd, mavl, float(sl), float(dg)))  
  
 # update tong tien (IFNULL / COALESCE for safety)  
 cur.execute("""  
 UPDATE hoadon  
 SET TongTien = (SELECT IFNULL(SUM(ThanhTien),0) FROM ct\_hoadon WHERE MaHD=%s)  
 WHERE MaHD=%s  
 """, (mahd, mahd))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_table()  
 self.clear\_inputs()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
  
 # ----------------------  
 # load selected row into form (for edit)  
 # ----------------------  
 def load\_selected\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chọn dòng!", "Hãy chọn chi tiết hóa đơn để sửa!")  
  
 item = sel[0]  
 mahd, tenvl, sl, dg, tt, mavl = self.tree.item(item)["values"]  
  
 # set comboboxes / entries  
 self.cbb\_mahd.set(mahd)  
 self.cbb\_tenvl.set(tenvl)  
  
 self.entry\_soluong.delete(0, tk.END)  
 self.entry\_soluong.insert(0, str(sl))  
  
 # DonGia may be displayed formatted in grid; convert to plain numeric for edit if needed  
 try:  
 dg\_plain = str(dg).replace(".", "").replace(",", "")  
 self.entry\_dongia.delete(0, tk.END)  
 self.entry\_dongia.insert(0, dg\_plain)  
 except:  
 self.entry\_dongia.delete(0, tk.END)  
 self.entry\_dongia.insert(0, str(dg))  
  
 # show ThanhTien formatted  
 self.var\_thanhtien.set(self.format\_money\_display(tt))  
  
 self.\_editing\_original = {"MaHD": mahd, "MaVL": mavl}  
  
 # ----------------------  
 # save edit  
 # ----------------------  
 def save\_edit(self):  
 if not self.\_editing\_original:  
 return messagebox.showwarning("Lỗi", "Bạn chưa chọn dòng để sửa!")  
  
 mahd\_old = self.\_editing\_original["MaHD"]  
 mavl\_old = self.\_editing\_original["MaVL"]  
  
 mahd\_new = self.cbb\_mahd.get().strip()  
 tenvl = self.cbb\_tenvl.get().strip()  
  
 if not mahd\_new:  
 return messagebox.showwarning("Thiếu dữ liệu", "Chưa chọn Mã hóa đơn!")  
  
 if tenvl not in self.vl\_name\_to\_code:  
 return messagebox.showerror("Lỗi", "Tên vật liệu không tồn tại!")  
  
 mavl\_new = self.vl\_name\_to\_code[tenvl]  
  
 try:  
 sl\_text = self.entry\_soluong.get().strip()  
 dg\_text = self.entry\_dongia.get().strip()  
 sl = decimal.Decimal(sl\_text.replace(".", "").replace(",", ""))  
 dg = decimal.Decimal(dg\_text.replace(".", "").replace(",", ""))  
 except:  
 return messagebox.showwarning("Sai", "Số lượng / giá không hợp lệ!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 UPDATE ct\_hoadon  
 SET MaHD=%s, MaVL=%s, SoLuong=%s, DonGia=%s  
 WHERE MaHD=%s AND MaVL=%s  
 """, (mahd\_new, mavl\_new, float(sl), float(dg), mahd\_old, mavl\_old))  
  
 cur.execute("""  
 UPDATE hoadon  
 SET TongTien = (SELECT IFNULL(SUM(ThanhTien),0) FROM ct\_hoadon WHERE MaHD=%s)  
 WHERE MaHD=%s  
 """, (mahd\_new, mahd\_new))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_table()  
 self.clear\_inputs()  
 self.\_editing\_original = None  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
  
 # ----------------------  
 # delete  
 # ----------------------  
 def delete\_ct(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn!", "Chọn dòng để xóa!")  
  
 item = sel[0]  
 mahd, tenvl, sl, dg, tt, mavl = self.tree.item(item)["values"]  
  
 if not messagebox.askyesno("Xác nhận", f"Xóa vật liệu '{tenvl}' khỏi hóa đơn {mahd}?"):  
 return  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("DELETE FROM ct\_hoadon WHERE MaHD=%s AND MaVL=%s", (mahd, mavl))  
  
 cur.execute("""  
 UPDATE hoadon  
 SET TongTien = (  
 SELECT IFNULL(SUM(ThanhTien), 0)  
 FROM ct\_hoadon  
 WHERE MaHD=%s  
 )  
 WHERE MaHD=%s  
 """, (mahd, mahd))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_table()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
  
 # ----------------------  
 # search  
 # ----------------------  
 def search(self):  
 key = self.entry\_search.get().strip()  
 if not key:  
 return self.load\_table()  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 SELECT c.MaHD, v.TenVL, c.SoLuong, c.DonGia, c.ThanhTien, c.MaVL  
 FROM ct\_hoadon c  
 LEFT JOIN vatlieu v ON c.MaVL=v.MaVL  
 WHERE c.MaHD LIKE %s OR v.TenVL LIKE %s  
 """, ("%" + key + "%", "%" + key + "%"))  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 r = list(r)  
 # format money cols  
 try:  
 r[3] = self.format\_money\_display(r[3])  
 except:  
 pass  
 try:  
 r[4] = self.format\_money\_display(r[4])  
 except:  
 pass  
 self.tree.insert("", "end", values=r)  
 except Exception as e:  
 messagebox.showerror("Lỗi DB", str(e))  
  
 # ----------------------  
 # clear inputs  
 # ----------------------  
 def clear\_inputs(self):  
 self.cbb\_mahd.set("")  
 self.cbb\_tenvl.set("")  
 self.entry\_soluong.delete(0, tk.END)  
 self.entry\_dongia.delete(0, tk.END)  
 self.var\_thanhtien.set("0")  
 self.\_editing\_original = None

### **Quản lý phiếu nhập**

Theo dõi các lần nhập hàng từ nhà cung cấp. Giúp doanh nghiệp kiểm soát lượng hàng vào kho, chi phí nhập hàng và phục vụ cho việc quản lý tồn kho, tính giá vốn.

**A screenshot of a computer

AI-generated content may be incorrect.**

Hình 16: Giao diện quản lý phiếu nhập

**Chi tiết code dùng để quản lý phiếu nhập:**

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
if BASE\_DIR not in sys.path:  
 sys.path.append(BASE\_DIR)  
  
import tkinter as tk  
from tkinter import ttk, messagebox  
from datetime import date  
from lienket import connect\_db  
  
  
class PhieuNhapFrame(tk.Frame):  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
  
 self.ncc\_map = {}  
 self.nv\_map = {}  
  
 self.load\_maps()  
 self.create\_ui()  
 self.load\_phieunhap()  
  
 # ============================================================  
 # LOAD MAP NCC + NV  
 # ============================================================  
 def load\_maps(self):  
 self.ncc\_map.clear()  
 self.nv\_map.clear()  
  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("SELECT MaNCC, TenNCC FROM nhacungcap ORDER BY TenNCC")  
 for ma, ten in cur.fetchall():  
 self.ncc\_map[ten] = ma  
  
 cur.execute("SELECT MaNV, HoTenNV FROM nhanvien ORDER BY HoTenNV")  
 for ma, ten in cur.fetchall():  
 self.nv\_map[ten] = ma  
  
 conn.close()  
  
 # ============================================================  
 # UI  
 # ============================================================  
 def create\_ui(self):  
 tk.Label(self, text="QUẢN LÝ PHIẾU NHẬP",  
 font=("Arial", 18, "bold"), bg="white", fg="#003A75").pack(pady=10)  
  
 frame = tk.Frame(self, bg="white")  
 frame.pack(fill="x", padx=20)  
  
 for i in range(4):  
 frame.columnconfigure(i, weight=1)  
  
 # Mã PN  
 tk.Label(frame, text="Mã phiếu nhập:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_mapn = tk.Entry(frame)  
 self.entry\_mapn.grid(row=0, column=1, padx=5, pady=5, sticky="we")  
  
 # Ngày nhập  
 tk.Label(frame, text="Ngày nhập:", bg="white").grid(row=0, column=2, sticky="w")  
 self.entry\_ngay = tk.Entry(frame)  
 self.entry\_ngay.insert(0, date.today().isoformat())  
 self.entry\_ngay.grid(row=0, column=3, padx=5, pady=5, sticky="we")  
  
 # Nhà cung cấp (COMBOBOX)  
 tk.Label(frame, text="Nhà cung cấp:", bg="white").grid(row=1, column=0, sticky="w")  
 self.cbb\_ncc = ttk.Combobox(frame, state="readonly",  
 values=list(self.ncc\_map.keys()))  
 self.cbb\_ncc.grid(row=1, column=1, padx=5, pady=5, sticky="we")  
  
 # Nhân viên (COMBOBOX)  
 tk.Label(frame, text="Nhân viên:", bg="white").grid(row=1, column=2, sticky="w")  
 self.cbb\_nv = ttk.Combobox(frame, state="readonly",  
 values=list(self.nv\_map.keys()))  
 self.cbb\_nv.grid(row=1, column=3, padx=5, pady=5, sticky="we")  
  
 # Ghi chú  
 tk.Label(frame, text="Ghi chú:", bg="white").grid(row=2, column=0, sticky="w")  
 self.entry\_ghichu = tk.Entry(frame)  
 self.entry\_ghichu.grid(row=2, column=1, padx=5, pady=5, sticky="we")  
  
 # Tổng tiền  
 tk.Label(frame, text="Tổng tiền:", bg="white").grid(row=2, column=2, sticky="w")  
 self.entry\_tongtien = tk.Entry(frame, state="readonly")  
 self.entry\_tongtien.grid(row=2, column=3, padx=5, pady=5, sticky="we")  
 self.entry\_tongtien.insert(0, "0")  
  
 # ============================================================  
 # SEARCH  
 # ============================================================  
 search\_frame = tk.Frame(self, bg="white")  
 search\_frame.pack(fill="x", padx=20, pady=10)  
  
 search\_frame.columnconfigure(0, weight=3)  
  
 tk.Label(search\_frame, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(search\_frame)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(  
 search\_frame,  
 values=[" ", "MaPN", "TenNCC", "TenNV"],  
 state="readonly"  
 )  
 self.cbb\_search\_type.set(" ")  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
  
 tk.Button(search\_frame, text="Tìm", bg="#003A75", fg="white",  
 command=self.search\_phieunhap).grid(row=1, column=2, padx=5)  
  
 # ============================================================  
 # TABLE  
 # ============================================================  
 cols = ("MaPN", "NgayNhap", "TenNCC", "TenNV", "TongTien", "GhiChu")  
 self.tree = ttk.Treeview(self, columns=cols, show="headings", height=12)  
  
 for c in cols:  
 self.tree.heading(c, text=c)  
 self.tree.column(c, width=160)  
  
 self.tree.pack(fill="both", padx=20, pady=10)  
  
 # ============================================================  
 # BUTTONS  
 # ============================================================  
 btn\_frame = tk.Frame(self, bg="white")  
 btn\_frame.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
  
 tk.Button(btn\_frame, text="Thêm", bg="#3498DB",  
 command=self.add\_phieunhap, \*\*btn).grid(row=0, column=0, padx=5)  
 tk.Button(btn\_frame, text="Sửa", bg="#E67E22",  
 command=self.load\_selected\_to\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
 tk.Button(btn\_frame, text="Lưu", bg="#27AE60",  
 command=self.save\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
 tk.Button(btn\_frame, text="Xóa", bg="#C0392B",  
 command=self.delete\_phieunhap, \*\*btn).grid(row=0, column=3, padx=5)  
 tk.Button(btn\_frame, text="Hủy", bg="#7D3C98",  
 command=self.clear\_input, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ============================================================  
 # FETCH  
 # ============================================================  
 def fetch\_all\_phieunhap(self):  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 SELECT   
 p.MaPN,  
 p.NgayNhap,  
 ncc.TenNCC,  
 nv.HoTenNV,  
 p.TongTien,  
 p.GhiChu  
 FROM phieunhap p  
 JOIN nhacungcap ncc ON p.MaNCC = ncc.MaNCC  
 JOIN nhanvien nv ON p.MaNV = nv.MaNV  
 ORDER BY p.MaPN  
 """)  
  
 rows = cur.fetchall()  
 conn.close()  
 return rows  
  
 # ============================================================  
 # LOAD TABLE  
 # ============================================================  
 def load\_phieunhap(self):  
 self.tree.delete(\*self.tree.get\_children())  
 for row in self.fetch\_all\_phieunhap():  
 self.tree.insert("", "end", values=row)  
  
 # ============================================================  
 # ADD  
 # ============================================================  
 def add\_phieunhap(self):  
 mapn = self.entry\_mapn.get().strip()  
 ngay = self.entry\_ngay.get().strip()  
 tenncc = self.cbb\_ncc.get().strip()  
 tennv = self.cbb\_nv.get().strip()  
 ghichu = self.entry\_ghichu.get().strip()  
  
 if not all([mapn, ngay, tenncc, tennv]):  
 return messagebox.showwarning("Thiếu dữ liệu", "Hãy nhập đủ thông tin!")  
  
 if tenncc not in self.ncc\_map:  
 return messagebox.showerror("Lỗi", "Tên nhà cung cấp không tồn tại!")  
  
 if tennv not in self.nv\_map:  
 return messagebox.showerror("Lỗi", "Tên nhân viên không tồn tại!")  
  
 mancc = self.ncc\_map[tenncc]  
 manv = self.nv\_map[tennv]  
  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 INSERT INTO phieunhap (MaPN, NgayNhap, MaNCC, MaNV, TongTien, GhiChu)  
 VALUES (%s, %s, %s, %s, 0, %s)  
 """, (mapn, ngay, mancc, manv, ghichu))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_phieunhap()  
 self.clear\_input()  
  
 # ============================================================  
 # DELETE  
 # ============================================================  
 def delete\_phieunhap(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn", "Hãy chọn phiếu nhập để xóa")  
  
 item = sel[0]  
 mapn = self.tree.item(item)["values"][0]  
  
 if not messagebox.askyesno("Xác nhận", f"Bạn có chắc muốn xóa phiếu nhập {mapn}?"):  
 return  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("DELETE FROM ct\_phieunhap WHERE MaPN=%s", (mapn,))  
 cur.execute("DELETE FROM phieunhap WHERE MaPN=%s", (mapn,))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_phieunhap()  
 messagebox.showinfo("OK", f"Đã xóa phiếu nhập {mapn}")  
  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
  
 # ============================================================  
 # LOAD SELECTED  
 # ============================================================  
 def load\_selected\_to\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn", "Hãy chọn dòng để sửa")  
  
 data = self.tree.item(sel)["values"]  
  
 self.entry\_mapn.delete(0, tk.END)  
 self.entry\_mapn.insert(0, data[0])  
  
 self.entry\_ngay.delete(0, tk.END)  
 self.entry\_ngay.insert(0, data[1])  
  
 self.cbb\_ncc.set(data[2])  
 self.cbb\_nv.set(data[3])  
  
 self.entry\_tongtien.config(state="normal")  
 self.entry\_tongtien.delete(0, tk.END)  
 self.entry\_tongtien.insert(0, data[4])  
 self.entry\_tongtien.config(state="readonly")  
  
 self.entry\_ghichu.delete(0, tk.END)  
 self.entry\_ghichu.insert(0, data[5])  
  
 # ============================================================  
 # SAVE EDIT  
 # ============================================================  
 def save\_edit(self):  
 mapn = self.entry\_mapn.get().strip()  
 ngay = self.entry\_ngay.get().strip()  
 tenncc = self.cbb\_ncc.get().strip()  
 tennv = self.cbb\_nv.get().strip()  
 ghichu = self.entry\_ghichu.get().strip()  
  
 if tenncc not in self.ncc\_map:  
 return messagebox.showerror("Lỗi", "Tên nhà cung cấp không tồn tại!")  
  
 if tennv not in self.nv\_map:  
 return messagebox.showerror("Lỗi", "Tên nhân viên không tồn tại!")  
  
 mancc = self.ncc\_map[tenncc]  
 manv = self.nv\_map[tennv]  
  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 cur.execute("""  
 UPDATE phieunhap  
 SET NgayNhap=%s, MaNCC=%s, MaNV=%s, GhiChu=%s  
 WHERE MaPN=%s  
 """, (ngay, mancc, manv, ghichu, mapn))  
  
 conn.commit()  
 conn.close()  
  
 self.load\_phieunhap()  
 self.clear\_input()  
  
 # ============================================================  
 # SEARCH  
 # ============================================================  
 def search\_phieunhap(self):  
 kw = self.entry\_search.get().strip()  
 field = self.cbb\_search\_type.get()  
  
 if kw == "":  
 return self.load\_phieunhap()  
  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 if field == "TenNCC":  
 sql = """  
 SELECT p.MaPN, p.NgayNhap, n.TenNCC, nv.HoTenNV, p.TongTien, p.GhiChu  
 FROM phieunhap p  
 JOIN nhacungcap n ON p.MaNCC = n.MaNCC  
 JOIN nhanvien nv ON p.MaNV = nv.MaNV  
 WHERE n.TenNCC LIKE %s  
 """  
 elif field == "TenNV":  
 sql = """  
 SELECT p.MaPN, p.NgayNhap, n.TenNCC, nv.HoTenNV, p.TongTien, p.GhiChu  
 FROM phieunhap p  
 JOIN nhacungcap n ON p.MaNCC = n.MaNCC  
 JOIN nhanvien nv ON p.MaNV = nv.MaNV  
 WHERE nv.HoTenNV LIKE %s  
 """  
 else:  
 sql = """  
 SELECT p.MaPN, p.NgayNhap, n.TenNCC, nv.HoTenNV, p.TongTien, p.GhiChu  
 FROM phieunhap p  
 JOIN nhacungcap n ON p.MaNCC = n.MaNCC  
 JOIN nhanvien nv ON p.MaNV = nv.MaNV  
 WHERE p.MaPN LIKE %s  
 """  
  
 cur.execute(sql, ('%' + kw + '%',))  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 self.tree.insert("", "end", values=r)  
  
 # ============================================================  
 # CLEAR INPUT  
 # ============================================================  
 def clear\_input(self):  
 self.entry\_mapn.delete(0, tk.END)  
 self.entry\_ngay.delete(0, tk.END)  
 self.entry\_ngay.insert(0, date.today().isoformat())  
 self.cbb\_ncc.set("")  
 self.cbb\_nv.set("")  
 self.entry\_ghichu.delete(0, tk.END)  
 self.entry\_tongtien.config(state="normal")  
 self.entry\_tongtien.delete(0, tk.END)  
 self.entry\_tongtien.insert(0, "0")  
 self.entry\_tongtien.config(state="readonly")

### **Quản lý ct\_phiếu nhập**

**A screenshot of a computer

AI-generated content may be incorrect.**Quản lý danh sách vật liệu/sản phẩm được nhập trong từng phiếu nhập. Đảm bảo hệ thống cập nhật chính xác số lượng tồn kho, giá nhập và giúp kiểm soát chất lượng thông tin nhập hàng.

Hình 17: Giao diện quản lý chi tiết phiếu nhập

**Chi tiết code dùng để quản lý ct\_phiếu nhập:**

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
if BASE\_DIR not in sys.path:  
 sys.path.append(BASE\_DIR)  
  
  
class CTPhieuNhapFrame(tk.Frame):  
  
 def \_\_init\_\_(self, parent, show\_main\_menu=None):  
 super().\_\_init\_\_(parent, bg="white")  
 self.show\_main\_menu = show\_main\_menu  
  
 # mapping từ Tên -> Mã (vật liệu)  
 self.\_name\_to\_code = {}  
 # MaPN list  
 self.\_mapn\_list = []  
 self.\_editing\_original = None  
  
 self.create\_ui()  
 self.load\_material\_map()  
 self.load\_mapn\_list()  
 self.load\_table()  
  
 # ======================= HELPERS =======================  
 def \_format\_thousands(self, value):  
 *"""Chuyển số (int/Decimal/str chứa số) thành chuỗi hiển thị 1.000"""* try:  
 # xử lý chuỗi có dấu . hoặc , trước  
 s = str(value)  
 s = s.replace(".", "").replace(",", "")  
 iv = int(decimal.Decimal(s))  
 return "{:,}".format(iv).replace(",", ".")  
 except Exception:  
 return str(value)  
  
 def \_parse\_number(self, s):  
 *"""Loại bỏ dấu phân cách và trả về Decimal. Nếu không parse được ném exception."""* if s is None:  
 return decimal.Decimal(0)  
 txt = str(s).replace(".", "").replace(",", "").strip()  
 if txt == "":  
 return decimal.Decimal(0)  
 return decimal.Decimal(txt)  
  
 def \_format\_entry\_thousands(self, entry\_widget):  
 *"""Dùng để bind KeyRelease: giữ lại các chữ số, định dạng hiển thị với dấu '.'"""* txt = entry\_widget.get()  
 # cho phép chỉ số nguyên (loại bỏ ký tự không phải số)  
 filtered = "".join(ch for ch in txt if ch.isdigit())  
 if filtered == "":  
 entry\_widget.delete(0, tk.END)  
 return  
 try:  
 iv = int(filtered)  
 formatted = "{:,}".format(iv).replace(",", ".")  
 # set mới  
 entry\_widget.delete(0, tk.END)  
 entry\_widget.insert(0, formatted)  
 # đặt con trỏ cuối  
 entry\_widget.icursor(tk.END)  
 except Exception:  
 # nếu lỗi giữ nguyên  
 pass  
  
 # ============================================================  
 # UI  
 # ============================================================  
 def create\_ui(self):  
 tk.Label(self, text="CHI TIẾT PHIẾU NHẬP",  
 font=("Arial", 18, "bold"), bg="white", fg="#003A75"  
 ).pack(pady=10)  
  
 frame\_info = tk.Frame(self, bg="white")  
 frame\_info.pack(padx=20, fill="x")  
  
 for i in range(4):  
 frame\_info.columnconfigure(i, weight=1)  
  
 # MaPN: COMBO (không cho nhập tay)  
 tk.Label(frame\_info, text="Mã Phiếu Nhập:", bg="white").grid(row=0, column=0, sticky="w")  
 self.cbb\_mapn = ttk.Combobox(frame\_info, state="readonly")  
 self.cbb\_mapn.grid(row=0, column=1, sticky="we", padx=5)  
  
 # TenVL: COMBO (không cho nhập tay)  
 tk.Label(frame\_info, text="Tên vật liệu:", bg="white").grid(row=0, column=2, sticky="w")  
 self.cbb\_tenvl = ttk.Combobox(frame\_info, state="readonly")  
 self.cbb\_tenvl.grid(row=0, column=3, sticky="we", padx=5)  
 self.cbb\_tenvl.bind("<<ComboboxSelected>>", lambda e: self.on\_tenvl\_selected())  
  
 # MaVL read-only (hiển thị)  
 tk.Label(frame\_info, text="Mã VL:", bg="white").grid(row=1, column=0, sticky="w")  
 self.entry\_mavl = tk.Entry(frame\_info, state="readonly")  
 self.entry\_mavl.grid(row=1, column=1, sticky="we", padx=5)  
  
 tk.Label(frame\_info, text="Số lượng:", bg="white").grid(row=1, column=2, sticky="w")  
 self.entry\_soluong = tk.Entry(frame\_info)  
 self.entry\_soluong.grid(row=1, column=3, sticky="we", padx=5)  
 # format khi gõ  
 self.entry\_soluong.bind("<KeyRelease>", lambda e: (self.\_format\_entry\_thousands(self.entry\_soluong), self.compute\_total()))  
  
 tk.Label(frame\_info, text="Đơn giá:", bg="white").grid(row=2, column=0, sticky="w")  
 self.entry\_dongia = tk.Entry(frame\_info)  
 self.entry\_dongia.grid(row=2, column=1, sticky="we", padx=5)  
 self.entry\_dongia.bind("<KeyRelease>", lambda e: (self.\_format\_entry\_thousands(self.entry\_dongia), self.compute\_total()))  
  
 tk.Label(frame\_info, text="Thành tiền:", bg="white").grid(row=2, column=2, sticky="w")  
 self.var\_thanhtien = tk.StringVar(value="0")  
 tk.Entry(frame\_info, textvariable=self.var\_thanhtien, state="readonly").grid(  
 row=2, column=3, sticky="we", padx=5  
 )  
  
 # SEARCH  
 frame\_search = tk.Frame(self, bg="white")  
 frame\_search.pack(padx=20, fill="x", pady=10)  
 frame\_search.columnconfigure(0, weight=3)  
  
 tk.Label(frame\_search, text="Tìm kiếm:", bg="white").grid(row=0, column=0, sticky="w")  
 self.entry\_search = tk.Entry(frame\_search)  
 self.entry\_search.grid(row=1, column=0, sticky="we", padx=10)  
  
 self.cbb\_search\_type = ttk.Combobox(frame\_search,  
 values=[" ", "MaPN", "TenVL", "SoLuongNhap", "DonGiaNhap"], state="readonly")  
 self.cbb\_search\_type.set(" ")  
 self.cbb\_search\_type.grid(row=1, column=1, padx=5)  
  
 tk.Button(frame\_search, text="Tìm", bg="#003A75", fg="white", width=12,  
 command=self.search).grid(row=1, column=2, padx=5)  
  
 # TABLE  
 cols = ("MaPN", "TenVL", "SoLuongNhap", "DonGiaNhap", "ThanhTien", "MaVL")  
 self.tree = ttk.Treeview(self, columns=cols, show="headings", height=13)  
 for c in cols:  
 self.tree.heading(c, text=c)  
 self.tree.column(c, width=150)  
 self.tree.pack(fill="both", padx=20, pady=10)  
 self.tree.bind("<Double-1>", lambda e: self.load\_for\_edit())  
  
 # BUTTONS  
 frame\_btn = tk.Frame(self, bg="white")  
 frame\_btn.pack(pady=10)  
  
 btn = dict(width=15, fg="white", font=("Arial", 11, "bold"))  
 tk.Button(frame\_btn, text="Thêm", bg="#3498DB", command=self.add\_ct, \*\*btn).grid(row=0, column=0, padx=5)  
 tk.Button(frame\_btn, text="Sửa", bg="#E67E22", command=self.load\_for\_edit, \*\*btn).grid(row=0, column=1, padx=5)  
 tk.Button(frame\_btn, text="Lưu", bg="#27AE60", command=self.save\_edit, \*\*btn).grid(row=0, column=2, padx=5)  
 tk.Button(frame\_btn, text="Xóa", bg="#C0392B", command=self.delete\_ct, \*\*btn).grid(row=0, column=3, padx=5)  
 tk.Button(frame\_btn, text="Hủy", bg="#7D3C98", command=self.clear\_inputs, \*\*btn).grid(row=0, column=4, padx=5)  
  
 # ============================================================  
 def on\_tenvl\_selected(self):  
 *"""Khi chọn tên vật liệu, set MaVL lên entry\_mavl"""* name = self.cbb\_tenvl.get()  
 mavl = self.\_name\_to\_code.get(name, "")  
 # set entry\_mavl (readonly)  
 self.entry\_mavl.config(state="normal")  
 self.entry\_mavl.delete(0, tk.END)  
 self.entry\_mavl.insert(0, mavl)  
 self.entry\_mavl.config(state="readonly")  
  
 # ============================================================  
 def compute\_total(self):  
 *"""Tính Thành tiền theo Số lượng và Đơn giá; hiển thị đã format"""* try:  
 sl = self.\_parse\_number(self.entry\_soluong.get())  
 dg = self.\_parse\_number(self.entry\_dongia.get())  
 tt = sl \* dg  
 # hiển thị với dấu chấm  
 self.var\_thanhtien.set(self.\_format\_thousands(tt))  
 except Exception:  
 self.var\_thanhtien.set("0")  
  
 # ============================================================  
 def load\_material\_map(self):  
 *"""Load tên vật liệu -> mã vật liệu và set combobox TenVL"""* conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("SELECT MaVL, TenVL FROM vatlieu")  
 rows = cur.fetchall()  
 conn.close()  
 self.\_name\_to\_code = {ten: ma for ma, ten in rows}  
 names = [""] + [ten for \_, ten in rows]  
 self.cbb\_tenvl['values'] = names  
 self.cbb\_tenvl.set("")  
  
 def load\_mapn\_list(self):  
 *"""Load danh sách MaPN vào combobox (không cho nhập tay)"""* conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("SELECT MaPN FROM phieunhap ORDER BY MaPN")  
 rows = cur.fetchall()  
 conn.close()  
 self.\_mapn\_list = [r[0] for r in rows]  
 vals = [""] + self.\_mapn\_list  
 self.cbb\_mapn['values'] = vals  
 self.cbb\_mapn.set("")  
  
 # ============================================================  
 def load\_table(self):  
 self.tree.delete(\*self.tree.get\_children())  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 SELECT c.MaPN, v.TenVL, c.SoLuongNhap, c.DonGiaNhap, c.ThanhTien, c.MaVL  
 FROM ct\_phieunhap c  
 JOIN vatlieu v ON c.MaVL = v.MaVL  
 ORDER BY c.MaPN  
 """)  
 rows = cur.fetchall()  
 conn.close()  
 for row in rows:  
 r = list(row)  
 # format số hiển thị  
 try:  
 r[2] = self.\_format\_thousands(r[2]) # SoLuongNhap  
 r[3] = self.\_format\_thousands(r[3]) # DonGiaNhap  
 r[4] = self.\_format\_thousands(r[4]) # ThanhTien  
 except:  
 pass  
 self.tree.insert("", "end", values=r)  
  
 # ============================================================  
 def update\_tong\_tien(self, mapn):  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 UPDATE phieunhap  
 SET TongTien = (SELECT COALESCE(SUM(ThanhTien),0) FROM ct\_phieunhap WHERE MaPN=%s)  
 WHERE MaPN=%s  
 """, (mapn, mapn))  
 conn.commit()  
 conn.close()  
  
 def check\_mapn\_exists(self, mapn):  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("SELECT MaPN FROM phieunhap WHERE MaPN=%s", (mapn,))  
 ok = cur.fetchone()  
 conn.close()  
 return ok is not None  
  
 # ============================================================  
 def add\_ct(self):  
 mapn = self.cbb\_mapn.get().strip()  
 tenvl = self.cbb\_tenvl.get().strip()  
  
 if not mapn:  
 return messagebox.showerror("Lỗi", "Vui lòng chọn Mã phiếu nhập!")  
 if not self.check\_mapn\_exists(mapn):  
 return messagebox.showerror("Lỗi", "Mã phiếu nhập không tồn tại!")  
  
 mavl = self.\_name\_to\_code.get(tenvl)  
 if not mavl:  
 return messagebox.showerror("Lỗi", "Vui lòng chọn Tên vật liệu!")  
  
 try:  
 sl = self.\_parse\_number(self.entry\_soluong.get())  
 dg = self.\_parse\_number(self.entry\_dongia.get())  
 except Exception:  
 return messagebox.showerror("Lỗi", "Sai định dạng số!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 INSERT INTO ct\_phieunhap (MaPN, MaVL, SoLuongNhap, DonGiaNhap)  
 VALUES (%s,%s,%s,%s)  
 """, (mapn, mavl, float(sl), float(dg)))  
 conn.commit()  
 messagebox.showinfo("OK", "Thêm thành công!")  
 conn.close()  
 except Exception as e:  
 return messagebox.showerror("Lỗi khi lưu", str(e))  
  
 self.update\_tong\_tien(mapn)  
 self.load\_table()  
 self.clear\_inputs()  
  
 # ============================================================  
 def load\_for\_edit(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Chưa chọn!", "Chọn vật liệu để sửa!")  
  
 mapn, tenvl, sl, dg, tt, mavl = self.tree.item(sel)["values"]  
  
 # Set combobox và entries (lưu ý: tree chứa dữ liệu đã format)  
 self.cbb\_mapn.set(mapn)  
 self.cbb\_tenvl.set(tenvl)  
  
 # set MaVL  
 self.entry\_mavl.config(state="normal")  
 self.entry\_mavl.delete(0, tk.END)  
 self.entry\_mavl.insert(0, mavl)  
 self.entry\_mavl.config(state="readonly")  
  
 # set số lượng & đơn giá (để dạng hiển thị đã format)  
 self.entry\_soluong.delete(0, tk.END)  
 self.entry\_soluong.insert(0, sl)  
 self.entry\_dongia.delete(0, tk.END)  
 self.entry\_dongia.insert(0, dg)  
 self.var\_thanhtien.set(tt)  
  
 self.\_editing\_original = {"MaPN": mapn, "MaVL": mavl}  
  
 # ============================================================  
 def save\_edit(self):  
 if not self.\_editing\_original:  
 return  
  
 oldpn = self.\_editing\_original["MaPN"]  
 oldvl = self.\_editing\_original["MaVL"]  
  
 mapn = self.cbb\_mapn.get().strip()  
 tenvl = self.cbb\_tenvl.get().strip()  
 mavl = self.\_name\_to\_code.get(tenvl)  
  
 if not mavl:  
 return messagebox.showerror("Lỗi", "Vui lòng chọn Tên vật liệu!")  
  
 try:  
 sl = self.\_parse\_number(self.entry\_soluong.get())  
 dg = self.\_parse\_number(self.entry\_dongia.get())  
 except Exception:  
 return messagebox.showerror("Lỗi", "Sai định dạng số!")  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 UPDATE ct\_phieunhap  
 SET SoLuongNhap=%s, DonGiaNhap=%s  
 WHERE MaPN=%s AND MaVL=%s  
 """, (float(sl), float(dg), oldpn, oldvl))  
 conn.commit()  
 conn.close()  
 except Exception as e:  
 return messagebox.showerror("Lỗi khi cập nhật", str(e))  
  
 self.update\_tong\_tien(mapn)  
 self.load\_table()  
 self.clear\_inputs()  
  
 # ============================================================  
 def delete\_ct(self):  
 sel = self.tree.selection()  
 if not sel:  
 return messagebox.showwarning("Thông báo", "Chọn dòng để xóa!")  
  
 item = sel[0]  
 mapn, tenvl, sl, dg, tt, mavl = self.tree.item(item)["values"]  
  
 if not messagebox.askyesno("Xác nhận", f"Xóa vật liệu '{tenvl}' khỏi phiếu nhập {mapn}?"):  
 return  
  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute(  
 "DELETE FROM ct\_phieunhap WHERE MaPN=%s AND MaVL=%s",  
 (mapn, mavl)  
 )  
 conn.commit()  
 conn.close()  
 self.update\_tong\_tien(mapn)  
 self.load\_table()  
 except Exception as e:  
 messagebox.showerror("Lỗi", str(e))  
  
 # ============================================================  
 def search(self):  
 key = self.entry\_search.get().strip()  
  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute("""  
 SELECT c.MaPN, v.TenVL, c.SoLuongNhap, c.DonGiaNhap, c.ThanhTien, c.MaVL  
 FROM ct\_phieunhap c  
 JOIN vatlieu v ON c.MaVL = v.MaVL  
 WHERE   
 c.MaPN LIKE %s OR   
 v.TenVL LIKE %s  
 """, (f"%{key}%", f"%{key}%"))  
  
 rows = cur.fetchall()  
 conn.close()  
  
 self.tree.delete(\*self.tree.get\_children())  
 for r in rows:  
 row = list(r)  
 try:  
 row[2] = self.\_format\_thousands(row[2])  
 row[3] = self.\_format\_thousands(row[3])  
 row[4] = self.\_format\_thousands(row[4])  
 except:  
 pass  
 self.tree.insert("", "end", values=row)  
  
 # ============================================================  
 def clear\_inputs(self):  
 self.cbb\_mapn.set("")  
 self.cbb\_tenvl.set("")  
 self.entry\_mavl.config(state="normal")  
 self.entry\_mavl.delete(0, tk.END)  
 self.entry\_mavl.config(state="readonly")  
 self.entry\_soluong.delete(0, tk.END)  
 self.entry\_dongia.delete(0, tk.END)  
 self.var\_thanhtien.set("0")  
 self.\_editing\_original = None

## **Các module liên kết khác**

* Module db\_functions.py

def connect\_db():  
 return mysql.connector.connect(  
 host="localhost",  
 user="root",  
 password="Minh@120705",  
 database="qlch\_vlxd",  
 port=3307   
 )  
  
  
# ============================  
# LOAD TABLE  
# ============================  
def load\_table(table):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
 cur.execute(f"SELECT \* FROM {table}")  
 rows = cur.fetchall()  
 conn.close()  
 return rows  
 except Error as e:  
 print("DB ERROR (load):", e)  
 return []  
  
  
# ============================  
# ADD RECORD  
# ============================  
def add\_record(table, fields, values):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 placeholders = ",".join(["%s"] \* len(values))  
 sql = f"INSERT INTO {table} ({','.join(fields)}) VALUES ({placeholders})"  
  
 cur.execute(sql, values)  
 conn.commit()  
 conn.close()  
  
 except Error as e:  
 print("DB ERROR (add):", e)  
 raise  
  
  
# ============================  
# DELETE RECORD  
# ============================  
def delete\_record(table, key\_field, key\_value):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 sql = f"DELETE FROM {table} WHERE {key\_field}=%s"  
 cur.execute(sql, (key\_value,))  
  
 conn.commit()  
 conn.close()  
  
 except Error as e:  
 print("DB ERROR (delete):", e)  
 raise  
  
  
# ============================  
# UPDATE RECORD  
# ============================  
def update\_record(table, key\_field, key\_value, fields, values):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 set\_clause = ",".join(f"{f}=%s" for f in fields)  
 sql = f"UPDATE {table} SET {set\_clause} WHERE {key\_field}=%s"  
  
 params = values + [key\_value]  
  
 cur.execute(sql, params)  
 conn.commit()  
 conn.close()  
  
 except Error as e:  
 print("DB ERROR (update):", e)  
 raise  
  
  
# ============================  
# SEARCH RECORD  
# ============================  
def search\_record(table, field, keyword):  
 try:  
 conn = connect\_db()  
 cur = conn.cursor()  
  
 sql = f"SELECT \* FROM {table} WHERE {field} LIKE %s"  
 cur.execute(sql, (f"%{keyword}%",))  
  
 rows = cur.fetchall()  
 conn.close()  
 return rows  
  
 except Error as e:  
 print("DB ERROR (search):", e)  
 return []

* Module main.py
* def test\_connection():  
   try:  
   conn = connect\_db()  
   conn.close()  
   print("KẾT NỐI MYSQL OK")  
   return True  
   except:  
   print("KẾT NỐI MYSQL THẤT BẠI")  
   return False  
    
  # =========================== CANH GIỮA ===========================  
  def center\_window(win, w=1200, h=700):  
   win.update\_idletasks()  
   ws = win.winfo\_screenwidth()  
   hs = win.winfo\_screenheight()  
   x = (ws // 2) - (w // 2)  
   y = (hs // 2) - (h // 2)  
   win.geometry(f"{w}x{h}+{x}+{y}")  
    
  # ====================== MENU CHÍNH ========================  
  def create\_menu(root, username, role):  
    
   main = tk.Frame(root, bg="#F0F2F5")  
   main.pack(fill="both", expand=True)  
    
   menu = tk.Frame(main, bg="#003A75", width=250)  
   menu.pack(side="left", fill="y")  
    
   content = tk.Frame(main, bg="white")  
   content.pack(side="left", fill="both", expand=True)  
    
   def show\_frame(form\_class):  
   for w in content.winfo\_children():  
   w.destroy()  
   frame = form\_class(content)  
   frame.pack(fill="both", expand=True)  
    
   # ===== TITLE =====  
   tk.Label(menu, text="MENU CHÍNH", bg="#003A75",  
   fg="white", font=("Arial", 18, "bold")).pack(pady=30)  
    
   # ===== STYLE =====  
   btn\_style = dict(bg="white", fg="#003A75",  
   font=("Arial", 12, "bold"),  
   width=22, height=2, relief="flat")  
    
   # ===== NHÂN VIÊN =====  
   def open\_nv():  
   if role != "Quản lý":  
   return messagebox.showwarning("Cấm truy cập", "Chỉ Quản lý được vào mục này")  
   show\_frame(NhanVienFrame)  
   def open\_ncc():  
   if role != "Quản lý":  
   return messagebox.showwarning("Cấm truy cập", "Chỉ Quản lý được vào mục này")  
   show\_frame(NhaCungCapFrame)  
    
   tk.Button(menu, text="1. Quản lý nhân viên",  
   command=open\_nv, \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="2. Quản lý khách hàng",  
   command=lambda: show\_frame(KhachHangFrame), \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="3. Quản lý nhà cung cấp",  
   command=open\_ncc, \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="4. Quản lý vật liệu",  
   command=lambda: show\_frame(VatLieuFrame), \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="5. Quản lý hóa đơn",  
   command=lambda: show\_frame(HoaDonFrame), \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="6. Quản lý chi tiết hóa đơn",  
   command=lambda: show\_frame(CTHoaDonFrame), \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="7. Quản lý phiếu nhập",  
   command=lambda: show\_frame(PhieuNhapFrame), \*\*btn\_style).pack(pady=5)  
     
   tk.Button(menu, text="8. Quản lý chi tiết phiếu nhập",  
   command=lambda: show\_frame(CTPhieuNhapFrame), \*\*btn\_style).pack(pady=5)  
    
   # ===== TRANG CHÀO =====  
   tk.Label(content,  
   text=f"Hệ thống quản lý cửa hàng VLXD", fg="#E01414",  
   bg="white", font=("Arial", 24, "bold")  
   ).place(relx=0.5, rely=0.40, anchor="center")  
    
   tk.Label(content,  
   text=f"Xin chào: {username} ({role})", fg="#003A75",  
   bg="white", font=("Arial", 24, "bold")  
   ).place(relx=0.5, rely=0.50, anchor="center")  
    
   tk.Label(content,  
   text="Vui lòng chọn chức năng bên trái",  
   bg="white", font=("Arial", 16)  
   ).place(relx=0.5, rely=0.55, anchor="center")  
    
   # ==================== NÚT DƯỚI CÙNG =====================  
   bottom\_frame = tk.Frame(menu, bg="#003A75")  
   bottom\_frame.pack(side="bottom", pady=25)  
    
   btn\_small\_style = dict(  
   font=("Arial", 10, "bold"),  
   width=12,  
   height=1,  
   relief="raised"  
  )  
    
    
  # ---- Quay lại ----  
   def go\_back():  
   for w in root.winfo\_children():  
   w.destroy()  
   login\_screen(root)  
    
   tk.Button(bottom\_frame,  
   text="Quay lại",  
   bg="#28A745", fg="white",  
   command=go\_back,  
   \*\*btn\_small\_style).grid(row=0, column=0, padx=5)  
    
   tk.Button(bottom\_frame,  
   text="Thoát",  
   bg="#C9302C", fg="white",  
   command=root.quit,  
   \*\*btn\_small\_style).grid(row=0, column=1, padx=5)  
    
    
    
  # ====================== LOGIN SCREEN ========================  
  def login\_screen(root):  
    
   frame = tk.Frame(root, bg="#F0F2F5")  
   frame.pack(fill="both", expand=True)  
    
   box = tk.Frame(frame, bg="white", bd=2, relief="groove")  
   box.place(relx=0.5, rely=0.5, anchor="center")  
    
   box.config(width=420, height=380)  
   box.pack\_propagate(False)  
    
   tk.Label(box, text="ĐĂNG NHẬP", bg="white",  
   font=("Arial", 22, "bold"), fg="#003A75").pack(pady=20)  
    
   accounts = {  
   "admin": ("123456", "Quản lý"),  
   "nv1": ("abc123", "Nhân viên"),  
   "nv2": ("xyz456", "Nhân viên")  
   }  
    
   # USERNAME  
   tk.Label(box, text="Tên đăng nhập:", bg="white").pack()  
   entry\_user = tk.Entry(box, width=30)  
   entry\_user.pack(pady=5)  
    
   # PASSWORD  
   tk.Label(box, text="Mật khẩu:", bg="white").pack()  
   entry\_pass = tk.Entry(box, show="\*", width=30)  
   entry\_pass.pack(pady=5)  
    
   # ROLE  
   tk.Label(box, text="Chức vụ:", bg="white").pack()  
   role\_var = tk.StringVar(value="Nhân viên")  
    
   tk.Radiobutton(box, text="Quản lý", variable=role\_var,  
   value="Quản lý", bg="white").pack()  
   tk.Radiobutton(box, text="Nhân viên", variable=role\_var,  
   value="Nhân viên", bg="white").pack()  
    
   # ========= LOGIN ===========  
   def do\_login():  
   user = entry\_user.get()  
   pw = entry\_pass.get()  
   role = role\_var.get()  
    
   if user in accounts:  
   correct\_pw, correct\_role = accounts[user]  
    
   if pw != correct\_pw:  
   return messagebox.showerror("Lỗi", "Sai mật khẩu!", "Vui lòng nhập lại")  
    
   if role != correct\_role:  
   return messagebox.showerror("Lỗi", "Sai chức vụ!")  
    
   messagebox.showinfo("Thành công", "Đăng nhập thành công!")  
    
   frame.destroy()  
   create\_menu(root, user, role)  
    
   else:  
   messagebox.showerror("Lỗi", "Không tồn tại tài khoản")  
    
   tk.Button(box, text="Đăng nhập", bg="#0056D6",  
   fg="white", width=20, font=("Arial", 12),  
   command=do\_login).pack(pady=20)  
    
    
  # =========================== MAIN ==========================  
  def main():  
    
   if not test\_connection():  
   return messagebox.showerror("Lỗi DB", "Không thể kết nối MySQL!")  
    
   root = tk.Tk()  
   root.title("Phần mềm quản lý cửa hàng VLXD")  
   center\_window(root, 1200, 700)  
    
   login\_screen(root)  
    
   root.mainloop()  
    
    
  if \_\_name\_\_ == "\_\_main\_\_":  
   main()
* Module lienket.py
* def connect\_db():  
   try:  
   conn = mysql.connector.connect(  
   host="localhost",  
   user="root",  
   password="Minh@120705",  
   database="qlch\_vlxd",  
   port=3307  
   )  
   return conn  
   except Error as e:  
   print("Lỗi kết nối:", e)  
   return None  
    
  # Test kết nối khi chạy trực tiếp file  
  if \_\_name\_\_ == "\_\_main\_\_":  
   conn = connect\_db()  
   if conn:  
   print("Kết nối thành công!")  
   conn.close()

## **Liên kết giao diện Tkinter với các hàm xử lý cơ sở dữ liệu đã xây dựng ở module db\_functions.py**

* Các nút trên giao diện được gắn chức năng tương ứng:

+ Thêm nhân viên: Lấy dữ liệu nhập từ Entry và gọi hàm add\_record() để thêm vào MySQL.

A computer screen shot of a computer

AI-generated content may be incorrect.

Hình 18: Giao diện quản lý nhân viên khi đã thêm nhân viên thành công

A screenshot of a computer

AI-generated content may be incorrect.

Hình 19: Giao diện quản lý nhân viên khi quản lý không nhập đầy đủ thông tin

* Sửa nhân viên: Khi chọn dòng và bấm Sửa, dữ liệu hiển thị lên Entry → người dùng chỉnh sửa → gọi update\_record().

A screenshot of a computer

AI-generated content may be incorrect.

Hình 20: Giao diện quản lý nhân viên thông báo yêu cầu chọn nhân viên để sửa

A screenshot of a computer

AI-generated content may be incorrect.

Hình 21: Giao diện quản lý nhân viên thông báo đã sửa và cập nhật thành công

* Hủy: dùng để xóa toàn bộ dữ liệu đang hiển thị trên các ô Entry, đưa giao diện về trạng thái trống để người dùng nhập mới. Chức năng này không tác động đến cơ sở dữ liệu, mà chỉ làm sạch các ô nhập liệu trên form. Trong chương trình, nút Hủy sẽ gọi hàm clear\_entries() để xóa nội dung các Entry như Mã NV, Tên NV, Giới tính, Lương,…

A screenshot of a computer

AI-generated content may be incorrect.

Hình 22: Giao diện quản lý nhân viên, các ô nhập liệu khi chưa dùng nút Hủy

A screenshot of a computer

AI-generated content may be incorrect.

Hình 23: Giao diện quản lý nhân viên, các ô nhập liệu sau khi đã dùng nút Hủy

* Xóa nhân viên: Gọi delete\_record() theo MaNV.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 24: Giao diện quản lý nhân viên hiện câu hỏi xác nhận muốn Xóa

A screenshot of a computer

AI-generated content may be incorrect.

Hình 25: Giao diện quản lý nhân viên sau khi xác nhận Xóa

* Tìm kiếm nhân viên: Khi nhập từ khóa, hàm search\_record() được gọi để truy vấn dữ liệu theo điều kiện.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 26: Giao diện quản lý nhân viên khi tìm kiếm nhân viên

* Thoát: Khi người dùng nhấn nút “Thoát”, chương trình sẽ hiển thị hộp thoại xác nhận. Nếu người dùng chọn “Yes”, ứng dụng sẽ đóng hoàn toàn bằng lệnh master.destroy().

A screenshot of a computer

AI-generated content may be incorrect.

Hình 27: Giao diện quản lý nhân viên hiển thị hộp thoại yêu cầu xác nhận Thoát

* Quay lại menu chính: Khi người dùng nhấn nút “Quay lại”, giao diện hiện tại sẽ đóng lại và ứng dụng chuyển về màn hình menu chính. Trong code Tkinter, nút này được gắn với hàm show\_main\_menu() để hiển thị lại giao diện chính.

A white background with red and blue text

AI-generated content may be incorrect.

Hình 28: Giao diện của quản lý quay lại màn hình menu chính

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 29: Giao diện quay lại phần đăng nhập