|  |
| --- |
| FPT POLYTECHNIC |
| **DỰ ÁN 1 Quản Lý NHÀ HÀNG** |
| NGÀNH CÔNG NGHỆ THÔNG TIN (PHÁT TRIỂN PHẦN MỀM) |



|  |
| --- |
| Tp Hồ Chí Minh 2023 |

|  |
| --- |
| GIẢNG VIÊN HƯỚNG DẪN: NGUYỄN VĂN THẦY |
| SINH VIÊN THỰC HIỆN |
| 1. Huỳnh Thiên Bảo (PS32589), Trưởng nhóm 2. Phạm Quang Gia Bảo (PS32974) 3. Lý Nguyên Hòa (PS32568) |

MỤC LỤC

[1 Phân tích 2](#_Toc520132211)

[1.1 Hiện trạng 2](#_Toc520132212)

[1.2 Yêu cầu hệ thống 2](#_Toc520132213)

[1.3 Use case 2](#_Toc520132214)

[2 Thiết kế 2](#_Toc520132215)

[2.1 Mô hình triển khai 2](#_Toc520132216)

[2.2 Thiết kế CSDL 2](#_Toc520132217)

[2.2.1 Sơ đồ quan hệ thực thể 2](#_Toc520132218)

[2.2.2 Thiết kế chi tiết các thực thể 2](#_Toc520132219)

[2.3 Thiết kế giao diện 2](#_Toc520132220)

[2.3.1 Sơ đồ tổ chức giao diện 2](#_Toc520132221)

[2.3.2 Thiết kế giao diện cho các chức năng nghiệp vụ 2](#_Toc520132222)

[3 Thực hiện viết mã 2](#_Toc520132223)

[3.1 Viết mã tạo CSDL 2](#_Toc520132224)

[3.1.1 Tạo CSDL 2](#_Toc520132225)

[3.1.2 SQL truy vấn và thao tác 2](#_Toc520132226)

[3.1.3 Các thủ tục lưu tổng hợp thống kê 2](#_Toc520132227)

[3.2 Lập trình JDBC 3](#_Toc520132228)

[3.2.1 Lớp hỗ trợ 3](#_Toc520132229)

[3.2.2 Model class - Các lớp mô tả dữ liệu 3](#_Toc520132230)

[3.2.3 DAO Class - Các lớp truy xuất dữ liệu 3](#_Toc520132231)

[3.3 Viết mã cho ứng dụng 3](#_Toc520132232)

[3.3.1 Xử lý Form X 3](#_Toc520132233)

[3.3.2 Xử lý Form Y 3](#_Toc520132234)

[4 Kiểm thử 3](#_Toc520132235)

[4.1 Kiểm thử form X 3](#_Toc520132236)

[4.2 Kiểm thử form Y 3](#_Toc520132237)

[5 Đóng gói và triển khai 3](#_Toc520132238)

[5.1 Hướng dẫn chuyển đổi jar thành exe 3](#_Toc520132239)

[5.2 Hướng dẫn cài đặt triển khai 3](#_Toc520132240)

[5.3 Hướng dẫn sử dụng phần mềm 3](#_Toc520132241)

# Phân tích

## Hiện trạng

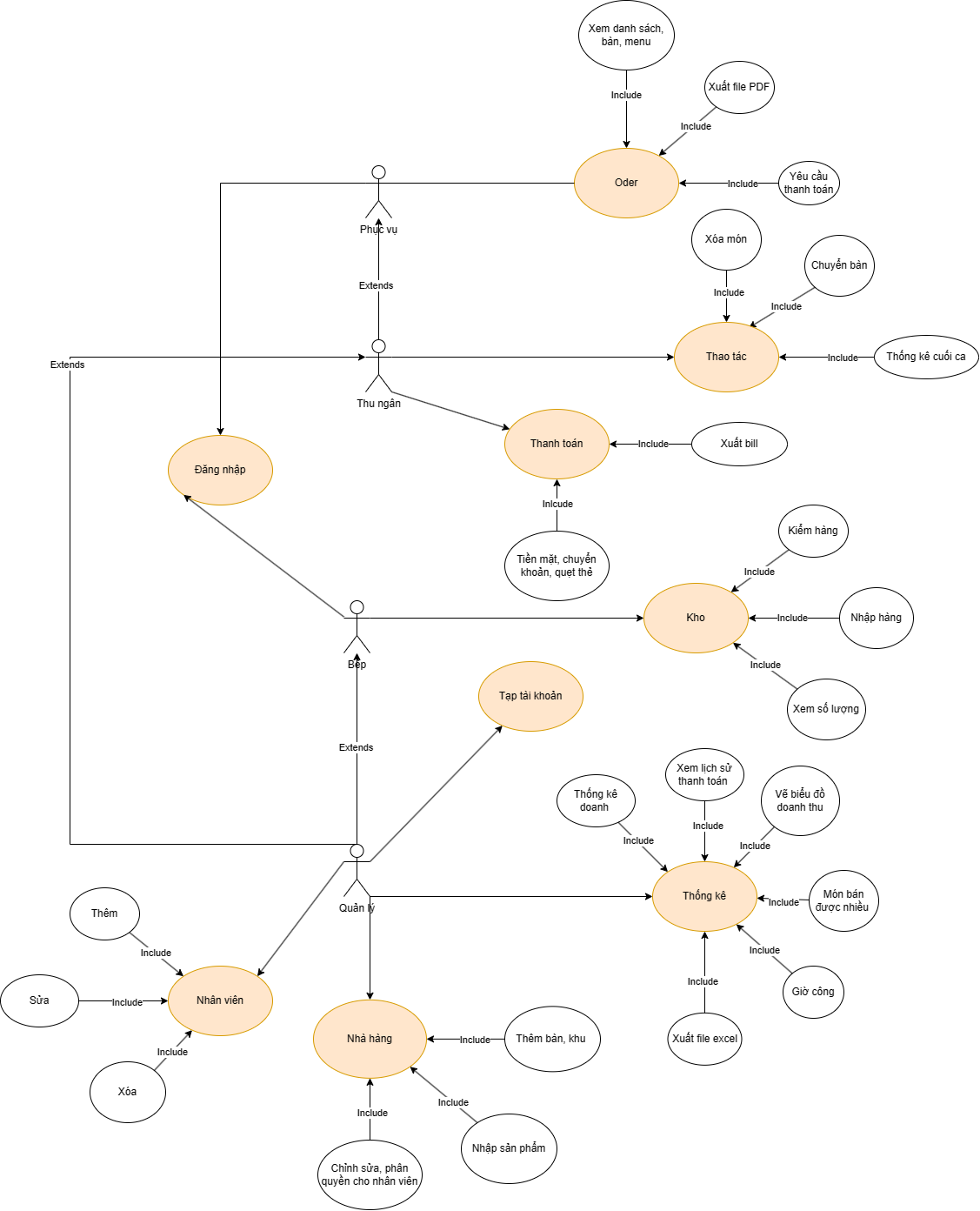
Phần mềm có thể quản lý nhà hàng/quán cà phê, được phát triển để hỗ trợ chủ nhà hàng quản lý dễ dàng kho hàng, nhân viên và theo dõi thông kê. Đồng thời giúp nhân viên order món nhanh chóng và chính xác, tạo điều kiện thuận lợi cho quầy pha chế và bếp. Chủ nhà hàng có khả năng theo dõi dòng tiền một cách minh bạch thông qua các báo cáo chi tiết về doanh thu, chi phí và lợi nhuận. Điều này giúp họ đưa ra quyết định dựa trên thông tin chính xác, từ chiến lược giảm giá đến tối ưu hóa chiến lược tiếp thị.

## Yêu cầu hệ thống

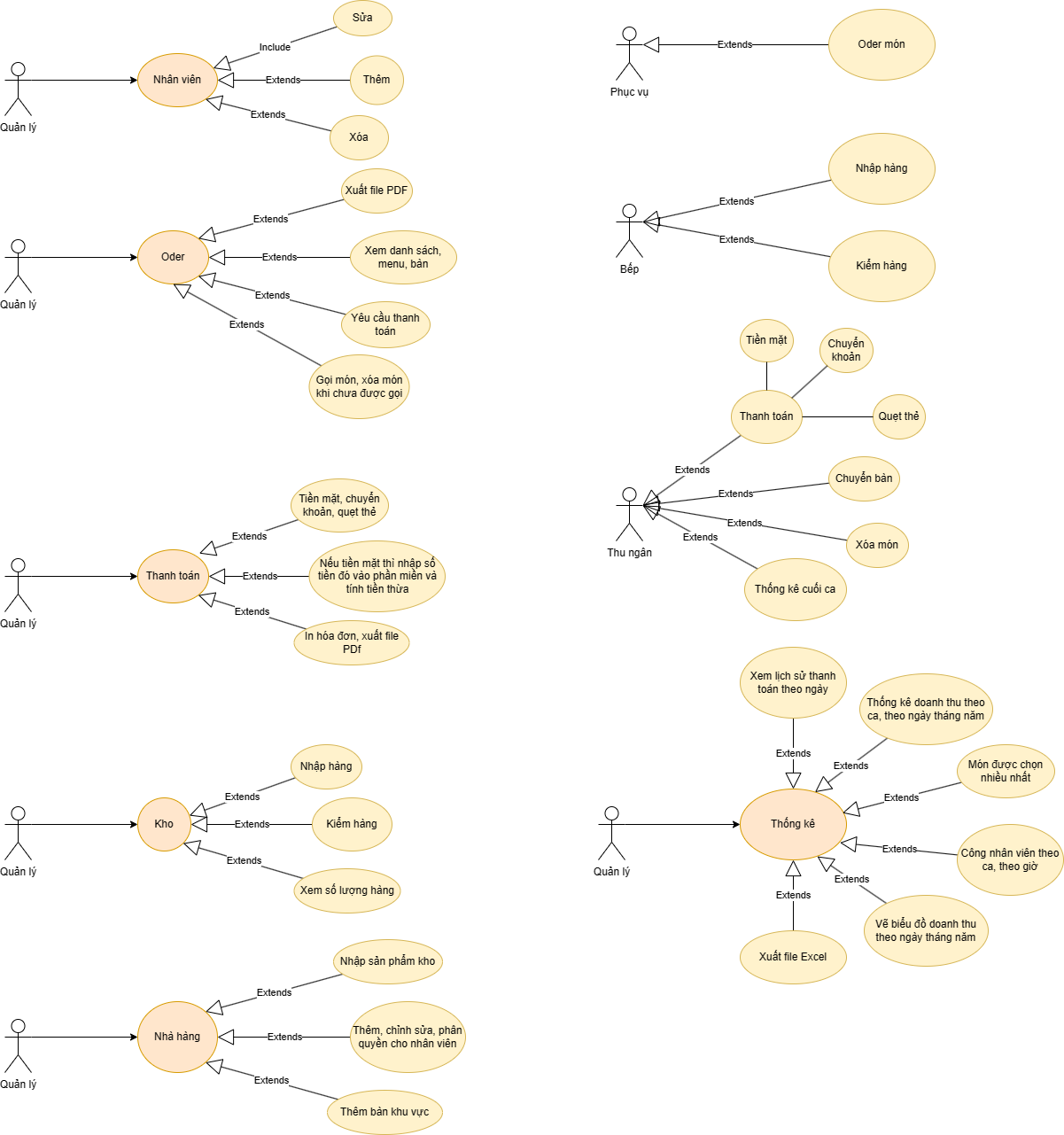
* Hệ Điều Hành: Windows 10, Windows 11.
* JDK: 19+.
* SQL Sever 2019.

## Use case

### Use case tổng quát

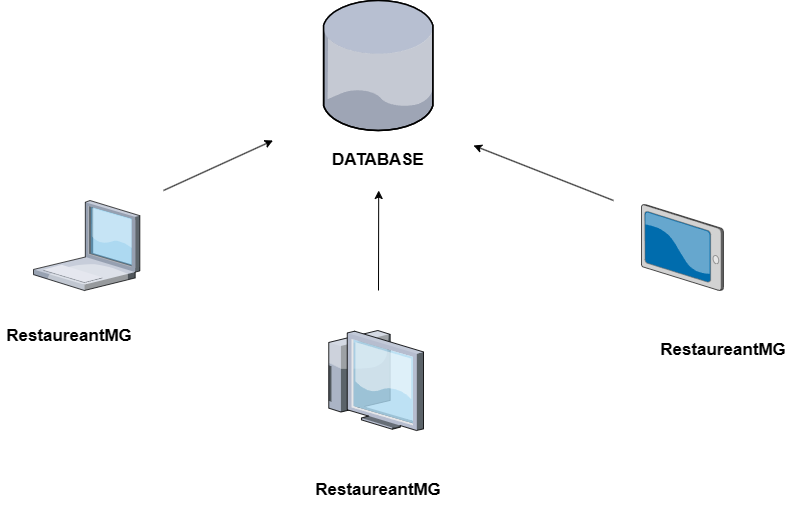


### Use case chi tiết



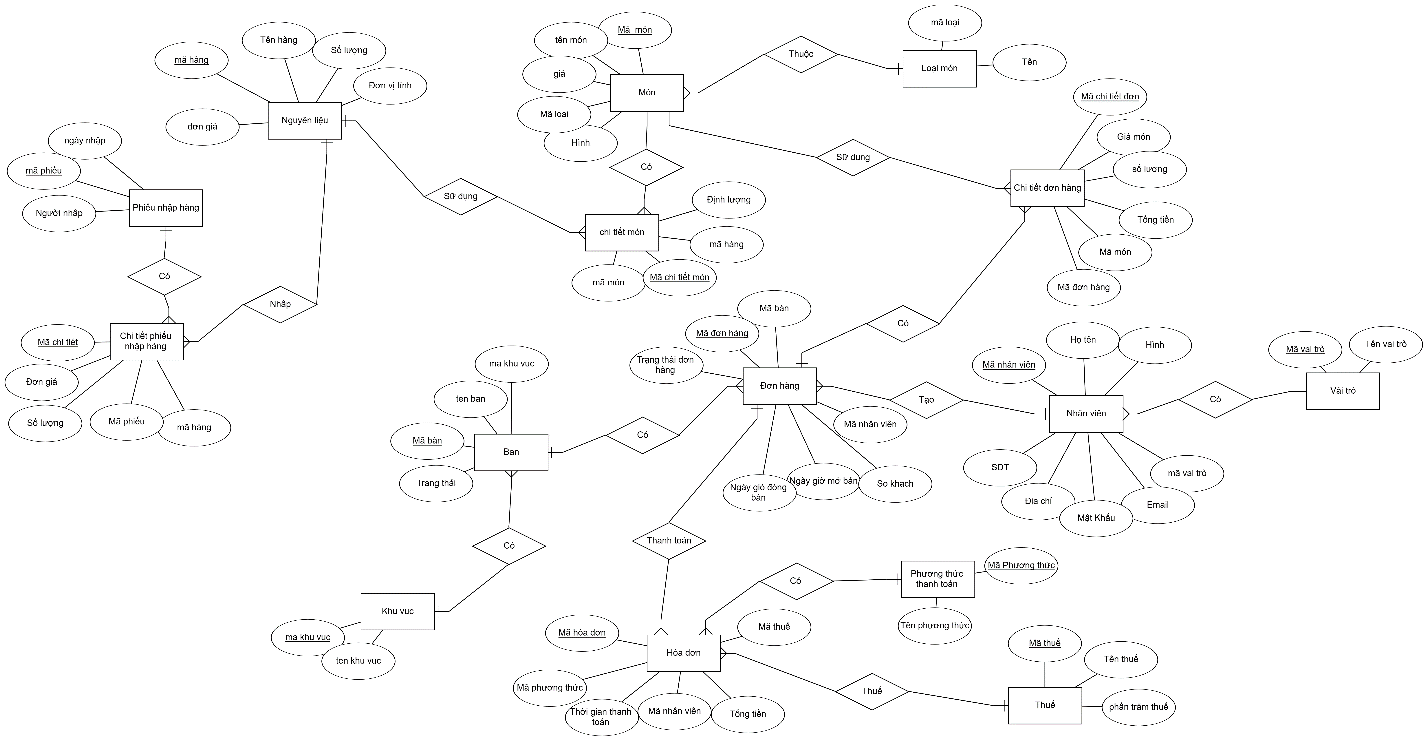
# Thiết kế

## Mô hình triển khai

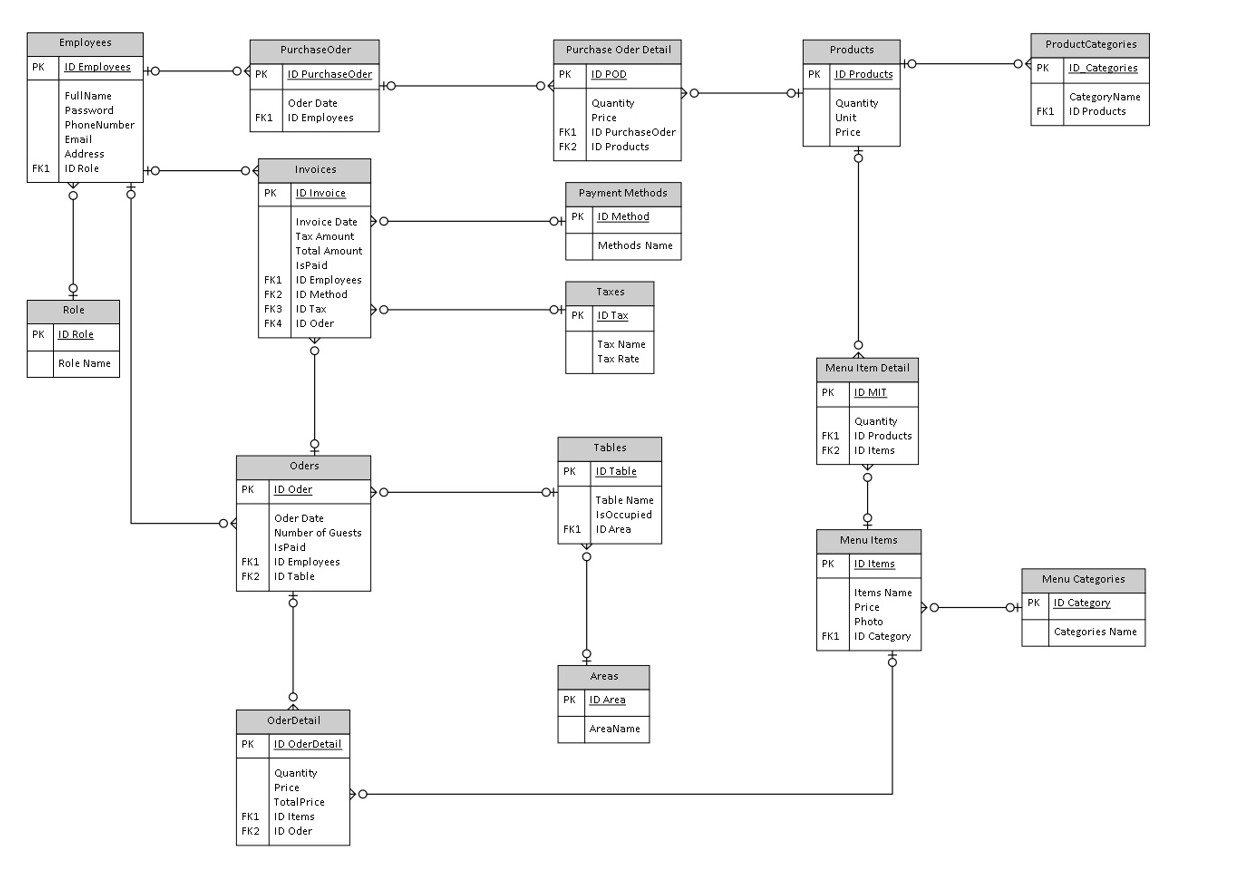


## Thiết kế CSDL

### Sơ đồ quan hệ thực thể

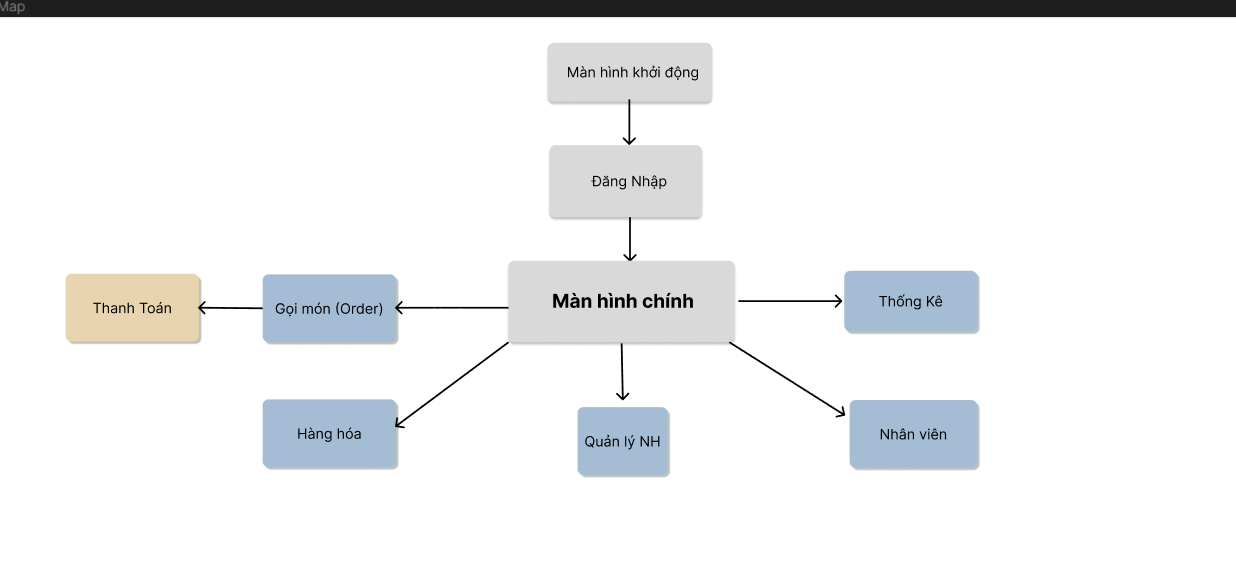


### Thiết kế chi tiết các thực thể



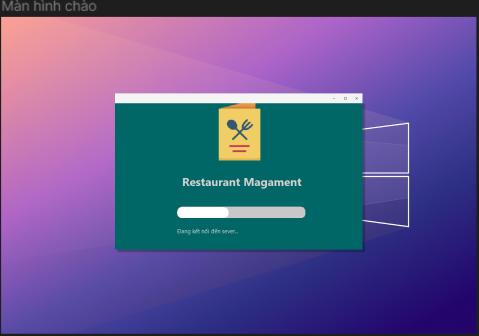
## Thiết kế giao diện

### Sơ đồ tổ chức giao diện

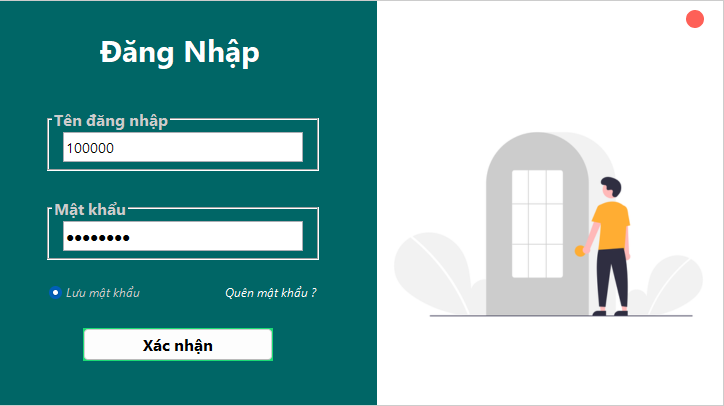


### Thiết kế giao diện cho các chức năng nghiệp vụ

#### Thiết kế Form màn hình chào



#### Thiết kế Form đăng nhập



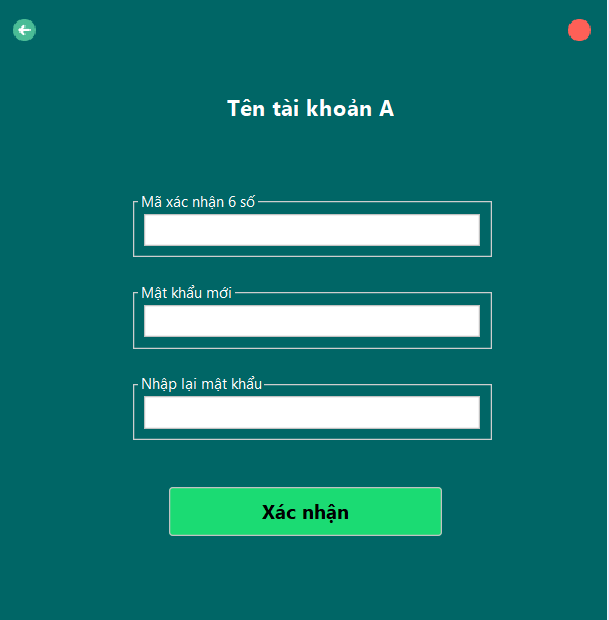
* Chức năng của form đăng nhập là để đăng nhập vào tài khoản nhân viên, quản lí, thu ngân …

#### Thiết kế Form quên mật khẩu



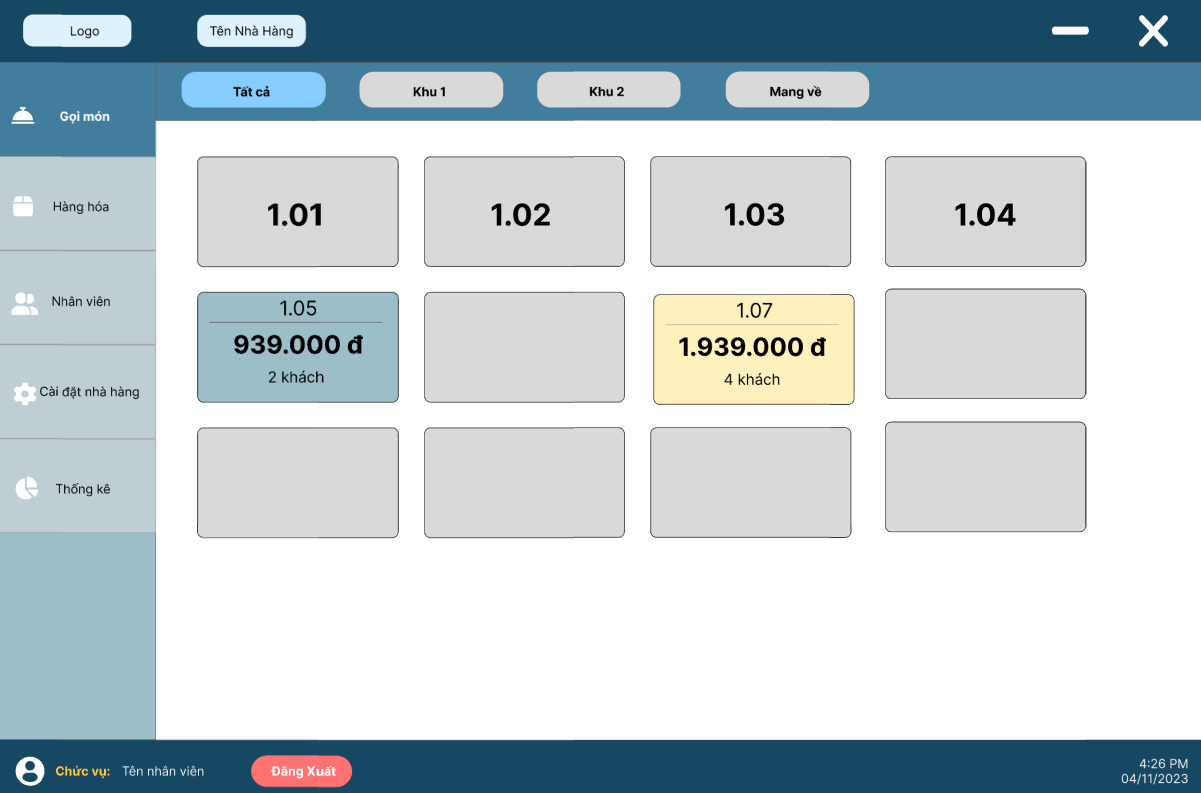
* Chức năng của form quên mật khẩu là khôi phục lại tài khoản đã đăng nhập khi quên mật khẩu

#### Thiết kế Form khôi phục mật khẩu



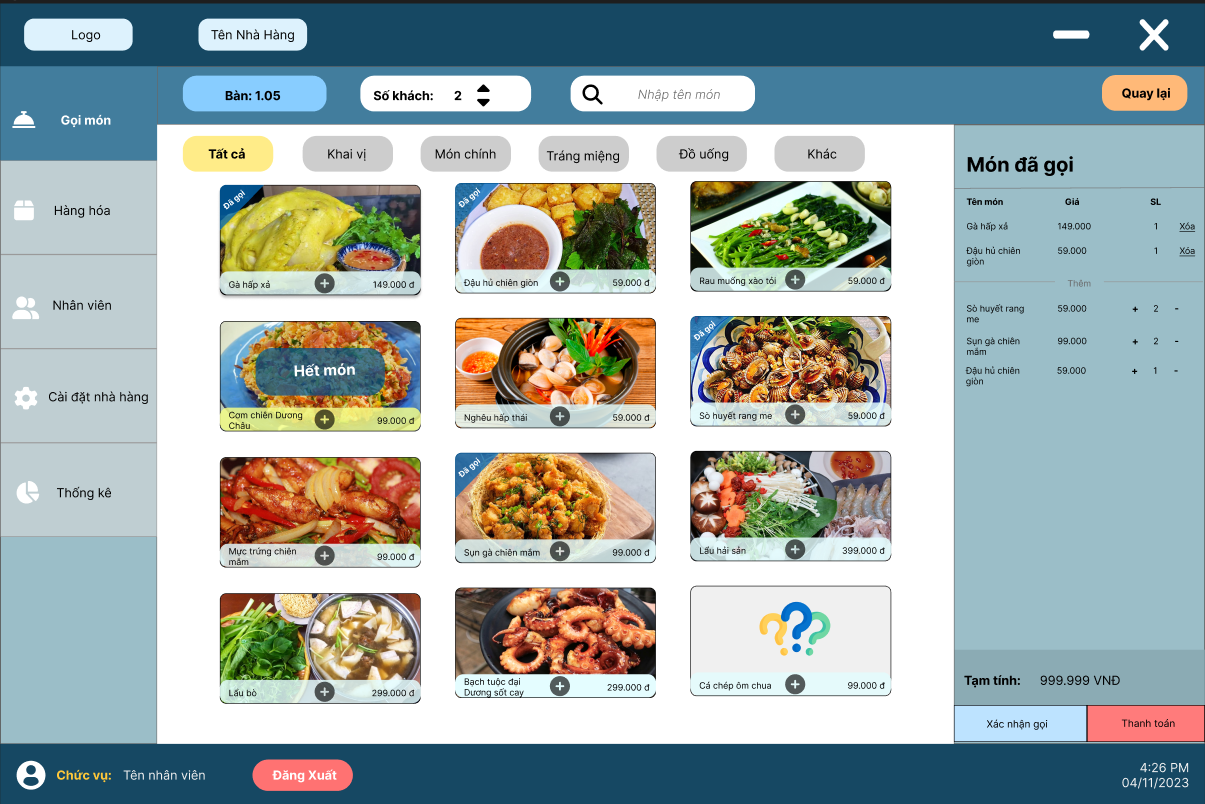
* Chức năng của form khôi phục mật khẩu là xác nhận mã được gửi qua mail để cấp quyền đặt lại mật khẩu mới cho tài khoản

#### Thiết kế Form gọi món



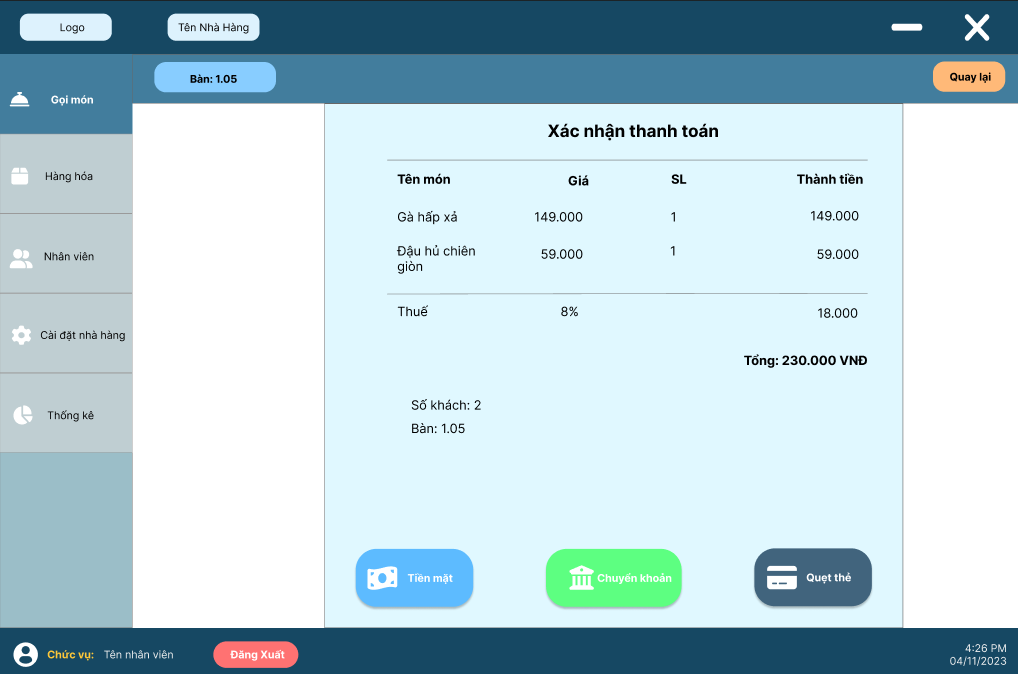
* Chức năng của form gọi món là hiển thị các khu và bàn có trong khu

#### Thiết kế Form gọi món chi tiết



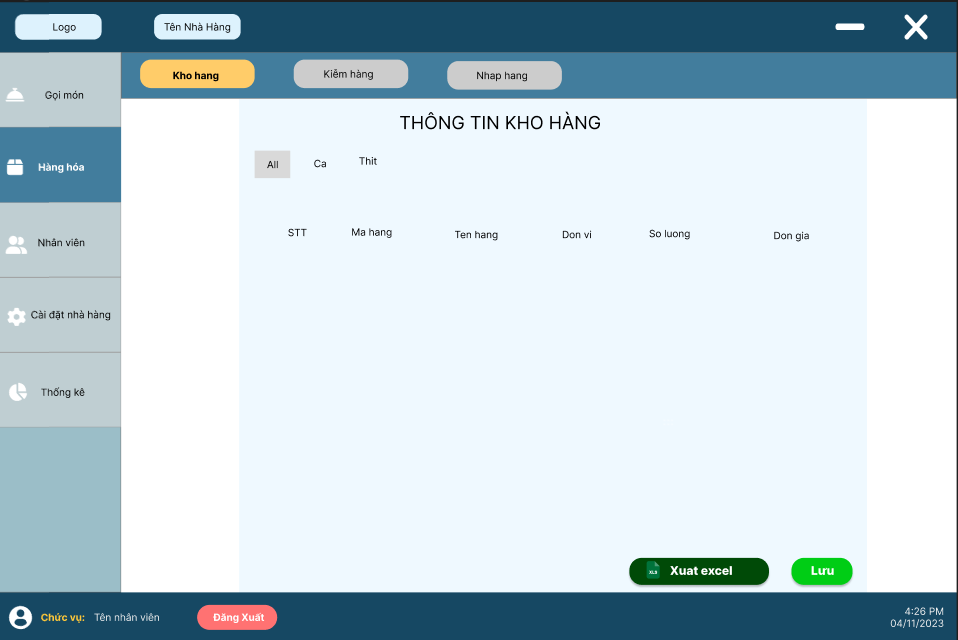
* Chức năng của Form gọi món chi tiết là hiện thị số bàn được chọn, số khách hàng có trong bàn và giao diện gọi món, bên cạnh là danh sách các món đã được thêm vào phần xác nhận gọi món

#### Thiết kế Form trang thanh toán



* Chức năng của form thanh toán là hiện thị tên, giá và số lượng các món đã được sử dụng, hiển thị số tổng hóa đơn và có thể thanh toán bằng 2 hình thức là tiền mặt, chuyển khoản.

#### Thiết kế Form hàng hóa



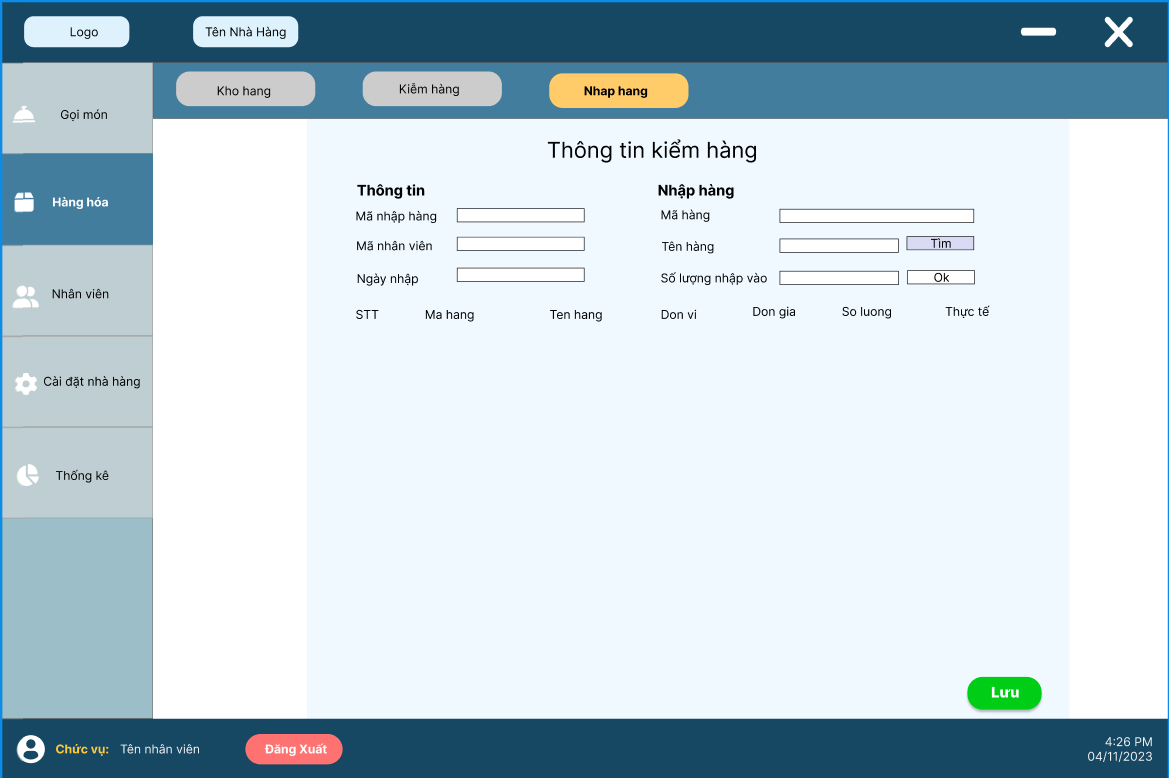
* Chức năng của from hàng hóa là hiển thị danh sách, kiểm hàng, nhập số hàng hóa trong kho

#### Thiết kế Form kiểm hàng



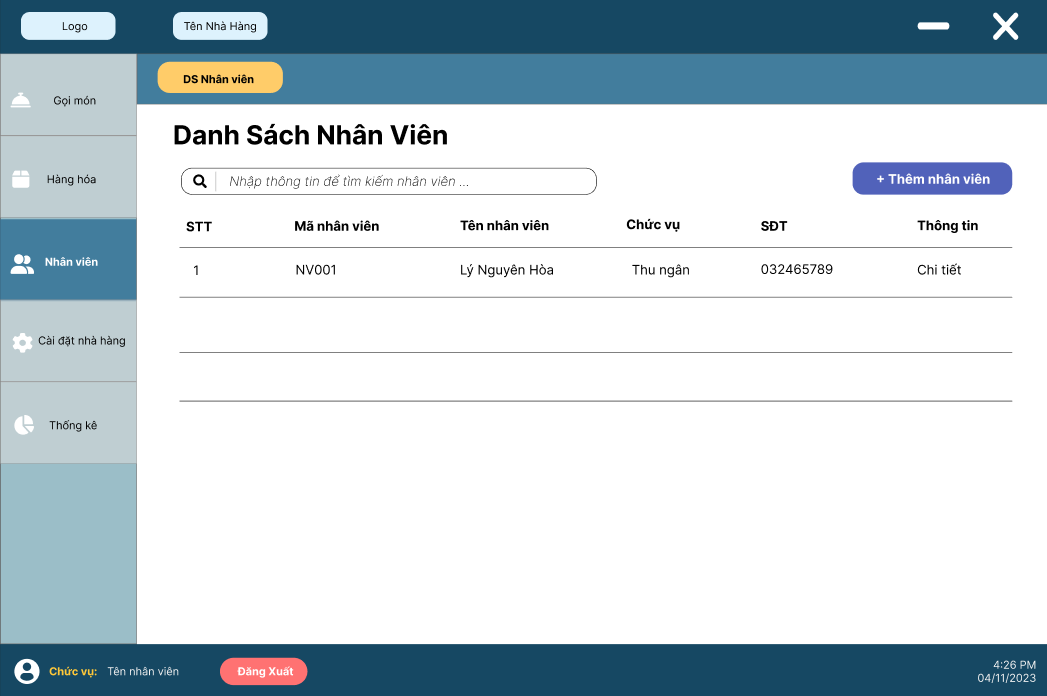
* Chức năng kiểm hàng cập nhật số lượng thực tế vào trong database.

#### Thiết kế Form nhập hàng



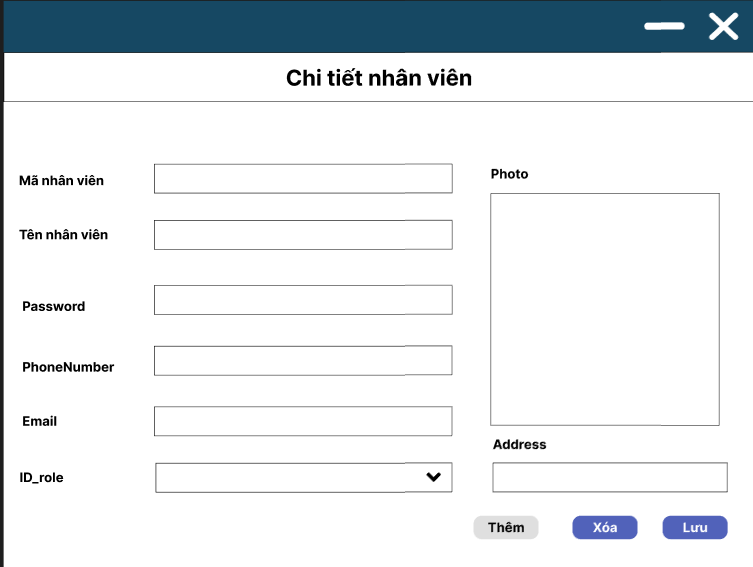
* Chức năng nhập hàng thêm hàng hóa cần thiết.

#### Thiết kế quản Form lý nhân viên



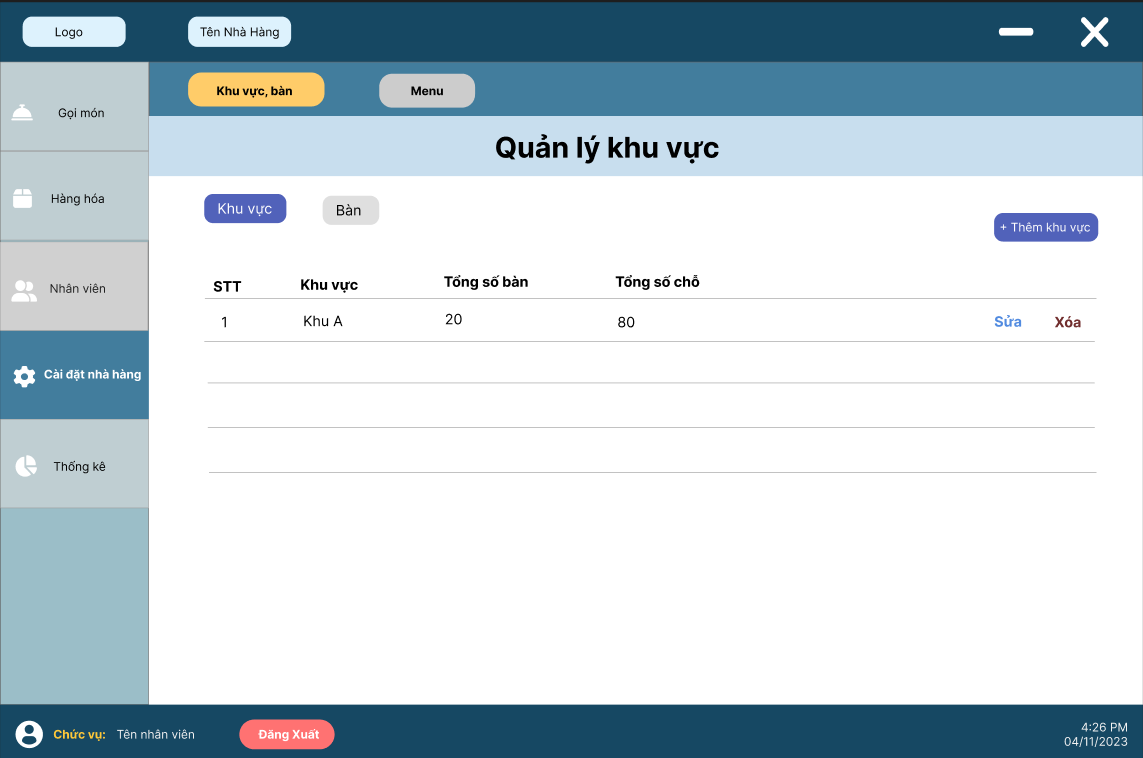
* Hiện thị thông tin nhân viên

#### Thiết kế Form chi tiết nhân viên

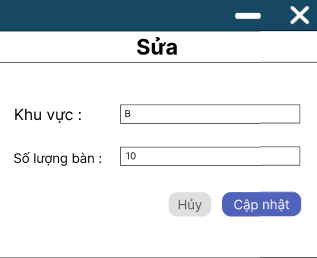


* Thêm, xóa, cập nhật thông tin nhân viên

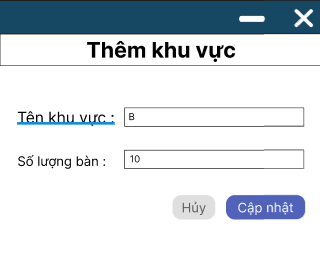
#### Thiết kế Form quản lý khu vực



* Hiện thị các thông tin liên quan cần thiết đến bàn

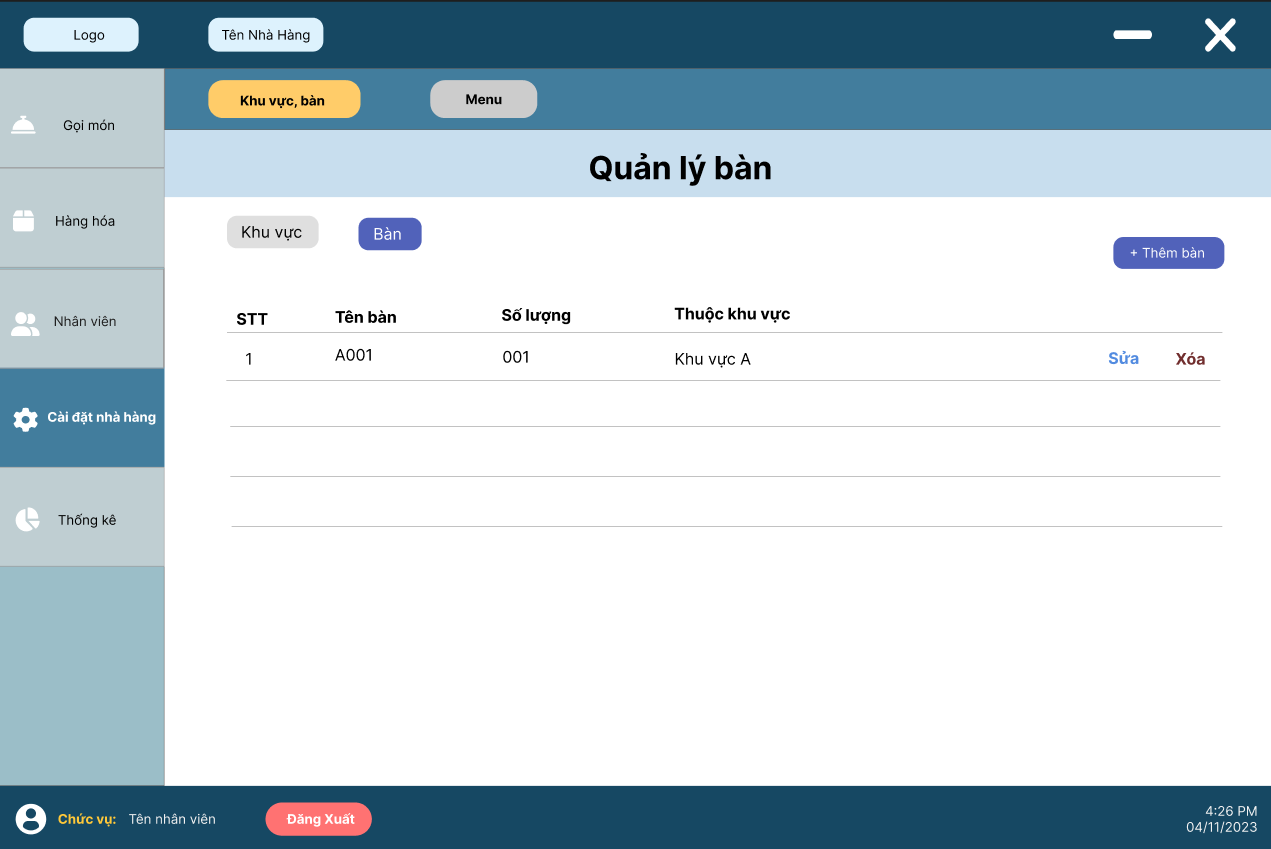


* Cập nhật tên khu vực

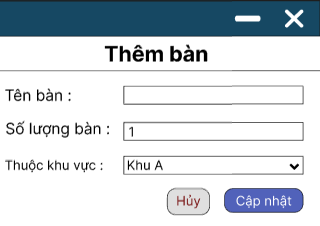


* Thêm khu vực

#### Thiết kế Form quản lý bàn



* Hiện thị thông tin cần thiết về bàn

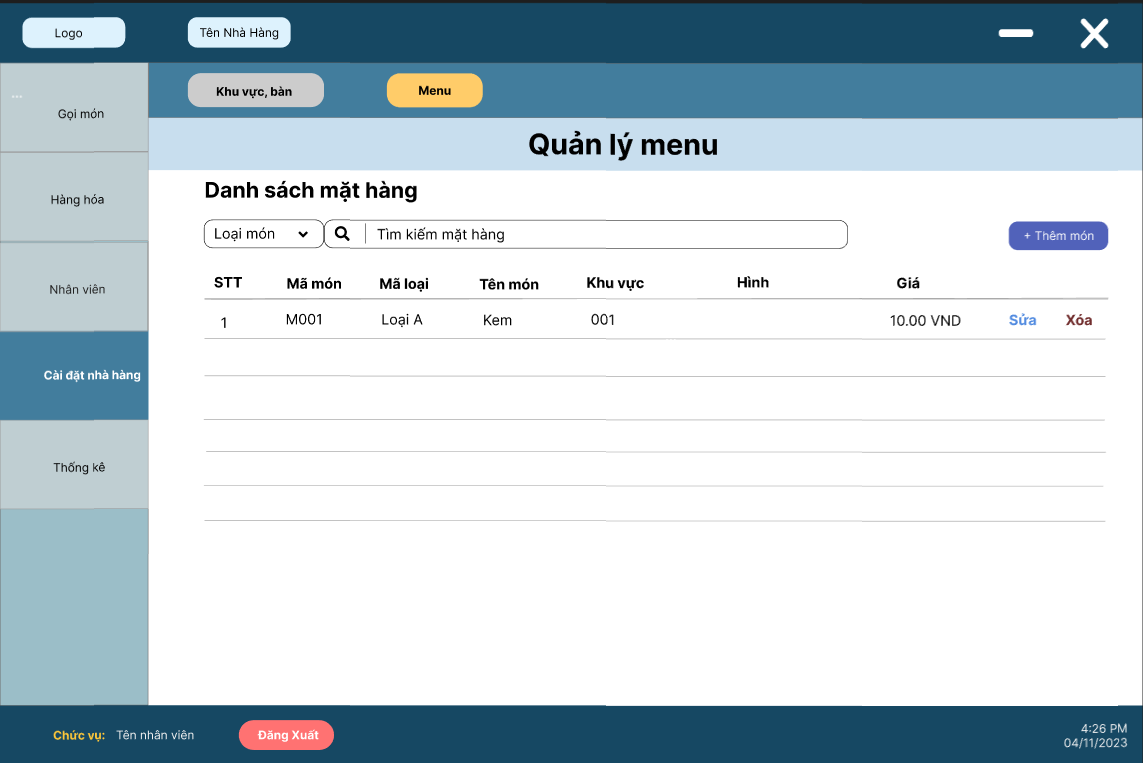


* Thêm bàn mới

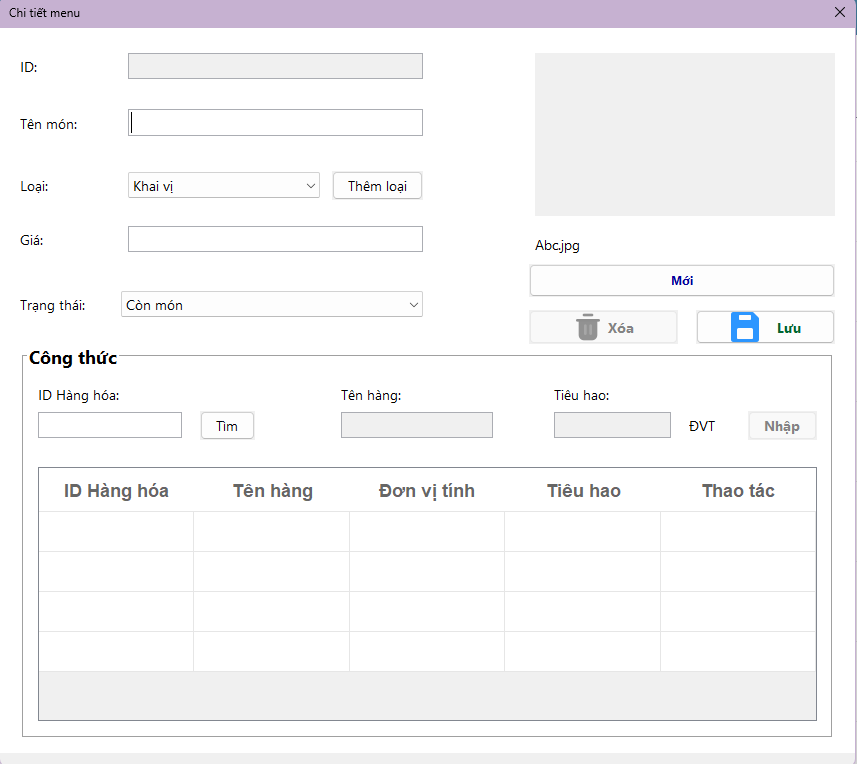


* Cập nhật thông tin cho bàn

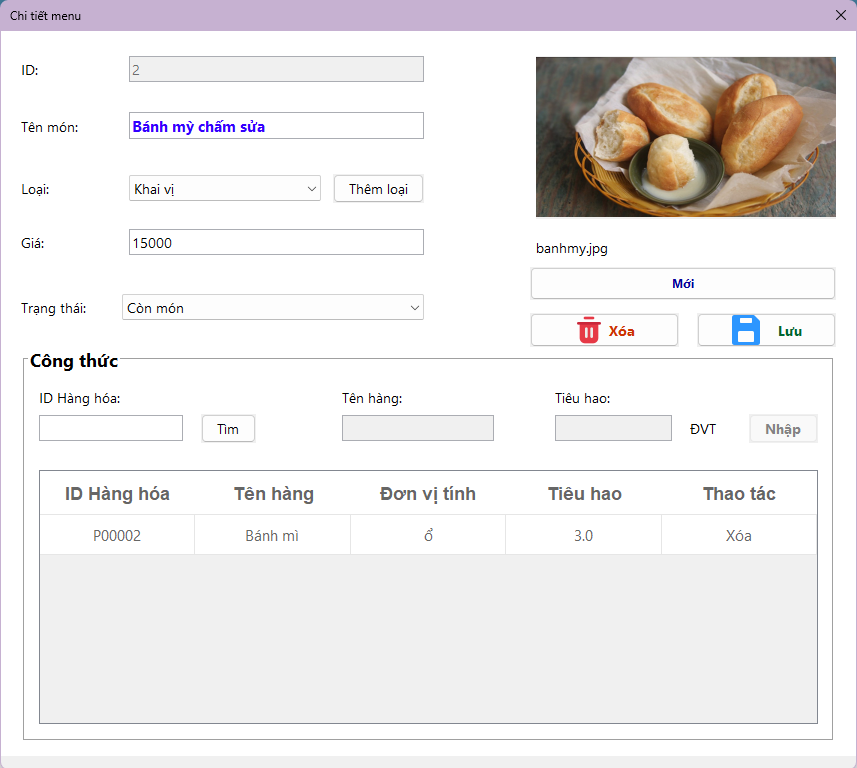
#### Thiết kế Form quản lý menu



* Hiện thị tất cả các món trong menu



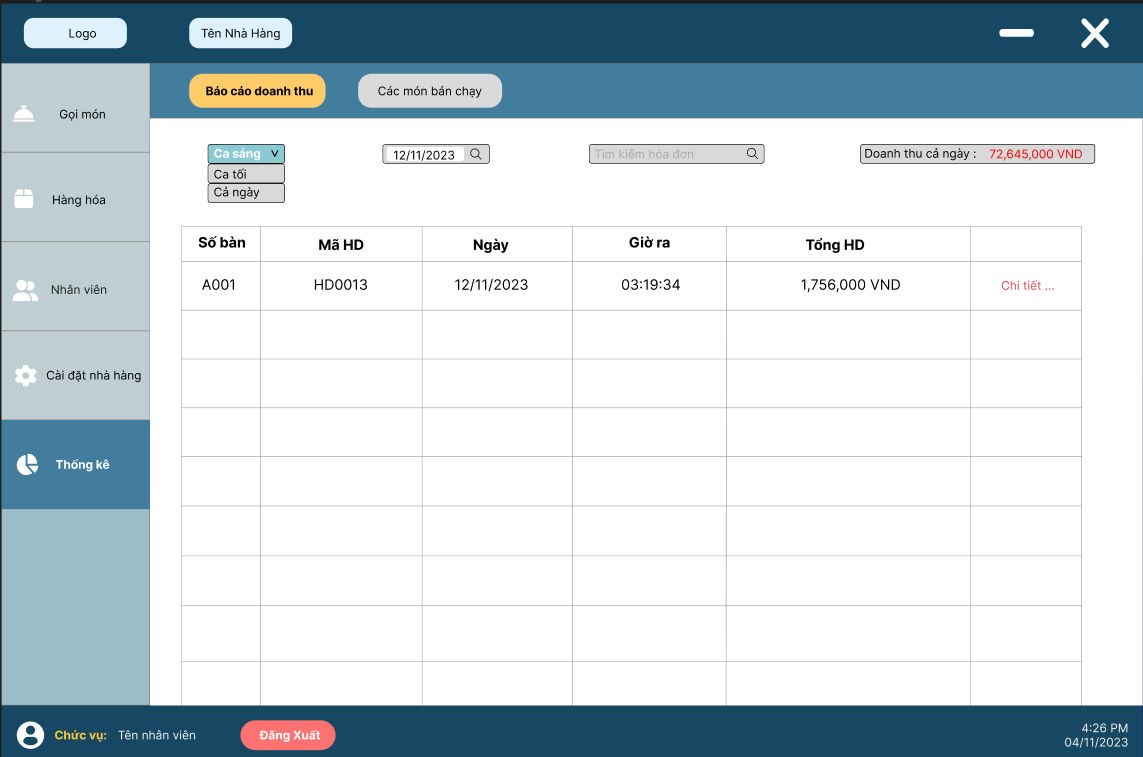
* Thêm món mới



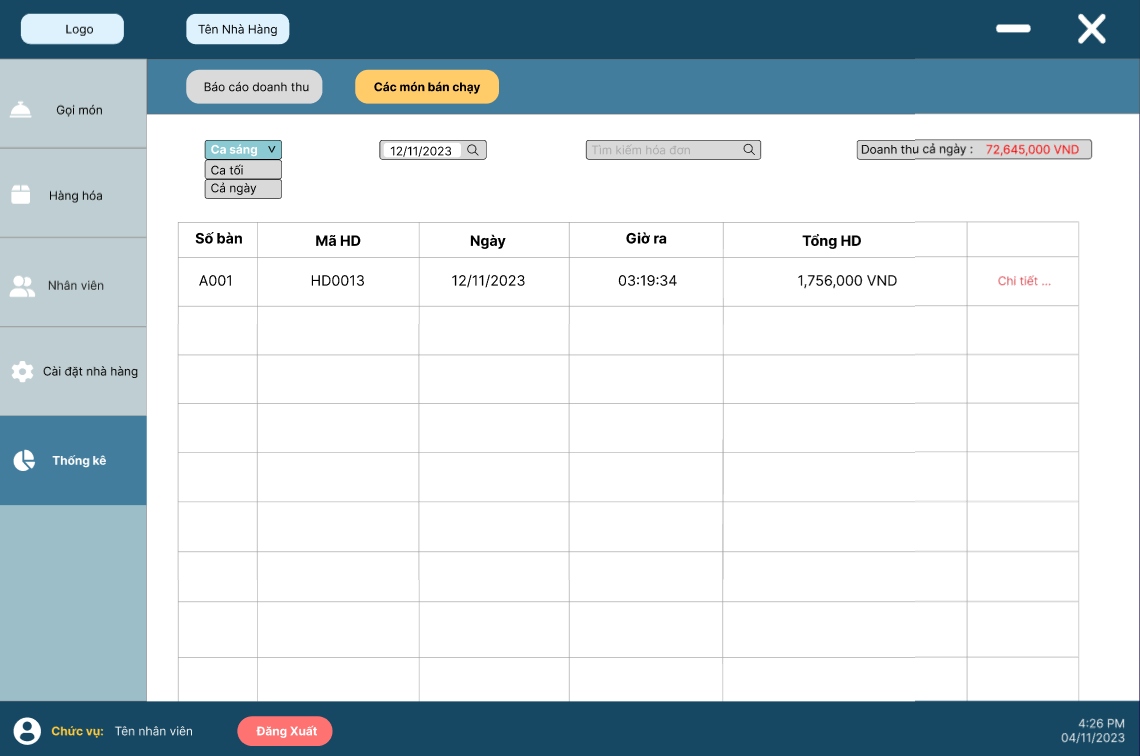
* Cập nhật thông tin của món có trong menu

#### Thiết kế Form thống kê

* Báo cáo doanh thu theo ca, ngày tháng năm



* Best seller, các món bán chạy nhất



#### Thiết kế Form giới thiệu

* Bổ sung

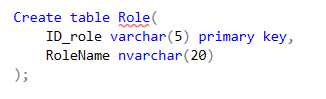
# Thực hiện viết mã

## Viết mã tạo CSDL



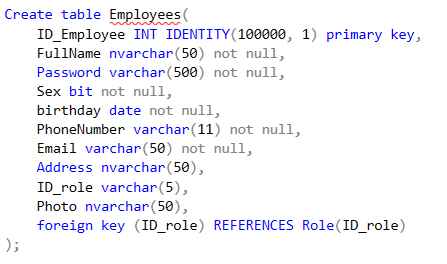
* Câu lệnh tạo database tên là restaurantData.

#### Tạo CSDL cho bảng Role



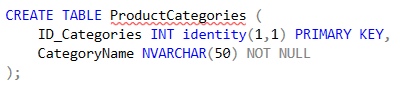
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_role | varchar(5) primary key |
| RoleName | RoleName nvarchar(20) |

#### Tạo CSDL cho bảng Employees



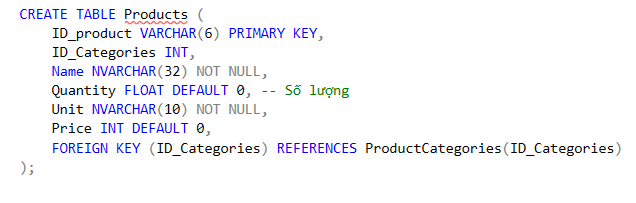
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Employee | INT IDENTITY(100000, 1) primary key |
| FullName | nvarchar(50) not null |
| Password | varchar(500) not null |
| Sex | bit not null |
| birthday | date not null, |
| PhoneNumber | varchar(11) not null |
| Email | varchar(50) not null |
| Address | nvarchar(50) |
| ID\_role | varchar(5) foreign key |
| Photo | nvarchar(50) |

#### Tạo CSDL cho bảng Table



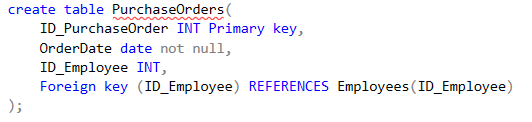
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Categories | INT identity(1,1) PRIMARY KEY |
| CategoryName | NVARCHAR(50) NOT NULL |

#### Tạo CSDL cho bảng Products



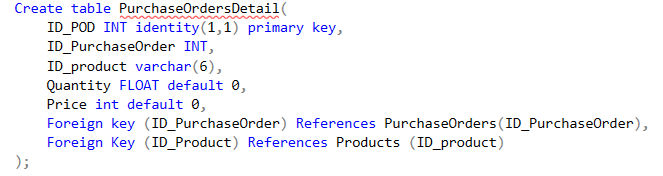
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_product | VARCHAR(6) PRIMARY KEY |
| ID\_Categories | FOREIGN KEY |
| Name | NVARCHAR(32) NOT NULL |
| Quantity | FLOAT DEFAULT 0 |
| Unit | NVARCHAR(10) NOT NULL |
| Price | INT DEFAULT 0 |

#### Tạo CSDL cho bảng PurchaseOrders



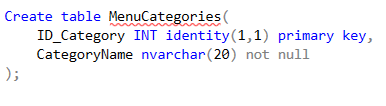
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_PurchaseOrder | INT Primary key |
| OrderDate | date not null |
| ID\_Employee | INT Foreign key |

#### Tạo CSDL cho bảng PurchaseOrdersDetail



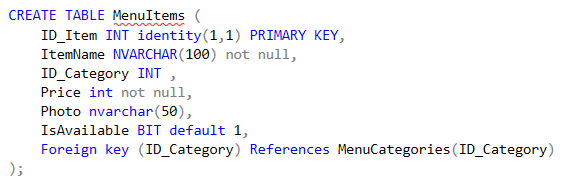
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_POD | INT identity(1,1) primary key |
| ID\_PurchaseOrder | INT Foreign key |
| ID\_product | varchar(6) Foreign Key |
| Quantity | FLOAT default 0 |
| Price | int default 0 |

#### Tạo CSDL cho bảng MenuCategories



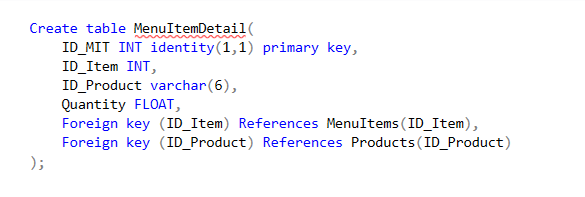
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Category | INT identity(1,1) primary key |
| CategoryName | nvarchar(20) not null |

#### Tạo CSDL cho bảng MenuItems



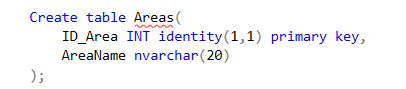
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Item | INT identity(1,1) PRIMARY KEY |
| ItemName | NVARCHAR(100) not null |
| ID\_Category | INT Foreign key |
| Price | int not null |
| Photo | nvarchar(50) |
| IsAvailable | BIT default 1 |

#### Tạo CSDL cho bảng MenuItemDetail



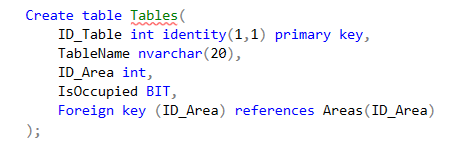
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_MIT | INT identity(1,1) primary key |
| ID\_Item | INT Foreign key |
| ID\_Product | varchar(6) Foreign key |
| Quantity | FLOAT |

#### Tạo CSDL cho bảng Areas



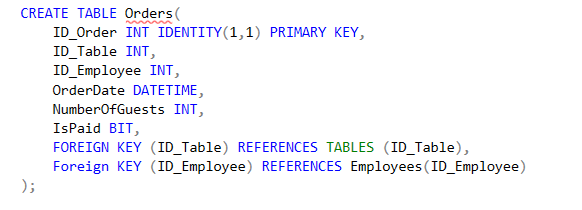
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Area | INT identity(1,1) primary key |
| AreaName | nvarchar(20) |

#### Tạo CSDL cho bảng Tables



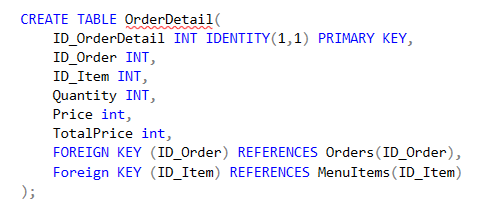
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Table | int identity(1,1) primary key |
| TableName | nvarchar(20) |
| ID\_Area | Int Foreign key |
| IsOccupied | BIT |

#### Tạo CSDL cho bảng Orders



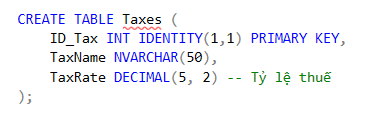
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Order | INT IDENTITY(1,1) PRIMARY KEY |
| ID\_Table | INT FOREIGN KEY |
| ID\_Employee | INT FOREIGN KEY |
| OrderDate | DATETIME |
| NumberOfGuests | INT |
| IsPaid | BIT |

#### Tạo CSDL cho bảng OrderDetail



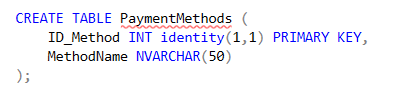
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_OrderDetail | INT IDENTITY(1,1) PRIMARY KEY |
| ID\_Order | INT FOREIGN KEY |
| ID\_Item | INT FOREIGN KEY |
| Quantity | INT |
| Price | INT |
| TotalPrice | INT |

#### Tạo CSDL cho bảng Taxes



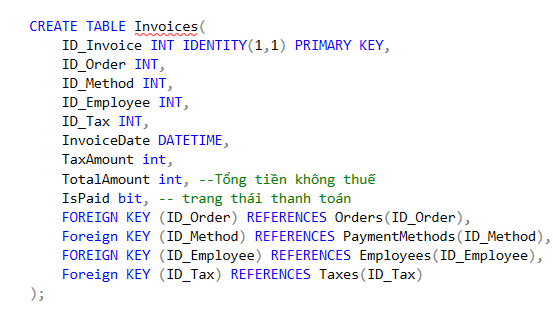
|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Tax | INT IDENTITY(1,1) PRIMARY KEY |
| TaxName | NVARCHAR(50) |
| TaxRate | DECIMAL(5, 2) |

#### Tạo CSDL cho bảng PaymentMethods



|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Method | INT identity(1,1) PRIMARY KEY |
| MethodName | NVARCHAR(50) |

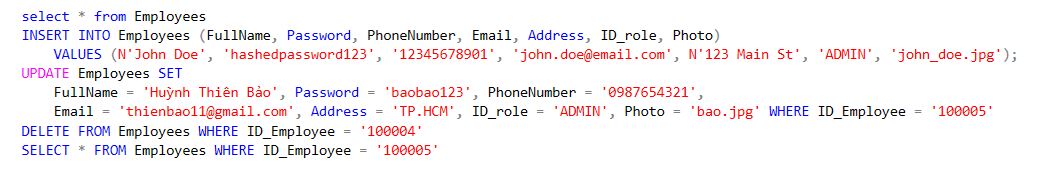
#### Tạo CSDL cho bảng Invoices



|  |  |
| --- | --- |
| **Tên cột** | **Kiểu dữ liệu** |
| ID\_Invoice | INT IDENTITY(1,1) PRIMARY KEY |
| ID\_Order | INT FOREIGN KEY |
| ID\_Method | INT FOREIGN KEY |
| ID\_Employee | INT FOREIGN KEY |
| ID\_Tax | INT FOREIGN KEY |
| InvoiceDate | DATETIME |
| TaxAmount | INT |
| TotalAmount | INT |
| IsPaid | INT |

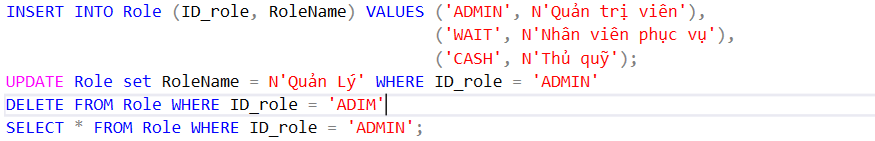
### SQL truy vấn và thao tác

#### SQL đối với Bảng Employees



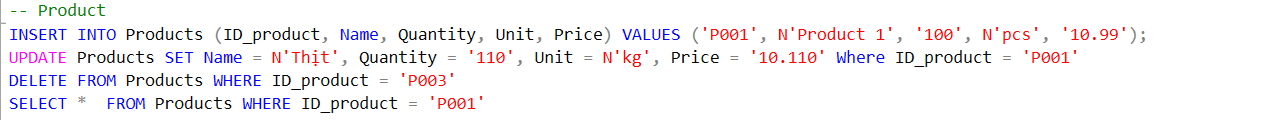
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Employees |
| Insert into | INSERT INTO Employees (FullName, Password, PhoneNumber, Email, Address, ID\_role, Photo) VALUES (?, ?, ?, ?, ?, ?, ?); |
| Update | UPDATE Employees SET FullName = ?, Password = ?, PhoneNumber = ?,  Email = ?, Address = ?, ID\_role = ?, Photo = ? WHERE ID\_Employee = ? |
| Delete | DELETE FROM Employees WHERE ID\_Employee = ? |
| Select by ID | SELECT \* FROM Employees WHERE ID\_Employee = ? |

#### SQL đối với Bảng Role



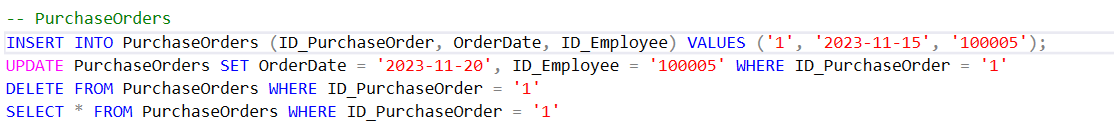
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Role |
| Insert into | INSERT INTO Role (ID\_role, RoleName) VALUES (?,?); |
| Update | UPDATE Role set RoleName = ? WHERE ID\_role = ? |
| Delete | DELETE FROM Role WHERE ID\_role = ? |
| Select by ID | SELECT \* FROM Role WHERE ID\_role = ? |

#### SQL đối với Bảng Product



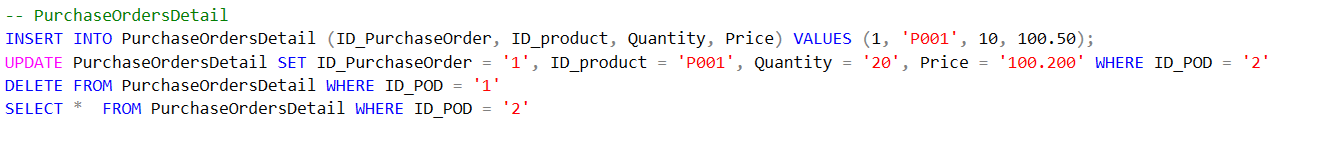
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Product |
| Insert into | INSERT INTO Products (ID\_product, Name, Quantity, Unit, Price) VALUES (?,?,?,?,?); |
| Update | UPDATE Products SET Name = ?, Quantity = ?, Unit = ?, Price = ? Where ID\_product = ? |
| Delete | DELETE FROM Products WHERE ID\_product = ? |
| Select by ID | SELECT \* FROM Products WHERE ID\_product = ? |

#### SQL đối với Bảng PurchaseOrder



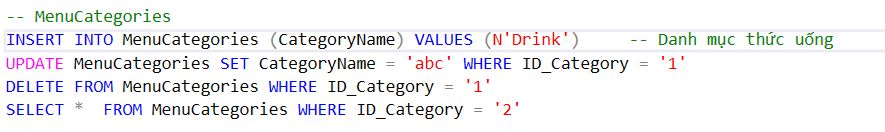
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from PurchaseOrder |
| Insert into | INSERT INTO PurchaseOrders (ID\_PurchaseOrder, OrderDate, ID\_Employee) VALUES (?, ?, ?); |
| Update | UPDATE PurchaseOrders SET OrderDate = ?, ID\_Employee = ? WHERE ID\_PurchaseOrder = ? |
| Delete | DELETE FROM PurchaseOrders WHERE ID\_PurchaseOrder = ? |
| Select by ID | SELECT \* FROM PurchaseOrders WHERE ID\_PurchaseOrder = ? |

#### SQL đối với Bảng PurchaseOrderDetail



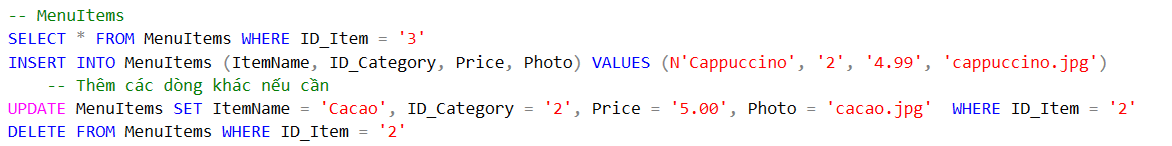
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from PurchaseOrderDetail |
| Insert into | INSERT INTO PurchaseOrdersDetail (ID\_PurchaseOrder, ID\_product, Quantity, Price) VALUES (?, ?, ?, ?); |
| Update | UPDATE PurchaseOrdersDetail SET ID\_PurchaseOrder = ?, ID\_product = ?, Quantity = ?, Price = ? WHERE ID\_POD = ? |
| Delete | DELETE FROM PurchaseOrdersDetail WHERE ID\_POD =? |
| Select by ID | SELECT \* FROM PurchaseOrdersDetail WHERE ID\_POD = ? |

#### SQL đối với Bảng MenuCategories



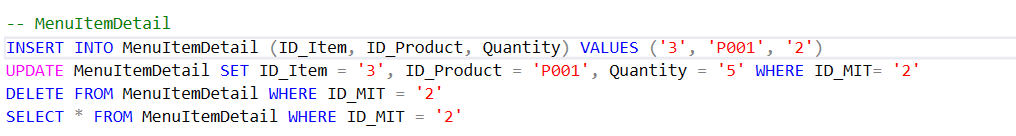
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from MenuCategories |
| Insert into | INSERT INTO MenuCategories (CategoryName) VALUES (?); |
| Update | UPDATE MenuCategories SET CategoryName = ? WHERE ID\_Category = ? |
| Delete | DELETE FROM MenuCategories WHERE ID\_Category = ? |
| Select by ID | SELECT \* FROM MenuCategories WHERE ID\_Category = ? |

#### SQL đối với Bảng MenuItems



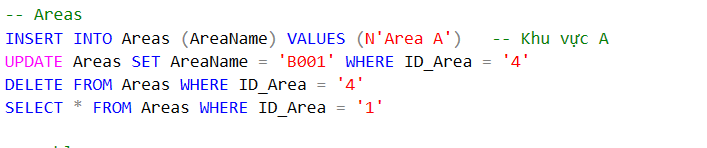
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from MenuItems |
| Insert into | INSERT INTO MenuItems (ItemName, ID\_Category, Price, Photo) VALUES (?, ?, ?, ?); |
| Update | UPDATE MenuItems SET ItemName = ?, ID\_Category = ?, Price = ?, Photo = ? WHERE ID\_Item = ? |
| Delete | DELETE FROM MenuItems WHERE ID\_Item = ? |
| Select by ID | SELECT \* FROM MenuItems WHERE ID\_Item = ? |

#### SQL đối với Bảng MenuItemDetail



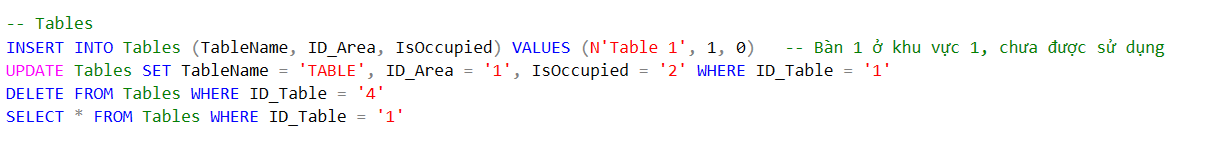
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from MenuItemDetail |
| Insert into | INSERT INTO MenuItemDetail (ID\_Item, ID\_Product, Quantity) VALUES (?, ?, ?); |
| Update | UPDATE MenuItemDetail SET ID\_Item = ?, ID\_Product = ?, Quantity = ? WHERE ID\_MIT= ? |
| Delete | DELETE FROM MenuItemDetail WHERE ID\_MIT = ? |
| Select by ID | SELECT \* FROM MenuItemDetail WHERE ID\_MIT = ? |

#### SQL đối với Bảng Areas



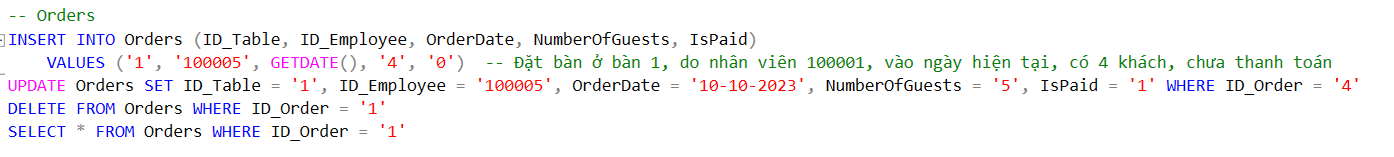
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Areas |
| Insert into | INSERT INTO Areas (AreaName) VALUES (?); |
| Update | UPDATE Areas SET AreaName = 'B001' WHERE ID\_Area = ? |
| Delete | DELETE FROM Areas WHERE ID\_Area = ? |
| Select by ID | SELECT \* FROM Areas WHERE ID\_Area = ? |

#### SQL đối với Bảng Tables



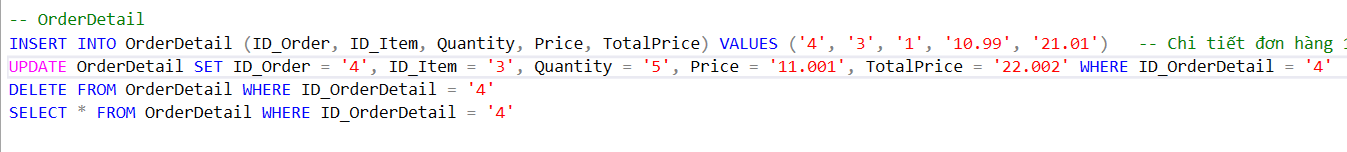
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Areas |
| Insert into | INSERT INTO Tables (TableName, ID\_Area, IsOccupied) VALUES (?, ? ,? ) |
| Update | UPDATE Tables SET TableName = ?, ID\_Area = ?, IsOccupied = ? WHERE ID\_Table = ? |
| Delete | DELETE FROM Tables WHERE ID\_Table = ? |
| Select by ID | SELECT \* FROM Tables WHERE ID\_Table = ? |

#### SQL đối với Bảng Orders



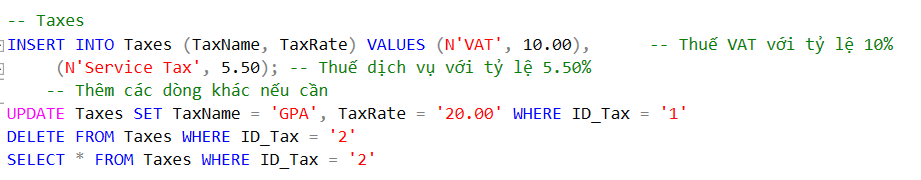
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Orders |
| Insert into | INSERT INTO Orders (ID\_Table, ID\_Employee, OrderDate, NumberOfGuests, IsPaid) VALUES (?, ?, ?, ?, ?) |
| Update | UPDATE Orders SET ID\_Table = ?, ID\_Employee = ?, OrderDate = ?, NumberOfGuests = ?, IsPaid = ? WHERE ID\_Order = ? |
| Delete | DELETE FROM Orders WHERE ID\_Order = ? |
| Select by ID | SELECT \* FROM Orders WHERE ID\_Order = ? |

#### SQL đối với Bảng OrderDetail



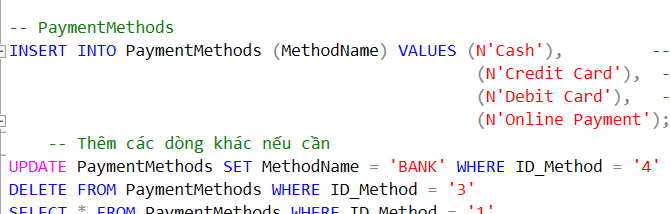
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from OrderDetail |
| Insert into | INSERT INTO OrderDetail (ID\_Order, ID\_Item, Quantity, Price, TotalPrice) VALUES (?, ?, ?, ?, ?) |
| Update | UPDATE OrderDetail SET ID\_Order = ?, ID\_Item = ?, Quantity = ?, Price = ?, TotalPrice = ? WHERE ID\_OrderDetail = ? |
| Delete | DELETE FROM OrderDetail WHERE ID\_OrderDetail = ? |
| Select by ID | SELECT \* FROM OrderDetail WHERE ID\_OrderDetail = ? |

#### SQL đối với Bảng Taxes



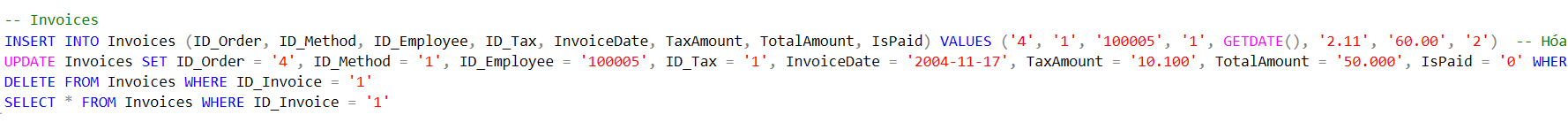
|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Taxes |
| Insert into | INSERT INTO Taxes (TaxName, TaxRate) VALUES (?,?); |
| Update | UPDATE Taxes SET TaxName = ?, TaxRate = ? WHERE ID\_Tax = ? |
| Delete | DELETE FROM Taxes WHERE ID\_Tax = ? |
| Select by ID | SELECT \* FROM Taxes WHERE ID\_Tax = ? |

#### SQL đối với Bảng PaymentMethods



|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from PaymentMethods |
| Insert into | INSERT INTO PaymentMethods (MethodName) VALUES (?); |
| Update | UPDATE PaymentMethods SET MethodName = ? WHERE ID\_Method = ? |
| Delete | DELETE FROM PaymentMethods WHERE ID\_Method = ? |
| Select by ID | SELECT \* FROM PaymentMethods WHERE ID\_Method = ? |

#### SQL đối với Bảng Invoices



|  |  |
| --- | --- |
| **Thực thi** | **Câu lệnh** |
| Select all | select \* from Invoices |
| Insert into | INSERT INTO Invoices (ID\_Order, ID\_Method, ID\_Employee, ID\_Tax, InvoiceDate, TaxAmount, TotalAmount, IsPaid) VALUES (?, ?, ?, ?, GETDATE(), ?, ?, ?); |
| Update | UPDATE Invoices SET ID\_Order = ?, ID\_Method = ?, ID\_Employee = ?, ID\_Tax = ?, InvoiceDate = ?, TaxAmount = ?, TotalAmount = ?, IsPaid = ? WHERE ID\_Invoice = ? |
| Delete | DELETE FROM Invoices WHERE ID\_Invoice = ? |
| Select by ID | SELECT \* FROM Invoices WHERE ID\_Invoice = ? |

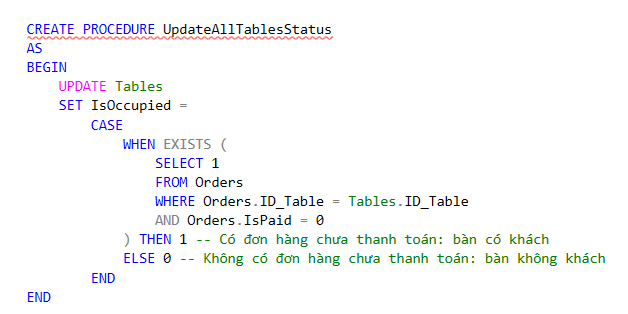
### Các thủ tục lưu tổng hợp thống kê

#### Procedure 1

#### Procedure 2

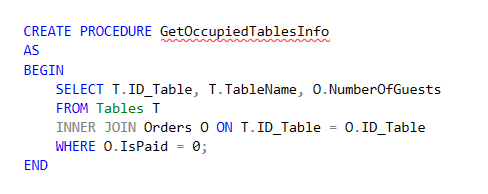
### Thủ tục lưu trữ cho đơn hàng, bàn

#### Procedure đơn hàng, bàn, món



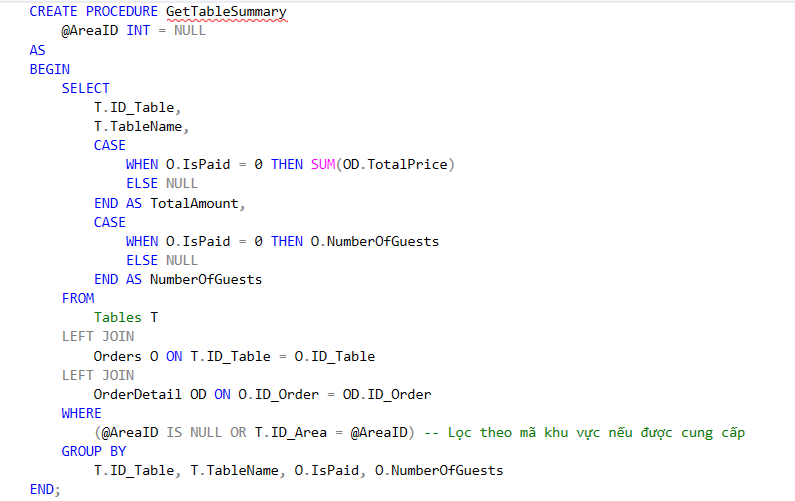
* Nhiệm vụ của thủ tục này là cập nhật trạng thái của tất cả các bàn trong bảng **Tables** dựa trên thông tin từ bảng **Orders**.

#### Procedure bảng chi tiết thông tin



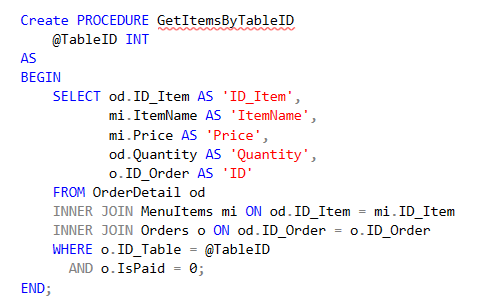
* Nhiệm vụ của thủ tục này là trả về thông tin về các bàn đang có khách (được sử dụng trong các đơn hàng chưa thanh toán) từ hai bảng là **Tables** và **Orders**.

#### Procedure bảng tóm tắt table



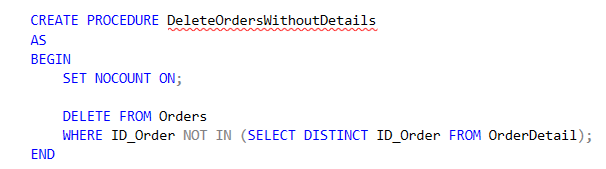
* Nhiệm vụ của thủ tục là trả về một tóm tắt về thông tin của các bàn trong một khu vực cụ thể hoặc toàn bộ nhà hàng.

#### Procedure bảng món, bàn



* Thủ tục này sẽ trả về một bảng kết quả với thông tin về các mục trong đơn hàng của một bàn cụ thể, bao gồm ID\_Item, ItemName, Price, Quantity, và ID.

#### Procedure bảng đơn hàng, chi tiết đơn hàng



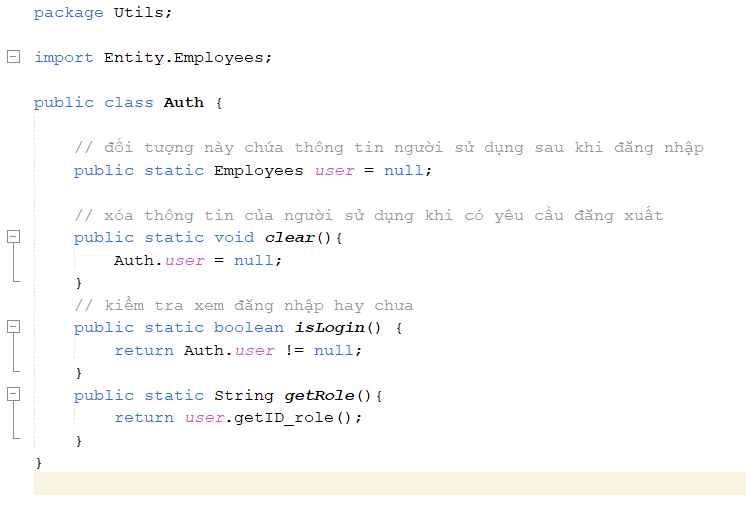
* Thủ tục này sẽ thực hiện việc xóa các đơn hàng từ bảng Orders mà không có chi tiết đặt hàng liên quan. Điều này có thể hữu ích để dọn dẹp dữ liệu và giữ cho cơ sở dữ liệu sạch sẽ bằng cách loại bỏ các đơn hàng không có nội dung chi tiết.

## Lập trình JDBC

### Lớp hỗ trợ

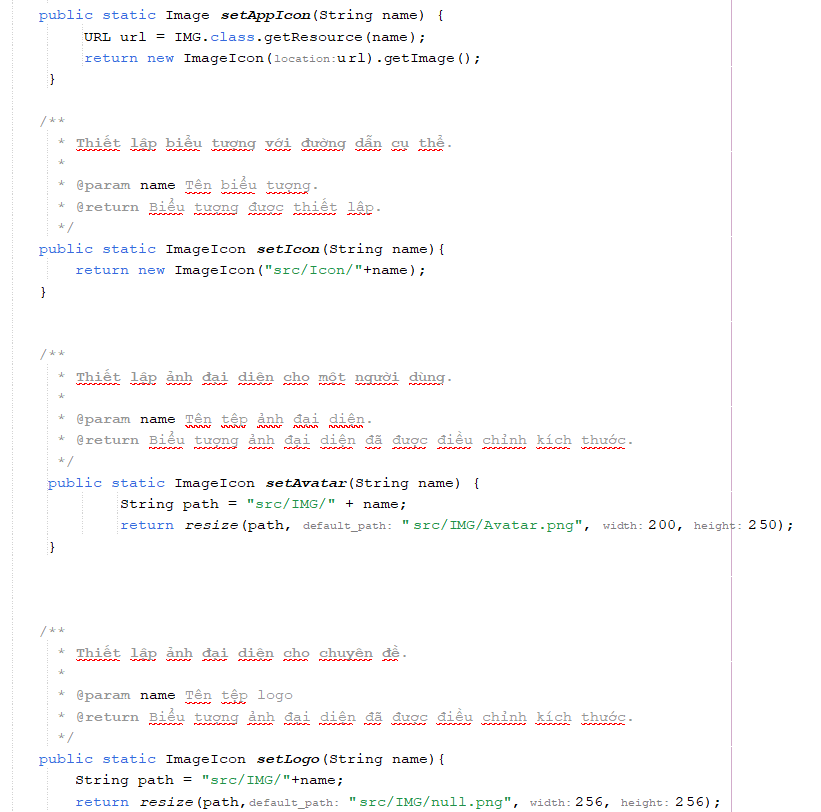
#### Lớp tiện ích Auth

* Cung cấp các chức năng cơ bản để quản lý thông tin người sử dụng sau khi đăng nhập, kiểm tra trạng thái đăng nhập, lấy vai trò của người sử dụng và xóa thông tin người sử dụng khi họ đăng xuất.



#### Lớp tiện ích EmailSender

* Có nhiệm vụ gửi email từ ứng dụng.



#### Lớp tiện ích Encryption

* Cung cấp một phương thức để băm mật khẩu sử dụng thuật toán SHA-256. Việc này thường được sử dụng để bảo vệ mật khẩu người dùng trong quá trình xác thực, nơi chỉ lưu trữ giá trị băm thay vì mật khẩu thực sự để tăng cường bảo mật.

#### Lớp tiện ích FileControl

* Cung cấp các phương thức để ghi, đọc và xóa tệp tin, giúp quản lý dữ liệu trong các tệp tin từ Java. Các phương thức này có thể hữu ích trong việc lưu trữ và quản lý dữ liệu với các tệp tin trong ứng dụng.

#### Lớp tiện ích IMG

* Cung cấp các tiện ích cho việc xử lý ảnh và biểu tượng, bao gồm thiết lập biểu tượng ứng dụng, thiết lập ảnh đại diện, thay đổi kích thước ảnh, và sao chép tệp tin.

|  |
| --- |
| package Utils;  import java.awt.Image;  import java.awt.image.BufferedImage;  import java.io.File;  import java.io.IOException;  import static java.lang.String.format;  import java.net.URL;  import java.nio.file.Files;  import java.nio.file.StandardCopyOption;  import javax.imageio.ImageIO;  import javax.swing.ImageIcon;  /\*\*  \* Thư viện `IMG` chứa các phương thức hỗ trợ xử lý ảnh và biểu tượng (icon) cho ứng dụng Java Swing.  \*/  public class IMG {    /\*\*  \* Đường dẫn mặc định cho biểu tượng ứng dụng.  \*/  private static final String DEFAULT\_ICON\_PATH = "/Icon/logo.png";  /\*\*  \* Thiết lập biểu tượng cho ứng dụng.  \*  \* @return Biểu tượng ứng dụng.  \*/    public static Image setAppIcon() {  URL url = IMG.class.getResource(DEFAULT\_ICON\_PATH);  return new ImageIcon(url).getImage();  }    /\*\*  \* Thiết lập biểu tượng cho ứng dụng với đường dẫn cụ thể.  \*  \* @param name Đường dẫn của biểu tượng.  \* @return Biểu tượng ứng dụng.  \*/  public static Image setAppIcon(String name) {  URL url = IMG.class.getResource(name);  return new ImageIcon(url).getImage();  }    /\*\*  \* Thiết lập biểu tượng với đường dẫn cụ thể.  \*  \* @param name Tên biểu tượng.  \* @return Biểu tượng được thiết lập.  \*/  public static ImageIcon setIcon(String name){  return new ImageIcon("src/Icon/"+name);  }      /\*\*  \* Thiết lập ảnh đại diện cho một người dùng.  \*  \* @param name Tên tệp ảnh đại diện.  \* @return Biểu tượng ảnh đại diện đã được điều chỉnh kích thước.  \*/  public static ImageIcon setAvatar(String name) {  String path = "src/IMG/" + name;  return resize(path, "src/IMG/Avatar.png", 200, 250);  }      /\*\*  \* Thiết lập ảnh đại diện cho chuyên đề.  \*  \* @param name Tên tệp logo  \* @return Biểu tượng ảnh đại diện đã được điều chỉnh kích thước.  \*/  public static ImageIcon setLogo(String name){  String path = "src/IMG/"+name;  return resize(path,"src/IMG/null.png", 256, 256);  }        /\*\*  \* Thay đổi kích thước của một ảnh ImageIcon.  \*  \* @param path Đường dẫn tới tệp ảnh.  \* @param width Chiều rộng mới.  \* @param height Chiều cao mới.  \* @return Biểu tượng ảnh đã được thay đổi kích thước.  \*/    public static ImageIcon resize(String path, int width, int height){  try {  // Đọc ảnh từ đường dẫn  File imageFile = new File(path);  if (!imageFile.exists()) {  System.out.println("Image not found: " + path);  imageFile = new File("src/IMG/null.png");  }  BufferedImage originalImg = ImageIO.read(imageFile);  // Thay đổi kích thước ảnh  Image scaledImage = originalImg.getScaledInstance(width, height, Image.SCALE\_SMOOTH);  // Tạo ImageIcon từ ảnh đã thay đổi kích thước  ImageIcon scaledIcon = new ImageIcon(scaledImage);  return scaledIcon;  } catch (IOException e) {  System.out.println(e.getMessage());  return null; // Trả về null nếu có lỗi  }  }    /\*\*  \* Thay đổi kích thước của một ảnh ImageIcon và cung cấp đường dẫn mặc định.  \*  \* @param path Đường dẫn tới tệp ảnh.  \* @param default\_path Đường dẫn mặc định nếu không tìm thấy ảnh.  \* @param width Chiều rộng mới.  \* @param height Chiều cao mới.  \* @return Biểu tượng ảnh đã được thay đổi kích thước.  \*/  public static ImageIcon resize(String path, String default\_path, int width, int height){  try {  // Đọc ảnh từ đường dẫn  File imageFile = new File(path);  if (!imageFile.exists()) {  System.out.println("Image not found: " + path);  imageFile = new File(default\_path);  }  BufferedImage originalImg = ImageIO.read(imageFile);  // Thay đổi kích thước ảnh  Image scaledImage = originalImg.getScaledInstance(width, height, Image.SCALE\_SMOOTH);  // Tạo ImageIcon từ ảnh đã thay đổi kích thước  ImageIcon scaledIcon = new ImageIcon(scaledImage);  return scaledIcon;  } catch (IOException e) {  e.printStackTrace();  System.out.println(e.getMessage());  return null; // Trả về null nếu có lỗi  }  }    /\*\*  \* Sao chép tệp từ một đường dẫn đến một vị trí đích.  \*  \* @param path Đường dẫn tới tệp nguồn.  \* @param to Đường dẫn đích.  \* @return true nếu sao chép thành công, false nếu có lỗi.  \*/  public static boolean save(String path, String to) {  try {  File sourceFile = new File(path);  File destinationDir = new File(to);  // Kiểm tra xem thư mục đích đã tồn tại chưa. Nếu chưa, tạo nó.  if (!destinationDir.exists()) {  destinationDir.mkdirs();  }  File destinationFile = new File(destinationDir, sourceFile.getName());  // Sao chép tệp ảnh từ vị trí đầu vào vào thư mục đích  Files.copy(sourceFile.toPath(), destinationFile.toPath(), StandardCopyOption.REPLACE\_EXISTING);    System.out.println("Save file " + sourceFile.getName() + " successfully.");  return true; // Trả về true nếu sao chép thành công  } catch (IOException e) {  e.printStackTrace();  System.out.println("Save file Error.");  return false; // Trả về false nếu có lỗi  }  }        } |

#### Lớp tiện ích Validate

* Phương thức hỗ trợ kiểm tra và xác nhận dữ liệu từ các thành phần giao diện người dùng.

|  |
| --- |
| package Utils;  import java.awt.Color;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.regex.Pattern;  import javax.swing.JOptionPane;  import javax.swing.JPasswordField;  import javax.swing.JTextField;  public class Validate {    public static boolean isNotEmpty(JTextField textField, String fieldName) {  String text = textField.getText().trim();  if (text.isEmpty()) {  msg.Warning("Không được bỏ trống!");  textField.setBackground(Color.yellow);    return false;  }  return true;  }    public static boolean isNotEmpty(JTextField textField, String fieldName, String lbl) {  String text = textField.getText().trim();  if (text.isEmpty()) {  msg.Warning("Không được bỏ trống "+lbl+"!");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  return true;  }  public static boolean isSoDuong(JTextField textField, String fieldName) {  String text = textField.getText().trim();  double num = Double.parseDouble(text);  if (num < 0) {  msg.Warning("Bạn phải nhập số dương!");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  return true;  }    public static boolean isSoDuong(JTextField textField, String fieldName, String lbl) {  String text = textField.getText().trim();  double num = Double.parseDouble(text);  if (num < 0) {  msg.Warning("Bạn phải nhập số dương "+lbl+"!");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  return true;  }    //danh cho text field không tiêu đề  public static boolean isLength(JTextField textFild, String str, int length){  if(str.trim().length() < length){  msg.Warning("Bạn phải nhập ít nhất "+length+" kí tự!");  textFild.setBackground(Color.yellow);  textFild.requestFocus();  return false;  }  return true;  }    //danh cho text field có tiêu đề  public static boolean isLength(JTextField textFild, String str, int length, String title){  if(str.trim().length() < length){  msg.Warning("Bạn phải nhập ít nhất "+length+" kí tự cho "+title+"!");  textFild.setBackground(Color.yellow);  textFild.requestFocus();  return false;  }  return true;  }    //Dành cho password không có tiêu đề  public static boolean isLength(JPasswordField textFild, String str, int length){  if(str.trim().length() < length){  msg.Warning("Bạn phải nhập ít nhất "+length+" kí tự!");  textFild.setBackground(Color.yellow);  textFild.requestFocus();  return false;  }  return true;  }  //Dành cho password có tiêu đề  public static boolean isLength(JPasswordField textFild, String str, int length, String title){  if(str.trim().length() < length){  msg.Warning("Bạn phải nhập ít nhất "+length+" kí tự cho "+title+"!");  textFild.setBackground(Color.yellow);  textFild.requestFocus();  return false;  }  return true;  }  public static boolean isEmail(JTextField textField, String fieldName) {  String text = textField.getText().trim();  if (!text.matches("^[a-zA-Z0-9+\_.-]+@[a-zA-Z0-9.-]+$")) {  msg.Error("Bạn phải nhập đúng định dang email!");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  return true;  }    public static boolean isPhoneNumber(JTextField textField, String fieldName) {  String text = textField.getText().trim();    // Sử dụng biểu thức chính quy để kiểm tra số điện thoại  if (!Pattern.matches("^\\d{10}$", text)) {  msg.Warning("Số điện thoại không hợp lệ!");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  return true;  }  public static boolean isDateDDMMYYYY(JTextField textField, String fieldName) {  String text = textField.getText().trim();    // Định dạng cho chuỗi ngày tháng "dd-MM-yyyy"  SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");  dateFormat.setLenient(false); // Tắt tính linh hoạt của định dạng    try {  // Chuyển chuỗi thành kiểu Date  Date date = dateFormat.parse(text);  return true;  } catch (ParseException e) {  msg.Warning("Ngày tháng năm không hợp lệ! Sử dụng định dạng 'dd-MM-yyyy'");  textField.setBackground(Color.yellow);  textField.requestFocus();  return false;  }  }    } |

#### Lớp tiện ích XDate

* Cung cấp các tiện ích cho việc chuyển đổi và xử lý ngày tháng trong ứng dụng Java, giúp làm cho các thao tác liên quan đến ngày tháng trở nên thuận tiện hơn.

#### Lớp tiện ích fNum

* Giúp quản lý việc định dạng và chuyển đổi số nguyên, làm cho hiển thị và xử lý số nguyên trở nên thuận tiện hơn trong ứng dụng của bạn.

#### Lớp tiện ích jdbc

* Quản lý kết nối và thao tác với cơ sở dữ liệu một cách thuận tiện trong ứng dụng của bạn.

|  |
| --- |
| package Utils;  import java.sql.\*;  public class jdbc {  //Giao thức kết nối  private static String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";  private static String dbUrl = "jdbc:sqlserver://localhost;database=restaurantData;encrypt=false";  private static String user = "sa";  private static String pass = "123123";  private static Connection cnn = null;  static {  try {  Class.forName(driver);  } catch (ClassNotFoundException e) {  System.out.println("Driver not found");  }  }  //Hàm connect  public static Connection getConnection() {  if (cnn == null) {  try {  cnn = (Connection) DriverManager.getConnection(dbUrl, user, pass);  System.out.println("Connection success to DataBase");  } catch (SQLException e) {  System.out.println("Connection error: " + e.getMessage());  }  }  return cnn;  }  //Hàm thực hiện truy vẫn  public static PreparedStatement getStmt(String sql, Object... args) throws SQLException {  //Kết nối  Connection connection = DriverManager.getConnection(dbUrl, user, pass);  PreparedStatement pstmt = null;  //Phân biệt câu truy vấn Proc và Bình thường  if (sql.trim().startsWith("{")) {  pstmt = connection.prepareCall(sql);  } else {  pstmt = connection.prepareStatement(sql);  }  //Gán các giá trị tham số vào câu truy vấn  for (int i = 0; i < args.length; i++) {  pstmt.setObject(i + 1, args[i]);  }  //Trả về câu truy vấn chưa run  return pstmt;  }  //Thực hiên update  public static int update(String sql, Object... args) {  try {  PreparedStatement stmt = getStmt(sql, args);  try {  return stmt.executeUpdate();  } finally {  stmt.getConnection().close();  }  } catch (Exception e) {  System.out.println(e.getMessage());  throw new RuntimeException();  }  }  //Câu lệnh truy vấn trả về tập giá trị  public static ResultSet query(String sql, Object... args) {  try {  PreparedStatement stmt = getStmt(sql, args);  return stmt.executeQuery();  } catch (SQLException e) {  throw new RuntimeException(e);  }  }  //Trả về một đối tượng duy nhất  public static Object value(String sql, Object... args) {  try {  //Dùng lại Hàm query  ResultSet rs = query(sql, args);  //Lấy giá trị đầu tiên, tránh trường hợp trả về nhiều giá trị  if (rs.next()) {  return rs.getObject(0);  }  rs.getStatement().getConnection().close();  } catch (SQLException e) {  throw new RuntimeException(e);  }  return null;  }          } |

#### Lớp tiện ích msg

* Các phương thức để hiển thị các hộp thoại thông báo (message dialog) của JOptionPane.

### Model class - Các lớp mô tả dữ liệu

#### Model Areas

|  |
| --- |
| package Entity;  import java.util.Objects;  public class Areas {  private int ID\_Area;  private String AreaName;  public Areas(int ID\_Area, String AreaName) {  this.ID\_Area = ID\_Area;  this.AreaName = AreaName;  }  public Areas() {  }  public int getID\_Area() {  return ID\_Area;  }  public void setID\_Area(int ID\_Area) {  this.ID\_Area = ID\_Area;  }  public String getAreaName() {  return AreaName;  }  public void setAreaName(String AreaName) {  this.AreaName = AreaName;  }    @Override  public String toString() {  return this.AreaName;  }  @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null || getClass() != obj.getClass()) {  return false;  }  Areas other = (Areas) obj;  return ID\_Area == other.ID\_Area && Objects.equals(AreaName, other.AreaName);  }  } |

#### Model Employees

|  |
| --- |
| package Entity;  import java.io.Serializable;  import java.util.Date;  public class Employees implements Serializable{  private int ID\_Employee;  private String FullName;  private String Password;  private boolean Sex;  private Date birthday;  private String PhoneNumber;  private String Email;  private String Address;  private String ID\_role;  private String Photo;  public Employees() {  }  public Employees(int ID\_Employee, String FullName, String Password, boolean Sex, Date birthday, String PhoneNumber, String Email, String Address, String ID\_role, String Photo) {  this.ID\_Employee = ID\_Employee;  this.FullName = FullName;  this.Password = Password;  this.Sex = Sex;  this.birthday = birthday;  this.PhoneNumber = PhoneNumber;  this.Email = Email;  this.Address = Address;  this.ID\_role = ID\_role;  this.Photo = Photo;  }  public int getID\_Employee() {  return ID\_Employee;  }  public void setID\_Employee(int ID\_Employee) {  this.ID\_Employee = ID\_Employee;  }  public String getFullName() {  return FullName;  }  public void setFullName(String FullName) {  this.FullName = FullName;  }  public String getPassword() {  return Password;  }  public void setPassword(String Password) {  this.Password = Password;  }  public boolean isSex() {  return Sex;  }  public void setSex(boolean Sex) {  this.Sex = Sex;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getPhoneNumber() {  return PhoneNumber;  }  public void setPhoneNumber(String PhoneNumber) {  this.PhoneNumber = PhoneNumber;  }  public String getEmail() {  return Email;  }  public void setEmail(String Email) {  this.Email = Email;  }  public String getAddress() {  return Address;  }  public void setAddress(String Address) {  this.Address = Address;  }  public String getID\_role() {  return ID\_role;  }  public void setID\_role(String ID\_role) {  this.ID\_role = ID\_role;  }  public String getPhoto() {  return Photo;  }  public void setPhoto(String Photo) {  this.Photo = Photo;  }      @Override  public String toString() {  return "Employee{" +  "idEmployee=" + ID\_Employee +  ", fullName='" + FullName + '\'' +  ", password='" + Password + '\'' +  ", sex='" + Sex + '\'' +  ", birthday='" + birthday + '\'' +  ", phoneNumber='" + PhoneNumber + '\'' +  ", email='" + Email + '\'' +  ", address='" + Address + '\'' +  ", idRole='" + ID\_role + '\'' +  ", photo='" + Photo + '\'' +  '}';  }  } |

#### Model Invoices

|  |
| --- |
| package Entity;  import java.util.Date;  public class Invoices {  private int ID\_Invoice;  private int ID\_Order;  private int ID\_Method;  private int ID\_Employee;  private int ID\_Tax;  private Date InvoiceDate;  private double TaxAmount;  private double TotalAmount;  private boolean paid;  public Invoices(int ID\_Invoice, int ID\_Order, int ID\_Method, int ID\_Employee, int ID\_Tax, Date InvoiceDate, double TaxAmount, double TotalAmount, boolean IsPaid) {  this.ID\_Invoice = ID\_Invoice;  this.ID\_Order = ID\_Order;  this.ID\_Method = ID\_Method;  this.ID\_Employee = ID\_Employee;  this.ID\_Tax = ID\_Tax;  this.InvoiceDate = InvoiceDate;  this.TaxAmount = TaxAmount;  this.TotalAmount = TotalAmount;  this.paid = IsPaid;  }  public Invoices() {  }  public int getID\_Order() {  return ID\_Order;  }  public void setID\_Order(int ID\_Order) {  this.ID\_Order = ID\_Order;  }  public int getID\_Method() {  return ID\_Method;  }  public void setID\_Method(int ID\_Method) {  this.ID\_Method = ID\_Method;  }  public int getID\_Employee() {  return ID\_Employee;  }  public void setID\_Employee(int ID\_Employee) {  this.ID\_Employee = ID\_Employee;  }  public int getID\_Tax() {  return ID\_Tax;  }  public void setID\_Tax(int ID\_Tax) {  this.ID\_Tax = ID\_Tax;  }    public int getID\_Invoice() {  return ID\_Invoice;  }  public void setID\_Invoice(int ID\_Invoice) {  this.ID\_Invoice = ID\_Invoice;  }  public Date getInvoiceDate() {  return InvoiceDate;  }  public void setInvoiceDate(Date InvoiceDate) {  this.InvoiceDate = InvoiceDate;  }  public double getTaxAmount() {  return TaxAmount;  }  public void setTaxAmount(double TaxAmount) {  this.TaxAmount = TaxAmount;  }  public double getTotalAmount() {  return TotalAmount;  }  public void setTotalAmount(double TotalAmount) {  this.TotalAmount = TotalAmount;  }  public boolean isPaid() {  return paid;  }  public void setPaid(boolean IsPaid) {  this.paid = IsPaid;  }  @Override  public String toString() {  return "Invoice{" +  "idInvoice=" + ID\_Invoice +  ", idOrder=" + ID\_Order +  ", idMethod=" + ID\_Method +  ", idEmployee=" + ID\_Employee +  ", idTax=" + ID\_Tax +  ", invoiceDate=" + InvoiceDate +  ", taxAmount=" + TaxAmount +  ", totalAmount=" + TotalAmount +  ", isPaid=" + paid +  '}';  }  } |

#### Model MenuCategories

|  |
| --- |
| package Entity;  public class MenuCategories {  private int ID\_Category;  private String CategoryName;  public MenuCategories(int ID\_Category, String CategoryName) {  this.ID\_Category = ID\_Category;  this.CategoryName = CategoryName;  }  public MenuCategories() {  }    public int getID\_Category() {  return ID\_Category;  }  public void setID\_Category(int ID\_Category) {  this.ID\_Category = ID\_Category;  }  public String getCategoryName() {  return CategoryName;  }  public void setCategoryName(String CategoryName) {  this.CategoryName = CategoryName;  }    @Override  public String toString() {  return this.CategoryName;  }    @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null || getClass() != obj.getClass()) {  return false;  }  MenuCategories other = (MenuCategories) obj;  // So sánh cả ID\_Category và CategoryName  return this.ID\_Category == other.ID\_Category &&  this.CategoryName.equals(other.CategoryName);  }  } |

#### Model MenuItem Detail

|  |
| --- |
| package Entity;  public class MenuItemDetail {  private int ID\_MIT;  private int ID\_Item;  private String ID\_Product;  private double Quantity;  public MenuItemDetail(int ID\_MIT, int ID\_Item, String ID\_Product, double Quantity) {  this.ID\_MIT = ID\_MIT;  this.ID\_Item = ID\_Item;  this.ID\_Product = ID\_Product;  this.Quantity = Quantity;  }  public MenuItemDetail() {  }  public int getID\_Item() {  return ID\_Item;  }  public void setID\_Item(int ID\_Item) {  this.ID\_Item = ID\_Item;  }  public String getID\_Product() {  return ID\_Product;  }  public void setID\_Product(String ID\_Product) {  this.ID\_Product = ID\_Product;  }    public int getID\_MIT() {  return ID\_MIT;  }  public void setID\_MIT(int ID\_MIT) {  this.ID\_MIT = ID\_MIT;  }  public double getQuantity() {  return Quantity;  }  public void setQuantity(double Quantity) {  this.Quantity = Quantity;  }    @Override  public String toString() {  return "MenuItemDetail{" +  "idMIT=" + ID\_MIT +  ", idItem=" + ID\_Item +  ", idProduct='" + ID\_Product + '\'' +  ", quantity=" + Quantity +  '}';  }  } |

#### Model Menu Items

|  |
| --- |
| package Entity;  public class MenuItems {  private int ID\_Item = 0;  private String ItemName;  private int ID\_Category;  private int Price;  private String Photo;  private boolean Available;  public MenuItems(int ID\_Item, String ItemName, int ID\_Category, int Price, String Photo, boolean Available) {  this.ID\_Item = ID\_Item;  this.ItemName = ItemName;  this.ID\_Category = ID\_Category;  this.Price = Price;  this.Photo = Photo;  this.Available = Available;  }      public MenuItems() {  }  public boolean isAvailable() {  return Available;  }  public void setAvailable(boolean Available) {  this.Available = Available;  }      public int getID\_Category() {  return ID\_Category;  }  public void setID\_Category(int ID\_Category) {  this.ID\_Category = ID\_Category;  }  public int getID\_Item() {  return ID\_Item;  }  public void setID\_Item(int ID\_Item) {  this.ID\_Item = ID\_Item;  }  public String getItemName() {  return ItemName;  }  public void setItemName(String ItemName) {  this.ItemName = ItemName;  }  public int getPrice() {  return Price;  }  public void setPrice(int Price) {  this.Price = Price;  }  public String getPhoto() {  return Photo;  }  public void setPhoto(String Photo) {  this.Photo = Photo;  }  @Override  public String toString() {  return "MenuItem{" +  "idItem=" + ID\_Item +  ", itemName='" + ItemName + '\'' +  ", idCategory=" + ID\_Category +  ", price=" + Price +  ", photo='" + Photo + '\'' +  '}';  }  } |

#### Model OrderDetail

|  |
| --- |
| package Entity;  public class OrderDetail {  private int ID\_OrderDetail;  private int ID\_Order;  private int ID\_Item;  private int Quantity;  private double Price;  private double TotalPrice;  public OrderDetail(int ID\_OrderDetail, int ID\_Order, int ID\_Item, int Quantity, double Price, double TotalPrice) {  this.ID\_OrderDetail = ID\_OrderDetail;  this.ID\_Order = ID\_Order;  this.ID\_Item = ID\_Item;  this.Quantity = Quantity;  this.Price = Price;  this.TotalPrice = TotalPrice;  }  public OrderDetail() {  }  public int getID\_Order() {  return ID\_Order;  }  public void setID\_Order(int ID\_Order) {  this.ID\_Order = ID\_Order;  }  public int getID\_Item() {  return ID\_Item;  }  public void setID\_Item(int ID\_Item) {  this.ID\_Item = ID\_Item;  }    public int getID\_OrderDetail() {  return ID\_OrderDetail;  }  public void setID\_OrderDetail(int ID\_OrderDetail) {  this.ID\_OrderDetail = ID\_OrderDetail;  }  public int getQuantity() {  return Quantity;  }  public void setQuantity(int Quantity) {  this.Quantity = Quantity;  }  public double getPrice() {  return Price;  }  public void setPrice(double Price) {  this.Price = Price;  }  public double getTotalPrice() {  return Quantity\*Price;  }  public void setTotalPrice() {  this.TotalPrice = Quantity\*Price;  }  @Override  public String toString() {  return "OrderDetail{" +  "idOrderDetail=" + ID\_OrderDetail +  ", idOrder=" + ID\_Order +  ", idItem=" + ID\_Item +  ", quantity=" + Quantity +  ", price=" + Price +  ", totalPrice=" + TotalPrice +  '}';  }  } |

#### Model Order

|  |
| --- |
| package Entity;  import java.util.Date;  public class Orders {  private int ID\_Order;  private int ID\_Table;  private int ID\_Employee;  private Date OrderDate;  private int NumberOfGuests;  private boolean IsPaid;  public Orders(int ID\_Order, int ID\_Table, int ID\_Employee, Date OrderDate, int NumberOfGuests, boolean IsPaid) {  this.ID\_Order = ID\_Order;  this.ID\_Table = ID\_Table;  this.ID\_Employee = ID\_Employee;  this.OrderDate = OrderDate;  this.NumberOfGuests = NumberOfGuests;  this.IsPaid = IsPaid;  }  public Orders() {  }  public int getID\_Table() {  return ID\_Table;  }  public void setID\_Table(int ID\_Table) {  this.ID\_Table = ID\_Table;  }  public int getID\_Employee() {  return ID\_Employee;  }  public void setID\_Employee(int ID\_Employee) {  this.ID\_Employee = ID\_Employee;  }    public int getID\_Order() {  return ID\_Order;  }  public void setID\_Order(int ID\_Order) {  this.ID\_Order = ID\_Order;  }  public Date getOrderDate() {  return OrderDate;  }  public void setOrderDate(Date OrderDate) {  this.OrderDate = OrderDate;  }  public int getNumberOfGuests() {  return NumberOfGuests;  }  public void setNumberOfGuests(int NumberOfGuests) {  this.NumberOfGuests = NumberOfGuests;  }  public boolean isIsPaid() {  return IsPaid;  }  public void setIsPaid(boolean IsPaid) {  this.IsPaid = IsPaid;  }    @Override  public String toString() {  return "Order{" +  "idOrder=" + ID\_Order +  ", idTable=" + ID\_Table +  ", idEmployee=" + ID\_Employee +  ", orderDate=" + OrderDate +  ", numberOfGuests=" + NumberOfGuests +  ", isPaid=" + IsPaid +  '}';  }  } |

#### Model PaymentMethods

|  |
| --- |
| package Entity;  public class PaymentMethods {  private int ID\_Method;  private String MethodName;  public PaymentMethods(int ID\_Method, String MethodName) {  this.ID\_Method = ID\_Method;  this.MethodName = MethodName;  }  public PaymentMethods() {  }  public int getID\_Method() {  return ID\_Method;  }  public void setID\_Method(int ID\_Method) {  this.ID\_Method = ID\_Method;  }  public String getMethodName() {  return MethodName;  }  public void setMethodName(String MethodName) {  this.MethodName = MethodName;  }  @Override  public String toString() {  return "PaymentMethod{" +  "idMethod=" + ID\_Method +  ", methodName='" + MethodName + '\'' +  '}';  }  } |

#### Model ProductsCategories

|  |
| --- |
| package Entity;  public class ProductCategories {  private int ID\_Categories;  private String CategoryName;  public ProductCategories() {  }  public ProductCategories(int ID\_Categories, String CategoryName) {  this.ID\_Categories = ID\_Categories;  this.CategoryName = CategoryName;  }  public int getID\_Categories() {  return ID\_Categories;  }  public void setID\_Categories(int ID\_Categories) {  this.ID\_Categories = ID\_Categories;  }  public String getCategoryName() {  return CategoryName;  }  public void setCategoryName(String CategoryName) {  this.CategoryName = CategoryName;  }    @Override  public String toString() {  return CategoryName;  }  } |

#### Model Products

|  |
| --- |
| package Entity;  public class Products {  private String ID\_product;  private String Name;  private double Quantity;  private String Unit;  private double Price;  public Products(String ID\_product, String Name, double Quantity, String Unit, double Price) {  this.ID\_product = ID\_product;  this.Name = Name;  this.Quantity = Quantity;  this.Unit = Unit;  this.Price = Price;  }  public Products() {  }    public String getID\_product() {  return ID\_product;  }  public void setID\_product(String ID\_product) {  this.ID\_product = ID\_product;  }  public String getName() {  return Name;  }  public void setName(String Name) {  this.Name = Name;  }  public double getQuantity() {  return Quantity;  }  public void setQuantity(double Quantity) {  this.Quantity = Quantity;  }  public String getUnit() {  return Unit;  }  public void setUnit(String Unit) {  this.Unit = Unit;  }  public double getPrice() {  return Price;  }  public void setPrice(double Price) {  this.Price = Price;  }    @Override  public String toString(){  return Name;  }  } |

#### Model ProductsOrder

|  |
| --- |
| package Entity;  import java.util.Date;  public class PurchaseOrders {  private int ID\_PurchaseOrder;  private Date OrderDate;  private int ID\_Employee;  public PurchaseOrders(int ID\_PurchaseOrder, Date OrderDate, int ID\_Emplyee) {  this.ID\_PurchaseOrder = ID\_PurchaseOrder;  this.OrderDate = OrderDate;  this.ID\_Employee = ID\_Emplyee;  }  public PurchaseOrders() {  }    public int getID\_PurchaseOrder() {  return ID\_PurchaseOrder;  }  public void setID\_PurchaseOrder(int ID\_PurchaseOrder) {  this.ID\_PurchaseOrder = ID\_PurchaseOrder;  }  public Date getOrderDate() {  return OrderDate;  }  public void setOrderDate(Date OrderDate) {  this.OrderDate = OrderDate;  }  public int getID\_Employee() {  return ID\_Employee;  }  public void setID\_Employee(int ID\_Employee) {  this.ID\_Employee = ID\_Employee;  }    @Override  public String toString() {  return "PurchaseOrder{" +  "idPurchaseOrder=" + ID\_PurchaseOrder +  ", orderDate=" + OrderDate +  ", idEmplyee=" + ID\_Employee +  '}';  }  } |

#### Model ProductsOrderDetail

|  |
| --- |
| package Entity;  public class PurchaseOrdersDetail {  private int ID\_POD;  private int ID\_PurchaseOrder;  private String ID\_Product;  private double Quantity;  private double Price;  public PurchaseOrdersDetail(int ID\_POD, int ID\_PurchaseOrder, String ID\_Product, double Quantity, double Price) {  this.ID\_POD = ID\_POD;  this.ID\_PurchaseOrder = ID\_PurchaseOrder;  this.ID\_Product = ID\_Product;  this.Quantity = Quantity;  this.Price = Price;  }  public PurchaseOrdersDetail() {  }    public void setID\_PurchaseOrder(int ID\_PurchaseOrder) {  this.ID\_PurchaseOrder = ID\_PurchaseOrder;  }  public int getID\_PurchaseOrder() {  return ID\_PurchaseOrder;  }    public String getID\_Product() {  return ID\_Product;  }  public void setID\_Product(String ID\_Product) {  this.ID\_Product = ID\_Product;  }  public int getID\_POD() {  return ID\_POD;  }  public void setID\_POD(int ID\_POD) {  this.ID\_POD = ID\_POD;  }  public double getQuantity() {  return Quantity;  }  public void setQuantity(double Quantity) {  this.Quantity = Quantity;  }  public double getPrice() {  return Price;  }  public void setPrice(double Price) {  this.Price = Price;  }    @Override  public String toString() {  return "PurchaseOrderDetail{" +  "idPOD=" + ID\_POD +  ", idPurchaseOrder=" + ID\_PurchaseOrder +  ", idProduct='" + ID\_Product + '\'' +  ", quantity=" + Quantity +  ", price=" + Price +  '}';  }  } |

#### Model Role

|  |
| --- |
| package Entity;  import java.io.Serializable;  public class Role implements Serializable{  private String ID\_role;  private String RoleNam;  public Role(String ID\_role, String RoleNam) {  this.ID\_role = ID\_role;  this.RoleNam = RoleNam;  }  public Role() {  }    public String getID\_role() {  return ID\_role;  }  public void setID\_role(String ID\_role) {  this.ID\_role = ID\_role;  }  public String getRoleNam() {  return RoleNam;  }  public void setRoleNam(String RoleNam) {  this.RoleNam = RoleNam;  }    @Override  public String toString(){  return ID\_role;  }  } |

#### Model Tables

|  |
| --- |
| package Entity;  public class Tables {  private int ID\_Table;  private String TableName;  private int ID\_Area;  private boolean IsOccupied;  public Tables(int ID\_Table, String TableName, int ID\_Area, boolean IsOccupied) {  this.ID\_Table = ID\_Table;  this.TableName = TableName;  this.ID\_Area = ID\_Area;  this.IsOccupied = IsOccupied;  }  public Tables() {  }  public int getID\_Area() {  return ID\_Area;  }  public void setID\_Area(int ID\_Area) {  this.ID\_Area = ID\_Area;  }    public int getID\_Table() {  return ID\_Table;  }  public void setID\_Table(int ID\_Table) {  this.ID\_Table = ID\_Table;  }  public String getTableName() {  return TableName;  }  public void setTableName(String TableName) {  this.TableName = TableName;  }  public boolean isIsOccupied() {  return IsOccupied;  }  public void setIsOccupied(boolean IsOccupied) {  this.IsOccupied = IsOccupied;  }    @Override  public String toString() {  return "Table{" +  "idTable=" + ID\_Table +  ", tableName='" + TableName + '\'' +  ", idArea=" + ID\_Area +  ", isOccupied=" + IsOccupied +  '}';  }  } |

#### Model Tax

|  |
| --- |
| package Entity;  public class Taxes {  private int ID\_Tax;  private String TaxName;  private double TaxRate;  public Taxes(int ID\_Tax, String TaxName, double TaxRate) {  this.ID\_Tax = ID\_Tax;  this.TaxName = TaxName;  this.TaxRate = TaxRate;  }  public Taxes() {  }  public int getID\_Tax() {  return ID\_Tax;  }  public void setID\_Tax(int ID\_Tax) {  this.ID\_Tax = ID\_Tax;  }  public String getTaxName() {  return TaxName;  }  public void setTaxName(String TaxName) {  this.TaxName = TaxName;  }  public double getTaxRate() {  return TaxRate;  }  public void setTaxRate(double TaxRate) {  this.TaxRate = TaxRate;  }  @Override  public String toString() {  return "Tax{" +  "idTax=" + ID\_Tax +  ", taxName='" + TaxName + '\'' +  ", taxRate=" + TaxRate +  '}';  }  } |

### DAO Class - Các lớp truy xuất dữ liệu

#### Lớp DAO làm việc với Bảng AreasDao

|  |
| --- |
| package Dao;  import Entity.Areas;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class AreasDao extends RestauranDao<Areas, Integer>{  final String INSERT\_SQL = "INSERT INTO Areas (AreaName) VALUES (?)";  final String UPDATE\_ALL = "UPDATE Areas SET AreaName = ? WHERE ID\_Area = ?";  final String DELETE\_SQL = "DELETE FROM Areas WHERE ID\_Area = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Areas";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Areas WHERE ID\_Area = ?";  @Override  public void insert(Areas entity) {  jdbc.update(INSERT\_SQL,  entity.getAreaName());  }  @Override  public void update(Areas entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Area(),  entity.getAreaName());  }  public int getTableCount(Integer id){  try {  String sql = "select COUNT(Tables.ID\_Area) as Num from Areas "  + "inner join Tables on Areas.ID\_Area = Tables.ID\_Area "  + "where Areas.ID\_Area = ? "  + "GROUP by Areas.ID\_Area ";  ResultSet r = jdbc.query(sql, id);  if(r.next()) return r.getInt("Num");  } catch (Exception e) {  System.out.println(e.getMessage());  return 0;  }  return 0;  }    public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Areas> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  public Areas selectById(Integer id) {  List<Areas> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Areas> selectBySql(String sql, Object... args) {  List<Areas> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Areas entity = new Areas();  entity.setID\_Area(rs.getInt("ID\_Area"));  entity.setAreaName(rs.getString("AreaName"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }      } |

#### Lớp DAO làm việc với Bảng EmployeesDao

|  |
| --- |
| package Dao;  import Entity.Employees;  import Utils.jdbc;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class EmployeesDao extends RestauranDao<Employees, Integer> {  final String INSERT\_SQL = "INSERT INTO Employees (FullName, Password, Sex, birthday, PhoneNumber, Email, Address, ID\_role, Photo)"  + "VALUES (?, ?, ?, ?, ?, ?, ?, ?,?)";  final String UPDATE\_ALL = "UPDATE Employees SET "  + "FullName = ?, Password = ?, Sex = ?, birthday = ?, PhoneNumber = ?, Email = ?, Address = ?, ID\_role = ?, Photo = ? WHERE ID\_Employee = ?";  final String UPDATE\_Not\_Pass = "UPDATE Employees SET "  + "FullName = ?, Sex = ?, birthday = ?, PhoneNumber = ?, Email = ?, Address = ?, ID\_role = ?, Photo = ? WHERE ID\_Employee = ?";  final String DELETE\_SQL = "DELETE FROM Employees WHERE ID\_Employee = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Employees";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Employees WHERE ID\_Employee = ?";  public int getNewID() {  try {  String sql = "Select MAX(ID\_Employee) ID From Employees";  ResultSet r = jdbc.query(sql);  int id = 0;  while (r.next()) {  id = r.getInt("ID");  }  return id + 1;  } catch (Exception e) {  return -100000;  }  }    @Override  public void insert(Employees entity) {  jdbc.update(INSERT\_SQL,  entity.getFullName(),  entity.getPassword(),  entity.isSex(),  entity.getBirthday(),  entity.getPhoneNumber(),  entity.getEmail(),  entity.getAddress(),  entity.getID\_role(),  entity.getPhoto());  }  @Override  public void update(Employees entity) {  jdbc.update(UPDATE\_ALL,  entity.getFullName(),  entity.getPassword(),  entity.isSex(),  entity.getBirthday(),  entity.getPhoneNumber(),  entity.getEmail(),  entity.getAddress(),  entity.getID\_role(),  entity.getPhoto(),  entity.getID\_Employee());  }  public List<Employees> Search(String key){  String sql = "select \* from Employees where ID\_Employee like ? OR FullName like ?";  return selectBySql(sql, "%"+key+"%","%"+key+"%");  }  public void updateNotPass(Employees entity) {  jdbc.update(UPDATE\_Not\_Pass,  entity.getFullName(),  entity.isSex(),  entity.getBirthday(),  entity.getPhoneNumber(),  entity.getEmail(),  entity.getAddress(),  entity.getID\_role(),  entity.getPhoto(),  entity.getID\_Employee());  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Employees> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public Employees selectById(Integer id) {  List<Employees> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if (list.isEmpty()) {  return null;  }  return list.get(0);  }  @Override  public List<Employees> selectBySql(String sql, Object... args) {  List<Employees> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while (rs.next()) {  Employees entity = new Employees();  entity.setID\_Employee(rs.getInt("ID\_Employee"));  entity.setFullName(rs.getString("FullName"));  entity.setPassword(rs.getString("Password"));  entity.setSex(rs.getBoolean("Sex"));  entity.setBirthday(rs.getDate("birthday"));  entity.setPhoneNumber(rs.getString("PhoneNumber"));  entity.setEmail(rs.getString("Email"));  entity.setAddress(rs.getString("Address"));  entity.setID\_role(rs.getString("ID\_role"));  entity.setPhoto(rs.getString("Photo"));  list.add(entity);  }  } catch (Exception e) {  System.out.println("Loi sql " + e.getMessage());  throw new RuntimeException(e);  }  return list;  }  } |

#### Lớp DAO làm việc với Bảng InvoicesDao

|  |
| --- |
| package Dao;  import Entity.Invoices;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class InvoicesDao extends RestauranDao<Invoices, String>{  final String INSERT\_SQL = "INSERT INTO Invoices (ID\_Order, ID\_Method, ID\_Employee, ID\_Tax, InvoiceDate, TaxAmount, TotalAmount, IsPaid)"  + " VALUES (?, ?, ?, ?, ?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE Invoices SET ID\_Order = ?, ID\_Method = ?, "  + "ID\_Employee = ?, ID\_Tax = ?, InvoiceDate = ?, TaxAmount = ?, "  + "TotalAmount = ?, IsPaid = ? WHERE ID\_Invoice = ?";  final String DELETE\_SQL = "DELETE FROM Invoices WHERE ID\_Invoice = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Invoices";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Invoices WHERE ID\_Invoice = ?";  final String UpdateThanhToan = "UPDATE Invoices set ID\_Method = ?, InvoiceDate = ?, IsPaid = ? where ID\_Invoice = ?";  @Override  public void insert(Invoices entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Order(),  entity.getID\_Method(),  entity.getID\_Employee(),  entity.getID\_Tax(),  entity.getInvoiceDate(),  entity.getTaxAmount(),  entity.getTotalAmount(),  entity.isPaid());  }  public int getID(int ID\_order){  try {  String sql = "select ID\_Invoice from Invoices where ID\_Order = ? and IsPaid = 0";  ResultSet r = jdbc.query(sql, ID\_order);  int id = -1;  while (r.next()) {  id = r.getInt("ID\_Invoice");  }  return id;  } catch (Exception e) {  return -1;  }  }    public int getNewID(){  try {  String sql = "Select MAX(ID\_Invoice) ID From Invoices";  ResultSet r = jdbc.query(sql);  int id = 0;  while (r.next()) {  id = r.getInt("ID");  }  return id;  } catch (Exception e) {  return -1;  }  }    @Override  public void update(Invoices entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Invoice(),  entity.getID\_Order(),  entity.getID\_Method(),  entity.getID\_Employee(),  entity.getID\_Tax(),  entity.getInvoiceDate(),  entity.getTaxAmount(),  entity.getTotalAmount(),  entity.isPaid());  }  public void updateThanhToan(Invoices e){  jdbc.update(UpdateThanhToan,  e.getID\_Method(),  e.getInvoiceDate(),  e.isPaid(),  e.getID\_Invoice()  );  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Invoices> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public Invoices selectById(String id) {  List<Invoices> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Invoices> selectBySql(String sql, Object... args) {  List<Invoices> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Invoices entity = new Invoices();  entity.setID\_Invoice(rs.getInt("ID\_Invoice"));  entity.setID\_Order(rs.getInt("ID\_Order"));  entity.setID\_Method(rs.getInt("ID\_Method"));  entity.setID\_Employee(rs.getInt("ID\_Employee"));  entity.setID\_Tax(rs.getInt("ID\_Tax"));  entity.setInvoiceDate(rs.getDate("InvoiceDate"));  entity.setTaxAmount(rs.getDouble("TaxAmount"));  entity.setTotalAmount(rs.getDouble("TotalAmount"));  entity.setPaid(rs.getBoolean("IsPaid"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng MenuCategoriesDao

|  |
| --- |
| package Dao;  import Entity.MenuCategories;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class MenuCategoriesDao extends RestauranDao<MenuCategories, Integer>{  final String INSERT\_SQL = "INSERT INTO MenuCategories (CategoryName) VALUES (?)";  final String UPDATE\_ALL = "UPDATE MenuCategories SET CategoryName = ? WHERE ID\_Category = ?";  final String DELETE\_SQL = "DELETE FROM MenuCategories WHERE ID\_Category = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM MenuCategories";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM MenuCategories WHERE ID\_Category = ?";  @Override  public void insert(MenuCategories entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Category(),  entity.getCategoryName());  }  @Override  public void update(MenuCategories entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Category(),  entity.getCategoryName());  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<MenuCategories> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public MenuCategories selectById(Integer id) {  List<MenuCategories> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<MenuCategories> selectBySql(String sql, Object... args) {  List<MenuCategories> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  MenuCategories entity = new MenuCategories();  entity.setID\_Category(rs.getInt("ID\_Category"));  entity.setCategoryName(rs.getString("CategoryName"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng MenuItemDetailDao

|  |
| --- |
| package Dao;  import Entity.MenuItemDetail;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class MenuItemDetailDao extends RestauranDao<MenuItemDetail, String>{  final String INSERT\_SQL = "INSERT INTO MenuItemDetail (ID\_Item, ID\_Product, Quantity) VALUES (?, ?, ?)";  final String UPDATE\_ALL = "UPDATE MenuItemDetail SET ID\_Item = ?, ID\_Product = ?, Quantity = ? WHERE ID\_MIT= ?";  final String DELETE\_SQL = "DELETE FROM MenuItemDetail WHERE ID\_MIT = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM MenuItemDetail";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM MenuItemDetail WHERE ID\_MIT = ?";  @Override  public void insert(MenuItemDetail entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_MIT(),  entity.getID\_Item(),  entity.getID\_Product(),  entity.getQuantity());  }  @Override  public void update(MenuItemDetail entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_MIT(),  entity.getID\_Item(),  entity.getID\_Product(),  entity.getQuantity());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<MenuItemDetail> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public MenuItemDetail selectById(String id) {  List<MenuItemDetail> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<MenuItemDetail> selectBySql(String sql, Object... args) {  List<MenuItemDetail> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  MenuItemDetail entity = new MenuItemDetail();  entity.setID\_MIT(rs.getInt("ID\_MIT"));  entity.setID\_Item(rs.getInt("ID\_Item"));  entity.setID\_Product(rs.getString("ID\_Product"));  entity.setQuantity(rs.getFloat("Quantity"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng MenuItemsDao

|  |
| --- |
| package Dao;  import Entity.MenuItems;  import Utils.jdbc;  import Utils.msg;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class MenuItemsDao extends RestauranDao<MenuItems, Integer>{  final String INSERT\_SQL = "INSERT INTO MenuItems (ItemName, ID\_Category, Price, Photo) "  + "VALUES (?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE MenuItems SET ItemName = ?, ID\_Category = ?, Price = ?, Photo = ?, IsAvailable = ? WHERE ID\_Item = ?";  final String DELETE\_SQL = "DELETE FROM MenuItems WHERE ID\_Item = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM MenuItems";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM MenuItems WHERE ID\_Item = ?";  @Override  public void insert(MenuItems entity) {  jdbc.update(INSERT\_SQL,  entity.getItemName(),  entity.getID\_Category(),  entity.getPrice(),  entity.getPhoto());  }  @Override  public void update(MenuItems entity) {  jdbc.update(UPDATE\_ALL,  entity.getItemName(),  entity.getID\_Category(),  entity.getPrice(),  entity.getPhoto(),  entity.isAvailable(),  entity.getID\_Item());  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<MenuItems> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  public List<MenuItems> selectCatory(int ID\_catory){  if(ID\_catory == 0){  return selectAll();  }    String sql = "Select \* from MenuItems where ID\_Category = ?";  return selectBySql(sql, ID\_catory);  }    @Override  public MenuItems selectById(Integer id) {  List<MenuItems> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<MenuItems> selectBySql(String sql, Object... args) {  List<MenuItems> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  MenuItems entity = new MenuItems();  entity.setID\_Item(rs.getInt("ID\_Item"));  entity.setItemName(rs.getString("ItemName"));  entity.setID\_Category(rs.getInt("ID\_Category"));  entity.setPrice(rs.getInt("Price"));  entity.setPhoto(rs.getString("Photo"));  entity.setAvailable(rs.getBoolean("isAvailable"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng OrderDetailDao

|  |
| --- |
| package Dao;  import Entity.OrderDetail;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class OrderDetailDao extends RestauranDao<OrderDetail, Integer>{  final String INSERT\_SQL = "INSERT INTO OrderDetail (ID\_Order, ID\_Item, Quantity, Price, TotalPrice) "  + "VALUES (?, ?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE OrderDetail SET "  + "ID\_Order = ?, ID\_Item = ?, Quantity = ?, Price = ?,"  + " TotalPrice = ? WHERE ID\_Order = ? and ID\_Item = ?";  final String DELETE\_SQL = "DELETE FROM OrderDetail WHERE ID\_OrderDetail = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM OrderDetail";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM OrderDetail WHERE ID\_OrderDetail = ?";  @Override  public void insert(OrderDetail entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Order(),  entity.getID\_Item(),  entity.getQuantity(),  entity.getPrice(),  entity.getTotalPrice());  }  @Override  public void update(OrderDetail entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Order(),  entity.getID\_Item(),  entity.getQuantity(),  entity.getPrice(),  entity.getTotalPrice(),  entity.getID\_Order(),  entity.getID\_Item()  );  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }    public void delete(Integer ID\_item, Integer ID\_DonHang){  jdbc.update("DELETE FROM OrderDetail WHERE ID\_Item = ? AND ID\_Order = ?", ID\_item, ID\_DonHang);  }  @Override  public List<OrderDetail> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public OrderDetail selectById(Integer id) {  List<OrderDetail> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<OrderDetail> selectBySql(String sql, Object... args) {  List<OrderDetail> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  OrderDetail entity = new OrderDetail();  entity.setID\_OrderDetail(rs.getInt("ID\_OrderDetail"));  entity.setID\_Order(rs.getInt("ID\_Order"));  entity.setID\_Item(rs.getInt("ID\_Item"));  entity.setQuantity(rs.getInt("Quantity"));  entity.setPrice(rs.getDouble("Price"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng OrdersDao

|  |
| --- |
| package Dao;  import Entity.Orders;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class OrdersDao extends RestauranDao<Orders, Integer>{  final String INSERT\_SQL = "INSERT INTO Orders (ID\_Table, ID\_Employee, OrderDate, NumberOfGuests, IsPaid)" +  " VALUES (?, ?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE Orders SET "  + "ID\_Table = ?, ID\_Employee = ?, OrderDate = ?, "  + "NumberOfGuests = ?, IsPaid = ? WHERE ID\_Order = ?";  final String DELETE\_SQL = "DELETE FROM Orders WHERE ID\_Order = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Orders";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Orders WHERE ID\_Order = ?";  public int GetID() {  try {  String sql = "Select MAX(ID\_Order) ID From Orders";  ResultSet r = jdbc.query(sql);  int id = 0;  while (r.next()) {  id = r.getInt("ID");  }  return id;  } catch (Exception e) {  System.out.println(e.getMessage());  return -1;  }  }    public void updateStatus(int ID\_Order, boolean paid){  String sql = "Update Orders set IsPaid = ? where ID\_Order = ?";  jdbc.update(sql, paid,ID\_Order);  }    @Override  public void insert(Orders entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Table(),  entity.getID\_Employee(),  entity.getOrderDate(),  entity.getNumberOfGuests(),  entity.isIsPaid());  }  @Override  public void update(Orders entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Table(),  entity.getID\_Employee(),  entity.getOrderDate(),  entity.getNumberOfGuests(),  entity.isIsPaid(),  entity.getID\_Order());  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Orders> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public Orders selectById(Integer id) {  List<Orders> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Orders> selectBySql(String sql, Object... args) {  List<Orders> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Orders entity = new Orders();  entity.setID\_Order(rs.getInt("ID\_Order"));  entity.setID\_Table(rs.getInt("ID\_Table"));  entity.setID\_Employee(rs.getInt("ID\_Employee"));  entity.setOrderDate(rs.getDate("OrderDate"));  entity.setNumberOfGuests(rs.getInt("NumberOfGuests"));  entity.setIsPaid(rs.getBoolean("IsPaid"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng PaymentMethodsDao

|  |
| --- |
| package Dao;  import Entity.PaymentMethods;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class PaymentMethodsDao extends RestauranDao<PaymentMethods, String>{  final String INSERT\_SQL = "INSERT INTO PaymentMethods (MethodName) VALUES (?)";  final String UPDATE\_ALL = "UPDATE PaymentMethods SET MethodName = ? WHERE ID\_Method = ?";  final String DELETE\_SQL = "DELETE FROM PaymentMethods WHERE ID\_Method = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM PaymentMethods";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM PaymentMethods WHERE ID\_Method = ?";  @Override  public void insert(PaymentMethods entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Method(),  entity.getMethodName());  }  @Override  public void update(PaymentMethods entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Method(),  entity.getMethodName());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<PaymentMethods> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public PaymentMethods selectById(String id) {  List<PaymentMethods> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<PaymentMethods> selectBySql(String sql, Object... args) {  List<PaymentMethods> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  PaymentMethods entity = new PaymentMethods();  entity.setID\_Method(rs.getInt("ID\_Method"));  entity.setMethodName(rs.getString("MethodName"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

#### Lớp DAO làm việc với Bảng ProductCategoriesDao

|  |
| --- |
| package Dao;  import Entity.ProductCategories;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class ProductCategoriesDao extends RestauranDao<ProductCategories, String>{  final String INSERT\_SQL = "INSERT INTO ProductCategories (CategoryName) VALUES (?)";  final String UPDATE\_ALL = "UPDATE ProductCategories SET CategoryName = ? WHERE ID\_Categories = ?";  final String DELETE\_SQL = "DELETE FROM ProductCategories WHERE ID\_Categories = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM ProductCategories";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM ProductCategories WHERE ID\_Categories = ?";  @Override  public void insert(ProductCategories entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Categories(),  entity.getCategoryName());  }  @Override  public void update(ProductCategories entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Categories(),  entity.getCategoryName());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<ProductCategories> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public ProductCategories selectById(String id) {  List<ProductCategories> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<ProductCategories> selectBySql(String sql, Object... args) {  List<ProductCategories> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  ProductCategories entity = new ProductCategories();  entity.setID\_Categories(rs.getInt("ID\_Categories"));  entity.setCategoryName(rs.getString("CategoryName"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }      } |

#### Lớp DAO làm việc với Bảng ProductsDao

|  |
| --- |
| package Dao;  import Entity.Products;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class ProductsDao extends RestauranDao<Products, String> {  final String INSERT\_SQL = "INSERT INTO Products (ID\_product, Name, Quantity, Unit, Price) "  + "VALUES (?, ?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE Products SET "  + "Name = ?, Quantity = ?, Unit = ?, Price = ? Where ID\_product = ?";  final String DELETE\_SQL = "DELETE FROM Products WHERE ID\_product = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Products";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Products WHERE ID\_product = ?";  final String FIND\_BY\_ID\_OR\_NAME = "SELECT TOP 1 \* "  + "FROM Products "  + "WHERE ID\_product LIKE ? OR Name LIKE ?";  @Override  public void insert(Products entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_product(),  entity.getName(),  entity.getQuantity(),  entity.getUnit(),  entity.getPrice());  }  @Override  public void update(Products entity) {  jdbc.update(UPDATE\_ALL,  entity.getName(),  entity.getQuantity(),  entity.getUnit(),  entity.getPrice(),  entity.getID\_product());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Products> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  public List<Products> getProductsByCategory(String categoryName) {  String sql = "SELECT p.\* FROM Products p "  + "INNER JOIN ProductCategories pc ON p.ID\_Categories = pc.ID\_Categories "  + "WHERE pc.CategoryName = ?";  return selectBySql(sql, categoryName);  }  public List<Products> Search(String key) {  String sql = "select \* from Products where ID\_product like ? OR Name like ?";  return selectBySql(sql, "%" + key + "%", "%" + key + "%");  }  public void updateQuantity(String idProduct, int quantity) {  try {  String sql = "UPDATE Products SET Quantity = Quantity + ? WHERE ID\_product = ?";  jdbc.update(sql, quantity, idProduct);  } catch (Exception e) {  throw new RuntimeException(e);  }  }  public Products SearchFirst(String maHang, String tenHang) {  if (maHang.isBlank()) {  maHang = "NULL";  }  if (tenHang.isBlank()) {  tenHang = "NULL";  }  List<Products> l = selectBySql(FIND\_BY\_ID\_OR\_NAME, "%" + maHang + "%", "%" + tenHang + "%");  if (l.isEmpty()) {  return null;  }  return l.get(0);  }  @Override  public Products selectById(String id) {  List<Products> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if (list.isEmpty()) {  return null;  }  return list.get(0);  }  @Override  public List<Products> selectBySql(String sql, Object... args) {  List<Products> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while (rs.next()) {  Products entity = new Products();  entity.setID\_product(rs.getString("ID\_product"));  entity.setName(rs.getString("Name"));  entity.setQuantity(rs.getFloat("Quantity"));  entity.setUnit(rs.getString("Unit"));  entity.setPrice(rs.getDouble("Price"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }  } |

#### Lớp DAO làm việc với Bảng PurchaseOrdersDao

|  |
| --- |
| package Dao;  import Entity.PurchaseOrders;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class PurchaseOrdersDao extends RestauranDao<PurchaseOrders, Integer> {  final String INSERT\_SQL = "INSERT INTO PurchaseOrders (ID\_PurchaseOrder, OrderDate, ID\_Employee) VALUES (?, ?, ?)";  final String UPDATE\_ALL = "UPDATE PurchaseOrders SET OrderDate = ?, ID\_Employee = ? WHERE ID\_PurchaseOrder = ?";  final String DELETE\_SQL = "DELETE FROM PurchaseOrders WHERE ID\_PurchaseOrder = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM PurchaseOrders";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM PurchaseOrders WHERE ID\_PurchaseOrder = ?";  @Override  public void insert(PurchaseOrders entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_PurchaseOrder(),  entity.getOrderDate(),  entity.getID\_Employee());  }  @Override  public void update(PurchaseOrders entity) {  jdbc.update(UPDATE\_ALL,  entity.getOrderDate(),  entity.getID\_Employee(),  entity.getID\_PurchaseOrder());  }  @Override  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<PurchaseOrders> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public PurchaseOrders selectById(Integer id) {  List<PurchaseOrders> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if (list.isEmpty()) {  return null;  }  return list.get(0);  }  @Override  public List<PurchaseOrders> selectBySql(String sql, Object... args) {  List<PurchaseOrders> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while (rs.next()) {  PurchaseOrders entity = new PurchaseOrders();  entity.setID\_PurchaseOrder(rs.getInt("ID\_PurchaseOrder"));  entity.setOrderDate(rs.getDate("OrderDate"));  entity.setID\_Employee(rs.getInt("ID\_Employee"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }  } |

#### Lớp DAO làm việc với Bảng PurchaseOrdersDetailDao

|  |
| --- |
| package Dao;  import Entity.PurchaseOrdersDetail;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class PurchaseOrdersDetailDao extends RestauranDao<PurchaseOrdersDetail, String> {  final String INSERT\_SQL = "INSERT INTO PurchaseOrdersDetail (ID\_PurchaseOrder, ID\_product, Quantity, Price) "  + "VALUES (?, ?, ?, ?)";  final String UPDATE\_ALL = "UPDATE PurchaseOrdersDetail SET"  + " ID\_PurchaseOrder = ?, ID\_product = ?, Quantity = ?, Price = ? WHERE ID\_POD = ?";  final String DELETE\_SQL = "DELETE FROM PurchaseOrdersDetail WHERE ID\_POD = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM PurchaseOrdersDetail";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM PurchaseOrdersDetail WHERE ID\_POD = ?";  @Override  public void insert(PurchaseOrdersDetail entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_POD(),  entity.getID\_PurchaseOrder(),  entity.getID\_Product(),  entity.getQuantity(),  entity.getPrice());  }  @Override  public void update(PurchaseOrdersDetail entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_PurchaseOrder(),  entity.getID\_Product(),  entity.getQuantity(),  entity.getPrice(),  entity.getID\_POD());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<PurchaseOrdersDetail> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public PurchaseOrdersDetail selectById(String id) {  List<PurchaseOrdersDetail> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if (list.isEmpty()) {  return null;  }  return list.get(0);  }  @Override  public List<PurchaseOrdersDetail> selectBySql(String sql, Object... args) {  List<PurchaseOrdersDetail> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while (rs.next()) {  PurchaseOrdersDetail entity = new PurchaseOrdersDetail();  entity.setID\_POD(rs.getInt("ID\_POD"));  entity.setID\_PurchaseOrder(rs.getInt("ID\_PurchaseOrder"));  entity.setID\_Product(rs.getString("ID\_product"));  entity.setQuantity(rs.getFloat("Quantity"));  entity.setPrice(rs.getDouble("Price"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }  } |

#### Lớp DAO làm việc với Bảng RestauranDao

|  |
| --- |
| package Dao;  import java.util.List;  public abstract class RestauranDao<EntityType, KeyType> {  public abstract void insert(EntityType entity);  public abstract void update(EntityType entity);  public abstract void delete(KeyType id);  public abstract List<EntityType> selectAll();  public abstract EntityType selectById(KeyType id);  public abstract List<EntityType> selectBySql(String sql, Object...args);  } |

* Dao chính để thực hiện truy xuất và thực hiện từ trong CSDL.

#### Lớp DAO làm việc với Bảng RoleDao

|  |
| --- |
| package Dao;  import Entity.Role;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class RoleDao extends RestauranDao<Role, String>{  final String INSERT\_SQL = "INSERT INTO Role (ID\_role, RoleName) VALUES (?, ?)";  final String UPDATE\_ALL = "UPDATE Role SET RoleName = ? WHERE ID\_role = ?";  final String DELETE\_SQL = "DELETE FROM Role WHERE ID\_role = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Role";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Role WHERE ID\_role = ?";  @Override  public void insert(Role entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_role(),  entity.getRoleNam());  }  @Override  public void update(Role entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_role(),  entity.getRoleNam());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Role> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public Role selectById(String id) {  List<Role> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Role> selectBySql(String sql, Object... args) {  List<Role> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Role entity = new Role();  entity.setID\_role(rs.getString("ID\_role"));  entity.setRoleNam(rs.getString("RoleName"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }  } |

#### Lớp DAO làm việc với Bảng TablesdDao

|  |
| --- |
| package Dao;  import Entity.Tables;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class TablesdDao extends RestauranDao<Tables, Integer>{  final String INSERT\_SQL = "INSERT INTO Tables (TableName, ID\_Area, IsOccupied) VALUES (?, ?, ?)";  final String UPDATE\_ALL = "UPDATE Tables SET TableName = ?, ID\_Area = ?, IsOccupied = ? WHERE ID\_Table = ?";  final String DELETE\_SQL = "DELETE FROM Tables WHERE ID\_Table = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Tables";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Tables WHERE ID\_Table = ?";  @Override  public void insert(Tables entity) {  jdbc.update(INSERT\_SQL,  entity.getTableName(),  entity.getID\_Area(),  entity.isIsOccupied());  }  @Override  public void update(Tables entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Table(),  entity.getTableName(),  entity.getID\_Area(),  entity.isIsOccupied());  }  public void delete(Integer id) {  jdbc.update(DELETE\_SQL, id);  }    @Override  public List<Tables> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }        public Tables selectById(Integer id) {  List<Tables> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Tables> selectBySql(String sql, Object... args) {  List<Tables> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Tables entity = new Tables();  entity.setID\_Table(rs.getInt("ID\_Table"));  entity.setTableName(rs.getString("TableName"));  entity.setID\_Area(rs.getInt("ID\_Area"));  entity.setIsOccupied(rs.getBoolean("IsOccupied"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }  public List<Tables> selectByArea(int ID\_area){  String sql;  if(ID\_area == -1){  sql = "Select \* from Tables";  return selectBySql(sql);  }else{  sql = "Select \* from Tables where ID\_Area = ?";  return selectBySql(sql, ID\_area);  }  }    } |

#### Lớp DAO làm việc với Bảng TaxesDao

|  |
| --- |
| package Dao;  import Entity.Taxes;  import Utils.jdbc;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  public class TaxesDao extends RestauranDao<Taxes, String>{  final String INSERT\_SQL = "INSERT INTO Taxes (TaxName, TaxRate) VALUES (?, ?)";  final String UPDATE\_ALL = "UPDATE Taxes SET TaxName = ?, TaxRate = ? WHERE ID\_Tax = ?";  final String DELETE\_SQL = "DELETE FROM Taxes WHERE ID\_Tax = ?";  final String SELECT\_ALL\_SQL = "SELECT \* FROM Taxes";  final String SELECT\_BY\_ID\_SQL = "SELECT \* FROM Taxes WHERE ID\_Tax = ?";  @Override  public void insert(Taxes entity) {  jdbc.update(INSERT\_SQL,  entity.getID\_Tax(),  entity.getTaxName(),  entity.getTaxRate());  }  @Override  public void update(Taxes entity) {  jdbc.update(UPDATE\_ALL,  entity.getID\_Tax(),  entity.getTaxName(),  entity.getTaxRate());  }  @Override  public void delete(String id) {  jdbc.update(DELETE\_SQL, id);  }  @Override  public List<Taxes> selectAll() {  return selectBySql(SELECT\_ALL\_SQL);  }  @Override  public Taxes selectById(String id) {  List<Taxes> list = selectBySql(SELECT\_BY\_ID\_SQL, id);  if(list.isEmpty()){  return null;  }  return list.get(0);  }  @Override  public List<Taxes> selectBySql(String sql, Object... args) {  List<Taxes> list = new ArrayList<>();  try {  ResultSet rs = jdbc.query(sql, args);  while(rs.next()) {  Taxes entity = new Taxes();  entity.setID\_Tax(rs.getInt("ID\_Tax"));  entity.setTaxName(rs.getString("TaxName"));  entity.setTaxRate(rs.getDouble("TaxRate"));  list.add(entity);  }  } catch (Exception e) {  throw new RuntimeException(e);  }  return list;  }    } |

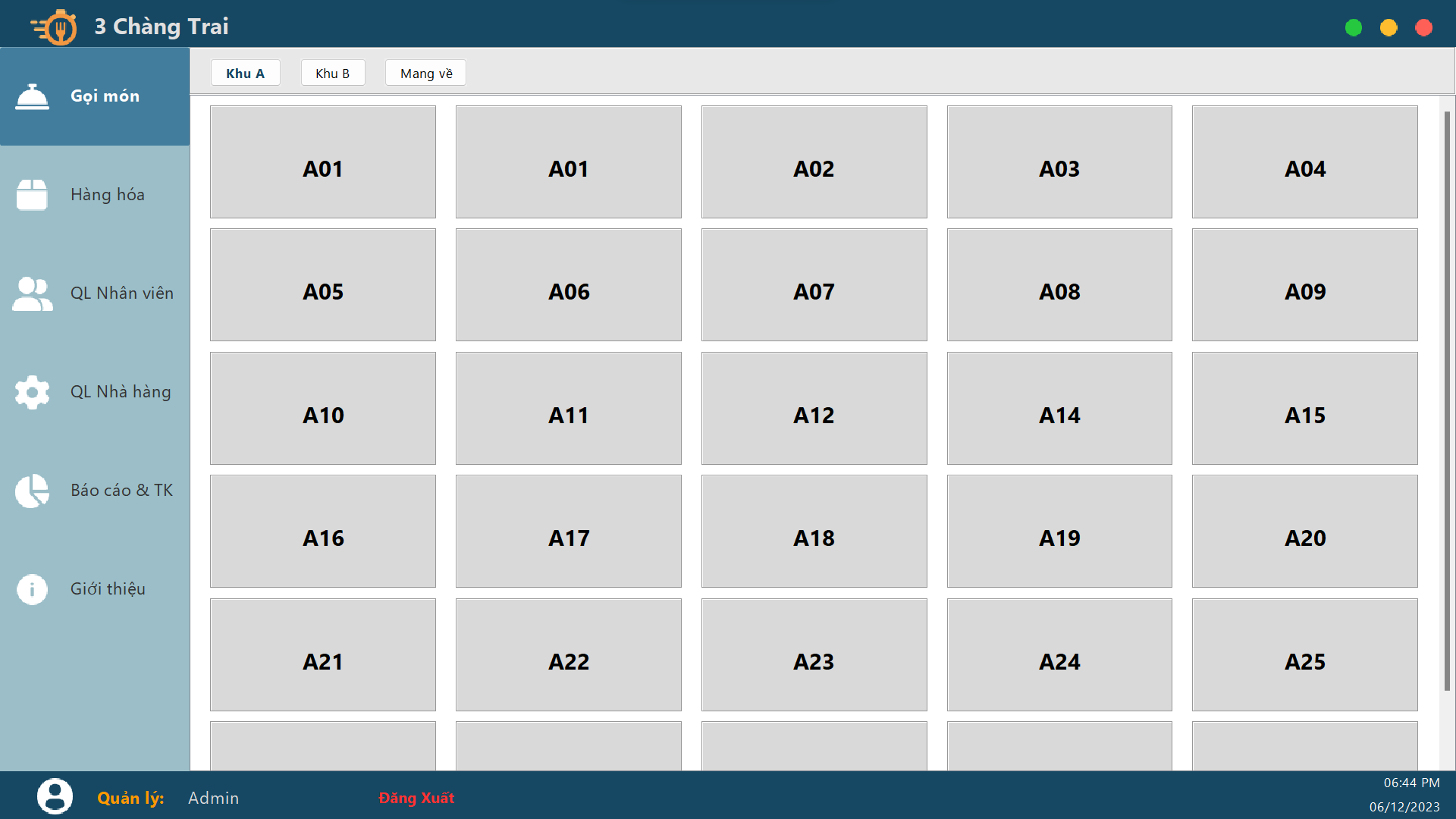
#### Lớp DAO làm việc với Bảng procDao

|  |
| --- |
| package Dao;  import UI.Model.Model\_Mon\_Da\_Goi;  import Utils.jdbc;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.LinkedList;  import java.util.List;  import java.util.Map;  public class procDao {    public static void DeleteOrdersWithoutDetails() {  try {  PreparedStatement stmt = jdbc.getStmt("{CALL DeleteOrdersWithoutDetails}");  stmt.execute();  } catch (Exception e) {  }  }    private List<Object[]> getListOfArray(String sql, String[] cols, Object...args) {  try {  List<Object[]> list = new ArrayList<>();  ResultSet rs = jdbc.query(sql, args);  while (rs.next()) {  Object[] vals = new Object[cols.length];  for (int i = 0; i < cols.length; i++) {  vals[i] = rs.getObject(cols[i]);  }  list.add(vals);  }  rs.getStatement().getConnection().close();  return list;  } catch (Exception e) {  throw new RuntimeException(e);  }  }    public List<Object[]> GetTableSummary(int ID\_Area) {  String sql = "{CALL GetTableSummary(?)}";  String[] cols = {"ID\_Table", "TableName", "TotalAmount", "NumberOfGuests"};  return getListOfArray(sql, cols, ID\_Area);  }          public Map<Integer, Model\_Mon\_Da\_Goi> GetGetItemsByTableID(int ID\_table){  try {  Map<Integer, Model\_Mon\_Da\_Goi> list = new HashMap<>();  String sql = "{CALL GetItemsByTableID(?)}";  ResultSet r = jdbc.query(sql, ID\_table);  while(r.next()){  Model\_Mon\_Da\_Goi i = new Model\_Mon\_Da\_Goi();  i.setId(r.getInt("ID\_item"));  i.setName(r.getString("ItemName"));  i.setPrice(r.getInt("Price"));  i.setSl(r.getInt("Quantity"));  i.ID\_Order = r.getInt("ID");  i.setType(Model\_Mon\_Da\_Goi.ItemType.ChuaGoi);  list.put(i.getId(), i);  }  return list;  } catch (Exception e) {  System.out.println(e.getMessage());  return null;  }  }    public List<Object[]> getDonHang(int ID\_Order){  String sql = "select od.ID\_Item, m.ItemName, od.Quantity, od.price, od.totalPrice from OrderDetail od "  + "inner join Orders o ON o.ID\_Order = od.ID\_Order "  + "inner join tables t ON t.ID\_Table = o.ID\_table "  +"inner join MenuItems m ON od.ID\_Item = m.ID\_Item "  + "where od.ID\_order = ?";  String cols[] = {"ID\_Item","ItemName","Quantity","Price","TotalPrice"};  return getListOfArray(sql, cols, ID\_Order);  }  } |

## Viết mã cho ứng dụng

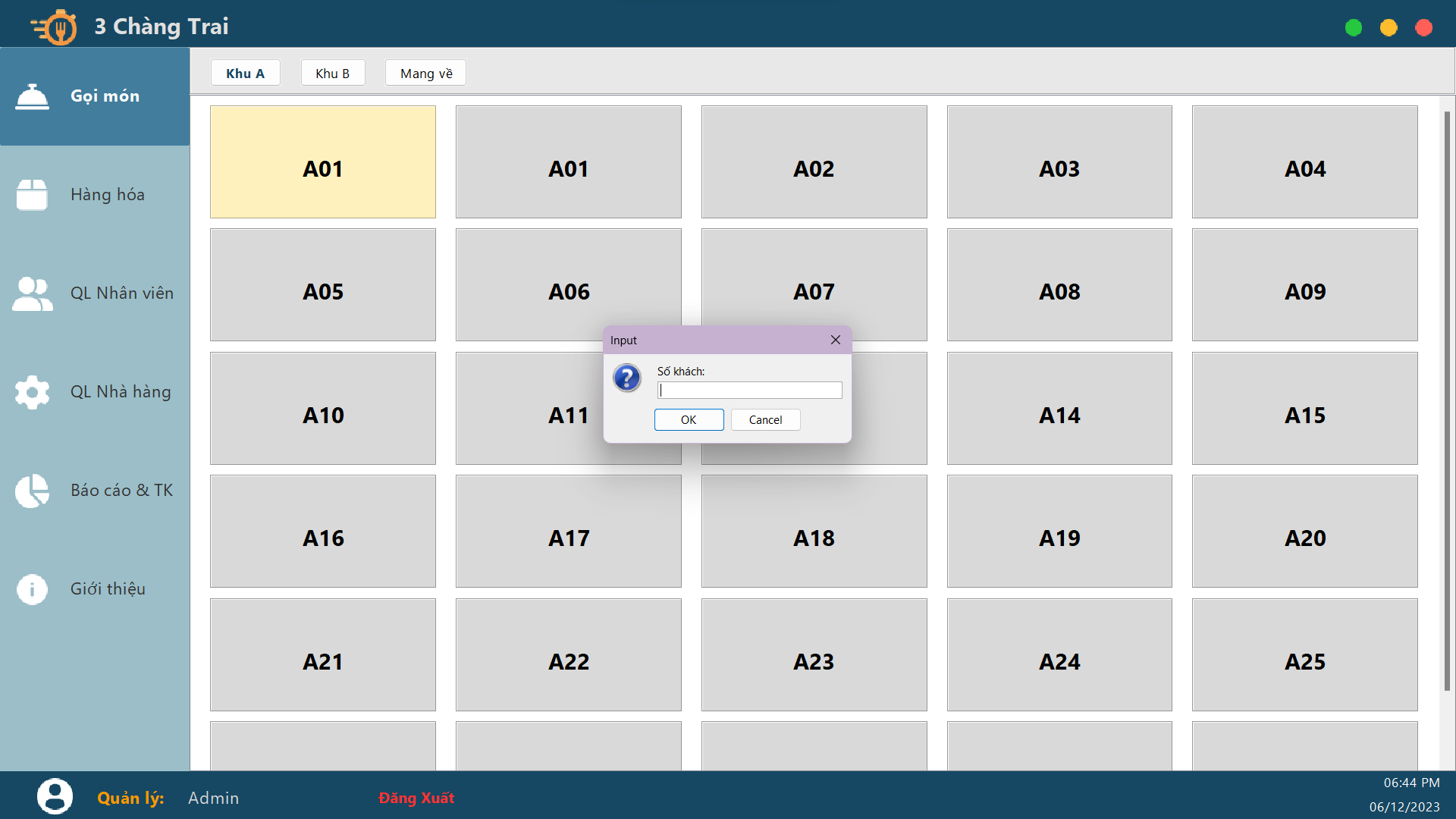
### Xử lý Form gọi món

#### Giao diện bàn, khu vực

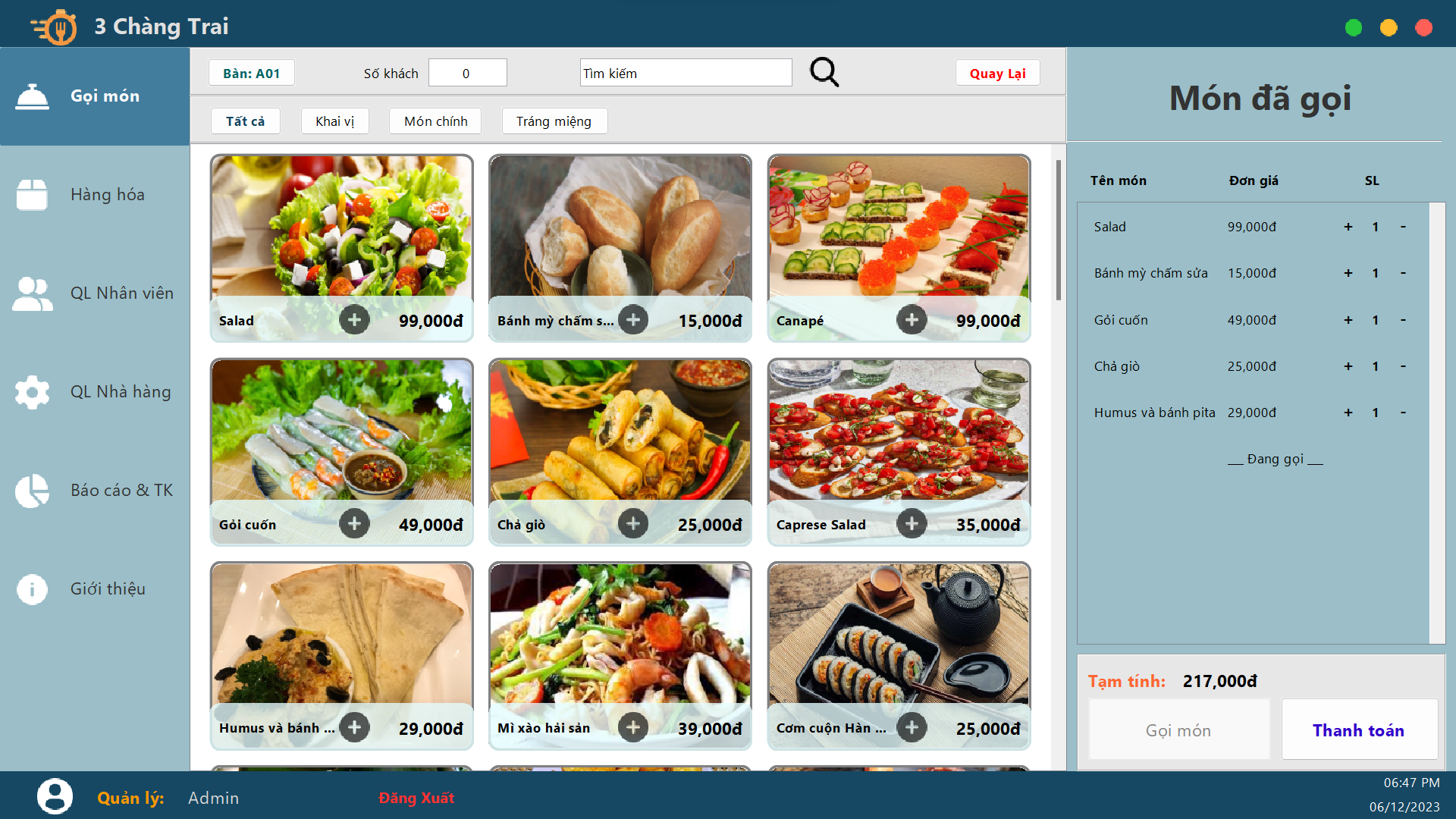


#### Giao diện goi món

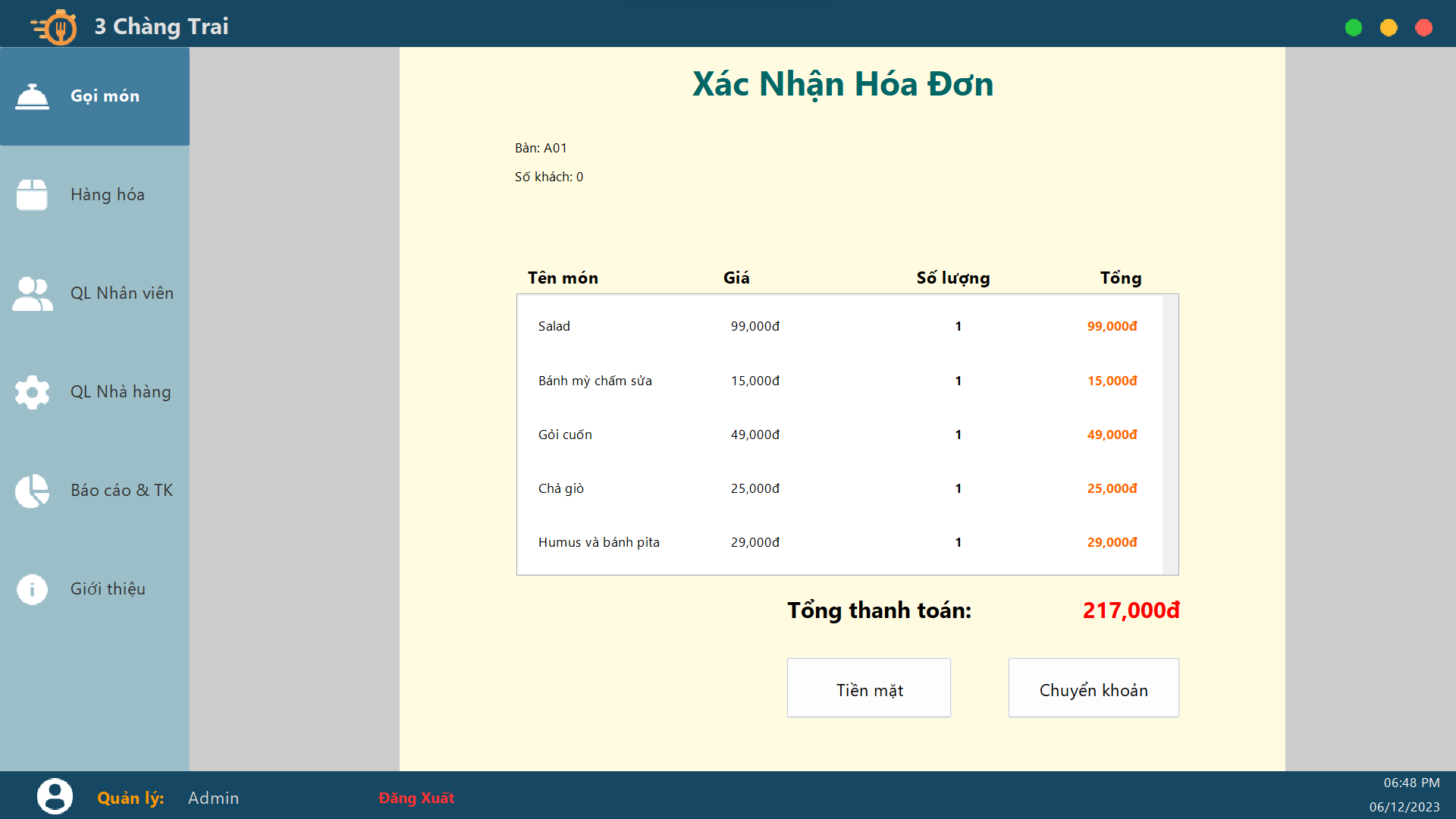
* Nhập số khách khi mở bàn



* Mở bàn và vào phần gọi món, chọn món cần dùng và nhấn gọi món



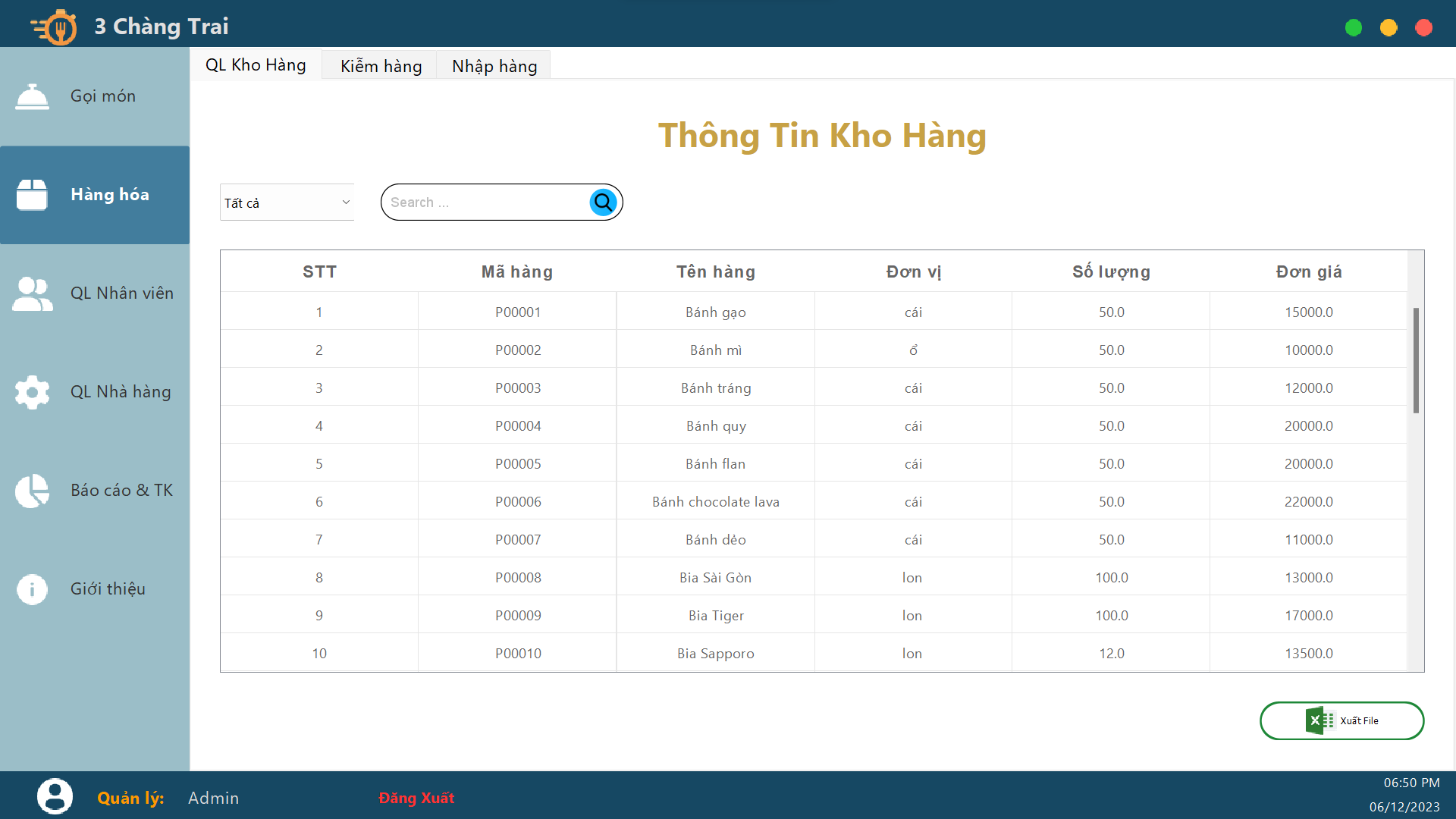
* Bấm thanh toán, có 2 hình thức là tiền mặt và chuyển khoản



### Xử lý Form hàng hóa

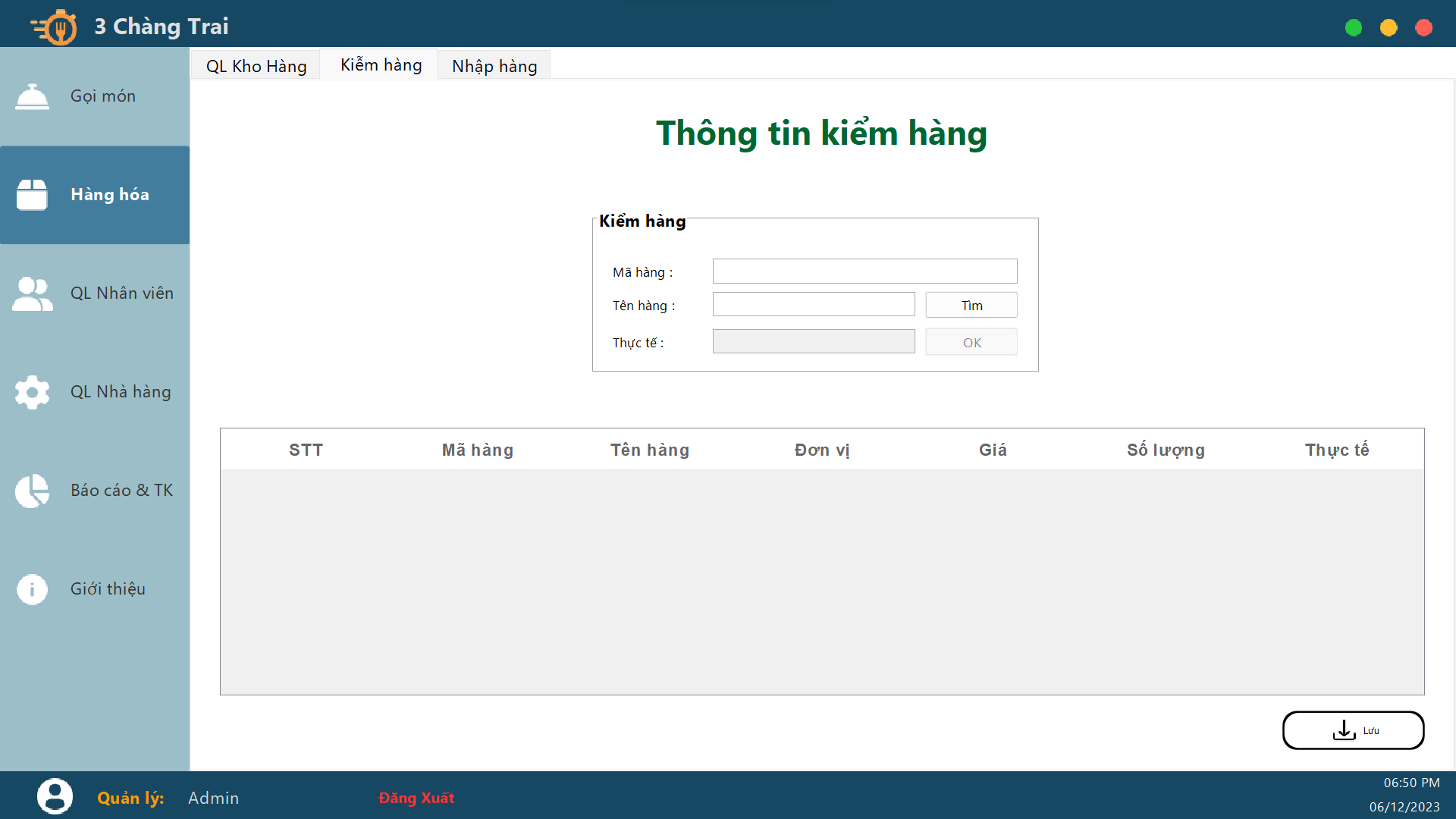
#### Quản lý kho hàng

* Tìm kiếm các mặt hàng cần thiết nếu cần và xuất ra file excel



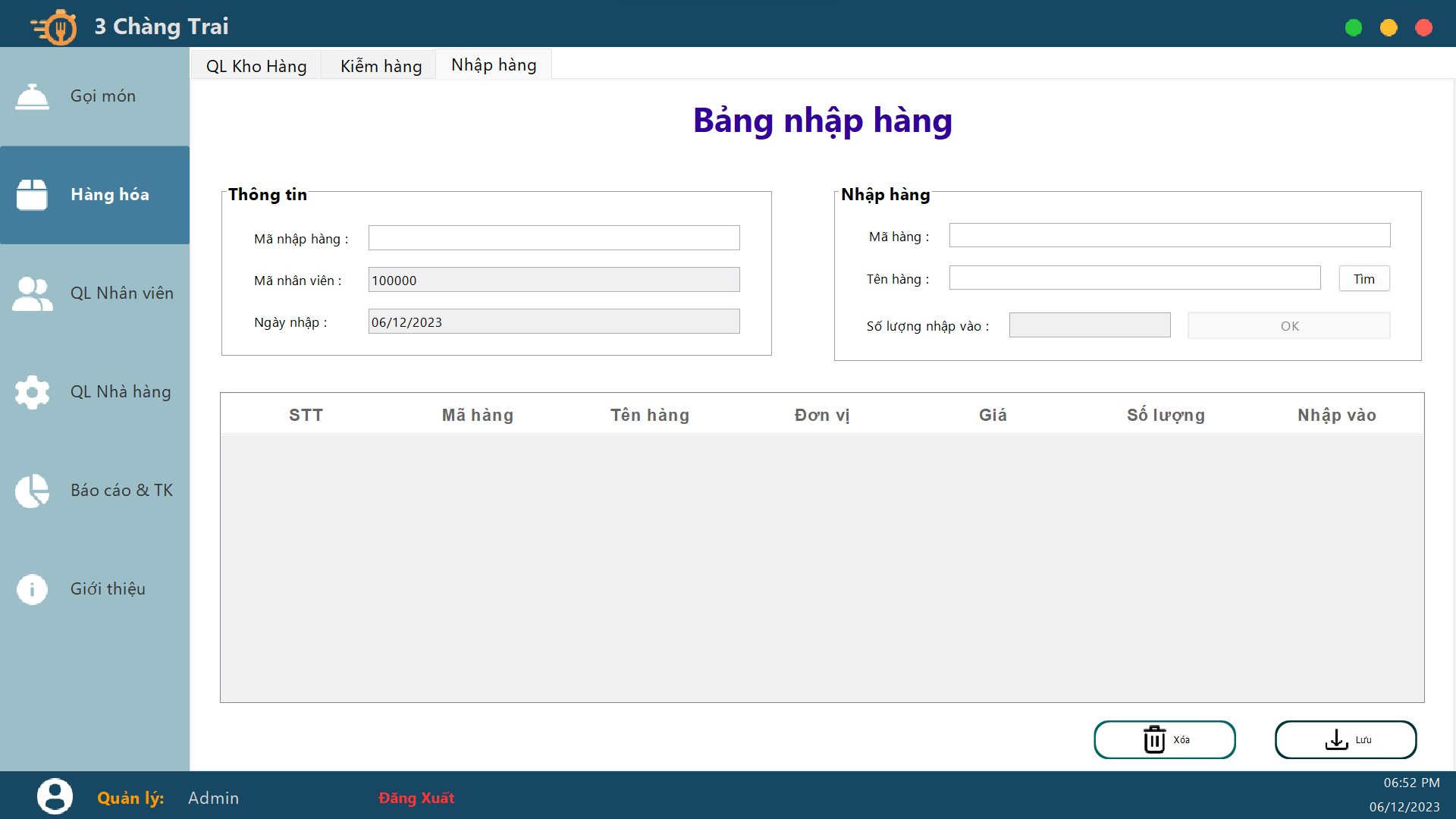
#### Quản lý kiểm hàng

* Kiểm tra hàng hóa trong kho và cập nhập trong CSDL



#### Quản lý nhập hàng

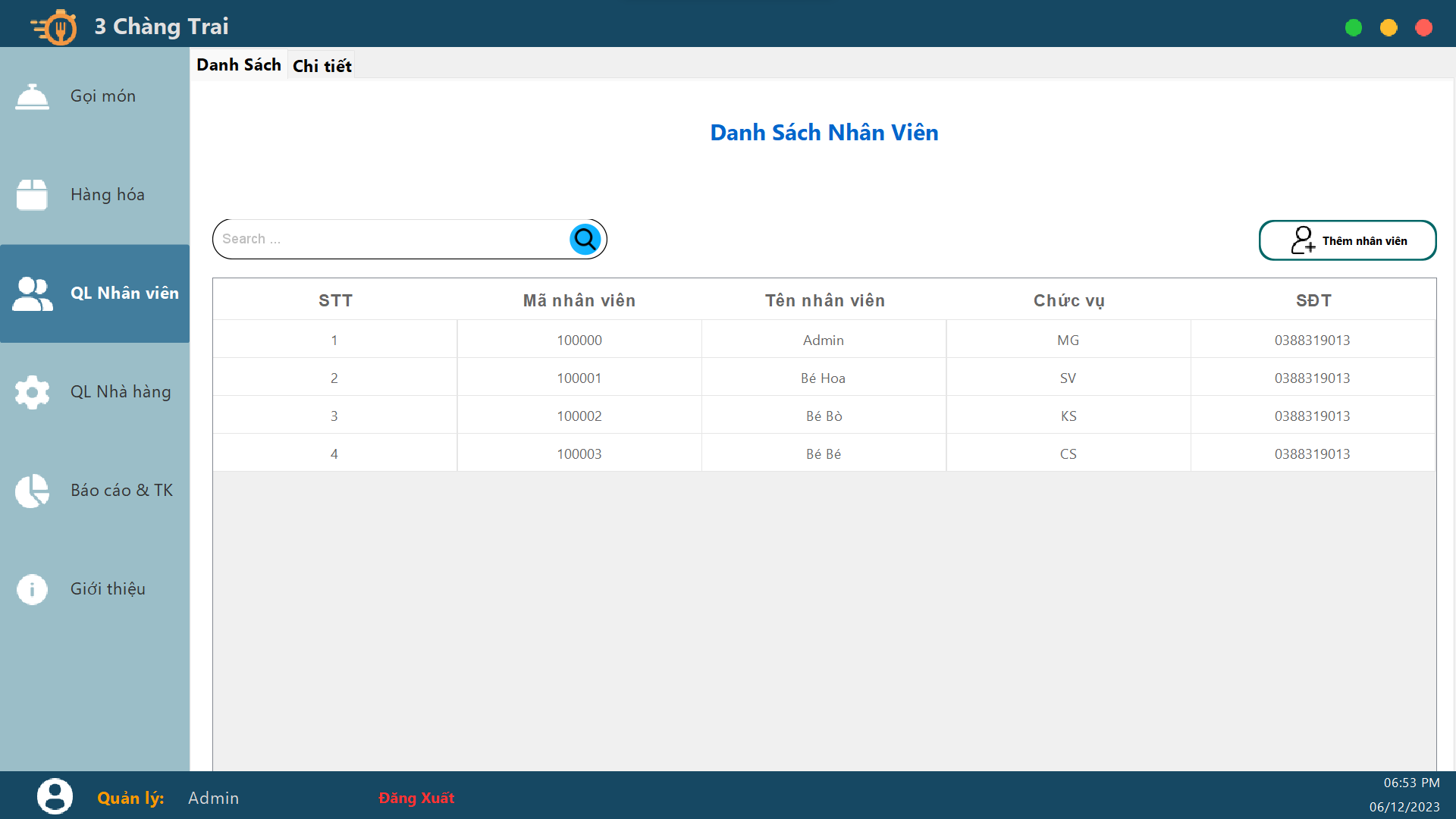
* Nhập vào các hàng hóa còn thiếu



### Xử lý Form quản lý nhân viên

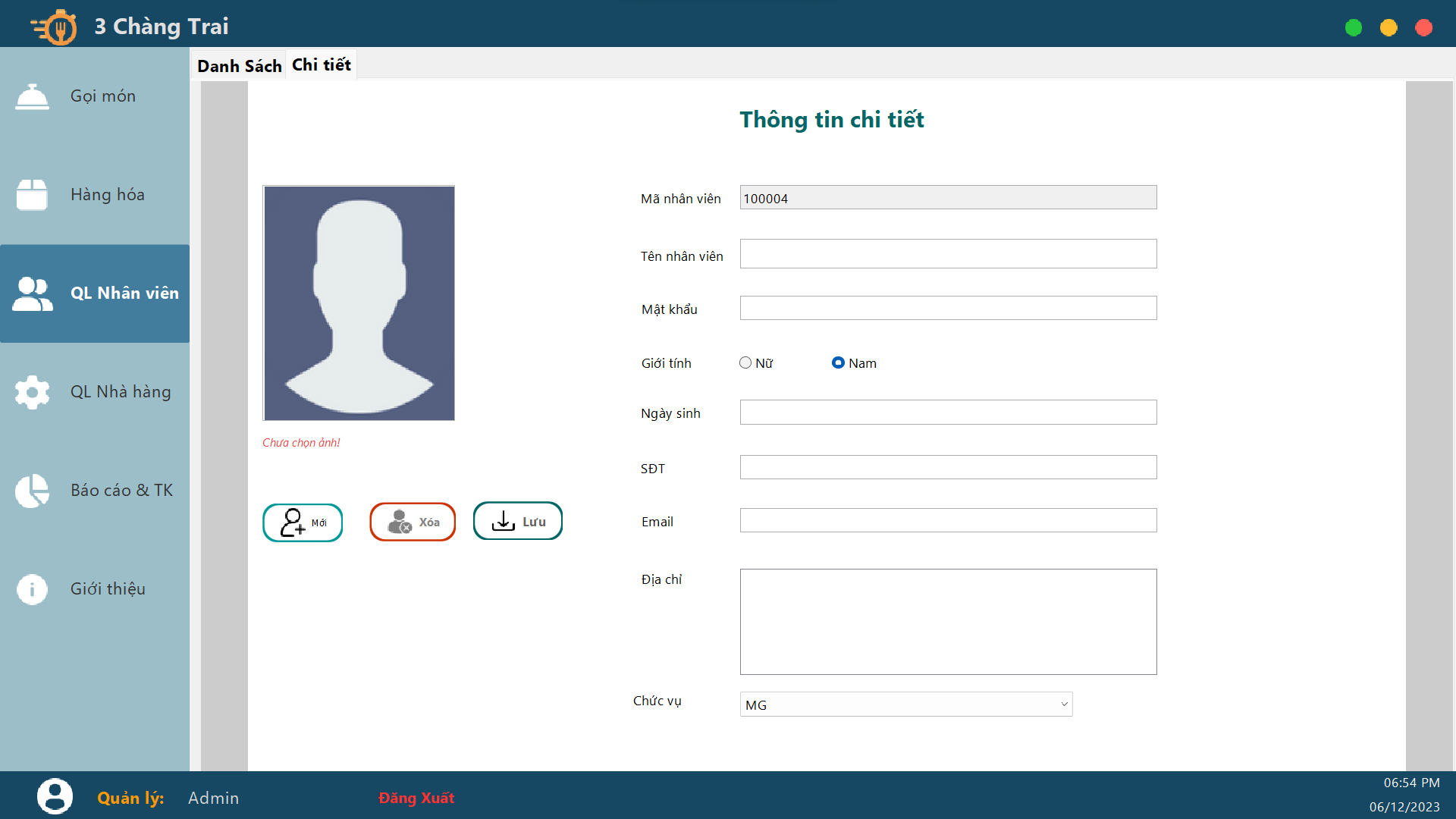
#### Quản lý danh sách nhân viên

* Hiện thị tất cả nhân viên và tìm kiếm có thể thêm nhân viên



#### Quản lý nhân viên chi tiết

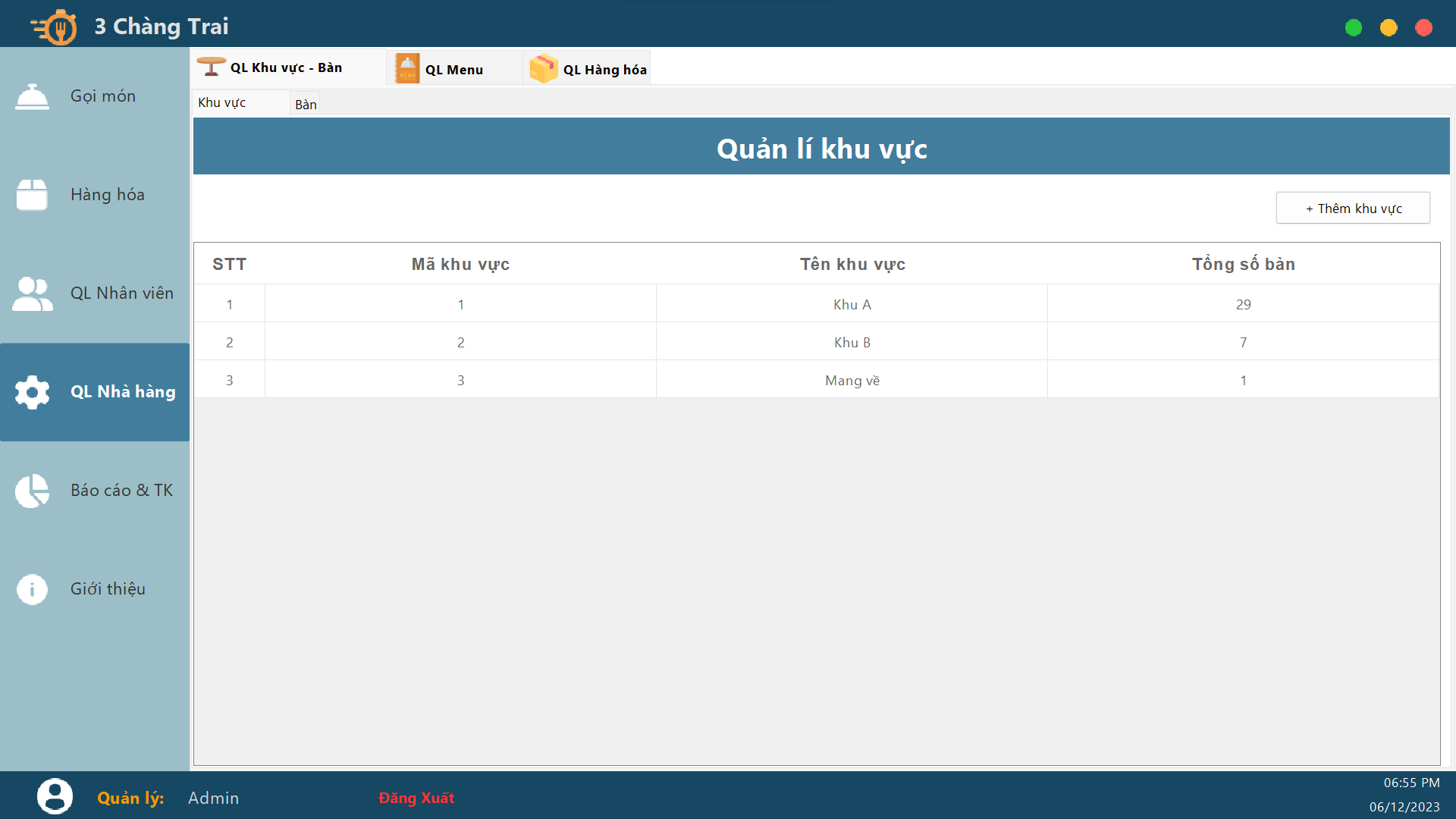
* Thêm, xóa, cập nhật thông tin nhân viên



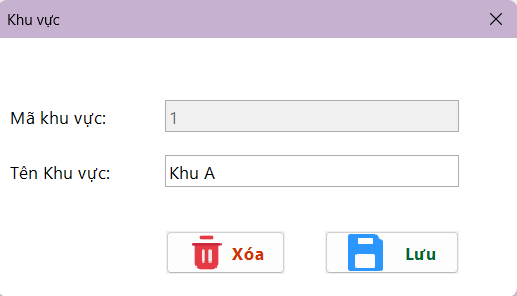
### Xử lý Form quản lý nhà hàng

#### Quản lý khu vực và bàn

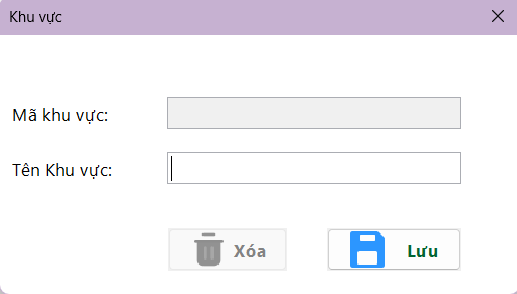
* **Quản lý khu vực**



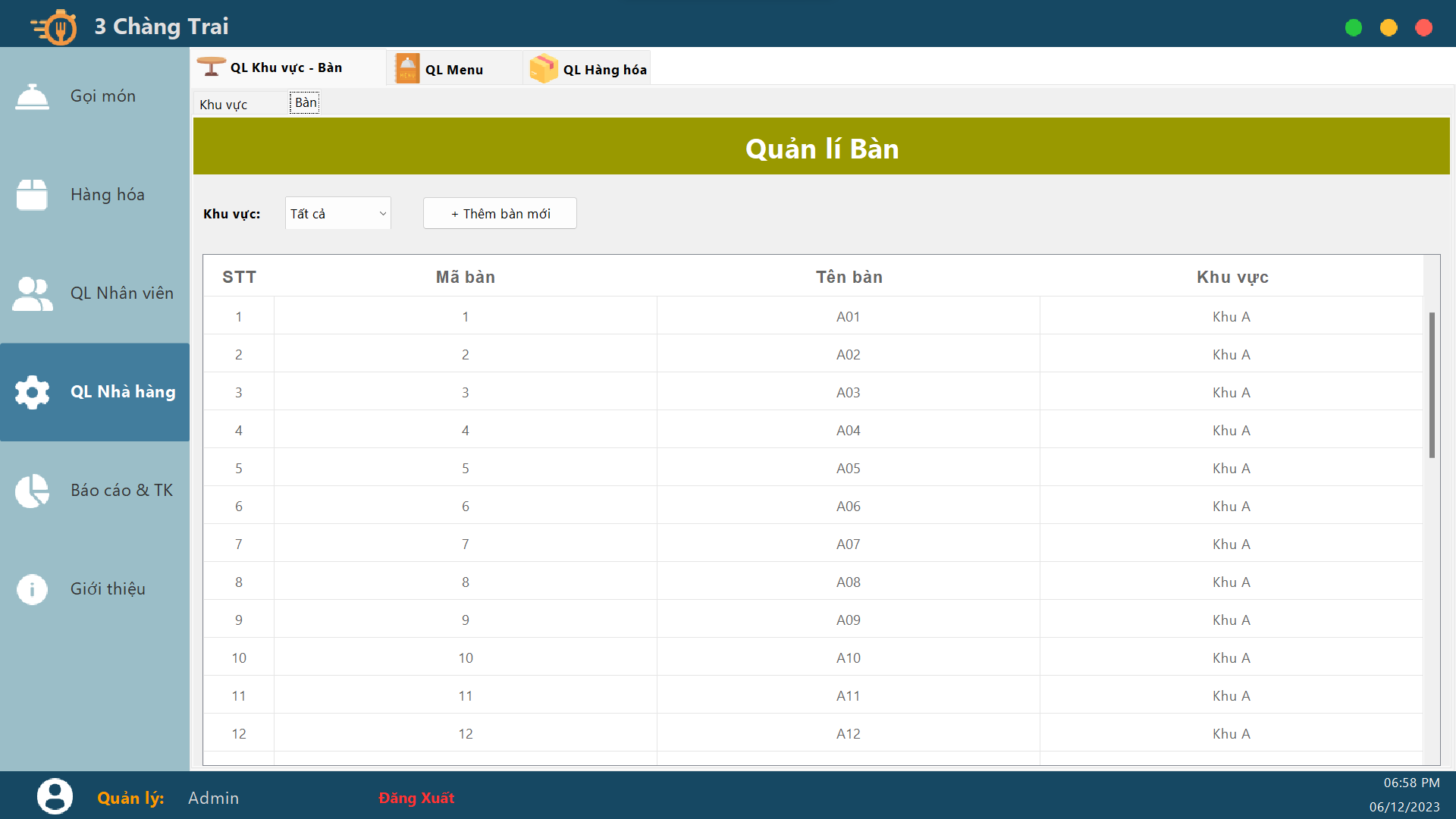
* Cập nhật tên khu vực



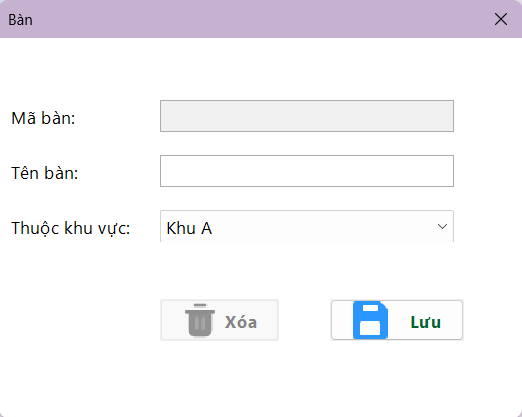
* Thêm khu vực



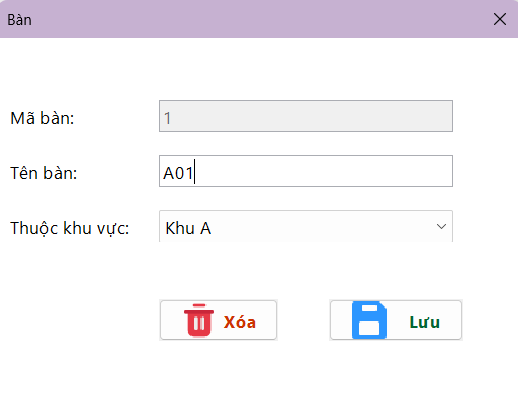
* **Quản lý bàn**

****

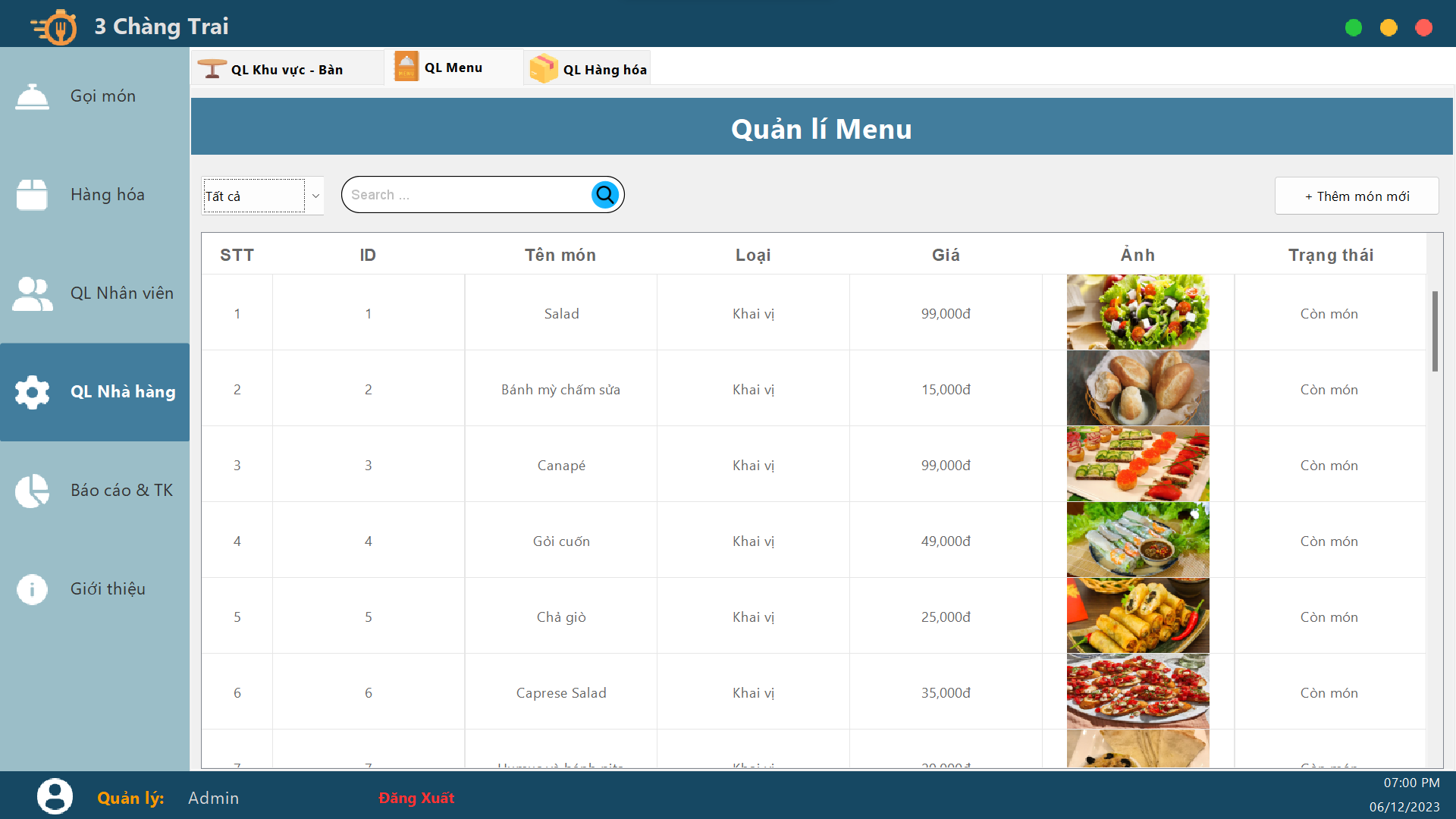
* Thêm bàn mới

****

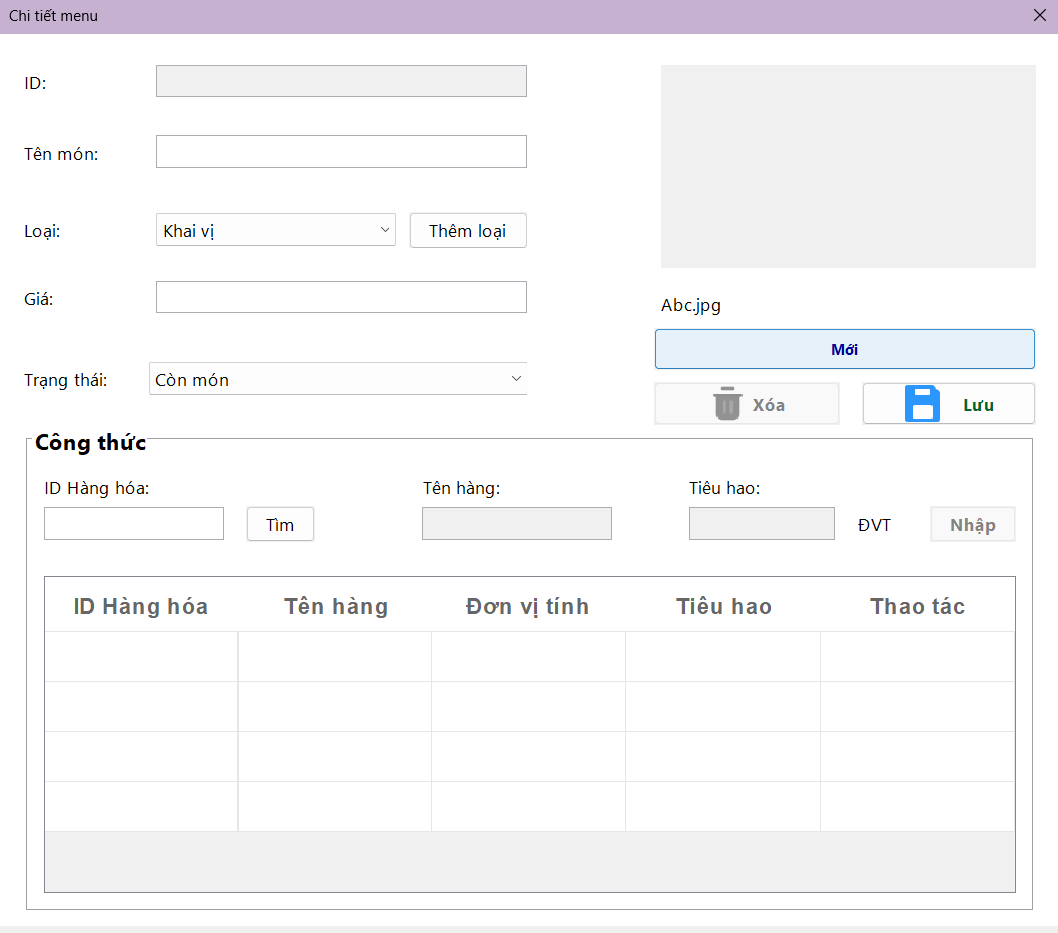
* Cập nhật tên bàn



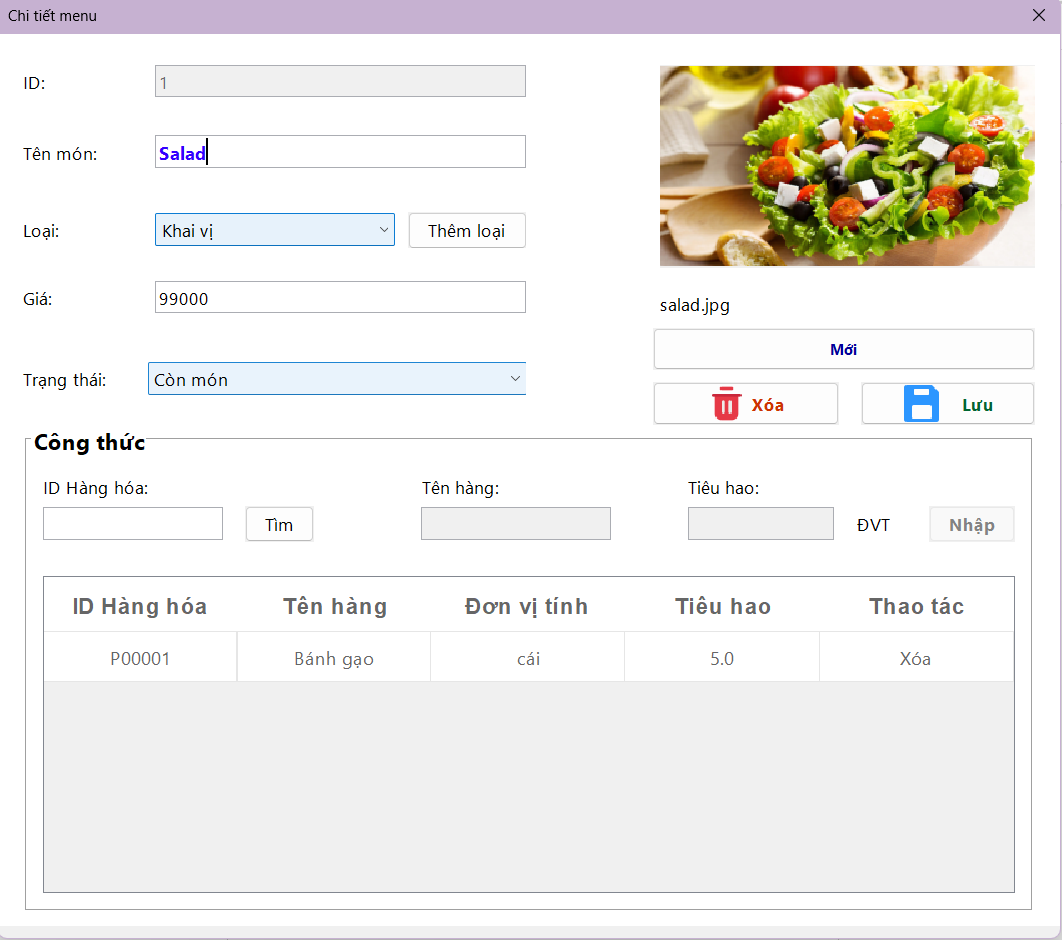
#### Quản lý menu



* Thêm món



* Cập nhật món

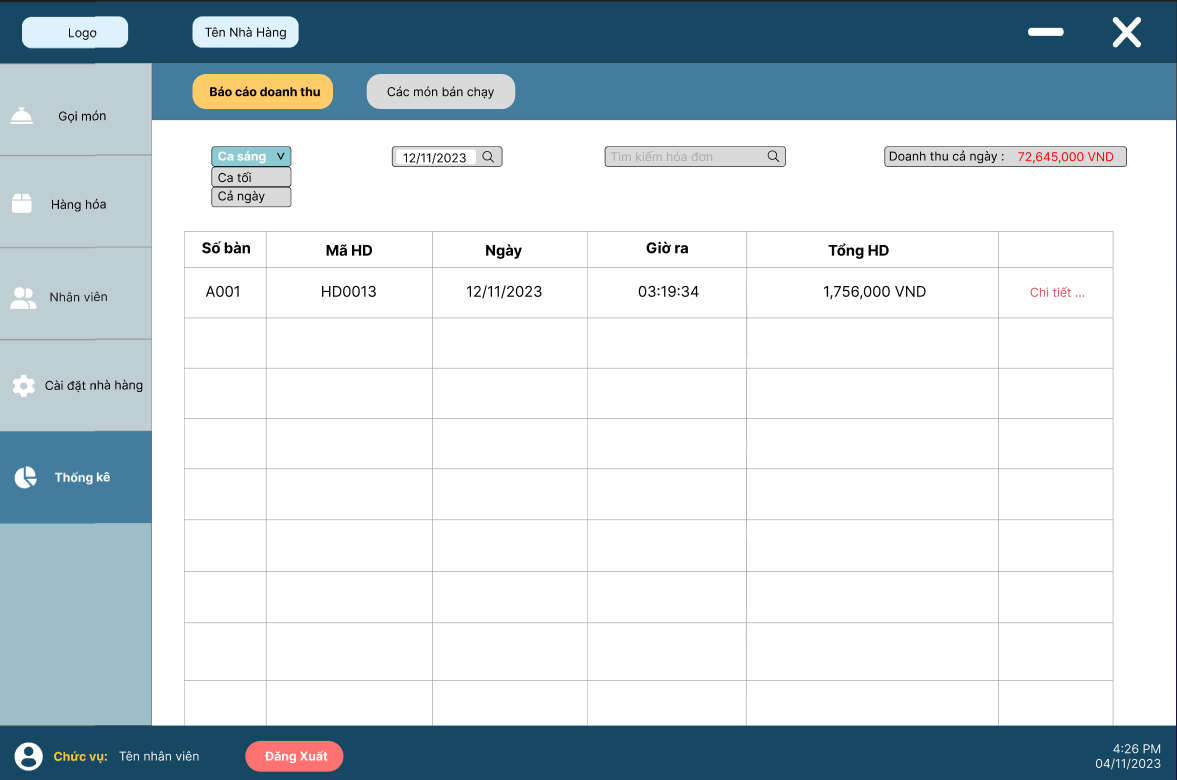


#### Quản lý hàng hóa

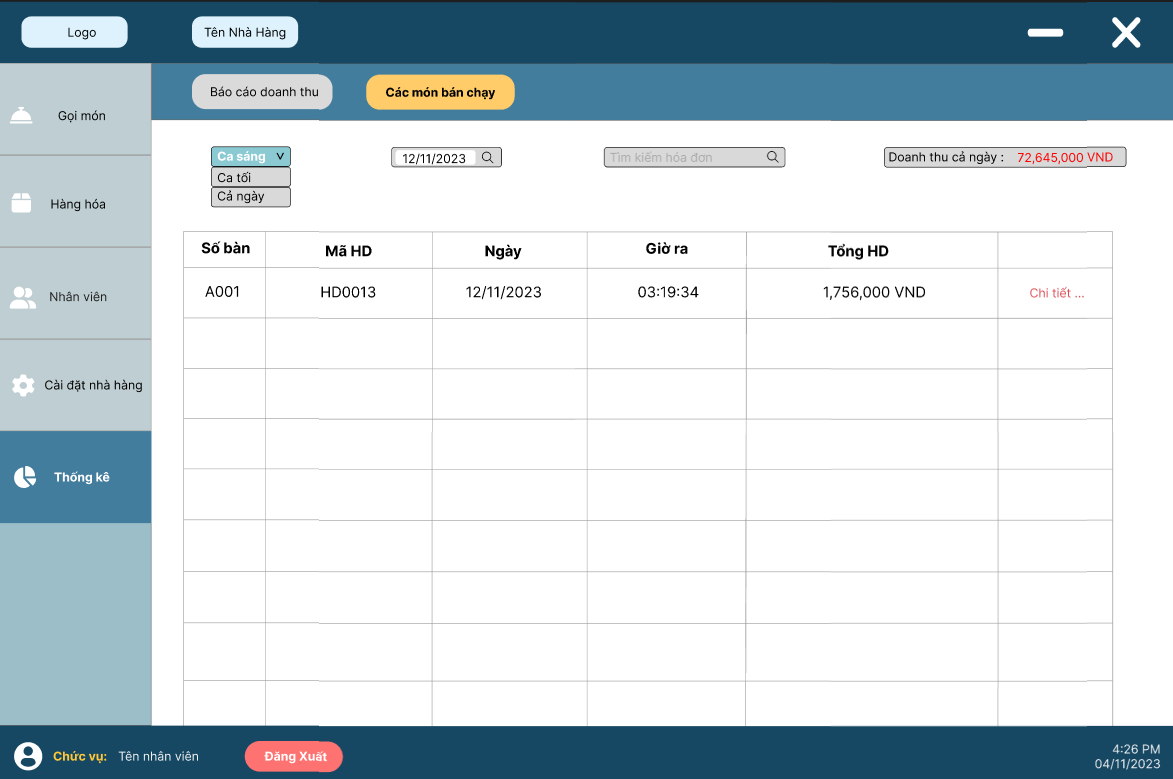
* Bổ sung

### Xử lý Form báo cáo và thống kê

#### Báo cáo doanh thu



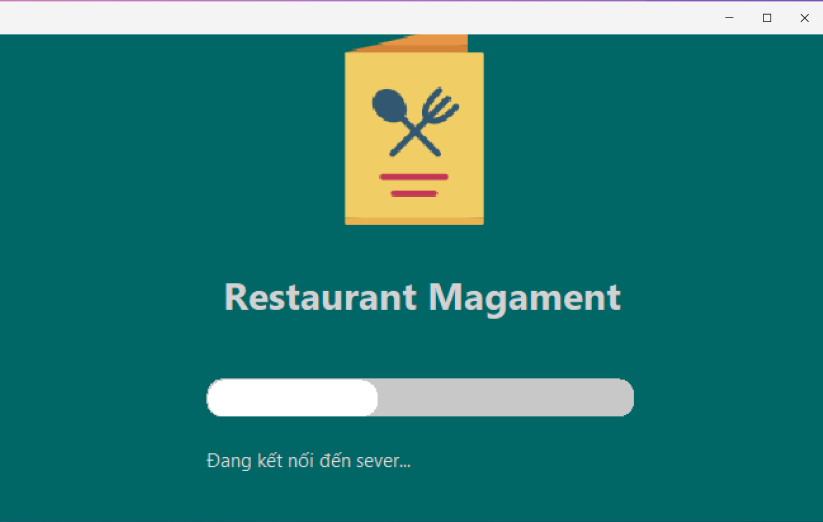
#### Thông tin mặt hàng bán chạy và được gọi nhiều nhất



### Xử lý Form giới thiệu

* Bổ sung

### Xử lý Form màng hình chào khi khởi động

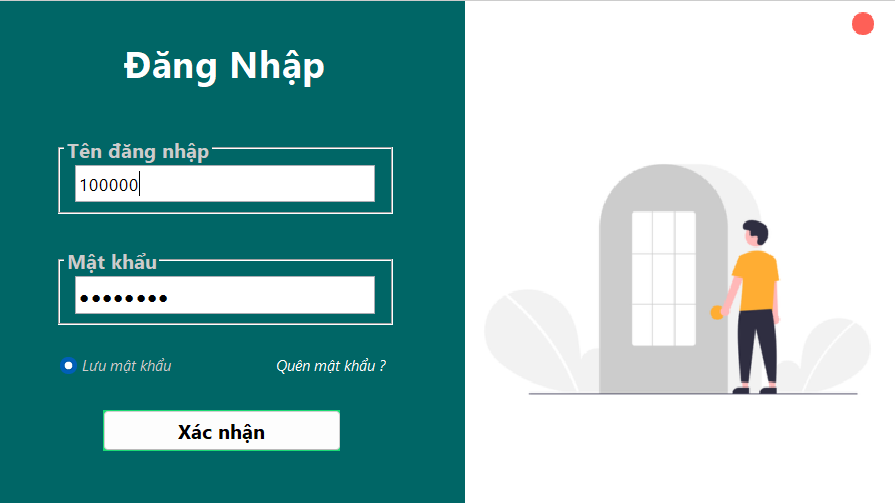


* Thực hiện

|  |
| --- |
| package UI.Form;  import Dao.procDao;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.Timer;  public class SpalshScreen extends javax.swing.JDialog {  public SpalshScreen(java.awt.Frame parent, boolean modal) {  super(parent, modal);  initComponents();  new Timer(10, new ActionListener() {  private int value = 0;  @Override  public void actionPerformed(ActionEvent e) {  prLoding.setValue(value);  value += 2;  if (value == 100) {  dispose();  };  }  }).start();  }  @SuppressWarnings("unchecked")  // <editor-fold defaultstate="collapsed" desc="Generated Code">  private void initComponents() {  jPanel1 = new javax.swing.JPanel();  jLabel1 = new javax.swing.JLabel();  jLabel2 = new javax.swing.JLabel();  prLoding = new UI.Compoment.ProgressBar();  lblTxt = new javax.swing.JLabel();  setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);  setUndecorated(true);  setResizable(false);  jPanel1.setBackground(new java.awt.Color(0, 102, 102));  jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);  jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/IMG/logo128.png"))); // NOI18N  jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N  jLabel2.setForeground(new java.awt.Color(210, 210, 210));  jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);  jLabel2.setText("Restaurant Magament");  prLoding.setForeground(new java.awt.Color(255, 255, 255));  prLoding.setValue(40);  prLoding.setBorderPainted(false);  prLoding.setColorString(new java.awt.Color(255, 153, 0));  lblTxt.setForeground(new java.awt.Color(215, 215, 215));  lblTxt.setText("Đang kết nối đến sever...");  javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  jPanel1.setLayout(jPanel1Layout);  jPanel1Layout.setHorizontalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addContainerGap(132, Short.MAX\_VALUE)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 271, Short.MAX\_VALUE)  .addComponent(prLoding, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))  .addComponent(lblTxt, javax.swing.GroupLayout.PREFERRED\_SIZE, 188, javax.swing.GroupLayout.PREFERRED\_SIZE))  .addGap(119, 119, 119))  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()  .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 119, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(199, 199, 199))))  );  jPanel1Layout.setVerticalGroup(  jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addGroup(jPanel1Layout.createSequentialGroup()  .addGap(21, 21, 21)  .addComponent(jLabel1)  .addGap(28, 28, 28)  .addComponent(jLabel2)  .addGap(37, 37, 37)  .addComponent(prLoding, javax.swing.GroupLayout.PREFERRED\_SIZE, 25, javax.swing.GroupLayout.PREFERRED\_SIZE)  .addGap(18, 18, 18)  .addComponent(lblTxt)  .addContainerGap(57, Short.MAX\_VALUE))  );  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  getContentPane().setLayout(layout);  layout.setHorizontalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  );  layout.setVerticalGroup(  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)  );  pack();  setLocationRelativeTo(null);  }// </editor-fold>  public static void main(String args[]) {  /\* Set the Nimbus look and feel \*/  //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  \*/  try {  for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {  if ("Windows".equals(info.getName())) {  javax.swing.UIManager.setLookAndFeel(info.getClassName());  break;  }  }  } catch (ClassNotFoundException ex) {  java.util.logging.Logger.getLogger(SpalshScreen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (InstantiationException ex) {  java.util.logging.Logger.getLogger(SpalshScreen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (IllegalAccessException ex) {  java.util.logging.Logger.getLogger(SpalshScreen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  } catch (javax.swing.UnsupportedLookAndFeelException ex) {  java.util.logging.Logger.getLogger(SpalshScreen.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);  }  //</editor-fold>  /\* Create and display the dialog \*/  java.awt.EventQueue.invokeLater(new Runnable() {  public void run() {  SpalshScreen dialog = new SpalshScreen(new javax.swing.JFrame(), true);  dialog.addWindowListener(new java.awt.event.WindowAdapter() {  @Override  public void windowClosing(java.awt.event.WindowEvent e) {  System.exit(0);  }  });  dialog.setVisible(true);  }  });  }  // Variables declaration - do not modify  private javax.swing.JLabel jLabel1;  private javax.swing.JLabel jLabel2;  private javax.swing.JPanel jPanel1;  private javax.swing.JLabel lblTxt;  private UI.Compoment.ProgressBar prLoding;  // End of variables declaration  } |

### Xử lý Form đăng nhập

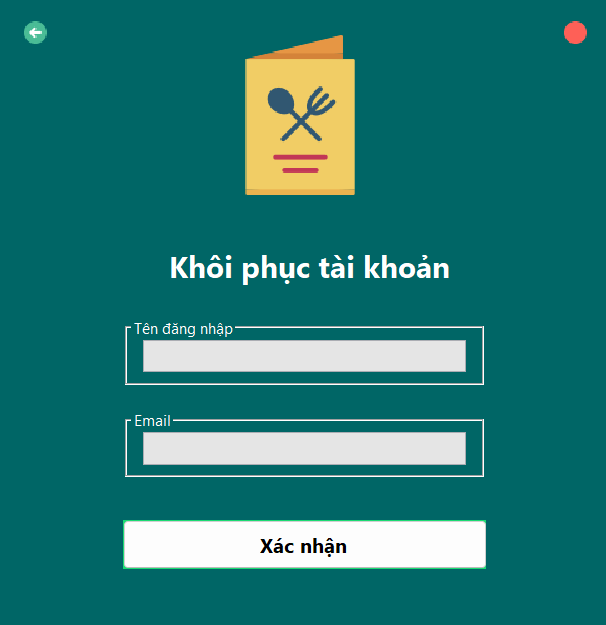
* Form đăng nhập



|  |
| --- |
| public class Login extends javax.swing.JDialog {  EmployeesDao EmDao = new EmployeesDao();  public Login(java.awt.Frame parent, boolean modal) {  super(parent, modal);  initComponents();    //Kiễm tra nếu có lưu mật khẩu  Employees el = new Employees();  try {  el =(Employees) FileControl.readFile("src/Auth.dat");  if(el != null){  txtUserName.setText(el.getID\_Employee()+"");  txtPassword.setText(el.getPassword());  }  } catch (Exception e) {  }      //Bắt sự kiện nút exit  btnClose.addMouseListener(new MouseAdapter() {  @Override  public void mouseEntered(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/xDot.png"));  }  @Override  public void mouseExited(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/redDot.png"));  }  });  }  public boolean Validate(){  boolean check = true;  int UserName = Integer.valueOf(txtUserName.getText());  String Password = new String(txtPassword.getPassword());    try{  int UsernameINT = Integer.valueOf(UserName);  if(!Validate.isLength(txtUserName, UserName + "", 6, "User name")){  txtUserName.setBackground(Color.YELLOW);  check = false;  }  }catch(Exception e){  msg.Warning("Vui lòng nhập user name là các số.");  txtUserName.setBackground(Color.YELLOW);  check = false;  }    if(!Validate.isLength(txtPassword, Password, 6, "mật khẩu")) check = false;  return check;  }    public void submitLogin() {  int UserName = Integer.valueOf(txtUserName.getText());  String Password = new String(txtPassword.getPassword());    String HashPassword = null;  try {  HashPassword = Encryption.hashPassword(Password);  } catch (NoSuchAlgorithmException ex) {  msg.Error("Có lỗi bất định "+ ex.getMessage());  }    Employees nv = EmDao.selectById(UserName);  System.out.println(nv);  if (nv != null) {  if (nv.getPassword().equals(HashPassword)) {  Auth.user = nv;  if (rdSavePass.isSelected()) {  nv.setPassword(Password);  FileControl.writeFile(nv, "src/Auth.dat");  } else {  FileControl.deleteFile("src/Auth.dat");  }  this.dispose();  return;  }  }  msg.Error("Sai tên đăng nhập hoặc mật khẩu");  } |

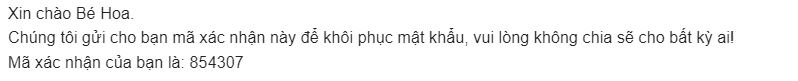
* Quên mật khẩu

|  |
| --- |
| public class ForgetPass extends javax.swing.JDialog {  public ForgetPass(java.awt.Frame parent, boolean modal) {  super(parent, modal);  initComponents();  //Bắt sự kiện nút exit  btnClose.addMouseListener(new MouseAdapter() {  @Override  public void mouseEntered(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/xDot.png"));  }  @Override  public void mouseExited(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/redDot.png"));  }  });  //Bắt sự kiện nút exit  btnBack.addMouseListener(new MouseAdapter() {  @Override  public void mouseEntered(MouseEvent e) {  btnBack.setIcon(new ImageIcon("src/IMG/backHover.png"));  }  @Override  public void mouseExited(MouseEvent e) {  btnBack.setIcon(new ImageIcon("src/IMG/back.png"));  }  });  }  public boolean Validate() {  boolean check = true;  int UserName = Integer.valueOf(txtUserName.getText());  String Email = txtEmail.getText();  try {  int UsernameINT = Integer.valueOf(UserName);  if (!Validate.isLength(txtUserName, UserName + "", 6, "User name")) {  txtUserName.setBackground(Color.YELLOW);  check = false;  }  } catch (Exception e) {  msg.Warning("Vui lòng nhập user name là các số.");  txtUserName.setBackground(Color.YELLOW);  check = false;  }  if (!Validate.isEmail(txtEmail, Email)) {  check = false;  }  return check;  }  public void submit() {  if (Validate()) {  //Thực hiện kiễm tra và gửi maill  EmployeesDao dao = new EmployeesDao();  int UserName = Integer.valueOf(txtUserName.getText());  Employees nv = dao.selectById(UserName);  String randomNumber = "";  if (nv != null && txtEmail.getText().equalsIgnoreCase(nv.getEmail())) {  Random random = new Random();  for (int i = 0; i < 6; i++) {  int digit = random.nextInt(10); // Tạo số ngẫu nhiên từ 0 đến 9  randomNumber += digit; // Ghép số vào chuỗi  }  EmailSender.sendEmail(nv.getEmail(), "Restaurant MG - Cấp lại mật khẩu",  "Xin chào " + nv.getFullName()  + ".\nChúng tôi gửi cho bạn mã xác nhận này để khôi phục mật khẩu, vui lòng không chia sẽ cho bất kỳ ai!"  + "\nMã xác nhận của bạn là: " + randomNumber);  } else {  msg.Error("Sai tên đăng nhập hoặc Email!!!");  return;  }  // Loại bỏ thành phần cũ  this.setLayout(new BorderLayout());  this.remove(pnBg);  JPanel pn = new resetPassForm(nv.getFullName(), nv.getID\_Employee(), randomNumber, this);  this.add(pn);  // Cập nhật giao diện  this.revalidate();  this.repaint();  }  } |



* Gửi email khi quên mật khẩu, đổi mật khẩu và quay trở lại màng mình để đăng nhập





|  |
| --- |
| public class resetPassForm extends javax.swing.JPanel {  JDialog d = new JDialog();  String code;  int ID;  public resetPassForm(String name,int ID, String code, JDialog d) {  initComponents();  lbUser.setText(name);  this.code = code;  this.ID = ID;  this.d = d;  //Bắt sự kiện nút exit  btnClose.addMouseListener(new MouseAdapter() {  @Override  public void mouseEntered(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/xDot.png"));  }  @Override  public void mouseExited(MouseEvent e) {  btnClose.setIcon(new ImageIcon("src/IMG/redDot.png"));  }  });      //Bắt sự kiện nút exit  btnBack.addMouseListener(new MouseAdapter() {  @Override  public void mouseEntered(MouseEvent e) {  btnBack.setIcon(new ImageIcon("src/IMG/backHover.png"));  }  @Override  public void mouseExited(MouseEvent e) {  btnBack.setIcon(new ImageIcon("src/IMG/back.png"));  }  });  }    public boolean Check(){  if(!Validate.isLength(txtCode, txtCode.getText(), 6)) return false;  try{  int code = Integer.valueOf(txtCode.getText());  }catch(Exception e){  msg.Error("Mã xác nhận không đúng định dạng!");  return false;  }    if(!Validate.isLength(txtPass, new String(txtPass.getPassword()), 6)) return false;    if(!new String(txtNhapLai.getPassword()).equals(new String(txtPass.getPassword()))){  msg.Error("Mật khẩu nhập lại không chính xác");  return false;  }    if(!code.equals(txtCode.getText())){  msg.Error("Mã xác nhận không chính xác!. Vui lòng thử lại");  return false;  }    return true;  } |

# Kiểm thử

## Kiểm thử form X

## Kiểm thử form Y

# Đóng gói và triển khai

## Hướng dẫn chuyển đổi jar thành exe

## Hướng dẫn cài đặt triển khai

## Hướng dẫn sử dụng phần mềm