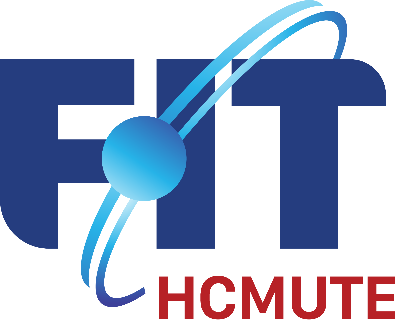


**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**



**ĐỀ TÀI CUỐI KỲ MÔN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**TÊN ĐỀ TÀI:**

**QUẢN LÝ BÁN HÀNG THỂ THAO**

**GVHD:** TS. Nguyễn Thành Sơn

**Lớp HP:** DBMS330284\_23\_2\_03CLC

**Nhóm thực hiện:** Nhóm 06

**Học kỳ:** 2

**Năm học:** 2023 – 2024

Thành phố Hồ Chí Minh, … tháng … năm 2024

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ THỐNG THÔNG TIN**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA**

**VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

TP.HCM, ngày … tháng … năm 2024

**DANH SÁCH SINH VIÊN THỰC HIỆN**

**HỌC KỲ 2 NĂM HỌC 2023 – 2024**

**Nhóm 06**

**Đề tài: Quản lý cửa hàng bán đồ thể thao**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên sinh viên** | **Mã số sinh viên** | **Tỉ lệ % tham gia** |
| 1 | Võ Thị Kim Anh | 22110105 | 100% |
| 2 | Nguyễn Hoài Bảo | 22110108 | 100% |
| 3 | Lê Thanh Phong | 22110198 | 100% |
| 4 | Nguyễn Chí Thanh | 22110226 | 100% |

**Ghi chú:**

* Tỉ lệ % = 100%
* Trưởng nhóm: Nguyễn Chí Thanh

Nhận xét của giáo viên

Ngày … tháng … năm 2024

Giáo viên chấm điểm

Mục lục

**[LỜI CẢM ƠN 4](#_Toc25292)**

**[LỜI NÓI ĐẦU 5](#_Toc11426)**

**[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG 6](#_Toc17380)**

[1.1 Mô tả bài toán: 6](#_Toc8898)

[1.2 Nghiệp vụ bài toán: 7](#_Toc20331)

[- Quản lý nhân viên: 7](#_Toc2069)

[- Quy trình phục vụ khách hàng: 7](#_Toc4824)

[- Quy trình quản lý hàng hóa: 8](#_Toc16950)

**[CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG](#_Toc15189)** [9](#_Toc15189)

[2.1 Thiết kế cơ sở dữ liệu mức quan niệm: 9](#_Toc6078)

[2.2 Thiết kế cơ sở dữ liệu mức logic: 10](#_Toc21217)

[2.2 Thiết kế cơ sở dữ liệu ở mức vật lý: 11](#_Toc31058)

[2.3 Các ràng buộc cho cơ sở dữ liệu: 11](#_Toc25511)

[2.4 Cài đặt cơ sở dữ liệu và các ràng buộc: 14](#_Toc7195)

[2.5 Các View: 18](#_Toc6933)

[2.6 Các Trigger: 23](#_Toc14617)

**[CHƯƠNG 3: THIẾT KẾ CÁC CHỨC NĂNG](#_Toc4709)** [30](#_Toc4709)

[3.1 Lớp kết nối và hàm truy vấn cơ sở dữ liệu 30](#_Toc8174)

[3.2 Các hàm và thủ tục về Sản Phẩm 34](#_Toc20666)

[3.3 Các hàm và thủ tục về Khách Hàng 38](#_Toc272)

[3.4 Các hàm và thủ tục về Hóa Đơn 42](#_Toc31502)

[3.5 Các hàm và thủ tục về Chi Tiết Hóa Đơn 46](#_Toc12935)

[3.6 Các hàm và thủ tục về Nhân Viên 47](#_Toc2161)

[3.7 Các hàm và thủ tục về Nhà Cung Cấp 53](#_Toc2978)

[3.8 Các hàm và thủ tục về Biên Lai 57](#_Toc17779)

[3.9 Các hàm và thủ tục về Chi Tiết Biên Lai 61](#_Toc5957)

[3.10 Các hàm và thủ tục về Doanh Thu 62](#_Toc1966)

[3.11 Các hàm và thủ tục về Lợi Nhuận 63](#_Toc6595)

**[Chương 4: TẠO USER VÀ PHÂN QUYỀN](#_Toc15048)** [64](#_Toc15048)

[4.1 Tạo Role 64](#_Toc3675)

[4.2 Tạo user 66](#_Toc31339)

[4.3 Kiểm tra user 68](#_Toc8660)

**[Chương 5 :THIẾT KẾ GIAO DIỆN HỆ THỐNG](#_Toc18684)** [70](#_Toc18684)

[5.1 Giao diện quản lý 70](#_Toc6864)

[5.2 Giao diện nhân viên bán hàng 79](#_Toc12574)

**LỜI CẢM ƠN**

Trong suốt quá trình thực hiện đồ án môn học Hệ Quản Trị Cơ Sở Dữ Liệu, chúng em may mắn nhận được sự hướng dẫn và động viên từ thầy. Thầy đã không ngừng truyền đạt kiến thức, tạo điều kiện cho chúng em phát triển và hoàn thiện kỹ năng trong lĩnh vực này. Chúng em xin bày tỏ lòng biết ơn sâu sắc và gửi đến thầy những lời chúc tốt đẹp nhất. Chúc thầy sẽ tiếp tục thành công trong sự nghiệp giáo dục, trên hành trình trao truyền tri thức và tiếp tục truyền cảm hứng cho thế hệ sinh viên tương lai. Dù đã cố gắng hoàn thiện đồ án, chúng em không tránh khỏi những thiếu sót do kiến thức còn hạn chế. Chúng em trân trọng nhận mọi ý kiến đóng góp của thầy để chúng em có thể hoàn thiện hơn và phát triển tốt hơn trong tương lai. Một lần nữa, chúng em xin chân thành cảm ơn thầy vì sự hỗ trợ nhiệt tình dành cho thế hệ sinh viên. Trân trọng cảm ơn thầy!

# **TỔNG QUAN VỀ HỆ THỐNG**

## **1.1 Mô tả bài toán:**

Để kinh doanh một cửa hàng thể thao một cách hiệu quả và linh hoạt, việc triển khai một hệ thống quản lý là điều thiết yếu. Hệ thống này được thiết kế để đáp ứng đầu đủ các yêu cầu liên quan đến quản lí nhân viên, hàng hoá, giao dịch mua bán trong cửa hàng. Trọng tâm của hệ thống là tập trung vào hai đối tượng chính: người quản lý, có trách nhiệm quản lý toàn bộ hoạt động; nhân viên, thực hiện việc trao đổi mua bán các sản phẩm trong cửa hàng. Để đảm bảo được tính an toàn và linh hoạt, cả hai đối tượng này sẽ được gán những quyền hạn và truy cập khác nhau.

Cửa hàng sẽ có nhiều nhân viên, mỗi nhân viên sẽ được phân biệt bằng mã nhân viên, mỗi nhân viên sẽ có họ tên, số điện thoại, ngày sinh, giới tính, địa chỉ, mã chức vụ . Mỗi nhân viên sẽ có một tài khoản đăng nhập với các thông tin là mã nhân viên, mật khẩu, trạng thái. Khách hàng sẽ được lưu trữ bằng số điện thoại, mỗi khách hàng sẽ có các thông tin như họ tên, số điện thoại, ngày sinh, giới tính, điểm tính luỹ (điểm tích luỹ dùng để tính khuyến mãi).

Sản phẩm sẽ được chia thành nhiều loại sản phẩm và có các thương hiệu khác nhau. Loại sản phẩm sẽ được phân biệt bằng mã loại, tên. Mỗi loại sản phẩm có thể có nhiều sản phẩm, và mỗi sản phẩm chỉ thuộc một loại sản phẩm. Mỗi sản phẩm sẽ thuộc một thương hiệu, các thương hiệu được phân biệt bằng mã thương hiệu, tên thương hiệu. Về sản phẩm, có các thông tin như mã sản phẩm, tên sản phẩm, số lượng, đơn giá bán, mã loại sản phẩm, mã thương hiệu (đối với các sản phẩm khác size và khác màu sẽ có mã sản phẩm khác nhau).

Để nhập sản phẩm về cửa hàng, thông tin về nhà cung cấp cũng là quan trọng. Mỗi nhà cung cấp có thông tin như mã nhà cung cấp, tên nhà cung cấp, địa chỉ và số điện thoại, email. Mỗi lần nhập hàng thì sẽ có Biên lai nhập hàng sẽ được đánh số với mã biên lai nhập hàng và thông tin như ngày nhập hàng, trị giá biên lai nhập hàng và mã nhà cung cấp.

Mỗi khách hàng sẽ có thể mua sản phẩm nhiều lần nên sẽ có nhiều hóa đơn nên mỗi hóa đơn được phân biệt bằng mã hóa đơn, gồm các thông tin như mã hóa đơn, ngày thanh toán, số điện thoại khách hàng và trị giá hóa đơn và khuyến mãi (nếu có). Chi tiết khuyến mãi sẽ được quản lý qua bảng khuyến mãi, bao gồm mã khuyến mãi, phần trăm khuyến mãi, thời gian bắt đầu và kết thúc.

## **Nghiệp vụ bài toán:**

Bài toán quản lý cửa hàng bán đồ thể thao là một nghiệp vụ quản lý kinh doanh trong lĩnh vực đồ dùng thể thao. Nó liên quan đến việc quản lý hàng hóa, quản lý doanh thu và điều hành cửa hàng. Nhân viên là bộ mặt của cửa hàng và giao tiếp với khách hàng từ lúc bước vào cho đến khi khách ra về. Vì vậy chúng ta cần xây dựng tài liệu đào tạo nghiệp vụ cho nhân viên về quy trình tác nghiệp, văn hóa phục vụ, cách ứng xử, ...

**- Quản lý nhân viên:**

Khi tuyển dụng nhân sự, cửa hàng cần đưa ra các yêu cầu cũng như các điều kiện (ví dụ: làm ca đêm, không đi trễ, ...) mà nhân sự đồng ý trong suốt quá trình làm việc. Cửa hàng cần trả cho nhân viên mức lương hợp lý. Nhân viên đã được tuyển dụng sẽ trải qua đợt đào tạo về nghiệp vụ, quy trình ở vị trí của bản thân. Người quản lý cần theo dõi và quan sát các nhân sự của mình, chỉ ra những thiếu sót để nhân sự khắc phục và lời khen cho những điểm tốt của nhân sự lấy đó làm động lực.

**- Quy trình phục vụ khách hàng:**

Trước tiên, nhân viên sẽ chào khách trước khi bước vào cửa hàng. Sau đó nhân viên sẽ hỏi nhu cầu của khách là gì và tư vấn cho khách về những sản phẩm phù hợp với mình, các chương trình khuyến mãi, ... kiểm tra số lượng sản phẩm tư vấn cho khách hàng có còn hay không, sau khi đã hoàn thành việc chọn sản phẩm thì nhân viên sẽ tiến hành thêm các sản phẩm vào đơn hàng mới tạo. Tiếp theo hỏi khách đã từng mua ở cửa hàng chưa và có mã giảm giá không, nếu chưa thì tạo khách hàng mới bằng cách hỏi tên và số điện thoại, còn có rồi thì sử dụng số điện thoại của khách đã từng lưu. Tiếp theo là nhân viên tạo hóa đơn, ở đây nhân viên sẽ nhập mã sản phẩm trên sản phẩm, số lượng, số điện thoại khách hàng,... Cuối cùng nhân viên sẽ xuất hoá đơn cho khách và nhận tiền thanh toán cùng với sản phẩm. Sau khi hoàn thành thì nói lời cảm ơn khách hàng, mở cửa và tiễn khách về.

**- Quy trình quản lý hàng hóa:**

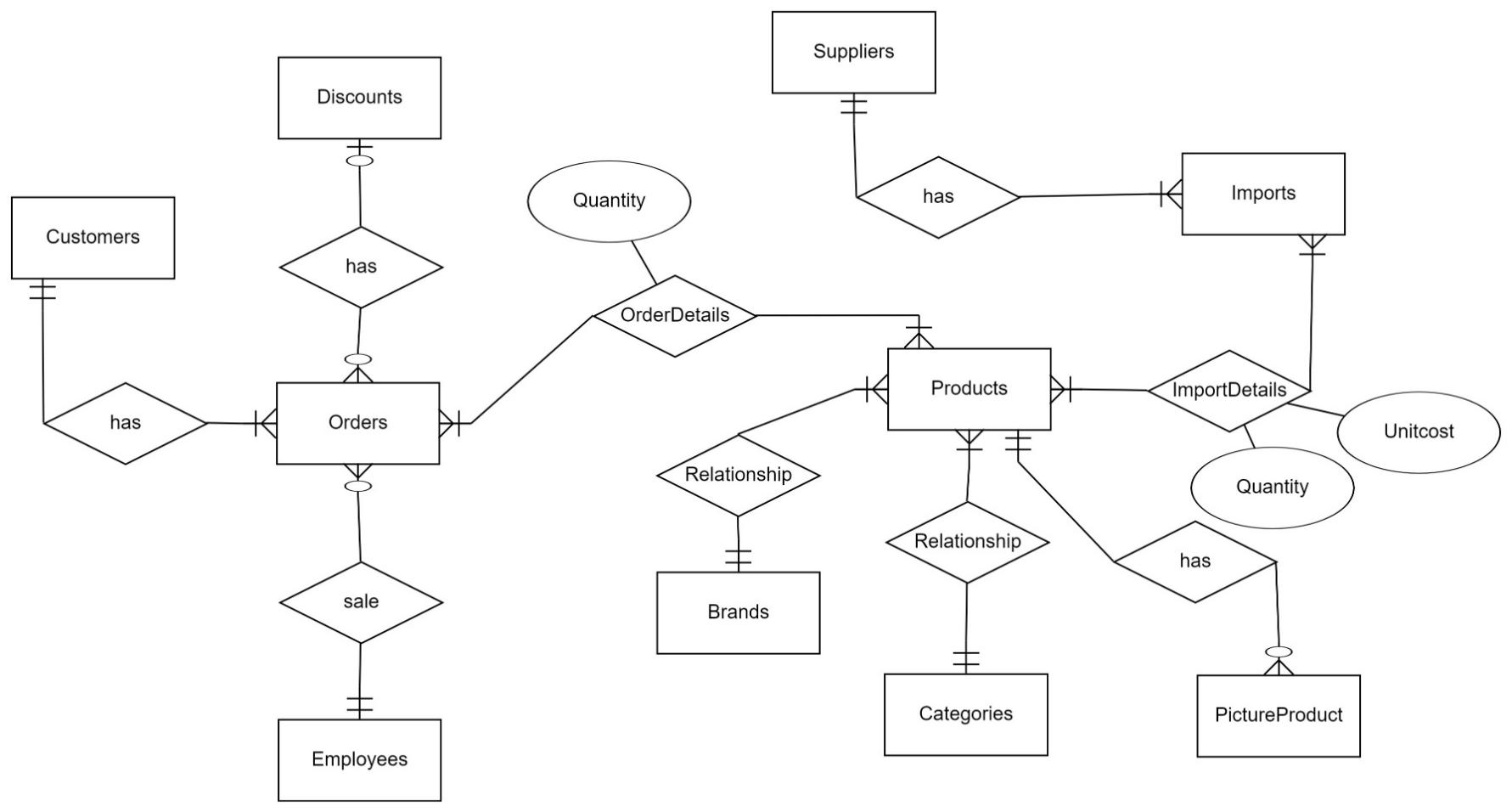
Đầu tiên cửa hàng phải tìm các nhà cung cấp uy tín, có chính sách giá cả, giao hàng, chứng từ rõ ràng để tiện cho việc kiểm soát. Khi nhập hàng cần kiểm tra hàng hóa có khớp với hóa đơn hay không, nếu khớp thì tiến hành nhập hàng.

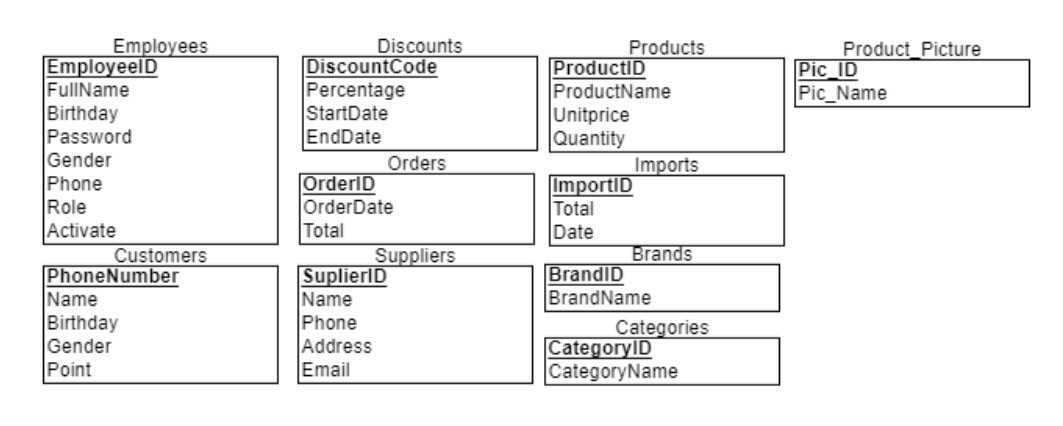
Quy trình vận hành, việc cửa hàng có nhiều sản phẩm, hay sai sót trong việc xuất hóa đơn của nhân viên có thể dẫn đến bị lệch mẫu sản phẩm (cùng loại cùng giá nhưng khác màu, khác size, ...) thì cần có cơ chế kiểm kho hàng tuần, hàng tháng để cần chính xác về kho sản phẩm

Cuối cùng, cần kiểm soát nghiệp vụ của nhân viên dựa trên những đánh giá của khách hàng về chất lượng phục vụ, trích xuất camera. Ngoài ra cần kiểm tra thu chi hằng ngày. Nhân viên sẽ được phân quyền để xem bảng doanh thu theo ngày để kết ca vào cuối ngày. Nhân viên có thể kiểm tra số tiền ca làm việc của mình rồi mới bàn giao ca cho nhân viên khác. Người quản lý cần phải nắm rõ doanh thu từng ngày, từng tháng, phân tích để nắm rõ lợi nhuận và có kế hoạch điều chỉnh sao cho đạt chất lượng tốt nhất

# **PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

## **Thiết kế cơ sở dữ liệu mức quan niệm:**





## **2.2 Thiết kế cơ sở dữ liệu mức logic:**

**CUSTOMERS** ( PhoneNumber, Name, Birthday, Gender , Point)  
**EMPLOYEES** ( EmployeeID, FullName, Birthday, Gender , Address, Phone,Role, Activate,Password)  
**BRANDS** ( Brand\_ID, BrandName)

**CATEGORIES** ( Category\_ID, CategoryName)

**DISCOUNTS** ( DiscountCode, Percentage, StartDay, EndDay)

**SUPPLIERS** ( Suppler\_ID, CompanyName, Phone, Address, Email)

**IMPORTS**( Import\_ID, Supplier\_ID(FK), ImportDate, Total)

**ORDERS**( Order\_ID, PhoneNumber(FK), Employee\_ID(FK), OrderDate, Total, DiscountCode(FK))

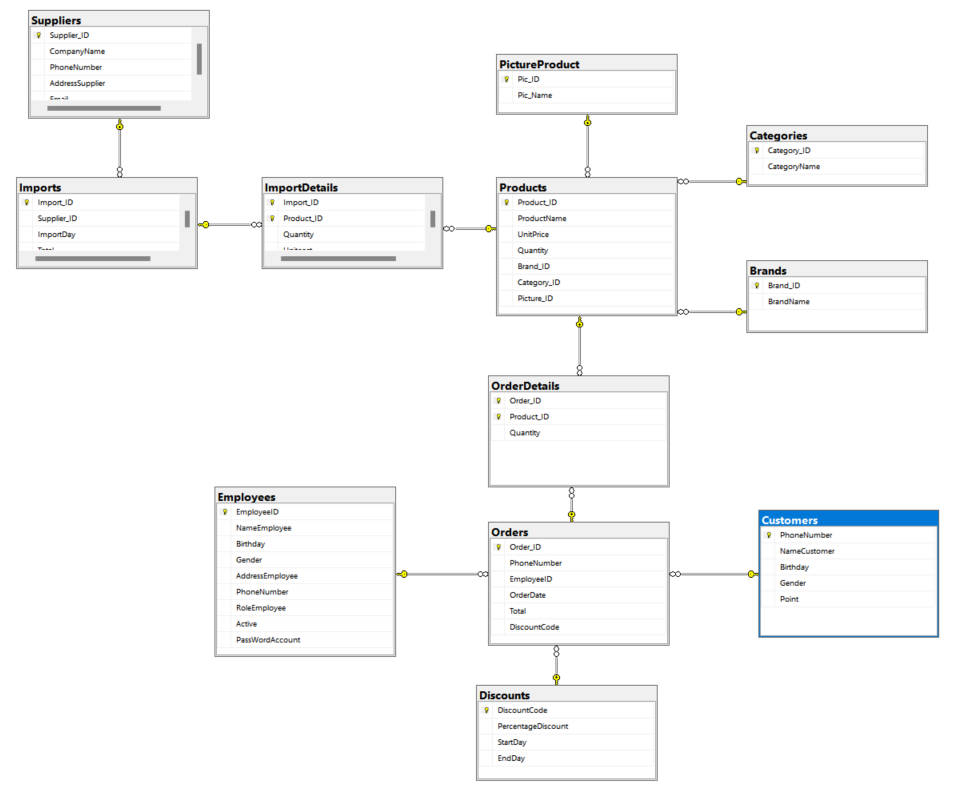
**PRODUCTS**( Product\_ID, ProductName, UnitPrice, Brand\_ID(FK), Category\_ID(FK), Quantity)

**ODERDETAILS**( Order\_ID(FK), Product\_ID(FK), Quantity)

**IMPORTDETAILS**( ImportID, Product\_ID(FK), Quantity, Unitcost)

**PICTUREPRODUCT**( Pic\_ID, Pic\_Name)

## **Thiết kế cơ sở dữ liệu ở mức vật lý:**



## **Các ràng buộc cho cơ sở dữ liệu:**

|  |  |  |
| --- | --- | --- |
| STT | TABLE |  |
| 1 | CUSTOMERS | - Ràng buộc khóa chính: PhoneNumber  - Ràng buộc not null:   * NameCustomer * Gender   - Ràng buộc check:   * LEN(PhoneNumber) = 10 * PhoneNumber NOT LIKE '%[^0-9]%' * GENDER IN ('Nam', 'Nu') |
| 2 | EMPLOYEES | - Ràng buộc khóa chính: EmployeeID  - Ràng buộc not null:   * NameEmployee * Gender * AddressEmployee * PhoneNumber * RoleEmployee * Active * PassWordAccount   - Ràng buộc check:   * LEN(PhoneNumber) = 10 * PhoneNumber NOT LIKE '%[^0-9]%' * GENDER IN ('Nam', 'Nu') * LEN(PassWordAccount) >= 6 |
| 3 | BRANDS | - Ràng buộc khóa chính: Brand\_ID  - Ràng buộc not null:   * BrandName |
| 5 | CATEGORIES | - Ràng buộc khóa chính: CategoryID  - Ràng buộc not null:   * CategoryName |
| 6 | DISCOUNTS | - Ràng buộc khóa chính: DiscountCode  - Ràng buộc not null:   * Percentage * StartDay * EndDay |
| 7 | SUPPLIERS | - Ràng buộc khóa chính: Suppler\_ID  - Ràng buộc not null:   * CompanyName * PhoneNumber * Address   - Ràng buộc check:   * LEN(PhoneNumber) = 10 * PhoneNumber NOT LIKE '%[^0-9]%' |
| 8 | IMPORTS | - Ràng buộc khóa chính: Import\_ID  - Ràng buộc not null:   * Supplier\_ID * ImportDate * Total   - Ràng buộc khóa ngoại: Supplier\_ID |
| 9 | ORDERS | - Ràng buộc khóa chính: Order\_ID  - Ràng buộc not null:   * OrderDate * Total * PhoneNumber * Employee\_ID   - Ràng buộc khóa ngoại:   * PhoneNumber * EmployeeID * DiscountCode |
| 10 | PRODUCTS | - Ràng buộc khóa chính: Product\_ID  - Ràng buộc not null:   * ProductName * UnitPrice * Brand\_ID * Category\_ID   - Ràng buộc khóa ngoại:   * Brand\_ID(FK) * Category\_ID(FK) |
| 11 | ORDERDETAILS | -Ràng buộc khóa chính: Order\_ID, Product\_ID  - Ràng buộc not null:   * ProductName * Order\_ID   -Ràng buộc khóa ngoại:  Order\_ID(FK), Product\_ID(FK) |
| 12 | IMPORTDETAILS | - Ràng buộc khóa chính: Import\_ID, Product\_ID  -Ràng buộc not null:   * Quantity * Unitcost   - Ràng buộc khóa ngoại:   * Import\_ID * Product\_ID |
| 13 | PICTUREPRODUCT | - Ràng buộc khóa chính: Pic\_ID  -Ràng buộc not null: Pic\_Name |

## **Cài đặt cơ sở dữ liệu và các ràng buộc:**

**2.5.1 Tạo database**

|  |
| --- |
| use master;  DROP DATABASE IF EXISTS QuanLyBanHangTheThao;  CREATE DATABASE QuanLyBanHangTheThao; |

**2.5.2 Bảng CUSTOMER**

|  |
| --- |
| CREATE TABLE Customers (  PhoneNumber VARCHAR(12) CONSTRAINT PhoneKey PRIMARY KEY,  NameCustomer NVARCHAR(50) NOT NULL,  Birthday DATE,  Gender CHAR(3) NOT NULL CONSTRAINT GenderCheck\_Customer CHECK (GENDER IN ('Nam', 'Nu')),  Point INT,  CONSTRAINT PhoneNumber\_Customer CHECK (LEN(PhoneNumber) = 10 AND PhoneNumber NOT LIKE '%[^0-9]%')  ); |

**2.5.3 Bảng EMPLOYEE**

|  |
| --- |
| CREATE TABLE Employees (  EmployeeID VARCHAR(5) CONSTRAINT EmployeeIDKey PRIMARY KEY,  NameEmployee NVARCHAR(50) NOT NULL,  Birthday DATE,  Gender CHAR(3) NOT NULL CONSTRAINT GenderCheck\_Employee CHECK (GENDER IN ('Nam', 'Nu')),  AddressEmployee NVARCHAR(100) NOT NULL,  PhoneNumber VARCHAR(12) NOT NULL,  RoleEmployee NVARCHAR(50) NOT NULL,  Active CHAR(1) NOT NULL,  PassWordAccount VARCHAR(10) NOT NULL,  CONSTRAINT PhoneNumber\_Employee CHECK (LEN(PhoneNumber) = 10 AND PhoneNumber NOT LIKE '%[^0-9]%'),  CONSTRAINT LengthPassWordAccount CHECK (LEN(PassWordAccount) >= 6)  ); |

**2.5.5 Bảng BRAND**

|  |
| --- |
| CREATE TABLE Brands (  Brand\_ID VARCHAR(10) CONSTRAINT Brand\_IDKey PRIMARY KEY,  BrandName NVARCHAR(50) NOT NULL  ); |

**2.5.6 Bảng CATEGORY**

|  |
| --- |
| CREATE TABLE Categories (  Category\_ID VARCHAR(10) CONSTRAINT Category\_IDKey PRIMARY KEY,  CategoryName NVARCHAR(50) NOT NULL  ); |

**2.5.7 Bảng DISCOUNT**

|  |
| --- |
| CREATE TABLE Discounts (  DiscountCode VARCHAR(10) CONSTRAINT DiscountCodeKey PRIMARY KEY,  PercentageDiscount INT NOT NULL,  StartDay DATE NOT NULL,  EndDay DATE NOT NULL  ); |

**2.5.8 Bảng SUPPLIER**

|  |
| --- |
| CREATE TABLE Suppliers (  Supplier\_ID VARCHAR(10) CONSTRAINT Supplier\_IDKey PRIMARY KEY,  CompanyName NVARCHAR(30) NOT NULL,  PhoneNumber VARCHAR(12) CONSTRAINT PhoneNumber\_Supplier CHECK (LEN(PhoneNumber) = 10 AND PhoneNumber NOT LIKE '%[^0-9]%'),  AddressSupplier NVARCHAR(100) NOT NULL,  Email VARCHAR(50)  ); |

**2.5.9 Bảng IMPORT**

|  |
| --- |
| CREATE TABLE Imports (  Import\_ID VARCHAR(10) CONSTRAINT Import\_IDKey PRIMARY KEY,  Supplier\_ID VARCHAR(10) Not null ,  ImportDay DATE NOT NULL,  Total INT NOT NULL,  CONSTRAINT FK\_Supplier\_Import FOREIGN KEY (Supplier\_ID)  REFERENCES Suppliers(Supplier\_ID) On Update Cascade  ); |

**2.5.10 Bảng ORDERS**

|  |
| --- |
| CREATE TABLE Orders (  Order\_ID VARCHAR(15) CONSTRAINT Order\_IDKey PRIMARY KEY,  PhoneNumber VARCHAR(12) NOT NULL,  EmployeeID VARCHAR(5) NOT NULL,  OrderDate DATE NOT NULL,  Total INT NOT NULL,  DiscountCode VARCHAR(10),  CONSTRAINT PhoneNumber\_Orders CHECK (LEN(PhoneNumber) = 10 AND PhoneNumber NOT LIKE '%[^0-9]%'),  CONSTRAINT FK\_PhoneNumber\_Orders FOREIGN KEY (PhoneNumber)  REFERENCES Customers(PhoneNumber) On Update Cascade,  CONSTRAINT FK\_Employee\_Orders FOREIGN KEY (EmployeeID)  REFERENCES Employees(EmployeeID),  CONSTRAINT FK\_Discount\_Order FOREIGN KEY (DiscountCode)  REFERENCES Discounts(DiscountCode)  ); |

**2.5.11 Bảng PRODUCT**

|  |
| --- |
| CREATE TABLE Products (  Product\_ID VARCHAR(15) CONSTRAINT Product\_IDKey PRIMARY KEY,  ProductName NVARCHAR(100) NOT NULL,  UnitPrice INT NOT NULL,  Quantity INT NOT NULL,  Brand\_ID VARCHAR(10) NOT NULL,  Category\_ID VARCHAR(10) NOT NULL,  Picture\_ID int  CONSTRAINT FK\_Brand\_Product FOREIGN KEY (Brand\_ID)  REFERENCES Brands(Brand\_ID) ,  CONSTRAINT FK\_Category\_Product FOREIGN KEY (Category\_ID)  REFERENCES Categories(Category\_ID),  CONSTRAINT FK\_Pic\_Product FOREIGN KEY (Picture\_ID)  REFERENCES PictureProduct(Pic\_ID)  ); |

**2.5.12 Bảng ORDERDETAIL**

|  |
| --- |
| CREATE TABLE OrderDetails (      Order\_ID VARCHAR(15) NOT NULL,      Product\_ID VARCHAR(15) NOT NULL,      Quantity INT NOT NULL,      CONSTRAINT FK\_Order\_OrderDetail FOREIGN KEY (Order\_ID)  REFERENCES Orders(Order\_ID),      CONSTRAINT FK\_Product\_OrderDetail FOREIGN KEY (Product\_ID)  REFERENCES Products(Product\_ID) ON UPDATE CASCADE,  CONSTRAINT OrderDetail\_IDKey PRIMARY KEY (Order\_ID , Product\_ID)  ); |

**2.5.13 Bảng IMPORTDETAIL**

|  |
| --- |
| CREATE TABLE ImportDetails (      Import\_ID VARCHAR(10),      Product\_ID VARCHAR(15),      Quantity INT NOT NULL,      Unitcost INT NOT NULL,      CONSTRAINT FK\_Import\_ImportDetail FOREIGN KEY (Import\_ID)  REFERENCES Imports(Import\_ID),      CONSTRAINT FK\_Product\_ImportDetail FOREIGN KEY (Product\_ID)  REFERENCES Products(Product\_ID),  CONSTRAINT ImportDetail\_IDKey PRIMARY KEY (Import\_ID , Product\_ID)  ); |

**2.5.13 Bảng PICTUREPRODUCT**

|  |
| --- |
| CREATE TABLE PictureProduct (  Pic\_ID INT IDENTITY(1,1) PRIMARY KEY,  Pic\_Name varchar(100)  ); |

## **Các View:**

Xem danh sách sản phẩm

|  |
| --- |
| create VIEW View\_Product AS  SELECT  p.Product\_ID,  p.ProductName,  p.UnitPrice,  p.Quantity,  b.BrandName,  c.CategoryName,  pp.Pic\_Name,  p.Picture\_ID  FROM Products p  LEFT JOIN Brands b ON p.Brand\_ID = b.Brand\_ID  LEFT JOIN Categories c ON p.Category\_ID = c.Category\_ID  LEFT JOIN PictureProduct pp on p.Picture\_ID = pp.Pic\_ID; |

Thực hiện trên C#

|  |
| --- |
| public DataSet LaySanPham()  {  return db.ExecuteQueryDataSet(  "select Product\_ID, ProductName,UnitPrice,Quantity from View\_Product", CommandType.Text, null); // Method to retrieve products for general view  } |

Xem danh sách khách hàng

|  |
| --- |
| CREATE VIEW View\_Customer AS  SELECT  c.PhoneNumber,  c.NameCustomer,  c.Birthday,  c.Gender,  c.Point,  SUM(o.Total) AS Tong  FROM  Customers c LEFT JOIN Orders o  ON  c.PhoneNumber = o.PhoneNumber  GROUP BY  c.PhoneNumber,  c.NameCustomer,  c.Birthday,  c.Gender,  c.Point |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayKhachHang()  {  return db.ExecuteQueryDataSet(  "select \* from View\_Customer", CommandType.Text, null);  } |

Xem danh sách nhân viên còn làm việc

|  |
| --- |
| CREATE VIEW EMPLOYEES\_ACTIVE\_VIEW  AS  SELECT  e.EmployeeID,  e.NameEmployee,  e.Birthday,  e.Gender,  e.AddressEmployee,  e.PhoneNumber,  e.RoleEmployee,  Sum(o.Total) as Tong  FROM Employees e  Left Join Orders o on e.EmployeeID=o.EmployeeID  WHERE e.Active = 1  GROUP BY  e.EmployeeID,  e.NameEmployee,  e.Birthday,  e.Gender,  e.AddressEmployee,  e.PhoneNumber,  e.RoleEmployee |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayNhanVien()  {  return db.ExecuteQueryDataSet(  "select \* from EMPLOYEES\_ACTIVE\_VIEW", CommandType.Text, null);  } |

Xem toàn bộ danh sách nhân viên

|  |
| --- |
| CREATE VIEW EMPLOYEES\_All\_VIEW  AS  SELECT  e.EmployeeID,  e.NameEmployee,  e.Birthday,  e.Gender,  e.AddressEmployee,  e.PhoneNumber,  e.RoleEmployee,  e.Active,  e.PassWordAccount,  Sum(o.Total) as Tong  FROM Employees e  Left Join Orders o on e.EmployeeID=o.EmployeeID  GROUP BY  e.EmployeeID,  e.NameEmployee,  e.Birthday,  e.Gender,  e.AddressEmployee,  e.PhoneNumber,  e.RoleEmployee,  e.Active,  e.PassWordAccount |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayALLNhanVien()  {  return db.ExecuteQueryDataSet(  "select \* from EMPLOYEES\_All\_VIEW", CommandType.Text, null);  } |

Xem danh sách hoá đơn xuất ra

|  |
| --- |
| CREATE VIEW BILLS\_VIEW  AS  SELECT  o.Order\_ID,  c.NameCustomer,  e.NameEmployee,  o.OrderDate,  o.Total,  d.PercentageDiscount,  o.PhoneNumber,  o.DiscountCode,  o.EmployeeID  FROM Orders o  JOIN Employees e ON e.EmployeeID = o.EmployeeID  JOIN Customers c ON c.PhoneNumber = o.PhoneNumber  left JOIN Discounts d ON d.DiscountCode = o.DiscountCode |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayHoaDon()  {  return db.ExecuteQueryDataSet(  "select \* from BILLS\_VIEW", CommandType.Text, null);  } |

Xem danh sách hoá đơn nhập vào

|  |
| --- |
| CREATE VIEW IMPORTS\_VIEW  AS  SELECT  ips.Import\_ID,  ips.ImportDay,  ips.Total,  ips.Supplier\_ID,  s.CompanyName  FROM Imports ips  JOIN Suppliers s ON ips.Supplier\_ID = s.Supplier\_ID |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayBienLai()  {  return db.ExecuteQueryDataSet(  "select \* from IMPORTS\_VIEW", CommandType.Text, null);  } |

Xem danh sách nhà phân phối

|  |
| --- |
| Create VIEW SUPPLIER\_VIEW  AS  SELECT  s.Supplier\_ID,  s.CompanyName,  s.PhoneNumber,  s.AddressSupplier,  s.Email,  Sum(i.Total) as Tong  FROM Suppliers s  Left JOIN Imports i ON i.Supplier\_ID = s.Supplier\_ID  Group by  s.Supplier\_ID,  s.CompanyName,  s.PhoneNumber,  s.AddressSupplier,  s.Email |

Thực hiện trên C#

|  |
| --- |
| public DataSet LayNhaCungCap()  {  return db.ExecuteQueryDataSet(  "select \* from SUPPLIER\_VIEW", CommandType.Text, null);  } |

## **Các Trigger:**

Trigger cập nhật tổng tiền của hoá đơn nhập vào khi thêm vào chi tiết nhập

|  |
| --- |
| CREATE TRIGGER UpdateTotalImport\_InsertImportDetails ON ImportDetails  AFTER INSERT  AS  BEGIN  SET NOCOUNT ON;  BEGIN TRY  BEGIN TRANSACTION;    DECLARE @Import\_ID VARCHAR(10);  DECLARE @TotalAmount INT;  -- Lấy Import\_ID từ bảng inserted  SELECT @Import\_ID = Import\_ID FROM inserted;  -- Tính tổng số tiền cho đợt nhập hàng mới  SELECT @TotalAmount = SUM(Quantity \* Unitcost) FROM inserted;  -- Cập nhật tổng số tiền cho đợt nhập hàng trong bảng Imports  UPDATE Imports  SET Total = Total + @TotalAmount  WHERE Import\_ID = @Import\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  -- Nếu có lỗi xảy ra, rollback transaction và in ra thông báo lỗi  IF @@TRANCOUNT > 0  ROLLBACK TRANSACTION;    DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();    RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH;  END  GO |

Trigger cập nhật số lượng sản phẩm khi thêm vào chi tiết nhập

|  |
| --- |
| CREATE TRIGGER UpdateQuantityProduct\_InsertImportDetails ON ImportDetails  AFTER INSERT  AS  BEGIN  SET NOCOUNT ON;  BEGIN TRY  BEGIN TRANSACTION;  DECLARE @table TABLE(Product\_ID VARCHAR(15), Quantity INT);  -- Lưu thông tin từ bảng inserted vào bảng tạm  INSERT INTO @table  SELECT Product\_ID, Quantity FROM inserted;  -- Cập nhật số lượng sản phẩm trong bảng Products  UPDATE Products  SET Products.Quantity = Products.Quantity + T.Quantity  FROM Products  INNER JOIN @table AS T ON Products.Product\_ID = T.Product\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  -- Nếu có lỗi xảy ra, rollback transaction và in ra thông báo lỗi  IF @@TRANCOUNT > 0 –Kiểm tra xem có tran nào hd ko  ROLLBACK TRANSACTION;    DECLARE @ErrorMessage NVARCHAR(4000);  DECLARE @ErrorSeverity INT;  DECLARE @ErrorState INT;  SELECT @ErrorMessage = ERROR\_MESSAGE(),  @ErrorSeverity = ERROR\_SEVERITY(),  @ErrorState = ERROR\_STATE();    RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);  END CATCH;  END |

Trigger cập nhật số lượng sản phẩm khi thêm/xoá chi tiết hóa đơn

|  |
| --- |
| CREATE TRIGGER UpdateQuantityProduct\_IDOrdersDetails ON OrderDetails  FOR INSERT, DELETE  AS  BEGIN  SET NOCOUNT ON;  BEGIN TRY  BEGIN TRANSACTION;  IF EXISTS(SELECT \* FROM inserted)  BEGIN  IF EXISTS (SELECT \* FROM Products p INNER JOIN inserted i ON p.Product\_ID = i.Product\_ID WHERE p.Quantity < i.Quantity)  BEGIN  -- Nếu có sản phẩm không đủ để bán, phát sinh lỗi và rollback transaction  RAISERROR('San pham khong du de ban', 16, 1); --1:state(loi do ng dung gay ra); 16/25:severity  ROLLBACK TRANSACTION;  RETURN;  END  ELSE  BEGIN  -- Cập nhật số lượng sản phẩm sau khi thêm đơn hàng mới  UPDATE p  SET p.Quantity = p.Quantity - i.Quantity  FROM Products p  INNER JOIN inserted i ON p.Product\_ID = i.Product\_ID;  END  END  ELSE  BEGIN  -- Cập nhật số lượng sản phẩm sau khi xóa đơn hàng  UPDATE p  SET p.Quantity = p.Quantity + d.Quantity  FROM Products p  INNER JOIN deleted d ON p.Product\_ID = d.Product\_ID;  END  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  -- Nếu có lỗi xảy ra, rollback transaction và in ra thông báo lỗi  IF @@TRANCOUNT > 0  ROLLBACK TRANSACTION;  THROW;  END CATCH;  END |

Trigger cập nhật trạng thái của nhân viên

|  |
| --- |
| CREATE TRIGGER UpdateActive\_Employee ON Employees  INSTEAD OF DELETE  AS  BEGIN  SET NOCOUNT ON;  BEGIN TRY  BEGIN TRANSACTION;    -- Cập nhật trạng thái Active của nhân viên  UPDATE Employees  SET Active = 0  FROM Employees e  JOIN deleted d ON e.EmployeeID = d.EmployeeID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  -- Nếu có lỗi xảy ra, rollback transaction và in ra thông báo lỗi  IF @@TRANCOUNT > 0  ROLLBACK TRANSACTION;  THROW; --hiện lỗi ra message  END CATCH;  END |

Trigger kiểm tra ngày hợp lệ của Discount khi thêm hoá đơn xuất

|  |
| --- |
| CREATE TRIGGER CheckDateDiscount ON Orders  FOR INSERT  AS  BEGIN  SET NOCOUNT ON;  BEGIN TRY  BEGIN TRANSACTION;  DECLARE @DateOrder DATE;  DECLARE @DateEnd DATE;  DECLARE @DateStart DATE;  DECLARE @Code VARCHAR(10);  -- Lấy ngày đặt hàng và mã khuyến mãi từ dữ liệu được thêm mới hoặc cập nhật  SELECT @DateOrder = OrderDate, @Code = DiscountCode FROM inserted;  -- Lấy ngày bắt đầu và kết thúc của mã khuyến mãi từ bảng Discounts  SELECT @DateStart = StartDay, @DateEnd = EndDay FROM Discounts d WHERE d.DiscountCode = @Code ;  -- Kiểm tra xem ngày đặt hàng có nằm trong khoảng thời gian khuyến mãi không  IF (@DateOrder > @DateEnd OR @DateOrder < @DateStart)  BEGIN  -- Nếu ngày đặt hàng không nằm trong khoảng thời gian khuyến mãi, phát sinh lỗi và rollback transaction  RAISERROR('Ma khuyen mai het han', 16, 1);  ROLLBACK TRANSACTION;  RETURN;  END  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  -- Nếu có lỗi xảy ra, rollback transaction và in ra thông báo lỗi  IF @@TRANCOUNT > 0  ROLLBACK TRANSACTION;  THROW;  END CATCH;  END |

Trigger kiểm tra giá sản phẩm và giá sản phẩm bán ra có phù hợp hay không

|  |
| --- |
| CREATE TRIGGER CheckProductPrice\_OnInsertImportDetails  ON [dbo].[ImportDetails]  AFTER INSERT  AS  BEGIN  SET NOCOUNT ON;  -- Kiểm tra giá tiền sản phẩm từ đơn nhập hàng phải thấp hơn giá bán  IF EXISTS (  SELECT 1  FROM inserted AS i  INNER JOIN Products AS p ON i.Product\_ID = p.Product\_ID  WHERE i.Unitcost > p.UnitPrice  )  BEGIN  -- Nếu có sản phẩm có giá tiền cao hơn giá bán, rollback transaction và in ra thông báo lỗi  RAISERROR ('Giá tiền sản phẩm phải thấp hơn giá bán.', 16, 1);  ROLLBACK TRANSACTION;  END  END |

# **THIẾT KẾ CÁC CHỨC NĂNG**

**3.1 Lớp kết nối và hàm truy vấn cơ sở dữ liệu**

|  |
| --- |
| public class DAL  {  // Connection string to the database  public static SqlConnectionStringBuilder ConnStrBuilder;  public static int count = 0;  // Declaring SqlConnection, SqlCommand, and SqlDataAdapter objects  public SqlConnection conn = null;  public SqlCommand comm = null;  SqlDataAdapter da = null;    // Constructor for the DAL class  public DAL()  {  ConnStrBuilder = new SqlConnectionStringBuilder();  if (count == 0)  {  ConnStrBuilder.DataSource = "CHEESETHANK";  ConnStrBuilder.InitialCatalog = "QuanLyBanHangTheThao";  ConnStrBuilder.IntegratedSecurity = true;  ConnStrBuilder.Encrypt = false;  }  conn = new SqlConnection(ConnStrBuilder.ToString());  comm = conn.CreateCommand();  }  // Method to execute a query and return a DataSet  public DataSet ExecuteQueryDataSet(string strSQL, CommandType ct, params SqlParameter[] p)  {  // Closing the connection if it's already open  if (conn.State == ConnectionState.Open)  conn.Close();  // Opening the connection  conn.Open();  // Setting command text and type  comm.CommandText = strSQL;  comm.CommandType = ct;    // Initializing SqlDataAdapter object  da = new SqlDataAdapter(comm);  DataSet ds = new DataSet();  // Filling the DataSet with data from the query result  da.Fill(ds);  return ds;  }  // Method to execute non-query actions (insert, delete, update, stored procedure) and return success status  public bool MyExecuteNonQuery(string strSQL, CommandType ct, ref string error, params SqlParameter[] param)  {  bool f = false; // Initializing a flag to false  // Closing the connection if it's already open  if (conn.State == ConnectionState.Open)  conn.Close();  // Opening the connection  conn.Open();  // Clearing parameters and setting command text and type  comm.Parameters.Clear();  comm.CommandText = strSQL;  comm.CommandType = ct;  // Adding parameters if provided  if (param != null)  {  foreach (SqlParameter p in param)  {  comm.Parameters.Add(p);  }  }  try  {  // Executing the non-query command  comm.ExecuteNonQuery();  f = true; // Setting the flag to true indicating success  }  catch (SqlException ex)  {  // Catching any SQL exceptions and storing the error message  error = ex.Message;  }  finally  {  // Closing the connection in any case  conn.Close();  }  return f; // Returning the success status  }  public object ExecuteScalar(string strSQL, CommandType ct, params SqlParameter[] param)  {  // Declaring the result variable  object result = null;  try  {  if (conn.State == ConnectionState.Open)  conn.Close();  // Opening the connection  conn.Open();  // Clearing parameters and setting command text and type  comm.CommandText = strSQL;  comm.CommandType = ct;  // Adding parameters if provided  if (param != null)  {  foreach (SqlParameter p in param)  {  comm.Parameters.Add(p);  }  }  // Executing the command and getting the scalar result  result = comm.ExecuteScalar();  }  catch (SqlException ex)  {  // Handling SQL exceptions  Console.WriteLine("SQL Error: " + ex.Message);  }  finally  {  // Closing the connection  conn.Close();  }  // Returning the scalar result  return result;  }  } |

**3.2 Các hàm và thủ tục về Sản Phẩm**

**3.2.1 Thêm Sản phẩm**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertProduct  @Product\_ID VARCHAR(15),  @ProductName NVARCHAR(100),  @UnitPrice INT,  @Quantity INT,  @BrandName NVARCHAR(10),  @CategoryName NVARCHAR(10),  @Pic\_Name VARCHAR(100)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  DECLARE @Brand\_ID VARCHAR(10), @Category\_ID VARCHAR(10), @Pic\_ID INT  SELECT @Brand\_ID = Brand\_ID FROM Brands WHERE BrandName like N'%'+@BrandName+'%';  SELECT @Category\_ID = Category\_ID FROM Categories WHERE CategoryName like N'%'+@CategoryName+'%';  SELECT @Pic\_ID = Picture\_ID FROM PictureProduct WHERE @Pic\_Name like @Pic\_Name;  IF (@Product\_ID = '' OR @ProductName = '' OR @UnitPrice IS NULL OR @Pic\_Name = '' or @CategoryName='' or @BrandName = '')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Products (Product\_ID, ProductName, UnitPrice, Quantity, Brand\_ID, Category\_ID, Picture\_ID)  VALUES (@Product\_ID, @ProductName, @UnitPrice, @Quantity, @Brand\_ID, @Category\_ID, @Pic\_ID);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool TaoSanPham(ref string err, string ma, string ten, int gia, string th, string dm, int sl, string Img)  {  return db.MyExecuteNonQuery("spInsertProduct",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Product\_ID", ma),  new SqlParameter("@ProductName", ten),  new SqlParameter("@UnitPrice", gia),  new SqlParameter("@Quantity", sl),  new SqlParameter("@BrandName", th),  new SqlParameter("@CategoryName", dm),  new SqlParameter("@Pic\_Name", Img)  );  } |

**3.2.2 Cập nhật Sản phẩm**

|  |
| --- |
| CREATE or ALTER PROCEDURE spUpdateProduct  @Product\_ID VARCHAR(15),  @ProductName NVARCHAR(100),  @UnitPrice INT,  @Quantity INT,  @BrandName NVARCHAR(50),  @CategoryName NVARCHAR(50),  @Pic\_ID INT  AS  BEGIN    DECLARE @Brand\_ID VARCHAR(10), @Category\_ID VARCHAR(10)  SELECT @Brand\_ID = Brand\_ID FROM Brands WHERE BrandName = @BrandName;  SELECT @Category\_ID = Category\_ID FROM Categories WHERE CategoryName = @CategoryName;  BEGIN TRANSACTION;  BEGIN TRY  IF (@Product\_ID = '' OR @ProductName = '' OR @UnitPrice IS NULL OR @Pic\_ID = '' or @CategoryName='' or @BrandName = '')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  UPDATE Products  SET ProductName = @ProductName,  UnitPrice = @UnitPrice,  Quantity = @Quantity,  Brand\_ID = @Brand\_ID,  Category\_ID = @Category\_ID,  Picture\_ID = @Pic\_ID  WHERE Product\_ID = @Product\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatSanPham(ref string err, string ma, string ten, int gia, string th, string dm, int sl, int idImg)  {  return db.MyExecuteNonQuery("spUpdateProduct",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Product\_ID", ma),  new SqlParameter("@ProductName", ten),  new SqlParameter("@UnitPrice", gia),  new SqlParameter("@Quantity", sl),  new SqlParameter("@BrandName", th),  new SqlParameter("@CategoryName", dm),  new SqlParameter("@Pic\_ID", idImg)  );  } |

**3.2.3 Tìm Sản phẩm theo danh mục, thương hiệu, tên**

|  |
| --- |
| CREATE FUNCTION Find\_Product  (  @BrandName NVARCHAR(50),  @CategoryName NVARCHAR(50),  @Name NVARCHAR(100)  )  RETURNS TABLE  AS  RETURN  (  SELECT Product\_ID, ProductName, UnitPrice, Quantity  FROM View\_Product  WHERE BrandName LIKE '%' + @BrandName + '%'  AND CategoryName LIKE '%' + @CategoryName + '%'  AND LOWER(ProductName) LIKE '%' + @Name + '%'  ); |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimSanPham(string a, string b, string c)  {  return db.ExecuteQueryDataSet(  "SELECT \* FROM Find\_Product(N'" + a + "',N'" + b + "',N'" + c + "')",  CommandType.Text, null);  } |

**3.3 Các hàm và thủ tục về Khách Hàng**

**3.3.1 Thêm Khách Hàng**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertCustomer  @PhoneNumber VARCHAR(12),  @NameCustomer NVARCHAR(50),  @Birthday DATE,  @Gender CHAR(3),  @Point INT  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@NameCustomer ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Customers (PhoneNumber, NameCustomer, Birthday, Gender, Point)  VALUES (@PhoneNumber, @NameCustomer, @Birthday, @Gender, @Point);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemKhachHang(ref string err, string sdt, string name, DateTime birthday, string gender, int point)  {  return db.MyExecuteNonQuery("spInsertCustomer",  CommandType.StoredProcedure, ref err,  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@NameCustomer", name),  new SqlParameter("@Birthday", birthday),  new SqlParameter("@Gender", gender),  new SqlParameter("@Point", point)  );  } |

**3.3.2 Cập nhật Khách Hàng**

|  |
| --- |
| CREATE or ALTER PROCEDURE spUpdateCustomer  @PhoneNumber VARCHAR(12),  @NameCustomer NVARCHAR(50),  @Birthday DATE,  @Gender CHAR(3),  @Point INT  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@PhoneNumber ='' or @NameCustomer ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  UPDATE Customers  SET NameCustomer = @NameCustomer,  Birthday = @Birthday,  Gender = @Gender,  Point = @Point  WHERE PhoneNumber = @PhoneNumber;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatKhachHang(ref string err, string sdt, string name, DateTime birthday, string gender, int point)  {  return db.MyExecuteNonQuery("spUpdateCustomer",  CommandType.StoredProcedure, ref err,  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@NameCustomer", name),  new SqlParameter("@Birthday", birthday),  new SqlParameter("@Gender", gender),  new SqlParameter("@Point", point)  );  } |

**3.3.3 Tìm Khách Hàng theo số điện thoại và tên**

|  |
| --- |
| CREATE FUNCTION Find\_Customer  (  @Phone VARCHAR(12),  @Name NVARCHAR(50)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM View\_Customer  WHERE LOWER(NameCustomer) LIKE '%' + @Name + '%'  AND PhoneNumber LIKE '%' + @Phone + '%'  );  go |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimKhachHang(string Phone, string Name)  {  return db.ExecuteQueryDataSet("SELECT \* FROM Find\_Customer('" + Phone + "',N'" + Name + "')",  CommandType.Text, null);  } |

**3.3.4 Tìm các sản phẩm Khách Hàng đã mua**

|  |
| --- |
| CREATE or alter FUNCTION ProductOfCustomer  (  @Phone VARCHAR(12)  )  RETURNS TABLE  AS  RETURN  (  SELECT od.Product\_ID, p.ProductName, SUM(od.Quantity) AS Quantity, p.UnitPrice  FROM OrderDetails od  INNER JOIN Orders o ON od.Order\_ID = o.Order\_ID  INNER JOIN Products p ON od.Product\_ID = p.Product\_ID  WHERE o.PhoneNumber = @Phone  GROUP BY od.Product\_ID, p.ProductName, p.UnitPrice  ); |

Thực hiện trên C#

|  |
| --- |
| public DataSet SPcuaKhachHang(string Phone)  {  return db.ExecuteQueryDataSet("SELECT \* FROM ProductOfCustomer('" + Phone + "')",  CommandType.Text, null);  } |

**3.4 Các hàm và thủ tục về Hóa Đơn**

**3.4.1 Thêm Hóa Đơn**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertOrder  @Order\_ID VARCHAR(15),  @PhoneNumber VARCHAR(12),  @EmployeeID VARCHAR(5),  @OrderDate DATE,  @Total INT,  @DiscountCode VARCHAR(10)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@PhoneNumber ='' )  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  IF @DiscountCode = ''  SET @DiscountCode = NULL;  INSERT INTO Orders (Order\_ID, PhoneNumber, EmployeeID, OrderDate, Total, DiscountCode)  VALUES (@Order\_ID, @PhoneNumber, @EmployeeID, @OrderDate, @Total, @DiscountCode);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemHoaDon(ref string err, string order\_ID, string sdt,string nv,  DateTime orderdate, int Total,string magiam)  {  return db.MyExecuteNonQuery("spInsertOrder",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Order\_ID", order\_ID),  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@EmployeeID", nv),  new SqlParameter("@OrderDate", orderdate),  new SqlParameter("@Total", Total),  new SqlParameter("@DiscountCode", magiam));  } |

**3.4.2 Xoá Hóa Đơn**

|  |
| --- |
| CREATE or ALTER PROCEDURE spDeleteOrder  @Order\_ID VARCHAR(15)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  DELETE FROM Orders WHERE Order\_ID = @Order\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatHoaDon(ref string err, string order\_ID, string sdt, string nv,  DateTime orderdate, int Total, string magiam)  {  return db.MyExecuteNonQuery("spUpdateOrder",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Order\_ID", order\_ID),  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@EmployeeID", nv),  new SqlParameter("@OrderDate", orderdate),  new SqlParameter("@Total", Total),  new SqlParameter("@DiscountCode", magiam));  } |

**3.4.3 Tìm Hóa Đơn theo mã hóa đơn và ngày thanh toán**

|  |
| --- |
| CREATE FUNCTION Find\_Order  (  @ID VARCHAR(15),  @date VARCHAR(15)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM BILLS\_VIEW  WHERE LOWER(Order\_ID) LIKE '%' + @ID + '%'  AND CONVERT(DATE, OrderDate, 103) LIKE '%' + @date + '%'  );  go |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimHoaDon(string HD,string date)  {  return db.ExecuteQueryDataSet(  "select \* from Find\_Order('"+HD+"','"+date+"')", CommandType.Text, null);  } |

**3.4.4 Tìm các sản phẩm của Hóa Đơn**

|  |
| --- |
| CREATE FUNCTION ProductOfOrder  (  @ID VARCHAR(15)  )  RETURNS TABLE  AS  RETURN  (  SELECT od.Product\_ID, p.ProductName, od.Quantity, p.UnitPrice  FROM OrderDetails od  INNER JOIN Products p ON od.Product\_ID = p.Product\_ID  WHERE od.Order\_ID = @ID  ); |

Thực hiện trên C#

|  |
| --- |
| public DataSet SPCuaHoaDon(string HD )  {  return db.ExecuteQueryDataSet(  "select \* from ProductOfOrder('" + HD + "')", CommandType.Text, null);  } |

**3.5 Các hàm và thủ tục về Chi Tiết Hóa Đơn**

**3.5.1 Thêm Chi Tiết Hóa Đơn**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertOrderDetail  @Order\_ID VARCHAR(15),  @Product\_ID VARCHAR(15),  @Quantity INT  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@Product\_ID ='' )  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO OrderDetails (Order\_ID, Product\_ID, Quantity)  VALUES (@Order\_ID, @Product\_ID, @Quantity);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemChiTietHoaDon(ref string err, string Order\_ID, string Product\_ID,  int Quantity)  {  return db.MyExecuteNonQuery("spInsertOrderDetail",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Order\_ID", Order\_ID),  new SqlParameter("@Product\_ID", Product\_ID),  new SqlParameter("@Quantity", Quantity));  } |

**3.6 Các hàm và thủ tục về Nhân Viên**

**3.6.1 Thêm Nhân Viên**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertEmployee  @EmployeeID VARCHAR(5),  @NameEmployee NVARCHAR(50),  @Birthday DATE,  @Gender CHAR(3),  @AddressEmployee NVARCHAR(100),  @PhoneNumber VARCHAR(12),  @RoleEmployee NVARCHAR(50),  @Active CHAR(1),  @PassWordAccount VARCHAR(10)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@EmployeeID ='' or @NameEmployee ='' or @AddressEmployee ='' or @PhoneNumber ='' or @RoleEmployee ='' or @PassWordAccount ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Employees (EmployeeID, NameEmployee, Birthday, Gender, AddressEmployee, PhoneNumber, RoleEmployee, Active, PassWordAccount)  VALUES (@EmployeeID, @NameEmployee, @Birthday, @Gender, @AddressEmployee, @PhoneNumber, @RoleEmployee, @Active, @PassWordAccount);  DECLARE @sqlString NVARCHAR(2000)  -- Tạo tài khoản login cho nhân viên, tên người dùng và mật khẩu là tài khoản được tạo trên bảng Account  SET @sqlString = 'CREATE LOGIN [' + @EmployeeID + '] WITH PASSWORD=''' +@PassWordAccount + ''', DEFAULT\_DATABASE=[QuanLyBanHangTheThao], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF'  EXEC (@sqlString)  -- Tạo tài khoản người dùng đối với nhân viên đó trên database (tên người dùng trùng với tên login)  SET @sqlString = 'CREATE USER ' +@EmployeeID + ' FOR LOGIN ' +@EmployeeID;  EXEC (@sqlString)  -- Thêm người dùng vào vai trò quyền tương ứng (Staff hoặc Manager(sysadmin))  IF (@RoleEmployee = N'Bán hàng')  SET @sqlString = 'ALTER ROLE BanHang ADD MEMBER ' + @EmployeeID;  ELSE  SET @sqlString = 'ALTER SERVER ROLE sysadmin ADD MEMBER ' + @EmployeeID;  EXEC (@sqlString)  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemNhanVien(ref string err, string id, string name,DateTime birthday, string gender, string address,string sdt,string role,int active, string password )  {  return db.MyExecuteNonQuery("spInsertEmployee",  CommandType.StoredProcedure, ref err,  new SqlParameter("@EmployeeID", id),  new SqlParameter("@NameEmployee", name),  new SqlParameter("@Birthday", birthday),  new SqlParameter("@Gender", gender),  new SqlParameter("@AddressEmployee", address),  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@RoleEmployee", role),  new SqlParameter("@Active", active),  new SqlParameter("@PassWordAccount", password)  );  } |

**3.6.2 Cập nhật nhân viên**

|  |
| --- |
| CREATE or ALTER PROCEDURE spUpdateEmployee  @EmployeeID VARCHAR(5),  @NameEmployee NVARCHAR(50),  @Birthday DATE,  @Gender CHAR(3),  @AddressEmployee NVARCHAR(100),  @PhoneNumber VARCHAR(12),  @RoleEmployee NVARCHAR(50),  @Active CHAR(1),  @PassWordAccount VARCHAR(15)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@EmployeeID ='' or @NameEmployee ='' or @AddressEmployee ='' or @PhoneNumber ='' or @RoleEmployee ='' or @PassWordAccount ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  UPDATE Employees  SET NameEmployee = @NameEmployee,  Birthday = @Birthday,  Gender = @Gender,  AddressEmployee = @AddressEmployee,  PhoneNumber = @PhoneNumber,  RoleEmployee = @RoleEmployee,  Active = @Active,  PassWordAccount = @PassWordAccount  WHERE EmployeeID = @EmployeeID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatNhanVien(ref string err, string id, string name, DateTime birthday, string gender, string address, string sdt, string role, int active, string password)  {  return db.MyExecuteNonQuery("spUpdateEmployee",  CommandType.StoredProcedure, ref err,  new SqlParameter("@EmployeeID", id),  new SqlParameter("@NameEmployee", name),  new SqlParameter("@Birthday", birthday),  new SqlParameter("@Gender", gender),  new SqlParameter("@AddressEmployee", address),  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@RoleEmployee", role),  new SqlParameter("@Active", active),  new SqlParameter("@PassWordAccount", password)  );  } |

**3.6.3 Xóa nhân viên**

|  |
| --- |
| CREATE or ALTER PROCEDURE spDeleteEmployee  @EmployeeID VARCHAR(5)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  DELETE FROM Employees WHERE EmployeeID = @EmployeeID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool XoaNhanVien(ref string err, string id)  {  return db.MyExecuteNonQuery("spDeleteEmployee",  CommandType.StoredProcedure, ref err,  new SqlParameter("@EmployeeID", id)  );  } |

**3.6.4 Tìm Nhân Viên theo mã nhân viên và tên**

|  |
| --- |
| CREATE FUNCTION Find\_Employee  (  @ID VARCHAR(12),  @Name NVARCHAR(50)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM EMPLOYEES\_ACTIVE\_VIEW  WHERE LOWER(NameEmployee) LIKE '%' + @Name + '%'  AND EmployeeID LIKE '%' + @ID + '%'  );  go |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimNhanVien(string ID,string name)  {  return db.ExecuteQueryDataSet(  "select \* from Find\_Employee(N'"+ID+"',N'"+name+"')", CommandType.Text, null);  } );  } |

**3.7 Các hàm và thủ tục về Nhà Cung Cấp**

**3.7.1 Thêm Nhà Cung Cấp**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertSupplier  @Supplier\_ID VARCHAR(10),  @CompanyName NVARCHAR(30),  @PhoneNumber VARCHAR(12),  @AddressSupplier NVARCHAR(100),  @Email VARCHAR(50)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF (@Supplier\_ID = '' OR @CompanyName = '' OR @PhoneNumber ='' or @AddressSupplier='' or @Email = '')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Suppliers (Supplier\_ID, CompanyName, PhoneNumber, AddressSupplier, Email)  VALUES (@Supplier\_ID, @CompanyName, @PhoneNumber, @AddressSupplier, @Email);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemNhaCungCap(ref string err, string Supplier\_ID, string CompanyName,  string PhoneNumber, string AddressSupplier, string Email)  {  return db.MyExecuteNonQuery("spInsertSupplier",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Supplier\_ID", Supplier\_ID),  new SqlParameter("@CompanyName", CompanyName),  new SqlParameter("@PhoneNumber", PhoneNumber),  new SqlParameter("@AddressSupplier", AddressSupplier),  new SqlParameter("@Email", Email));  } |

**3.7.2 Cập nhật Nhà Cung Cấp**

|  |
| --- |
| CREATE or ALTER PROCEDURE spUpdateSupplier  @Supplier\_ID VARCHAR(10),  @CompanyName NVARCHAR(30),  @PhoneNumber VARCHAR(12),  @AddressSupplier NVARCHAR(100),  @Email VARCHAR(50)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF (@Supplier\_ID = '' OR @CompanyName = '' OR @PhoneNumber = '' or @AddressSupplier='' or @Email = '')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  UPDATE Suppliers  SET CompanyName = @CompanyName,  PhoneNumber = @PhoneNumber,  AddressSupplier = @AddressSupplier,  Email = @Email  WHERE Supplier\_ID = @Supplier\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatNhaCungCap(ref string err, string Supplier\_ID, string CompanyName,  string PhoneNumber, string AddressSupplier, string Email)  {  return db.MyExecuteNonQuery("spUpdateSupplier",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Supplier\_ID", Supplier\_ID),  new SqlParameter("@CompanyName", CompanyName),  new SqlParameter("@PhoneNumber", PhoneNumber),  new SqlParameter("@AddressSupplier", AddressSupplier),  new SqlParameter("@Email", Email));  } |

**3.7.3 Tìm Nhà Cung Cấp theo mã nhà cung cấp và tên**

|  |
| --- |
| CREATE FUNCTION Find\_Supplier  (  @ID VARCHAR(10),  @Name NVARCHAR(30)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM SUPPLIER\_VIEW  WHERE CompanyName LIKE '%' + @Name + '%'  AND Supplier\_ID LIKE '%' + @ID + '%'  );  go |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimNhaCungCap(string ID, string name)  {  return db.ExecuteQueryDataSet(  "select \* from Find\_Supplier('" + ID + "',N'" + name + "')", CommandType.Text, null);  } |

**3.7.4 Tìm các sản phẩm mà Nhà Cung Cấp cung cấp**

|  |
| --- |
| CREATE FUNCTION ProductOfSupplier  (  @ID VARCHAR(10)  )  RETURNS TABLE  AS  RETURN  (  SELECT id.Product\_ID, p.ProductName, SUM(id.Quantity) AS Quantity, id.Unitcost  FROM ImportDetails id  INNER JOIN Products p ON id.Product\_ID = p.Product\_ID  INNER JOIN Imports i ON id.Import\_ID = i.Import\_ID  WHERE i.Supplier\_ID = @ID  GROUP BY id.Product\_ID, p.ProductName, id.Unitcost  ); |

Thực hiện trên C#

|  |
| --- |
| public DataSet SPCuaNhaCungCap(string ID)  {  return db.ExecuteQueryDataSet(  "select \* from ProductOfSupplier('" + ID + "')", CommandType.Text, null);  } |

**3.8 Các hàm và thủ tục về Biên Lai**

**3.8.1 Thêm Biên Lai**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertImport  @Import\_ID VARCHAR(10),  @Supplier\_ID VARCHAR(10),  @ImportDay DATE,  @Total INT  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF( @Supplier\_ID ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Imports (Import\_ID, Supplier\_ID, ImportDay, Total)  VALUES (@Import\_ID, @Supplier\_ID, @ImportDay, @Total);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemBienLai(ref string err, string Import\_ID, string Supplier\_ID,  DateTime ImportDay, int Total)  {  return db.MyExecuteNonQuery("spInsertImport",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Import\_ID", Import\_ID),  new SqlParameter("@Supplier\_ID", Supplier\_ID),  new SqlParameter("@ImportDay", ImportDay),  new SqlParameter("@Total", Total));  } |

**3.8.2 Cập nhật Biên Lai**

|  |
| --- |
| CREATE or ALTER PROCEDURE spDeleteImport  @Import\_ID VARCHAR(10)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  DELETE FROM Imports WHERE Import\_ID = @Import\_ID;  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool CapNhatBienLai(ref string err, string Import\_ID, string Supplier\_ID,  DateTime ImportDay, int Total)  {  return db.MyExecuteNonQuery("spInsertImport",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Import\_ID", Import\_ID),  new SqlParameter("@Supplier\_ID", Supplier\_ID),  new SqlParameter("@ImportDay", ImportDay),  new SqlParameter("@Total", Total));  } |

**3.8.3 Tìm Biên Lai theo mã biên lai và ngày thanh toán**

|  |
| --- |
| CREATE FUNCTION Find\_Import  (  @ID VARCHAR(15),  @date VARCHAR(15)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM IMPORTS\_VIEW  WHERE LOWER(Import\_ID) LIKE '%' + @ID + '%'  AND CONVERT(DATE, ImportDay, 103) LIKE '%' + @date + '%'  );  go |

Thực hiện trên C#

|  |
| --- |
| public DataSet TimBienLai(string HD, string date)  {  return db.ExecuteQueryDataSet(  "select \* from Find\_Import('" + HD + "','" + date + "')", CommandType.Text, null);  } |

**3.8.4 Tìm các sản phẩm trong Biên Lai**

|  |
| --- |
| CREATE FUNCTION ProductOfImport  (  @ID VARCHAR(15)  )  RETURNS TABLE  AS  RETURN  (  SELECT id.Product\_ID, p.ProductName, id.Quantity, id.Unitcost  FROM ImportDetails id  INNER JOIN Products p ON id.Product\_ID = p.Product\_ID  WHERE id.Import\_ID = @ID  ); |

Thực hiện trên C#

|  |
| --- |
| public DataSet SPCuaBienLai(string HD)  {  return db.ExecuteQueryDataSet(  "select \* from ProductOfImport('" + HD + "')", CommandType.Text, null);  } |

**3.9 Các hàm và thủ tục về Chi Tiết Biên Lai**

**3.9.1 Thêm Chi Tiết Biên Lai**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertImportDetail  @Import\_ID VARCHAR(10),  @Product\_ID VARCHAR(15),  @Quantity INT,  @Unitcost INT  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF( @Product\_ID ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO ImportDetails (Import\_ID, Product\_ID, Quantity, Unitcost)  VALUES (@Import\_ID, @Product\_ID, @Quantity, @Unitcost);  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemChiTietBienLai(ref string err, string Import\_ID, string Product\_ID,  int Quantity, int Unitcost)  {  return db.MyExecuteNonQuery("spInsertImportDetail",  CommandType.StoredProcedure, ref err,  new SqlParameter("@Import\_ID", Import\_ID),  new SqlParameter("@Product\_ID", Product\_ID),  new SqlParameter("@Quantity", Quantity),  new SqlParameter("@Unitcost", Unitcost));  } |

**3.10 Các hàm và thủ tục về Doanh Thu**

**3.10.1 Tính Doanh Thu theo ngày tùy chọn**

|  |
| --- |
| CREATE FUNCTION TotalRevenue  (  @fromDate Date,  @toDate Date  )  RETURNS TABLE  AS  RETURN  (  select OrderDate, SUM(Total) as Total From Orders where OrderDate between @fromDate and @toDate group by OrderDate  ); |

Thực hiện trên C#

|  |
| --- |
| // Retrieving total revenue by date  db.comm.CommandText = "SELECT \* FROM TotalRevenue(@fromDate,@todate)";  db.comm.Parameters.Add("@fromDate", System.Data.SqlDbType.Date).Value = startDate;  db.comm.Parameters.Add("@todate", System.Data.SqlDbType.Date).Value = endDate;  var reader = db.comm.ExecuteReader();  db.comm.Parameters.Clear();  var resultTable = new List<KeyValuePair<DateTime, int>>(); // Initializing a list to hold results  while (reader.Read())  {  resultTable.Add(new KeyValuePair<DateTime, int>((DateTime)reader[0], (int)reader[1])); // Adding results to the list  TotalRevenue += (int)reader[1]; // Calculating total revenue  }  reader.Close(); |

**3.11 Các hàm và thủ tục về Lợi Nhuận**

**3.11.1 Tính Lợi Nhuận theo ngày tùy chọn**

|  |
| --- |
| CREATE FUNCTION TotalProfit  (  @fromDate Date,  @toDate Date  )  RETURNS TABLE  AS  RETURN  ( -- CAST: ép kiểu sang DATE  SELECT CAST(o.OrderDate AS DATE) AS OrderDate, SUM((p.UnitPrice - id.Unitcost) \* od.Quantity) AS TotalProfit  FROM OrderDetails od  JOIN Products p ON od.Product\_ID = p.Product\_ID  JOIN ImportDetails id ON od.Product\_ID = id.Product\_ID  JOIN Orders o ON od.Order\_ID = o.Order\_ID  WHERE o.OrderDate BETWEEN @fromDate AND @toDate  GROUP BY CAST(o.OrderDate AS DATE)  ); |

Thực hiện trên C#

|  |
| --- |
| // Retrieving total profit by date  db.comm.CommandText = @"SELECT \* FROM TotalProfit(@fromDate,@todate)";  db.comm.Parameters.Add("@fromDate", System.Data.SqlDbType.Date).Value = startDate;  db.comm.Parameters.Add("@todate", System.Data.SqlDbType.Date).Value = endDate;  var reader1 = db.comm.ExecuteReader();  db.comm.Parameters.Clear();  while (reader1.Read())  {  decimal unitProfit = Convert.ToDecimal(reader1[1]);  TotalProfit += unitProfit; // Calculating total profit  }  reader1.Close(); |

# **Chương 4: TẠO USER VÀ PHÂN QUYỀN**

**4.1 Tạo Role**

|  |
| --- |
| CREATE ROLE BanHang;  GO  GRANT SELECT, REFERENCES ON Employees TO BanHang;  GRANT SELECT, INSERT, UPDATE, REFERENCES ON Customers TO BanHang;  GRANT SELECT, INSERT, REFERENCES ON Orders TO BanHang;  GRANT SELECT, INSERT, REFERENCES ON OrderDetails TO BanHang;  GRANT SELECT, INSERT, REFERENCES ON Products TO BanHang;  GRANT SELECT, INSERT, REFERENCES ON Brands TO BanHang;  GRANT SELECT, INSERT, REFERENCES ON Categories TO BanHang;  GRANT SELECT, REFERENCES ON Suppliers TO BanHang;  -- Cấp quyền EXECUTE cho vai trò BanHang  GRANT EXECUTE TO BanHang;  DENY EXECUTE ON spDeleteEmployee TO BanHang;  DENY EXECUTE ON spInsertEmployee TO BanHang;  DENY EXECUTE ON spDeleteOrder TO BanHang;  DENY EXECUTE ON spInsertSupplier TO BanHang;  DENY EXECUTE ON spUpdateSupplier TO BanHang;  DENY EXECUTE ON spInsertImportDetail TO BanHang;  -- Cấp quyền SELECT cho các hàm  GRANT SELECT ON Find\_Product TO BanHang;  GRANT SELECT ON Find\_Customer TO BanHang;  GRANT SELECT ON ProductOfCustomer TO BanHang;  GRANT SELECT ON Find\_Order TO BanHang;  GRANT SELECT ON ProductOfOrder TO BanHang;  GRANT SELECT ON Find\_Employee TO BanHang;  GRANT SELECT ON TotalRevenue TO BanHang;  GRANT SELECT ON TotalProfit TO BanHang;  GRANT SELECT ON Top5Product TO BanHang;  GRANT SELECT ON LoginCSDL TO BanHang;  DENY SELECT ON Find\_Supplier TO BanHang;  DENY SELECT ON ProductOfSupplier TO BanHang;  DENY SELECT ON Find\_Import TO BanHang;  DENY SELECT ON ProductOfImport TO BanHang;  -- Cấp quyền SELECT cho các view trừ IMPORTS\_VIEW và SUPPLIER\_VIEW  GRANT SELECT ON View\_Product TO BanHang;  GRANT SELECT ON View\_Customer TO BanHang;  GRANT SELECT ON EMPLOYEES\_ACTIVE\_VIEW TO BanHang;  DENY SELECT ON EMPLOYEES\_All\_VIEW TO BanHang;  GRANT SELECT ON BILLS\_VIEW TO BanHang; |

**4.2 Tạo user**

|  |
| --- |
| CREATE or ALTER PROCEDURE spInsertEmployee  @EmployeeID VARCHAR(5),  @NameEmployee NVARCHAR(50),  @Birthday DATE,  @Gender CHAR(3),  @AddressEmployee NVARCHAR(100),  @PhoneNumber VARCHAR(12),  @RoleEmployee NVARCHAR(50),  @Active CHAR(1),  @PassWordAccount VARCHAR(10)  AS  BEGIN  BEGIN TRANSACTION;  BEGIN TRY  IF(@EmployeeID ='' or @NameEmployee ='' or @AddressEmployee ='' or @PhoneNumber ='' or @RoleEmployee ='' or @PassWordAccount ='')  BEGIN  RAISERROR(N'Vui lòng nhập chính xác, đầy đủ thông tin', 16, 1)  return  END  INSERT INTO Employees (EmployeeID, NameEmployee, Birthday, Gender, AddressEmployee, PhoneNumber, RoleEmployee, Active, PassWordAccount)  VALUES (@EmployeeID, @NameEmployee, @Birthday, @Gender, @AddressEmployee, @PhoneNumber, @RoleEmployee, @Active, @PassWordAccount);  DECLARE @sqlString NVARCHAR(2000)  -- Tạo tài khoản login cho nhân viên, tên người dùng và mật khẩu là tài khoản được tạo trên bảng Account  SET @sqlString = 'CREATE LOGIN [' + @EmployeeID + '] WITH PASSWORD=''' +@PassWordAccount + ''', DEFAULT\_DATABASE=[QuanLyBanHangTheThao], CHECK\_EXPIRATION=OFF, CHECK\_POLICY=OFF'  EXEC (@sqlString)  -- Tạo tài khoản người dùng đối với nhân viên đó trên database (tên người dùng trùng với tên login)  SET @sqlString = 'CREATE USER ' +@EmployeeID + ' FOR LOGIN ' +@EmployeeID;  EXEC (@sqlString)  -- Thêm người dùng vào vai trò quyền tương ứng (Staff hoặc Manager(sysadmin))  IF (@RoleEmployee = N'Bán hàng')  SET @sqlString = 'ALTER ROLE BanHang ADD MEMBER ' + @EmployeeID;  ELSE  SET @sqlString = 'ALTER SERVER ROLE sysadmin ADD MEMBER ' + @EmployeeID;  EXEC (@sqlString)  COMMIT TRANSACTION;  END TRY  BEGIN CATCH  DECLARE @err NVARCHAR(MAX);  SELECT @err = N'Lỗi: ' + ERROR\_MESSAGE();  RAISERROR(@err, 16, 1)  ROLLBACK TRANSACTION;  END CATCH  END; |

Thực hiện trên C#

|  |
| --- |
| public bool ThemNhanVien(ref string err, string id, string name,DateTime birthday, string gender, string address,string sdt,string role,int active, string password )  {  return db.MyExecuteNonQuery("spInsertEmployee",  CommandType.StoredProcedure, ref err,  new SqlParameter("@EmployeeID", id),  new SqlParameter("@NameEmployee", name),  new SqlParameter("@Birthday", birthday),  new SqlParameter("@Gender", gender),  new SqlParameter("@AddressEmployee", address),  new SqlParameter("@PhoneNumber", sdt),  new SqlParameter("@RoleEmployee", role),  new SqlParameter("@Active", active),  new SqlParameter("@PassWordAccount", password)  );  } |

**4.3 Kiểm tra user**

|  |
| --- |
| public DAL()  {  ConnStrBuilder = new SqlConnectionStringBuilder();  if (count == 0)  {  ConnStrBuilder.DataSource = "CHEESETHANK";  ConnStrBuilder.InitialCatalog = "QuanLyBanHangTheThao";  ConnStrBuilder.IntegratedSecurity = true;  ConnStrBuilder.Encrypt = false;  }  conn = new SqlConnection(ConnStrBuilder.ToString());  comm = conn.CreateCommand();  } |
| bool isAuthenticated = dbnv.CheckLogin(textBox1.Text, textBox2.Text);  if (isAuthenticated)  {  DAL.ConnStrBuilder.IntegratedSecurity = false;  DAL.ConnStrBuilder.UserID = textBox1.Text;  DAL.ConnStrBuilder.Password = textBox2.Text;  DAL.count = 1;  this.Hide();  MainForm form2 = new MainForm(textBox1.Text);  form2.ShowDialog();  }  else  {  MessageBox.Show("Incorrect username or password", "Authentication Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);  } |

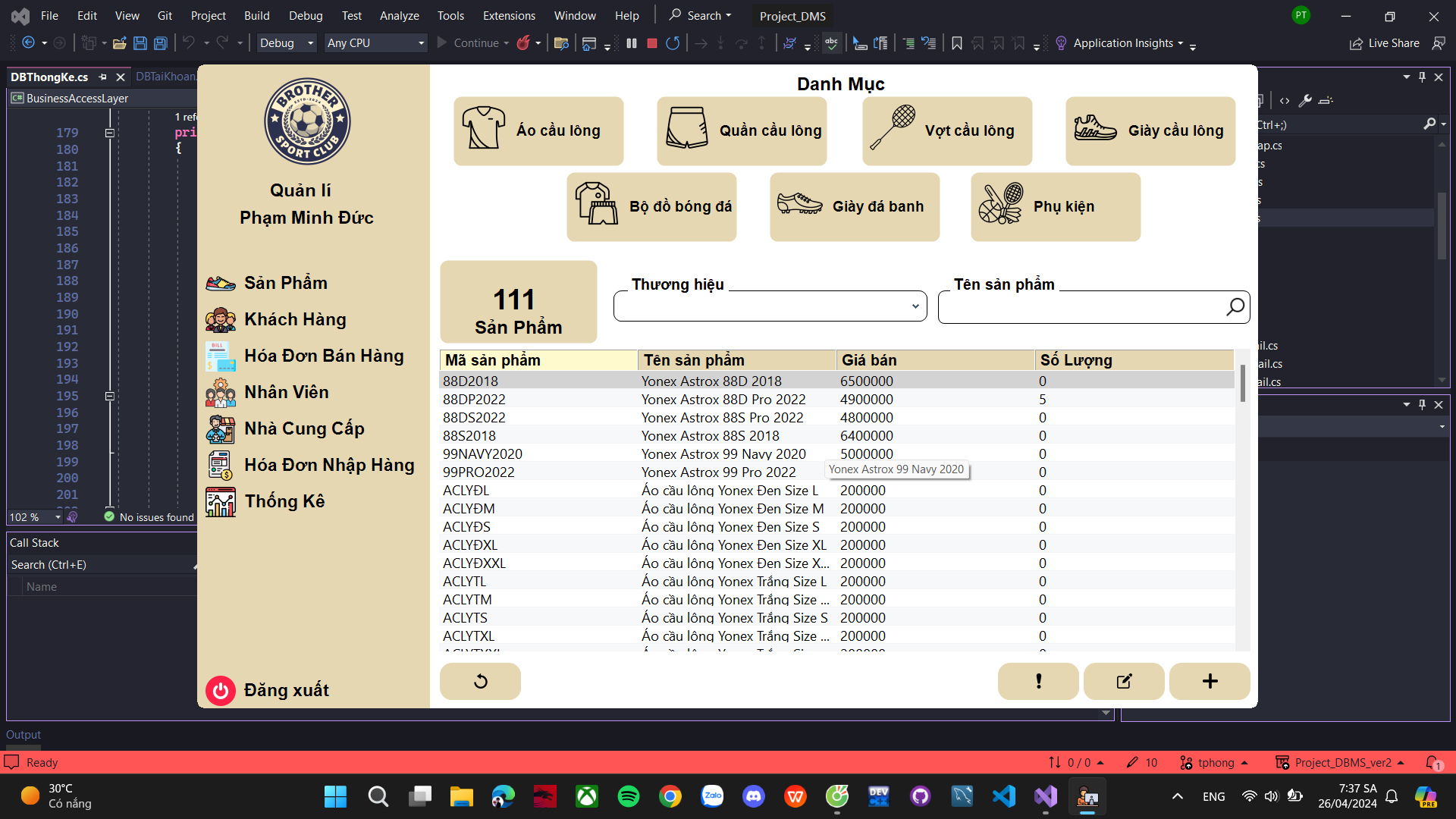
# **Chương 5 :THIẾT KẾ GIAO DIỆN HỆ THỐNG**

**5.1 Giao diện quản lý**

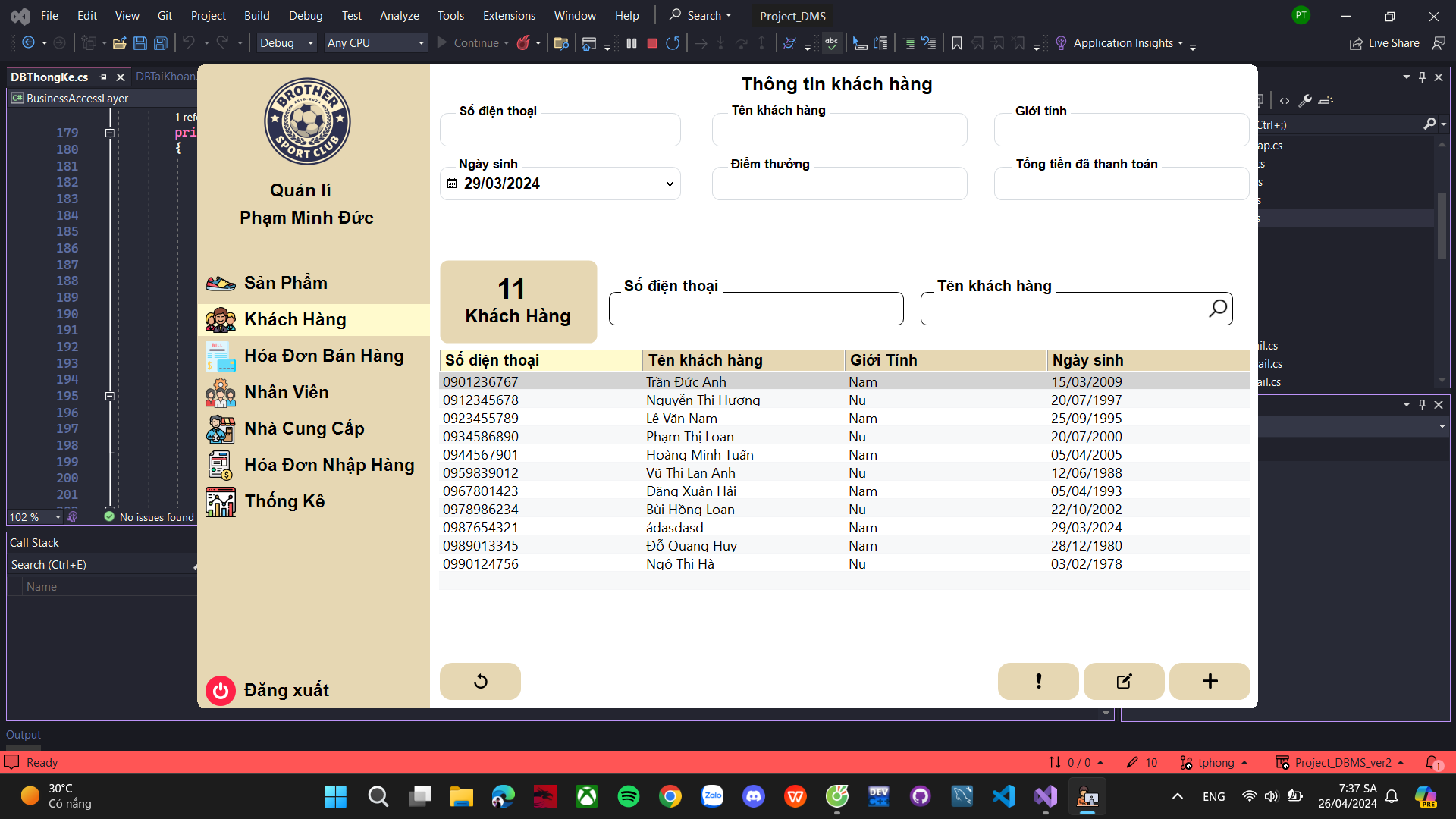
**5.1.1 Giao diện đăng nhập**

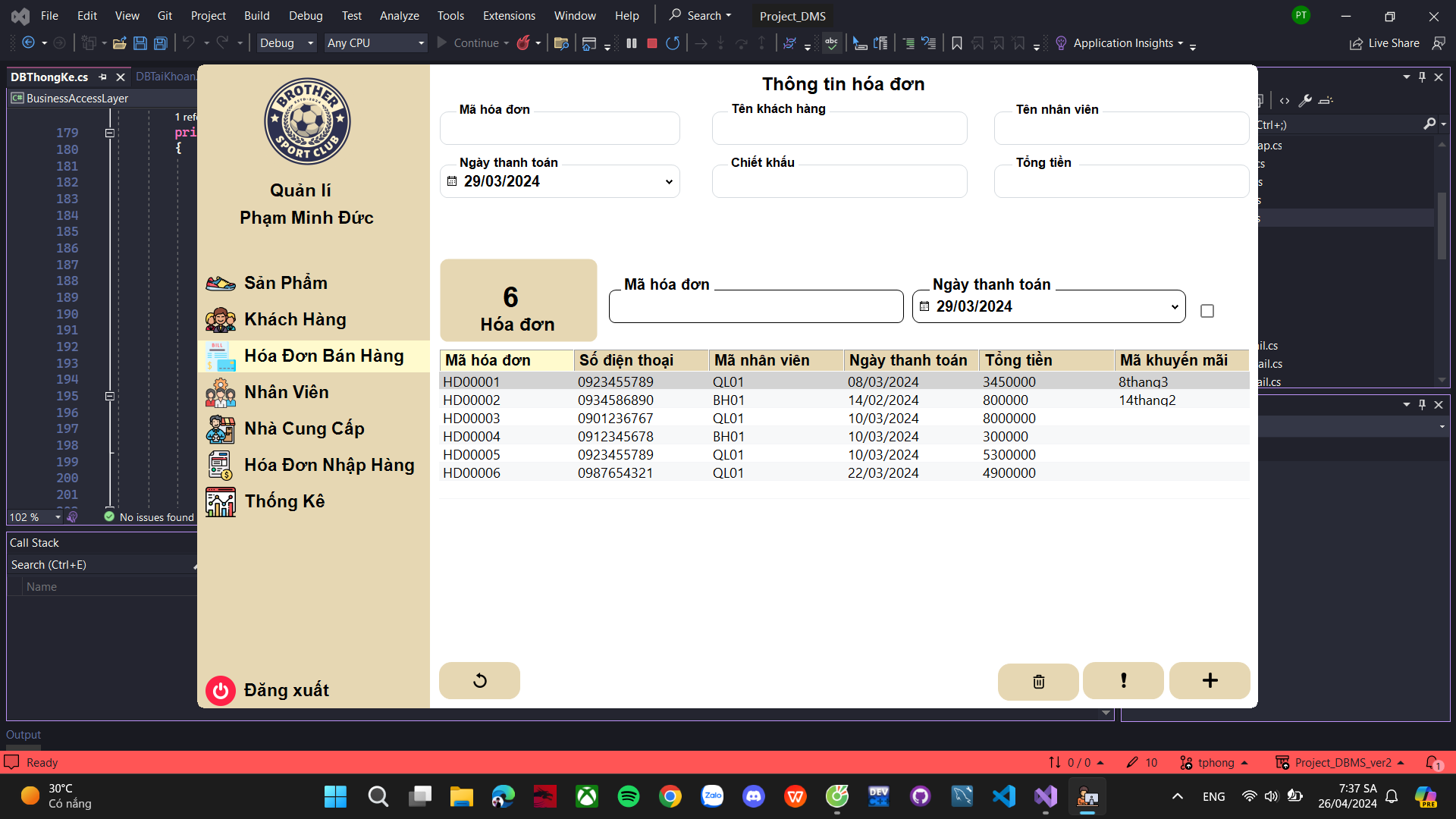
****

**5.1.2 Giao diện sản phẩm**

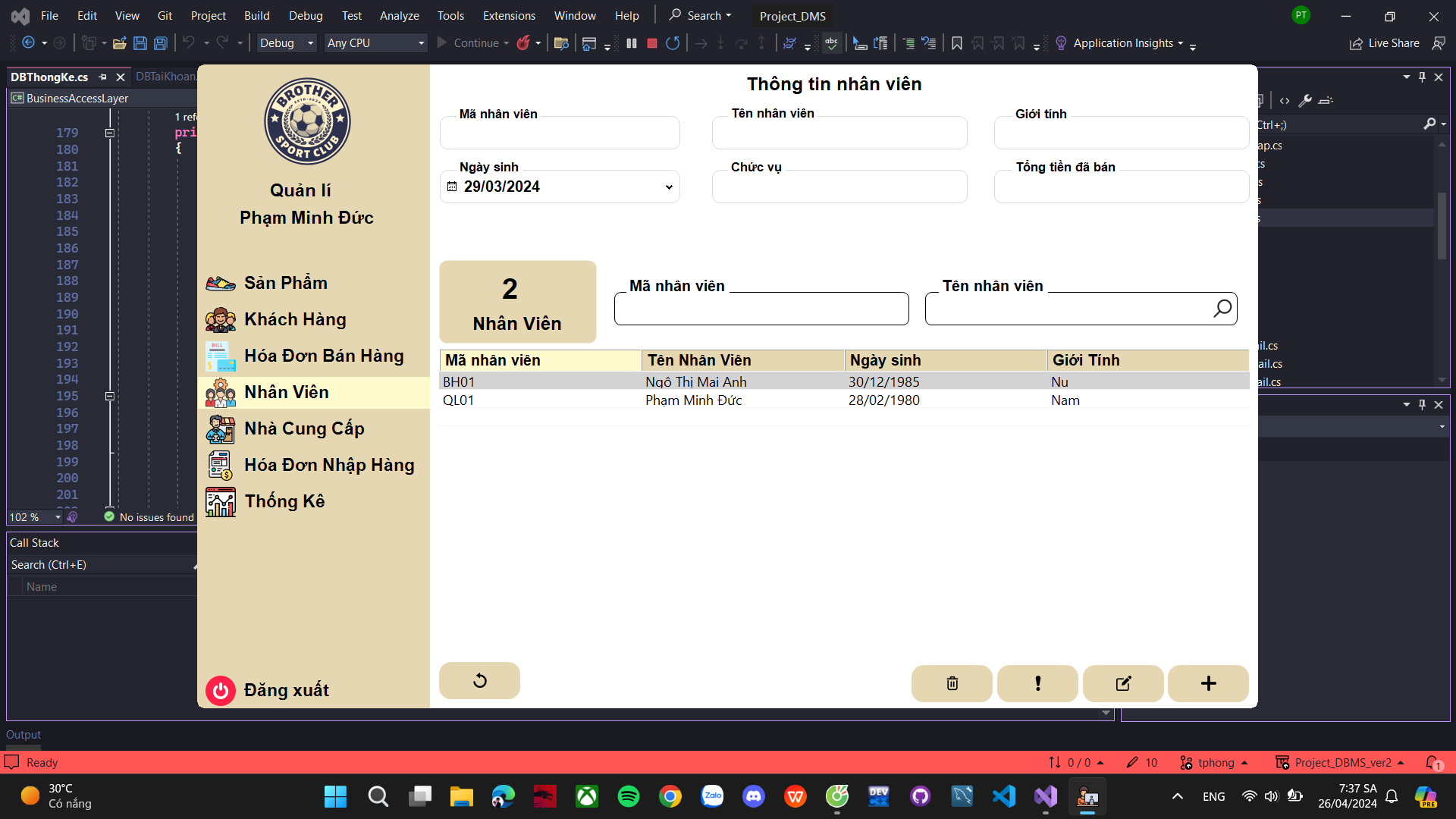
****

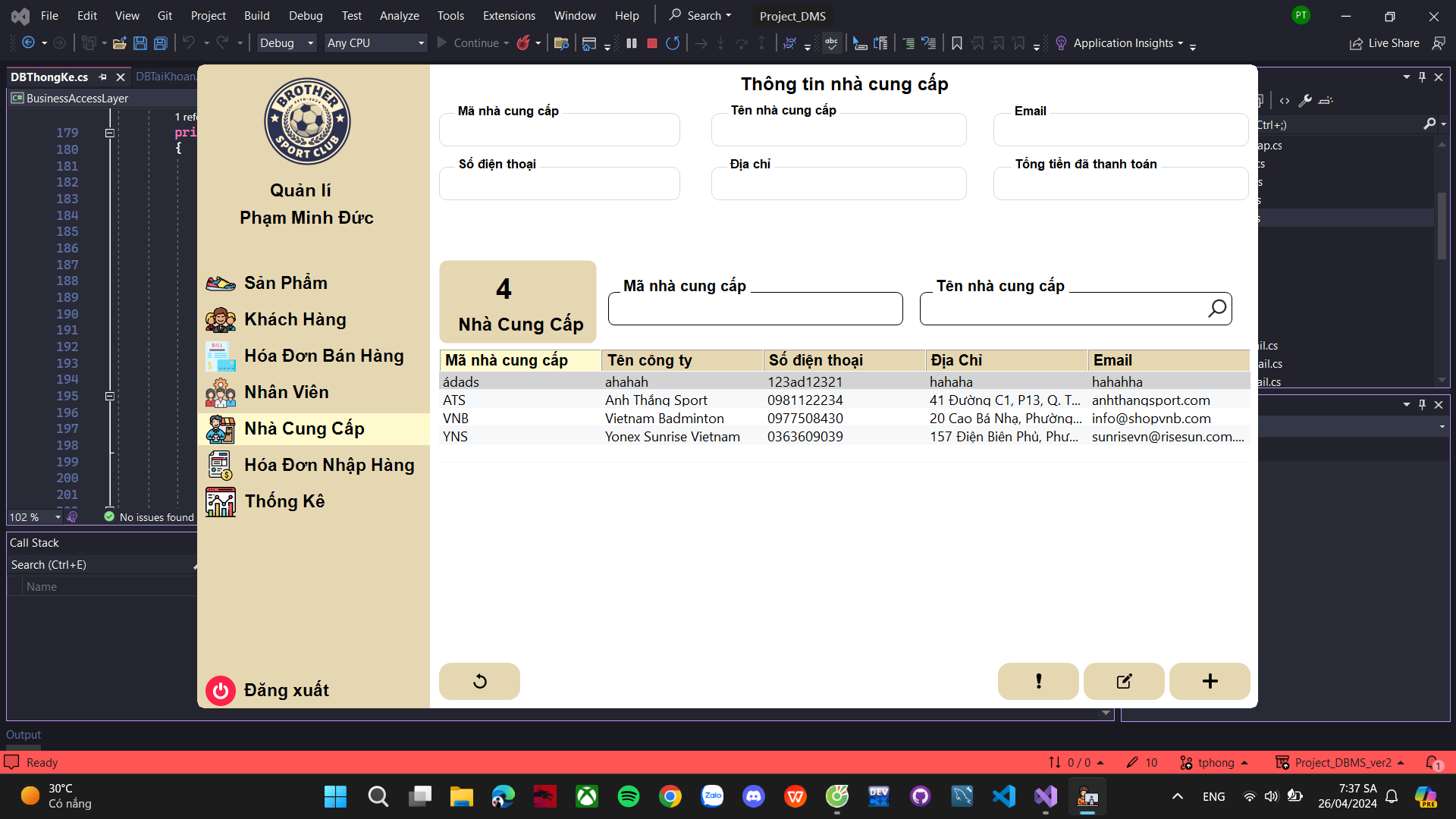
**5.1.3 Giao diện khách hàng**

****

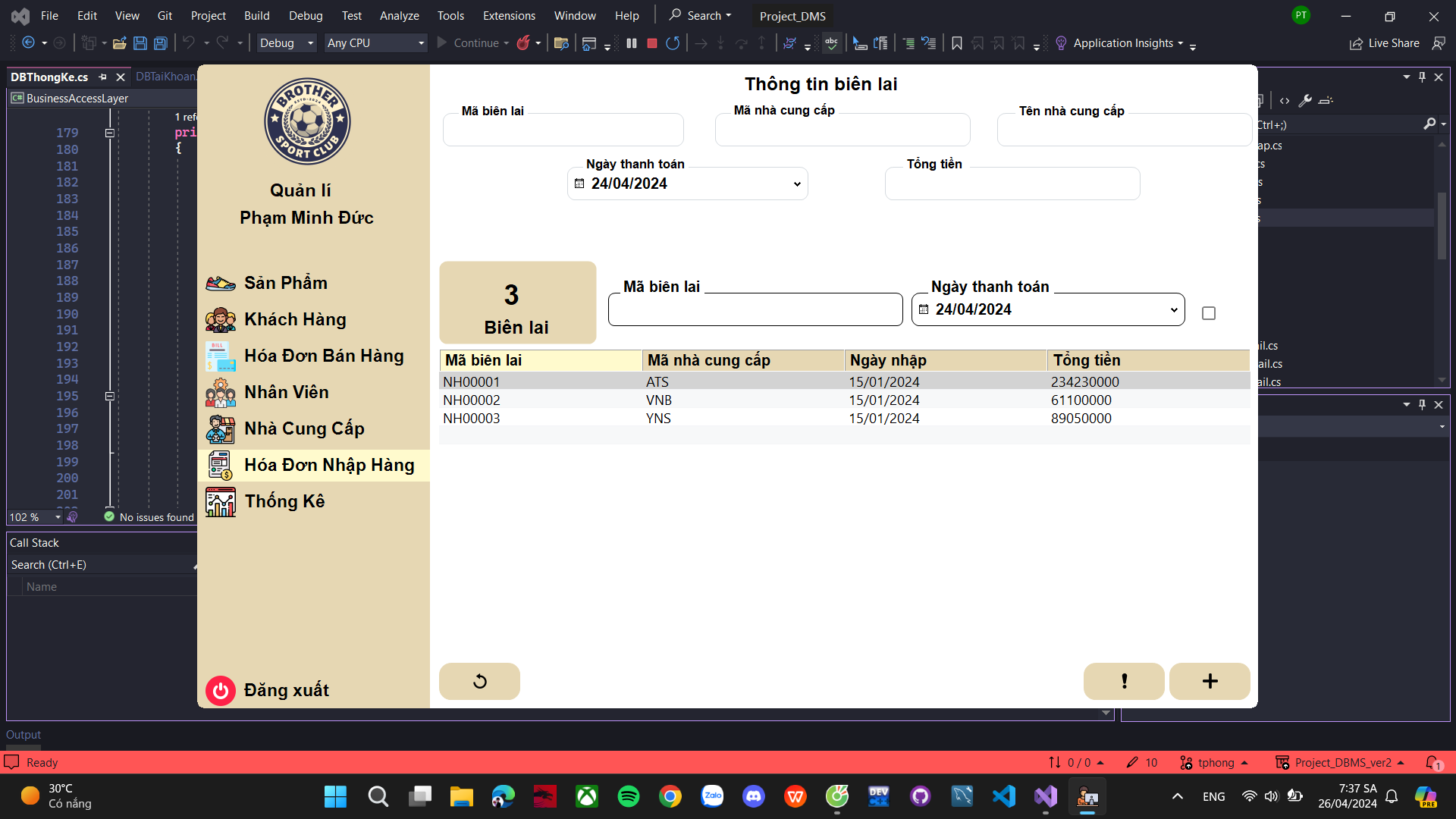
**5.1.4 Giao diện hóa đơn**

**5.1.6 Giao diện nhân viên**

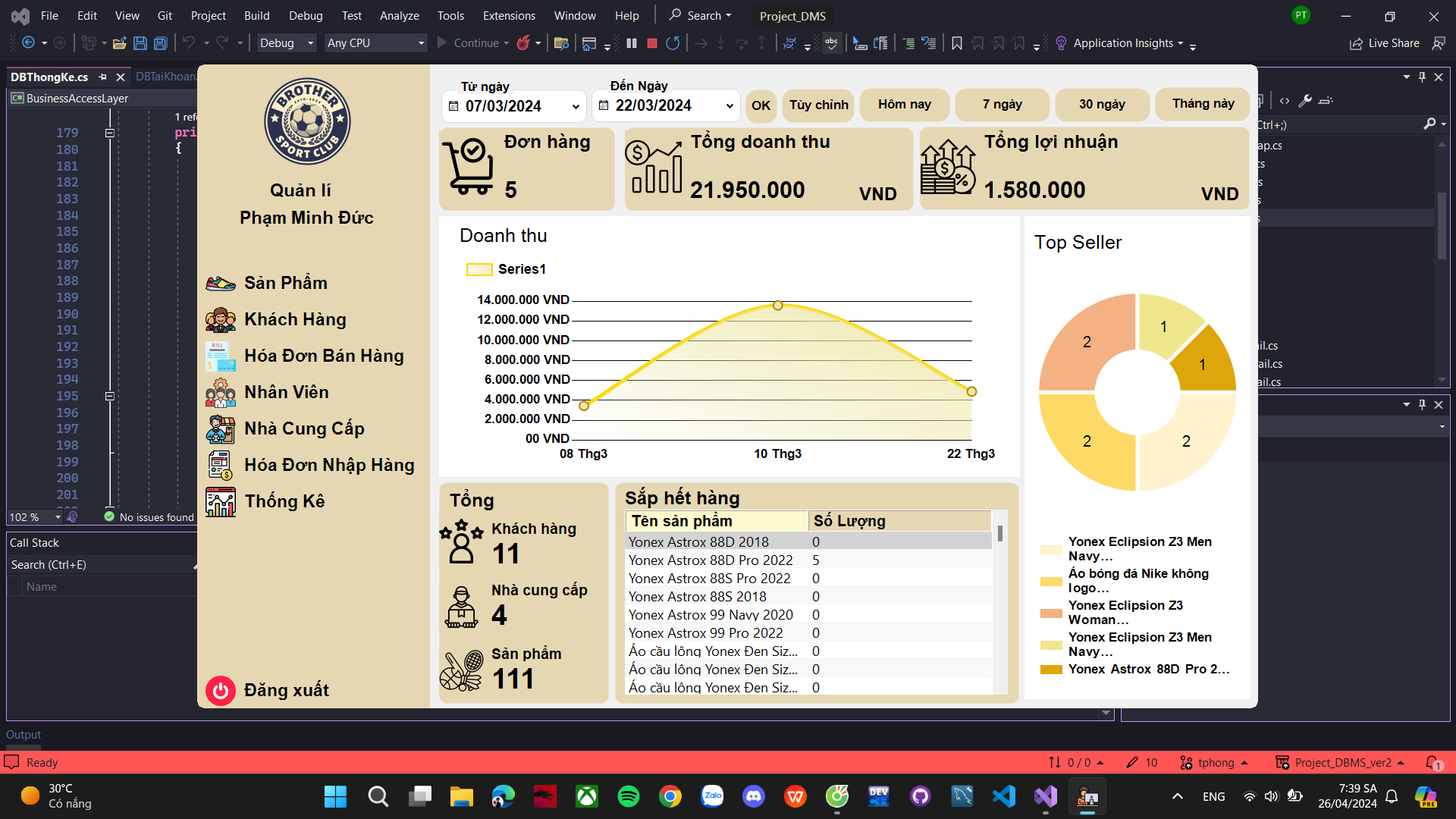
****

**5.1.7 Giao diện nhà cung cấp**

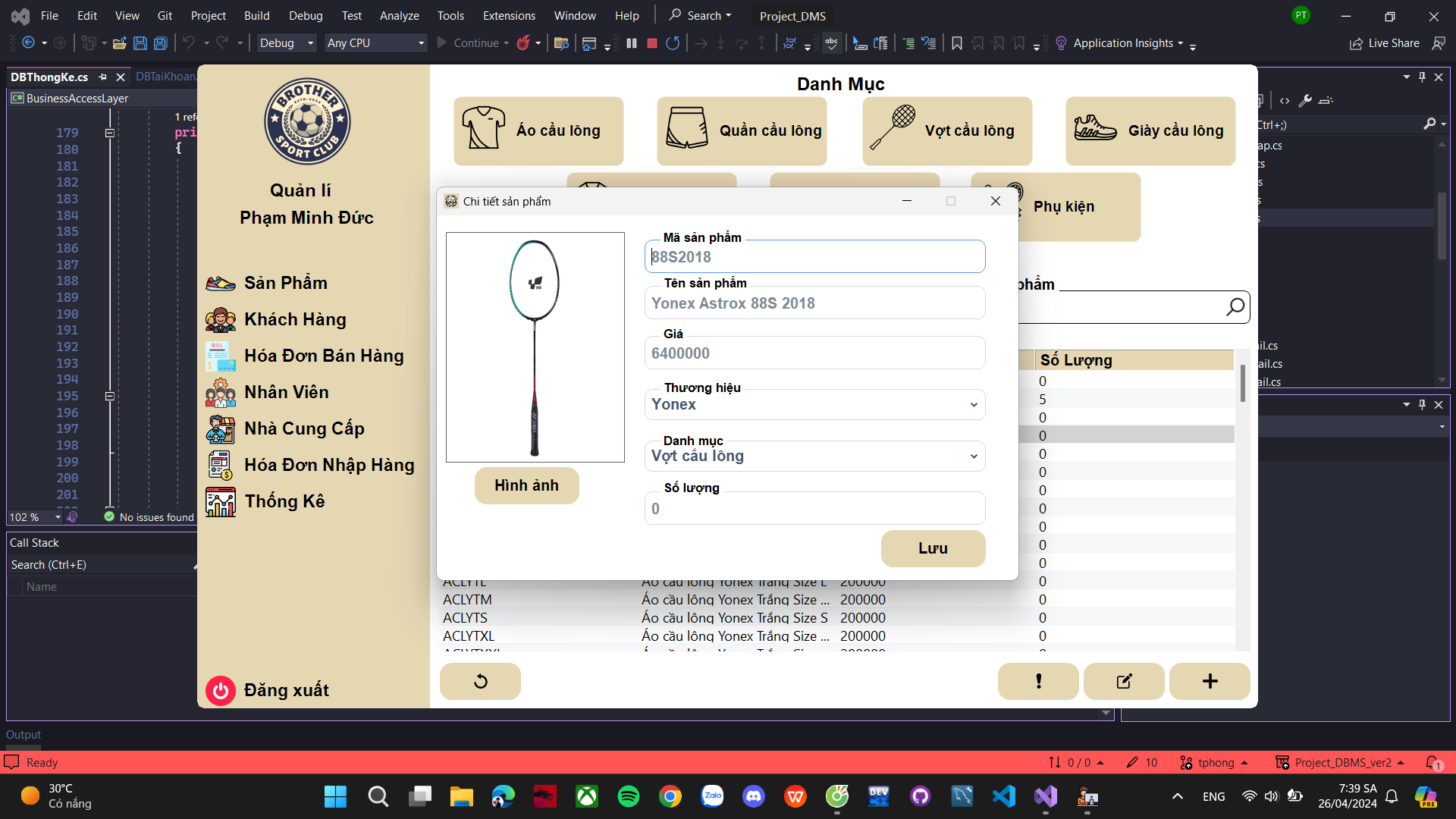
**5.1.8 Giao diện biên lai nhập hàng**

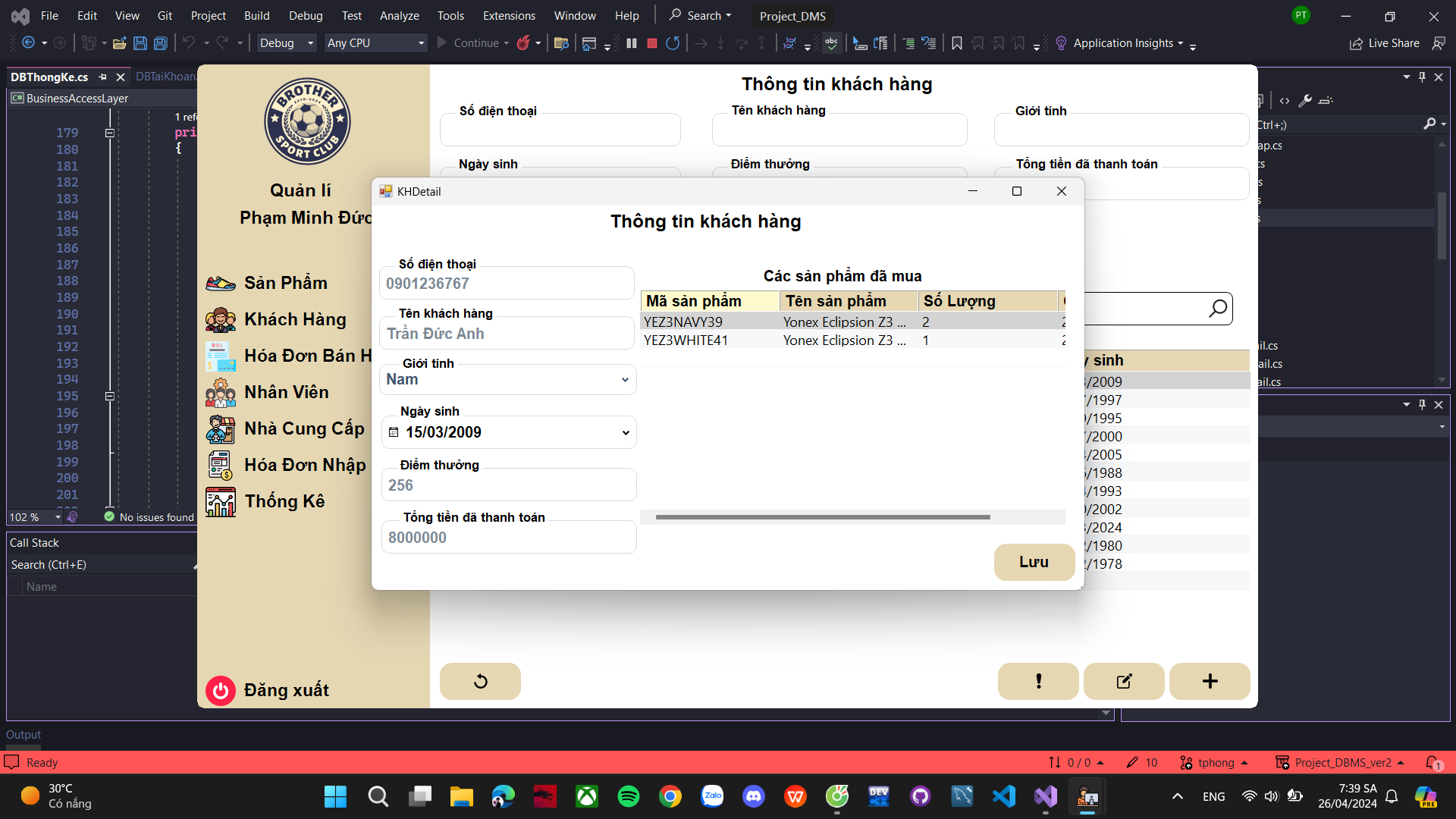
****

**5.1.9 Giao diện thống kê**

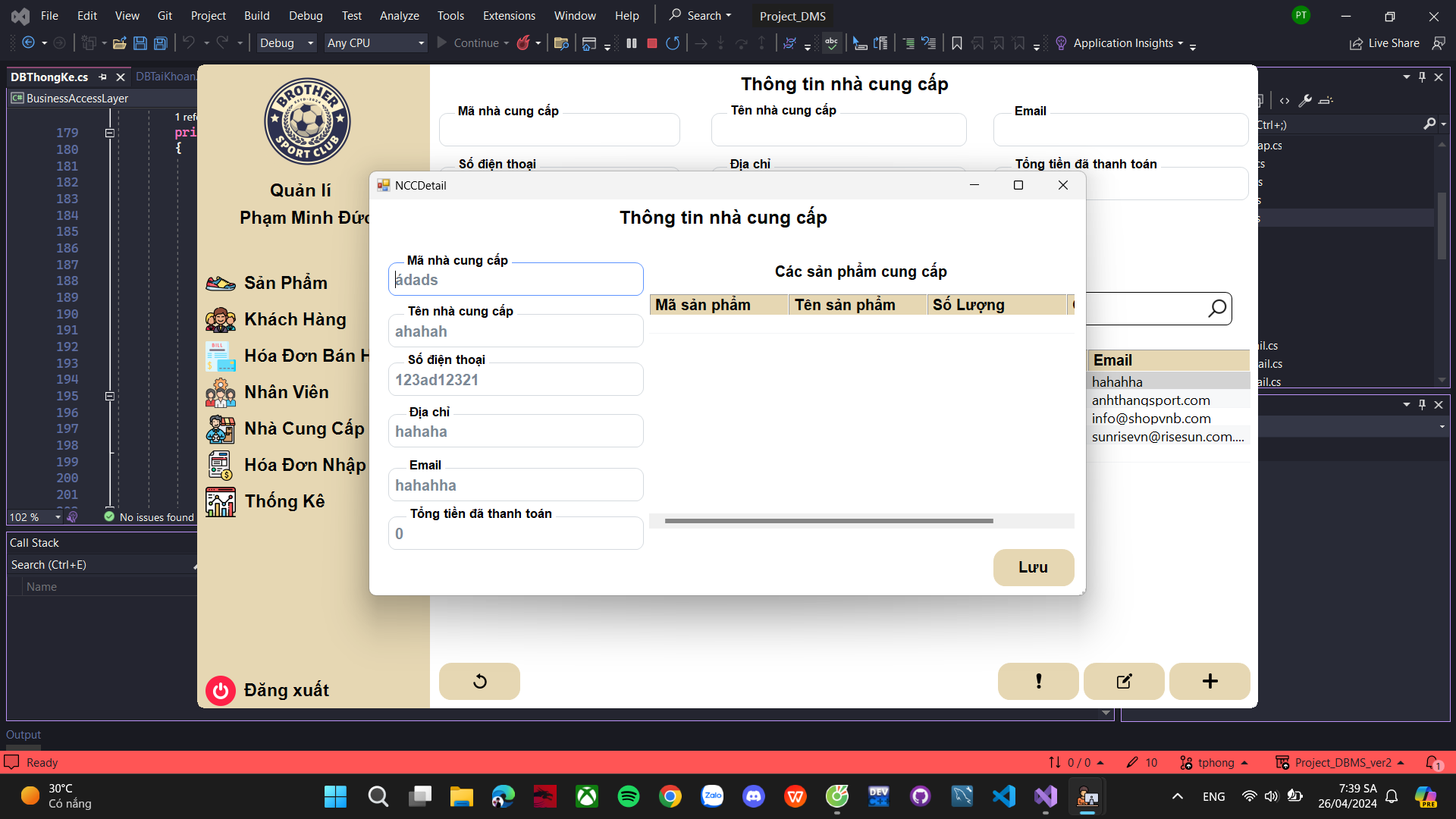
****

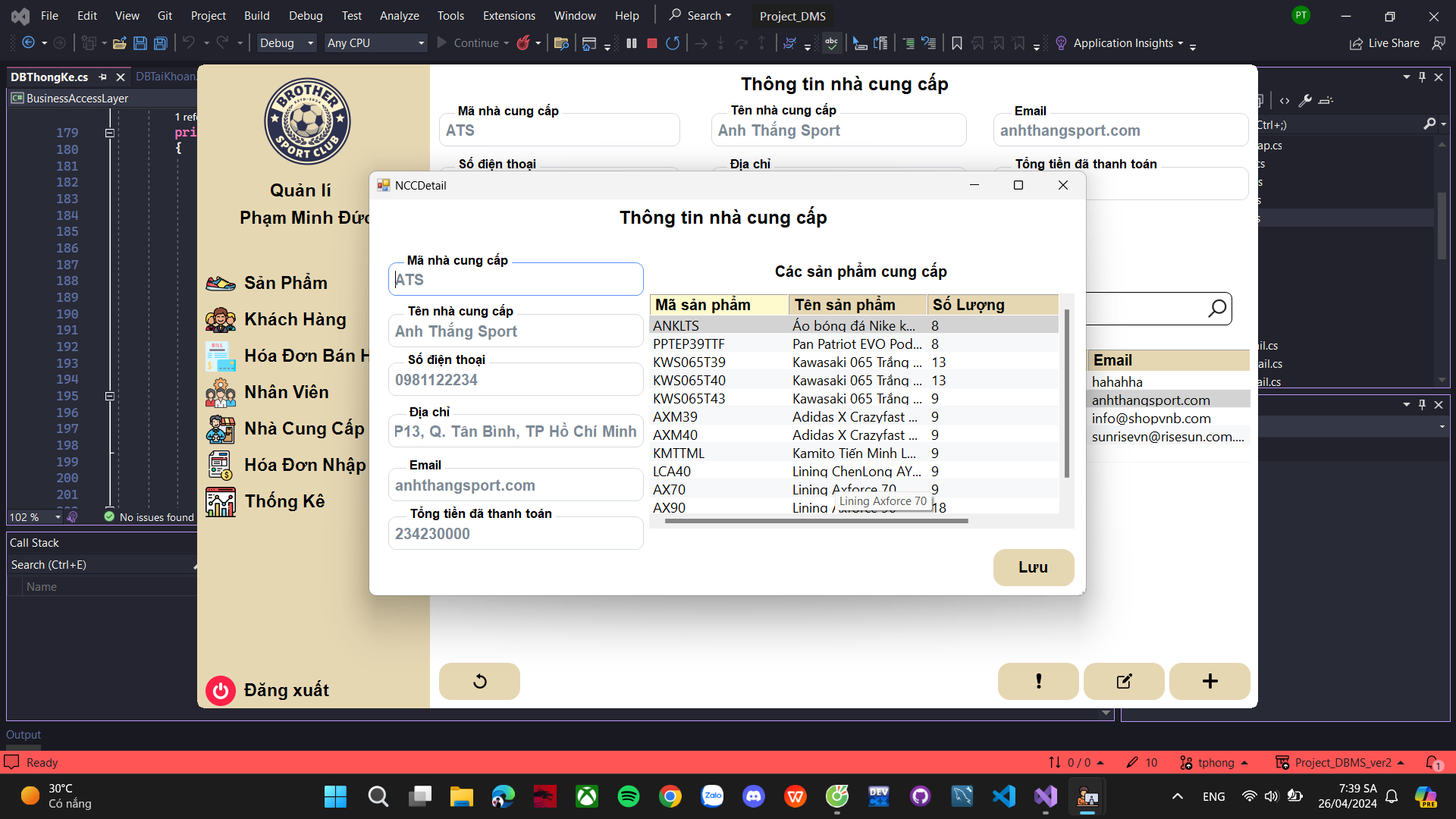
**5.1.10 Giao diện xem thông tin sản phẩm**

****

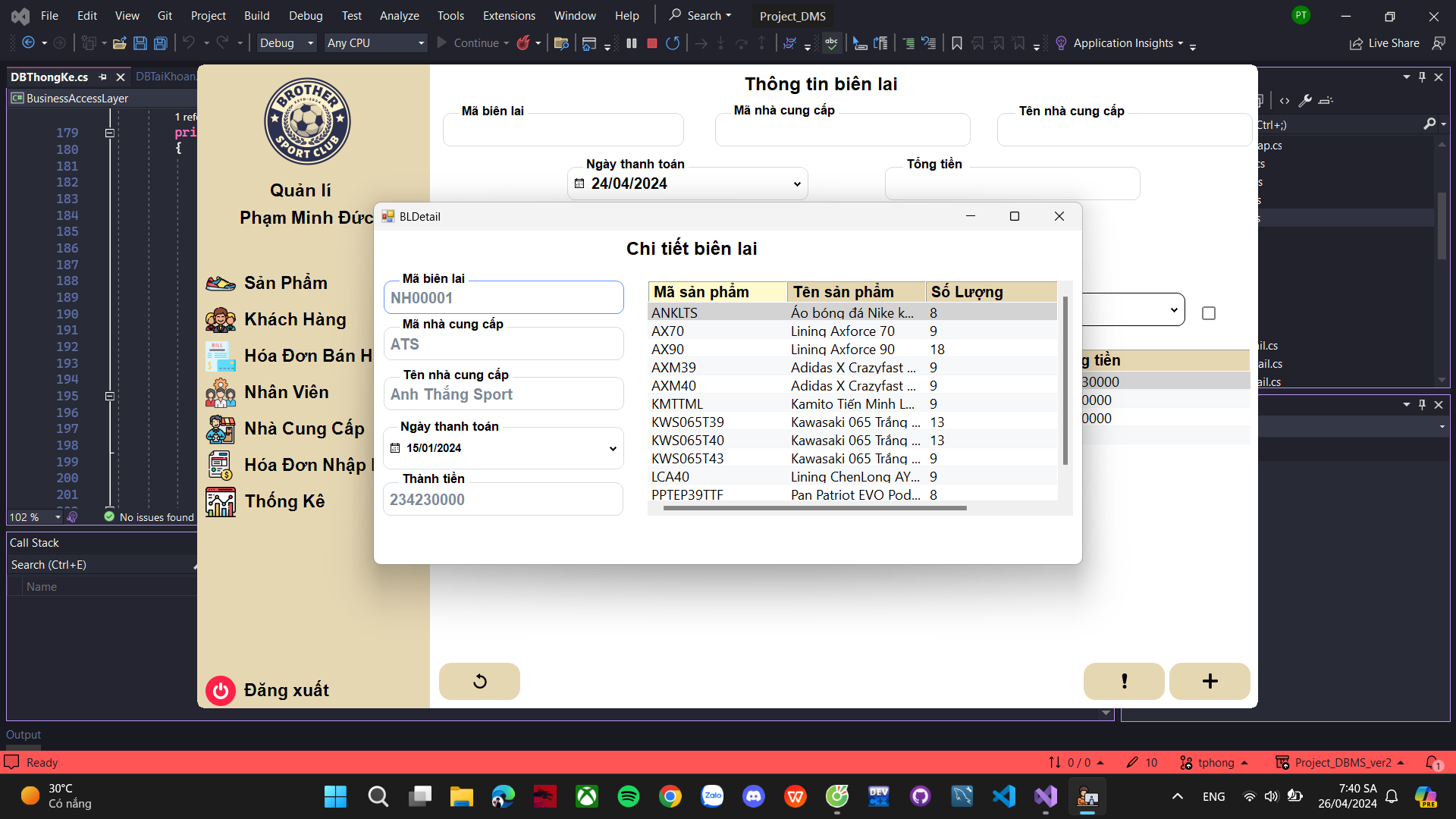
**5.1.11 Giao diện thông tin khách hàng**

**5.1.12 Giao diện thông tin nhà cung cấp**

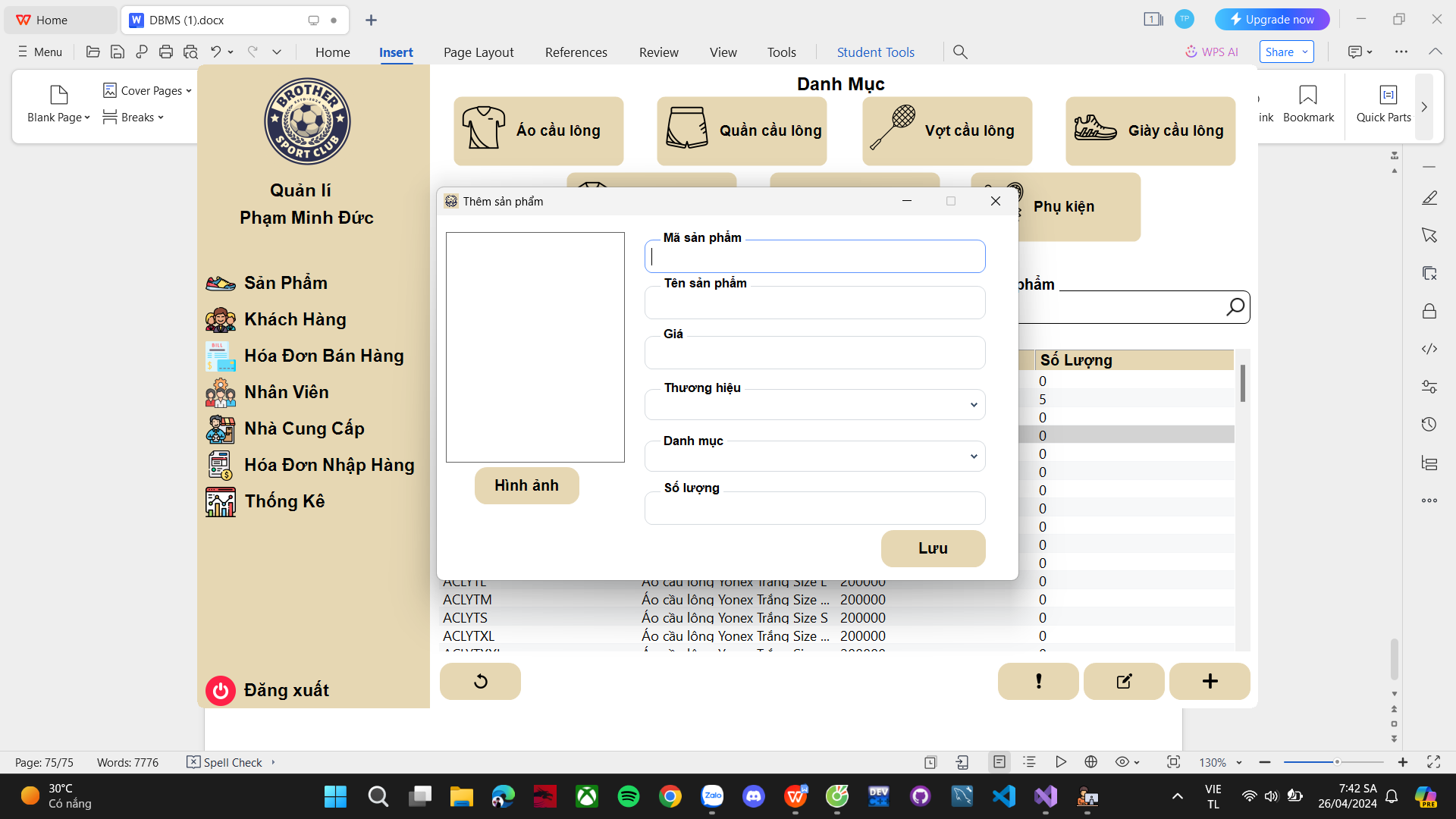
****

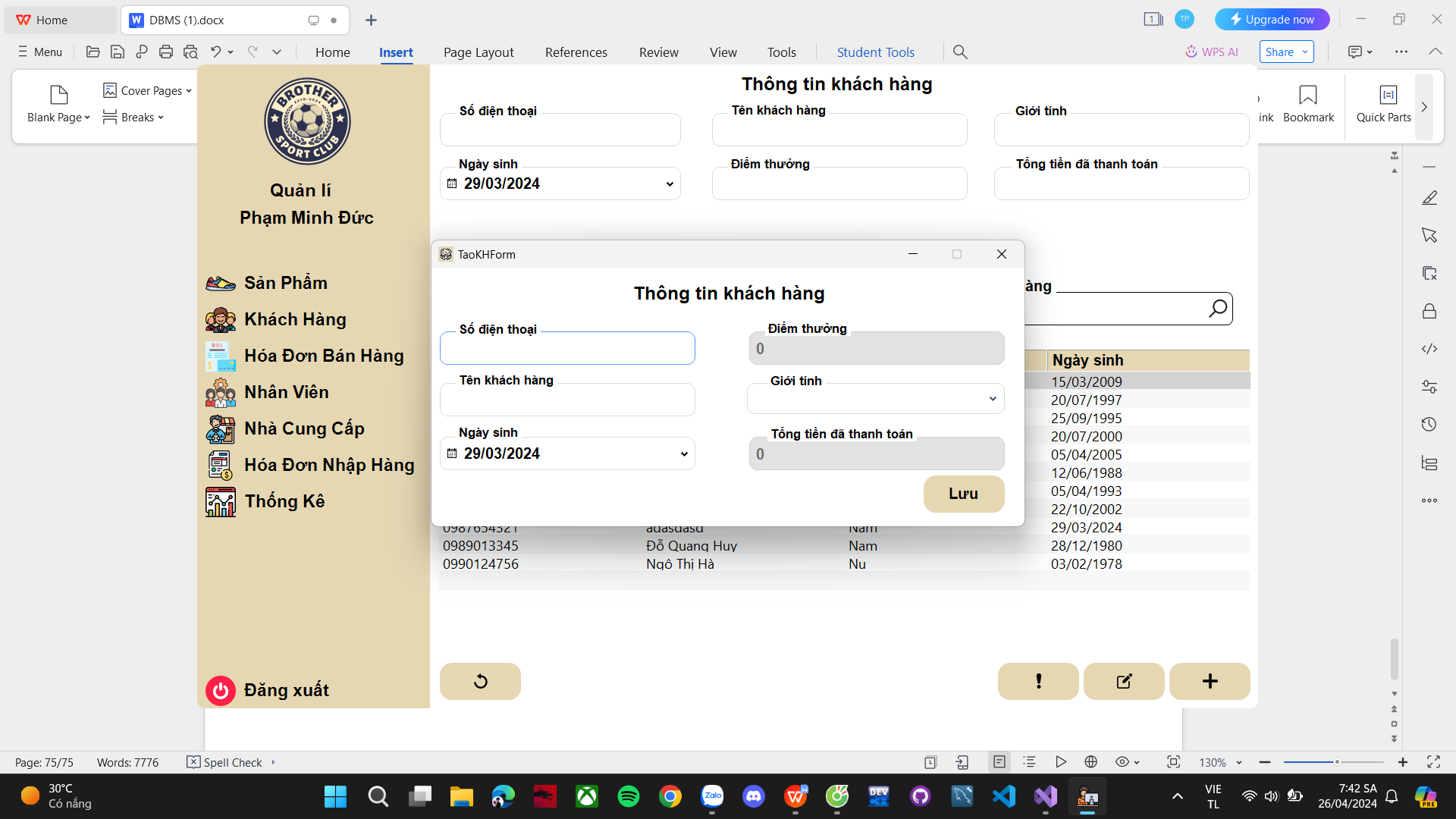
**5.1.13 Giao diện thông tin nhà cung cấp**

**5.1.14 Giao diện chi tiết biên lai**

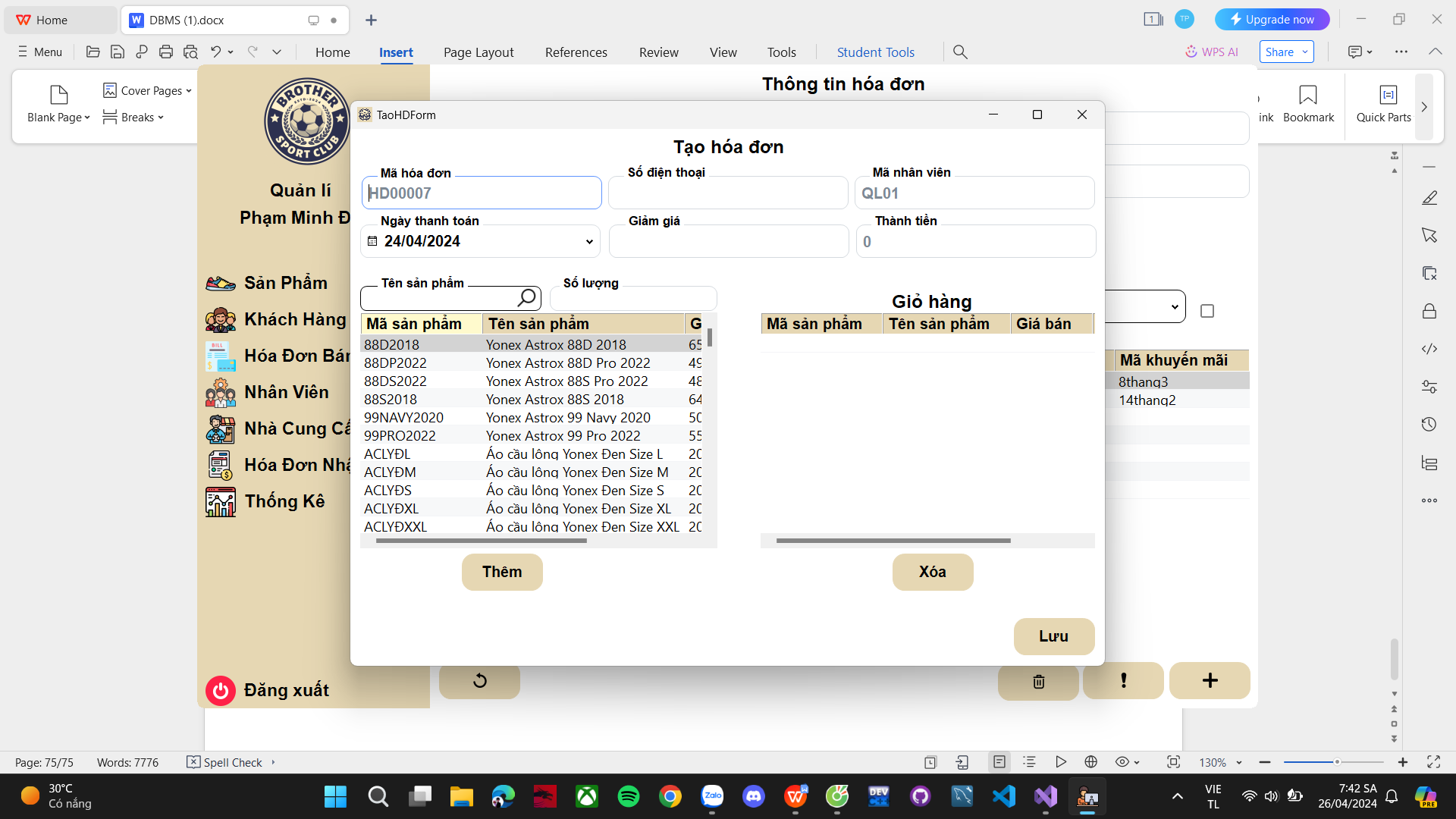
****

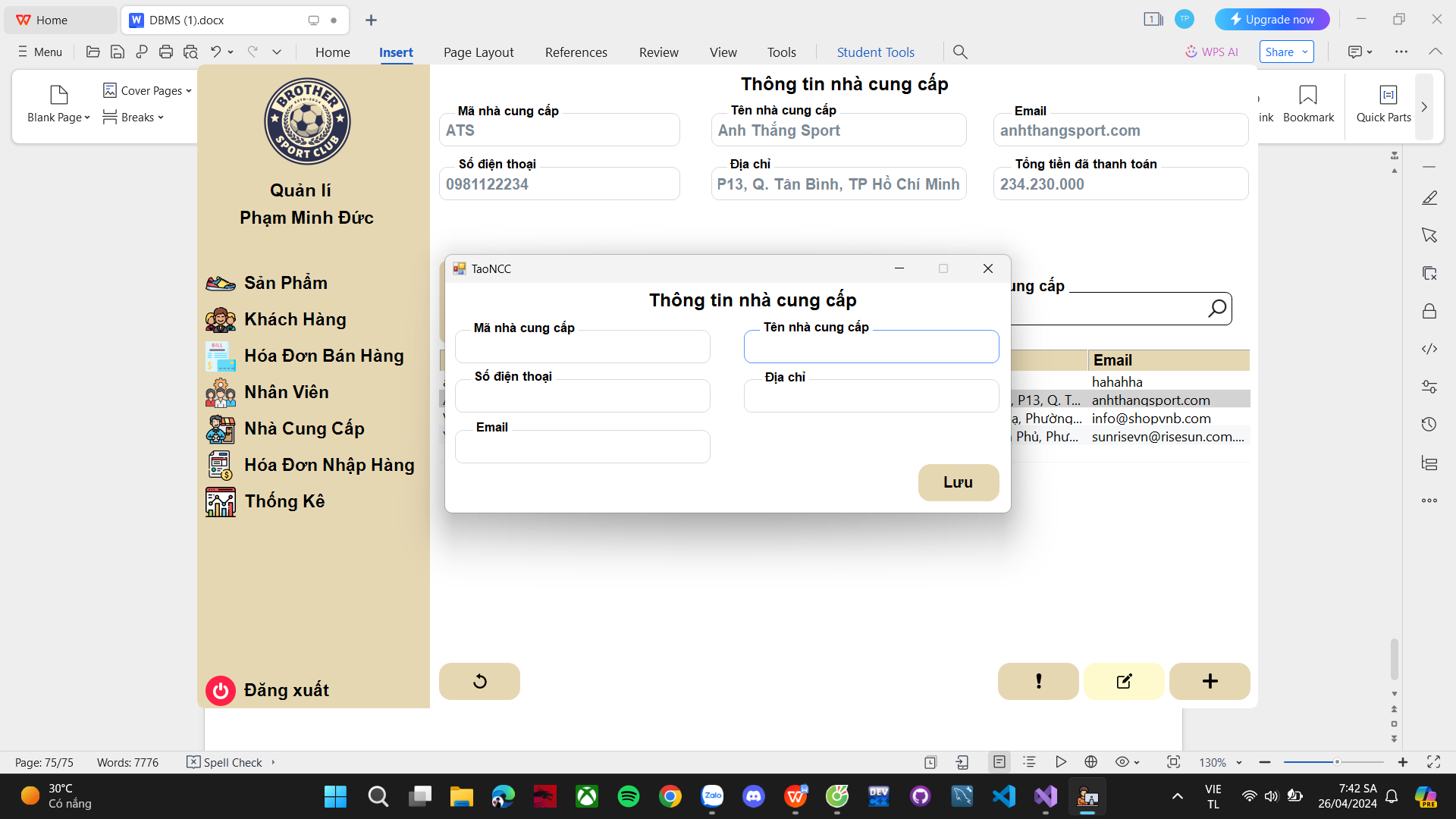
**5.1.15 Giao diện thêm sản phẩm**

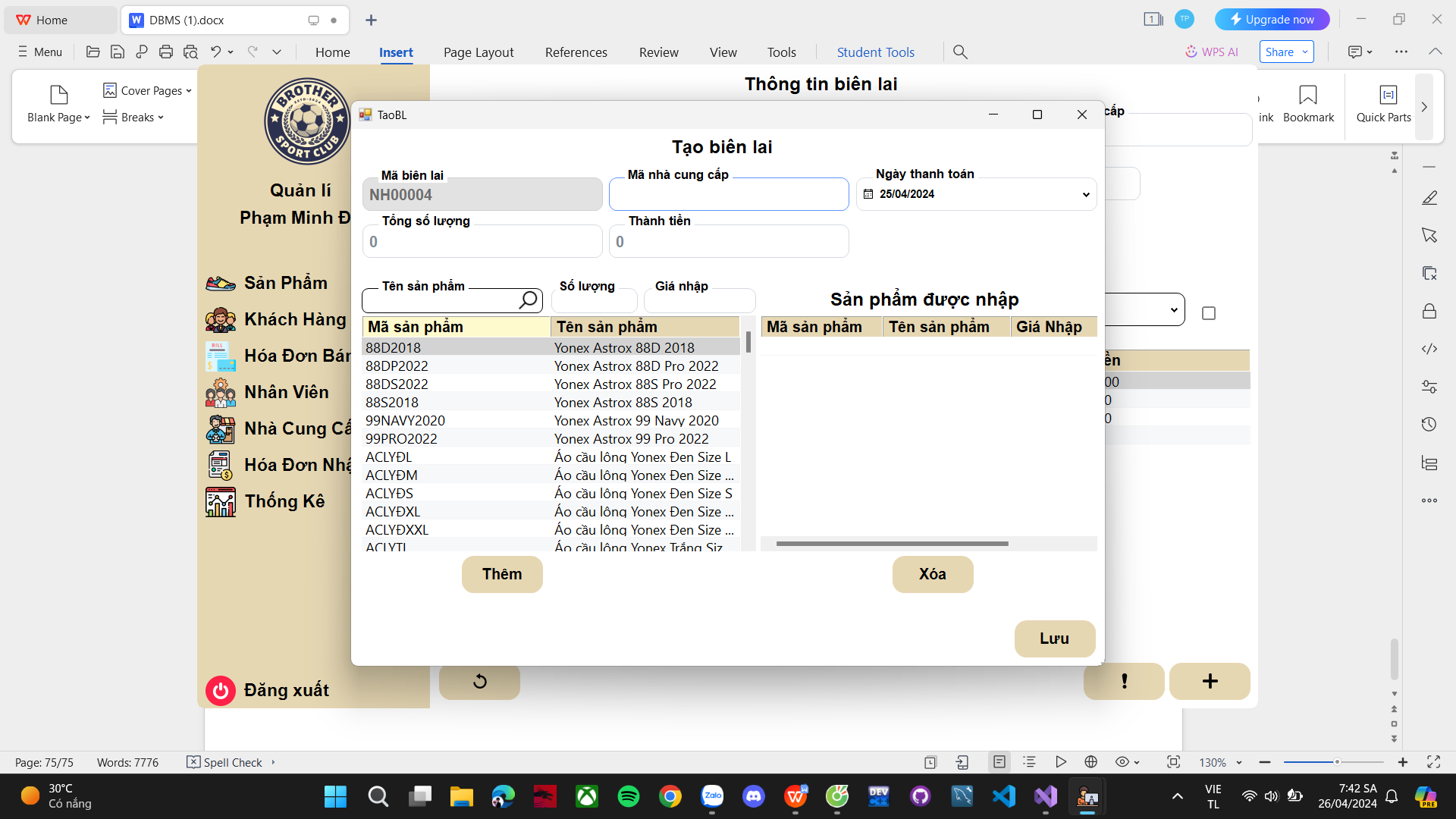
****

**5.1.16 Giao diện thêm khách hàng**

**5.1.17 Giao diện thêm hóa đơn**

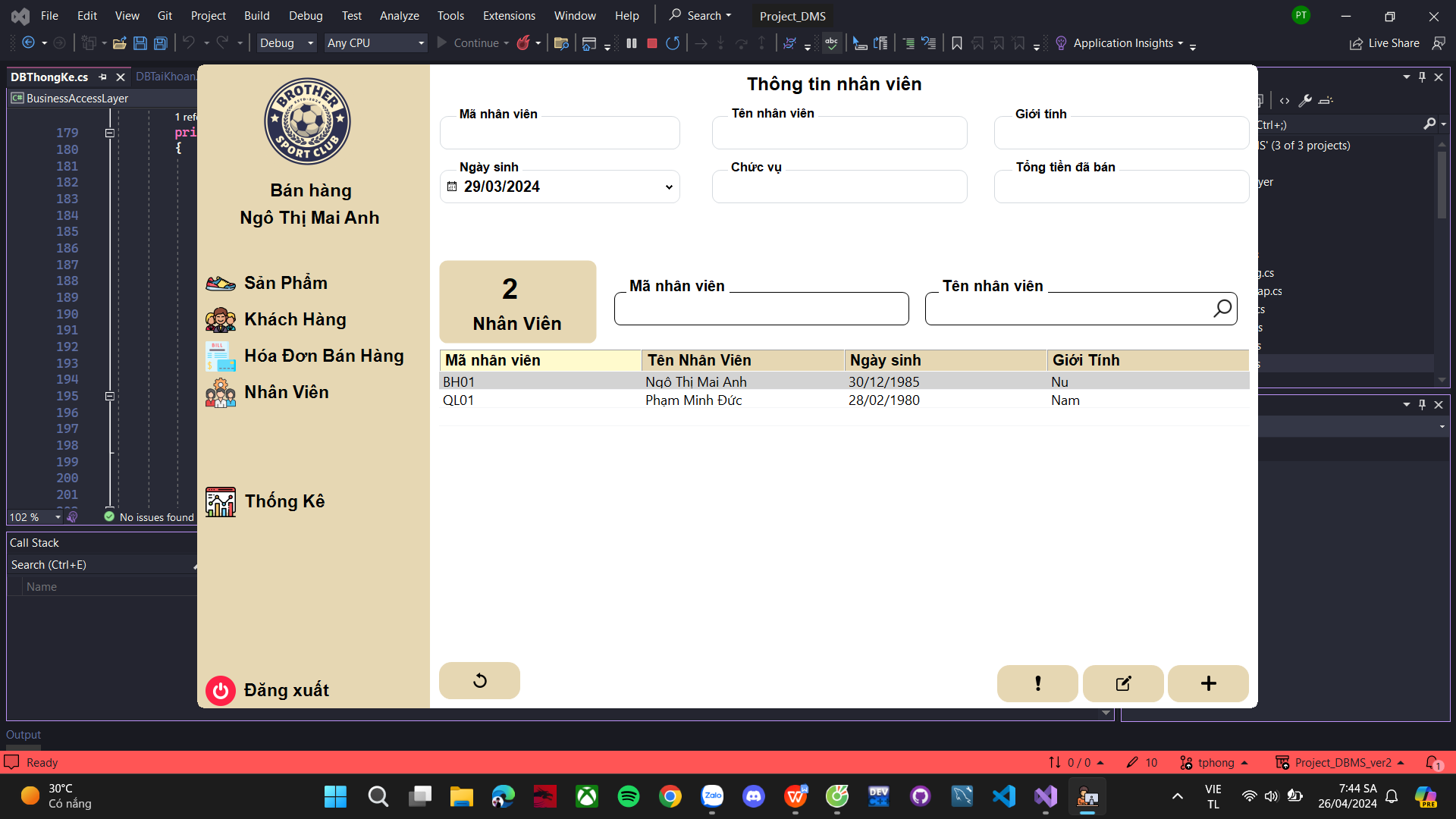
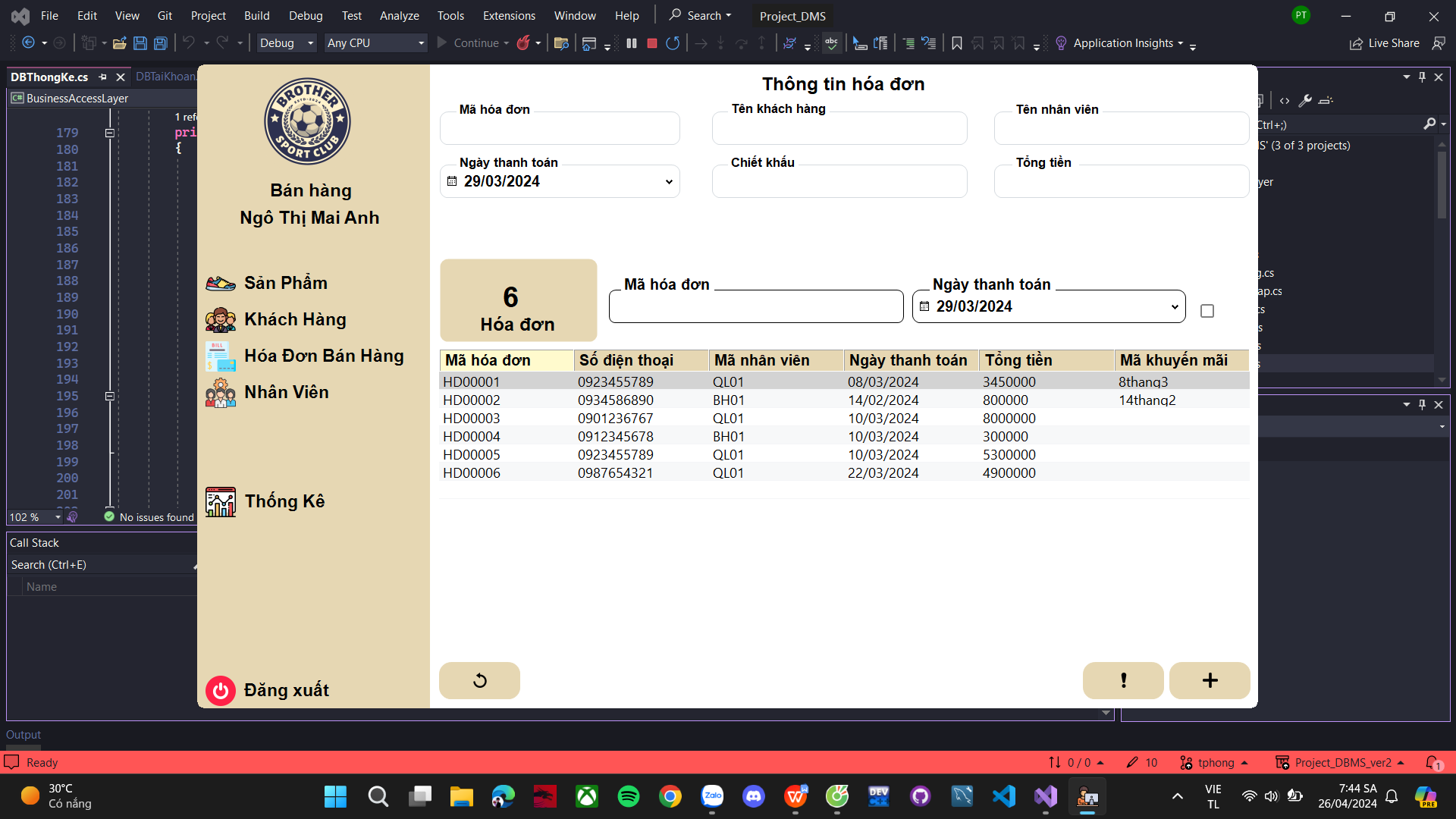
****

**5.1.18 Giao diện thêm cung cấp**

**5.1.19 Giao diện thêm biên lai**

**5.2 Giao diện nhân viên bán hàng**

**- Tương tự Quản lý nhưng ẩn nhà cung cấp và biên lai, không thể xóa hóa đơn, nhân viên, cũng như thêm nhân viên.**

****