# **DANH MỤC TỪ VIẾT TẮT**

# **DANH MỤC BẢNG BIỂU VÀ SƠ ĐỒ**

Contents

[**DANH MỤC TỪ VIẾT TẮT** 1](#_Toc148207746)

[**DANH MỤC BẢNG BIỂU VÀ SƠ ĐỒ** 2](#_Toc148207747)

[**Chương 1. Giới thiệu về đề tài** 4](#_Toc148207748)

[1.1. Giới thiệu về đề tài 4](#_Toc148207749)

[1.2. Kế hoạch làm đề tài 4](#_Toc148207750)

[1.3. Các công nghệ sử dụng 5](#_Toc148207751)

[1.4. Kết chương 1 11](#_Toc148207752)

[Chương 2. Mô hình hệ thống 12](#_Toc148207753)

[2.1. Mô hình chức năng 12](#_Toc148207754)

[2.2. Thiết kế giao diện 12](#_Toc148207755)

[2.3. Mô hình kiến trúc hệ thống 13](#_Toc148207756)

[2.4. Mô hình cơ sở dữ liệu 14](#_Toc148207757)

[2.5. Giao diện người dùng 15](#_Toc148207758)

[2.6 Các tiện ích 20](#_Toc148207759)

[Chương 3. Phát triển hệ thống 22](#_Toc148207760)

[3.1. Phát triển các giao diện 22](#_Toc148207761)

[3.2. Phát triển các lớp, triển khai các giao diện 23](#_Toc148207762)

[3.3. Phát triển giao diện người dùng 52](#_Toc148207763)

[3.4. Triển khai cơ sở dữ liệu 54](#_Toc148207764)

[Kết luận 58](#_Toc148207765)

[Kết quả đạt được 58](#_Toc148207766)

[Hạn chế và hướng phát triển của đề tài 58](#_Toc148207767)

[Danh mục sách tham khảo 59](#_Toc148207768)

# **Chương 1. Giới thiệu về đề tài**

* 1. Giới thiệu về đề tài

Phần mềm Quản lý Quán Cafe" là một ứng dụng hiện đại được phát triển để đáp ứng nhu cầu quản lý quán cafe trong thực tế ngày nay. Đây không chỉ là một hệ thống thông tin mà còn là một công cụ hữu ích giúp quán cafe hoạt động một cách hiệu quả và tối ưu hóa quá trình quản lý. Ứng dụng này sử dụng giao diện người dùng thân thiện để tạo ra một môi trường làm việc năng động và dễ quản lý. Dưới đây là một số điểm nổi bật về ứng dụng này:

Tích hợp đầy đủ cho hoạt động quản lý: Phần mềm này cung cấp tất cả công cụ cần thiết cho việc quản lý một quán cafe, bao gồm quản lý bàn ăn, thực đơn, hóa đơn và tài khoản nhân viên. Với tích hợp đầy đủ, quán cafe có thể dễ dàng theo dõi và cập nhật thông tin về hoạt động kinh doanh hàng ngày.

Ứng dụng cho phép quán cafe quản lý tài khoản nhân viên với các quyền hạn và phân quyền truy cập khác nhau. Điều này giúp kiểm soát việc sử dụng ứng dụng và đảm bảo rằng chỉ những người được ủy quyền có thể truy cập các tính năng quan trọng.

Thay vì sử dụng giấy và viết tay, ứng dụng này cho phép quán cafe ghi chép toàn bộ hoạt động kinh doanh một cách kỹ thuật số. Các thông tin về hóa đơn, bàn ăn và thực đơn được lưu trữ một cách an toàn và dễ dàng truy cập.

Một trong những tính năng quan trọng của phần mềm này là khả năng tạo báo cáo và thống kê. Quán cafe có thể theo dõi doanh thu, món ăn phổ biến, hoạt động của nhân viên và nhiều thông tin khác để ra quyết định kinh doanh thông minh.

Phần mềm giúp quán cafe tiết kiệm thời gian và nguồn lực bằng cách tự động tính toán tổng tiền cho mỗi hóa đơn, cập nhật trạng thái của bàn ăn và thực đơn, và loại bỏ công việc thủ công không hiệu quả.

Phần mềm này là một giải pháp tương lai sẵn sàng cho quán cafe. Với khả năng cải tiến và mở rộng, nó có thể điều chỉnh theo sự thay đổi trong nhu cầu kinh doanh của quán cafe.

Trong môi trường kinh doanh ngày nay, ứng dụng "Phần mềm Quản lý Quán Cafe" đóng vai trò quan trọng trong việc nâng cao hiệu suất quản lý, tạo ra trải nghiệm tốt hơn cho khách hàng và giúp quán cafe phát triển và thành công.

* 1. Kế hoạch làm đề tài

Kế hoạch làm đề tài "Phần mềm Quản lý Quán Cafe" có thể được chia thành các giai đoạn quan trọng dưới đây:

* Xác định yêu cầu và phân tích: Giai đoạn này sẽ bao gồm việc xác định rõ các yêu cầu cụ thể cho phần mềm quản lý quán cafe, bao gồm các tính năng cần thiết và giao diện người dùng. Nhiệm vụ cụ thể bao gồm:
* Làm rõ các tính năng quản lý, bao gồm quản lý bàn ăn, thực đơn, hóa đơn, tài khoản nhân viên và thống kê.
* Phân tích yêu cầu về cơ sở dữ liệu và cách lưu trữ thông tin.
* Xây dựng mô hình giao diện người dùng dự kiến.
* Thiết kế: Giai đoạn thiết kế sẽ bao gồm các công việc sau:
* Xây dựng cơ sở dữ liệu dựa trên yêu cầu.
* Thiết kế giao diện người dùng, bao gồm cách thức người dùng sẽ tương tác với phần mềm.
* Thiết kế cơ chế bảo mật và phân quyền truy cập.
* Xác định cách thức tích hợp các tính năng quản lý.
* Phát triển: Giai đoạn này bao gồm việc xây dựng phần mềm theo thiết kế đã hoàn chỉnh. Công việc cụ thể sẽ bao gồm:
* Phát triển cơ sở dữ liệu và các truy vấn cơ sở dữ liệu cần thiết.
* Xây dựng mã nguồn cho phần mềm, bao gồm các tính năng quản lý, giao diện người dùng và xử lý sự kiện.
* Kiểm tra và đảm bảo tính ổn định của phần mềm.
* Kiểm thử và sửa lỗi: Trong giai đoạn này, phần mềm sẽ được kiểm tra một cách kỹ lưỡng để xác định và sửa lỗi. Công việc kiểm thử bao gồm:
* Kiểm tra tính năng của phần mềm, đảm bảo chúng hoạt động đúng và đáp ứng các yêu cầu.
* Xác định và sửa lỗi gặp phải trong quá trình kiểm thử.
  1. Các công nghệ sử dụng
* Java Swing:

Java Swing là một thư viện (library) dành cho ngôn ngữ lập trình Java, được sử dụng để phát triển giao diện người dùng đồ họa (GUI) cho ứng dụng Java. Swing cung cấp một nền tảng mạnh mẽ để tạo ra các ứng dụng với giao diện đồ họa phong phú và tương tác. Dưới đây là một giới thiệu chi tiết và dài dòng về Java Swing:

* Lịch sử và Phiên bản:

Swing ra đời vào cuối thập kỷ 1990 và đã trở thành một phần của Java Foundation Classes (JFC).

Swing đã trải qua nhiều phiên bản và cải tiến, và hiện tại được tích hợp trong Java Standard Edition (SE).

* Ưu điểm của Java Swing:

Độc lập nền tảng: Ứng dụng Swing có thể chạy trên nhiều nền tảng khác nhau mà không cần sửa đổi mã nguồn, bao gồm Windows, macOS và Linux.

Thiết kế đa dạng: Swing cung cấp một loạt các thành phần giao diện người dùng đa dạng như nút bấm, ô văn bản, bảng, cây, cửa sổ, hộp thoại và nhiều thành phần tùy chỉnh khác.

Tích hợp dễ dàng: Bạn có thể tích hợp các thành phần Swing vào các ứng dụng Java một cách dễ dàng và tùy chỉnh chúng theo ý muốn.

Giao diện người dùng đẹp: Swing cho phép bạn tạo ra các giao diện người dùng đẹp và phức tạp với các đặc điểm giao diện đồ họa hiện đại.

Phong cách giao diện: Swing hỗ trợ nhiều phong cách giao diện, cho phép bạn tùy chỉnh giao diện người dùng theo các giao diện đồ họa khác nhau.

* Các thành phần chính trong Java Swing:

JFrame: Là cửa sổ chính của ứng dụng, được sử dụng để đặt các thành phần giao diện khác.

JPanel: Dùng để nhóm và quản lý các thành phần giao diện như nút bấm, hộp thoại, v.v.

JButton: Là nút bấm, được sử dụng để thực hiện hành động khi người dùng nhấn vào.

JLabel: Hiển thị văn bản hoặc hình ảnh.

JTextField và JTextArea: Cho phép nhập liệu văn bản.

JTable: Hiển thị dữ liệu trong bảng.

JComboBox: Dùng để chọn một giá trị từ danh sách thả xuống.

JCheckBox và JRadioButton: Dùng cho lựa chọn đa dạng.

JFileChooser: Cho phép chọn tệp và thư mục từ hệ thống tệp.

* Sự kiện và Tương tác:

Swing hỗ trợ xử lý sự kiện, cho phép người dùng đáp ứng khi người dùng tương tác với các thành phần giao diện.

Lập trình viên có thể thiết lập lắng nghe sự kiện cho nút bấm, ô văn bản, bảng và các thành phần khác.

* Gỡ lỗi và Thử nghiệm:

IDEs như NetBeans và Eclipse cung cấp trình gỡ lỗi tích hợp giúp lập trình viên xác định và sửa lỗi trong ứng dụng Swing.

Lập trình viên có thể xem trước giao diện người dùng để kiểm tra ứng dụng trước khi chạy.

* Điểm độc đáo:

Swing cho phép tạo các ứng dụng đa cửa sổ (MDI) và thực hiện kỹ thuật vẽ trên các thành phần giao diện.

Nó cung cấp khả năng tự thiết kế giao diện tùy chỉnh với sự hỗ trợ của Look and Feel (LAF).

* Thư viện bên ngoài:

Cộng đồng phát triển đã tạo ra nhiều thư viện bên ngoài (library) dựa trên Swing, giúp lập trình viên tạo ra các giao diện phức tạp một cách nhanh chóng.

* Hỗ trợ và Tài liệu:

Cộng đồng lập trình Java rộng lớn, và có nhiều tài liệu, hướng dẫn và ví dụ trực tuyến về Swing.

Swing là một phần của Java SE, do đó có sự hỗ trợ chính thức từ Oracle.

Java Swing là một công cụ mạnh mẽ cho việc phát triển ứng dụng đồ họa Java và có sức mạnh trong việc tạo ra giao diện người dùng phức tạp và đa dạng. Nó đã và đang được sử dụng rộng rãi trong việc phát triển ứng dụng máy tính và ứng dụng di động trên nhiều nền tảng.

* SQL Server:

SQL Server là một hệ quản trị cơ sở dữ liệu (DBMS) mạnh mẽ được phát triển bởi Microsoft. Đây là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) được sử dụng rộng rãi trong việc quản lý và lưu trữ dữ liệu trong các ứng dụng doanh nghiệp và phần mềm. SQL Server cung cấp một loạt các tính năng và khả năng mạnh mẽ, bao gồm quản lý cơ sở dữ liệu, truy vấn dữ liệu, bảo mật và tích hợp với các ứng dụng và dịch vụ khác của Microsoft. Dưới đây là một giới thiệu chi tiết và dài dòng về SQL Server:

* Lịch sử và Phiên bản:

SQL Server ra đời vào cuối những năm 1980 và đã trải qua nhiều phiên bản cải tiến. Phiên bản đầu tiên là SQL Server 1.0, và hiện tại là SQL Server 2019.

SQL Server hiện đang có hai phiên bản chính: SQL Server truyền thống (On-Premises) và SQL Server trong điện toán đám mây (Azure SQL Database).

* Ưu điểm của SQL Server:

Hiệu suất: SQL Server được tối ưu hóa cho hiệu suất và có khả năng xử lý tải cao, phù hợp với các ứng dụng doanh nghiệp lớn.

Bảo mật: SQL Server cung cấp các tính năng bảo mật mạnh mẽ để bảo vệ dữ liệu, bao gồm mã hóa dữ liệu, kiểm tra danh tính và quản lý quyền truy cập.

Dễ quản lý: SQL Server có giao diện quản lý đồ họa (SQL Server Management Studio) giúp quản trị viên dễ dàng quản lý cơ sở dữ liệu.

Tích hợp Azure: SQL Server tích hợp tốt với các dịch vụ điện toán đám mây của Microsoft, cho phép bạn di chuyển dữ liệu vào Azure hoặc sử dụng Azure SQL Database.

Hỗ trợ dữ liệu lớn: SQL Server hỗ trợ lưu trữ và truy vấn dữ liệu lớn (Big Data) thông qua tích hợp với Apache Hadoop.

* Các tính năng chính của SQL Server:

Ngôn ngữ truy vấn SQL: SQL Server sử dụng SQL (Structured Query Language) cho việc tạo và truy vấn cơ sở dữ liệu.

Truy vấn dữ liệu: SQL Server hỗ trợ các lệnh SELECT, INSERT, UPDATE, DELETE và JOIN để truy vấn và thay đổi dữ liệu.

Lập lịch công việc: SQL Server Agent cho phép lập lịch và tự động hóa các tác vụ quản lý cơ sở dữ liệu.

Bảo mật đa lớp: SQL Server cung cấp kiểm tra danh tính, mã hóa dữ liệu và quản lý quyền truy cập.

Khôi phục dữ liệu: Có khả năng sao lưu, phục hồi và quản lý lưu trữ dự phòng.

Replication: Hỗ trợ sao chép dữ liệu giữa các máy chủ SQL Server.

SQL Server Management Studio (SSMS):

SQL Server được quản lý và thao tác thông qua SQL Server Management Studio (SSMS), một ứng dụng đồ họa.

SSMS cho phép bạn thực hiện các nhiệm vụ quản lý như tạo cơ sở dữ liệu, thiết lập bảo mật, thực hiện truy vấn SQL, và theo dõi hiệu suất.

* Bộ dịch SQL Server và C#:

SQL Server hỗ trợ việc lập trình các ứng dụng sử dụng ngôn ngữ lập trình C# thông qua ADO.NET (ActiveX Data Objects .NET).

Điều này cho phép bạn kết nối và tương tác với cơ sở dữ liệu SQL Server từ các ứng dụng máy tính và di động.

* SQL Server và Cloud:

Microsoft cung cấp Azure SQL Database, dịch vụ quản lý cơ sở dữ liệu dựa trên SQL Server trong môi trường điện toán đám mây.

Điều này cho phép bạn triển khai và quản lý cơ sở dữ liệu SQL Server trên nền tảng đám mây của Microsoft.

* Ứng dụng của SQL Server:

SQL Server được sử dụng rộng rãi trong các ứng dụng doanh nghiệp, bao gồm hệ thống quản lý khách hàng (CRM), hệ thống quản lý nguồn nhân lực (HRM), quản lý kho hàng và các hệ thống thương mại điện tử.

* Học tập và Tài liệu:

Microsoft cung cấp nhiều tài liệu học tập và hướng dẫn về SQL Server trên trang web chính thức.

Cộng đồng lập trình cũng đã tạo ra nhiều tài liệu, video học, và tài liệu tham khảo về SQL Server.

* Kết luận:

SQL Server là một hệ quản trị cơ sở dữ liệu mạnh mẽ và phổ biến được sử dụng rộng rãi trong lĩnh vực công nghệ thông tin và doanh nghiệp.

Với nhiều tính năng mạnh mẽ và khả năng tích hợp với các dịch vụ Microsoft, SQL Server đóng một vai trò quan trọng trong việc quản lý và lưu trữ dữ liệu cho các ứng dụng và doanh nghiệp.

* JDBC:

JDBC (Java Database Connectivity) là một giao diện lập trình ứng dụng (API) trong ngôn ngữ lập trình Java, được sử dụng để kết nối và tương tác với các cơ sở dữ liệu quan hệ (RDBMS) như MySQL, Oracle, SQL Server, PostgreSQL và nhiều hệ thống quản trị cơ sở dữ liệu khác. JDBC cho phép các ứng dụng Java truy cập, truy vấn và cập nhật dữ liệu trong cơ sở dữ liệu. Dưới đây là một giới thiệu đầy đủ và dài dòng về JDBC:

* Tại sao JDBC quan trọng:

Cơ sở dữ liệu chơi một vai trò quan trọng trong hầu hết các ứng dụng doanh nghiệp và ứng dụng web.

JDBC giúp ứng dụng Java tương tác với cơ sở dữ liệu, thực hiện các thao tác như truy vấn, cập nhật, thêm, xóa dữ liệu và quản lý kết nối đến cơ sở dữ liệu.

Kiến thức cơ bản về JDBC:

Kết nối cơ sở dữ liệu: Để sử dụng JDBC, bạn cần thiết lập kết nối với cơ sở dữ liệu bằng cách cung cấp thông tin về URL kết nối, tên người dùng và mật khẩu.

Statement và PreparedStatement: JDBC cung cấp hai loại đối tượng để thực thi truy vấn SQL - Statement và PreparedStatement. PreparedStatement thường được ưa chuộng vì hiệu suất và tính bảo mật tốt hơn.

Truy vấn SQL: Bạn có thể sử dụng JDBC để thực hiện truy vấn SQL để lấy dữ liệu từ cơ sở dữ liệu.

Cập nhật dữ liệu: JDBC cũng cho phép bạn cập nhật dữ liệu trong cơ sở dữ liệu bằng cách sử dụng các truy vấn UPDATE hoặc INSERT.

Xử lý ngoại lệ: Xử lý ngoại lệ là một phần quan trọng của JDBC. Các lệnh SQL có thể gây ra các ngoại lệ, ví dụ như SQLException, và bạn cần xử lý chúng.

* Kiến thức nâng cao về JDBC:

Kết nối đa luồng: JDBC hỗ trợ kết nối đa luồng, cho phép nhiều luồng của ứng dụng truy cập cơ sở dữ liệu cùng một lúc.

Xử lý tham số: Sử dụng PreparedStatement, lập trình viên có thể dễ dàng thêm tham số vào các truy vấn SQL, giúp tránh các vấn đề bảo mật và tối ưu hóa hiệu suất.

Pooling kết nối: Khi làm việc với ứng dụng thời gian thực hoặc ứng dụng web, việc quản lý và tái sử dụng kết nối cơ sở dữ liệu bằng cơ chế pooling kết nối là quan trọng.

JDBC và Lập trình ứng dụng Java:

JDBC là một phần quan trọng của lập trình ứng dụng Java. Nó cho phép bạn xây dựng các ứng dụng quản lý dữ liệu, hệ thống quản lý cơ sở dữ liệu, và ứng dụng web động.

JDBC có thể kết hợp với các thư viện và framework khác trong hệ sinh thái Java như Spring và Hibernate.

* Hỗ trợ cơ sở dữ liệu:

JDBC hỗ trợ nhiều cơ sở dữ liệu phổ biến, bao gồm Oracle, MySQL, SQL Server, PostgreSQL, và nhiều hệ thống quản trị cơ sở dữ liệu khác.

Mỗi cơ sở dữ liệu có một trình điều khiển (driver) JDBC riêng, được sử dụng để thiết lập kết nối đến cơ sở dữ liệu cụ thể.

* Tài liệu và Học tập:

Cộng đồng Java cung cấp nhiều tài liệu học tập, ví dụ, và mã nguồn mẫu về JDBC.

Trang web chính thức của Oracle cũng chứa tài liệu và ví dụ về JDBC.

* Kết luận:

JDBC là một công cụ quan trọng cho các nhà phát triển Java để tương tác với cơ sở dữ liệu. Nó mở ra cánh cửa cho việc lập trình ứng dụng liên quan đến cơ sở dữ liệu và là một phần quan trọng của sự phát triển của ứng dụng Java trong thực tế.

* 1. Kết chương 1

Chương 1 đã giới thiệu về đề tài "Phần mềm Quản lý Quán Cafe" và các kế hoạch thực hiện dự án. Đã trình bày các lý do tại sao ứng dụng này quan trọng trong việc nâng cao hiệu suất quản lý quán cafe, cung cấp trải nghiệm tốt hơn cho khách hàng và giúp quán cafe phát triển và thành công. Đồng thời, chúng ta cũng đã xem xét kế hoạch làm dự án, bao gồm việc xác định yêu cầu, thiết kế, phát triển, kiểm thử, và sửa lỗi.

Trong chương này, cũng đã giới thiệu một số công nghệ quan trọng được sử dụng trong dự án, bao gồm Java Swing, SQL Server và JDBC. Java Swing được sử dụng để xây dựng giao diện người dùng đồ họa cho ứng dụng, SQL Server để lưu trữ dữ liệu quán cafe, và JDBC để kết nối và tương tác với cơ sở dữ liệu.

Chương 1 đã tạo nền tảng cho việc phát triển và triển khai "Phần mềm Quản lý Quán Cafe". Trong các chương tiếp theo, chúng ta sẽ đi sâu hơn vào việc thực hiện các giai đoạn được đề cập trong kế hoạch và xây dựng ứng dụng này một cách chi tiết.

# Chương 2. Mô hình hệ thống

## 2.1. Mô hình chức năng

A diagram of a company

Description automatically generated

## 2.2. Thiết kế giao diện

Giao diện Tài khoản:

Mô tả đối tượng tài khoản

+ Các thuộc tính: tenDangNhap, tenHienThi, matKhau, type

+ Các phương thức: thêm, sửa, xóa, lấy danh sách, tìm theo tên đăng nhập

Giao diện bàn:

Mô tả đối tượng bàn

+ Các thuộc tính: maBan, tenBan, trangThai

+ Các phương thức: thêm, sửa, xóa, lấy danh sách, tìm theo tên bàn, tìm theo mã bàn

Giao diện loại thức uống:

Mô tả đối tượng loại thức uống

+ Các thuộc tính: maLoai, tenLoai

+ Các phương thức: thêm, sửa, xóa, lấy danh sách, tìm theo tên loại, tìm theo mã loại

Giao diện thức uống:

Mô tả đối tượng thức uống

+ Các thuộc tính: maThucUong, tenThucUong, maLoai, gia

+ Các phương thức: thêm, sửa, xóa, lấy danh sách, tìm theo loại, tìm theo tên

Giao diện hóa đơn:

Mô tả đối tượng hóa đơn

+ Các thuộc tính: maHoaDon, thoiGianVao, thoiGianRa, maBan, tinhTrang

+ Các phương thức: thêm, sửa, xóa, lấy danh sách, tìm theo mã

## 2.3. Mô hình kiến trúc hệ thống

A diagram of a network

Description automatically generated

## 2.4. Mô hình cơ sở dữ liệu

A screenshot of a computer

Description automatically generated

Chèn dữ liệu:

-- Chèn dữ liệu cho bảng "Ban"

INSERT INTO Ban (TenBan, TrangThai)

VALUES (N'Bàn 1', N'Trống'),

(N'Bàn 2', N'Trống'),

(N'Bàn 3', N'Có người'),

(N'Bàn 4', N'Trống'),

(N'Bàn 5', N'Có người');

-- Chèn dữ liệu cho bảng "TaiKhoan"

INSERT INTO TaiKhoan (TenDangNhap, TenHienThi, MatKhau, Type)

VALUES (N'admin', N'Quản trị viên', N'123', 1),

(N'nhanvien', N'Nhân viên', N'123', 0);

-- Chèn dữ liệu cho bảng "Loai"

INSERT INTO Loai (TenLoai)

VALUES (N'Trà sữa'),

(N'Cà phê'),

(N'Sinh tố'),

(N'Nước uống lạnh');

-- Chèn dữ liệu cho bảng "ThucUong"

INSERT INTO ThucUong (TenThucUong, MaLoai, Gia)

VALUES (N'Capuchino', 2, 35000),

(N'Trà đào', 1, 25000),

(N'Espresso', 2, 32000),

(N'Sinh tố dâu', 3, 30000),

(N'Caramel Macchiato', 2, 38000),

(N'Trà bí đao', 1, 27000),

(N'Cà phê sữa đá', 2, 30000),

(N'Sinh tố bơ', 3, 29000),

(N'Latte', 2, 36000),

(N'Sinh tố xoài', 3, 28000);

## 2.5. Giao diện người dùng

Giao diện đăng nhập:

A screenshot of a computer screen

Description automatically generated

Giao diện tài khoản:

A screenshot of a computer

Description automatically generated

Giao diện bàn:

A screenshot of a computer

Description automatically generated

Giao diện loại thức uống:

A screenshot of a computer

Description automatically generated

Giao diện thức uống:

A screenshot of a computer

Description automatically generated

Giao diện bán hàng:

A screenshot of a computer

Description automatically generated

Giao diện lịch sử hóa đơn:

A screenshot of a computer

Description automatically generated

Giao diện thống kê:

A screenshot of a computer

Description automatically generated

## 2.6 Các tiện ích

Class method:

public static double tinhTongTienHoaDon(int maHoaDon);

public static List<ChiTietHoaDonDTO> layChiTietHoaDonChuaThanhToanCuaBan(int maBan)

public static HoaDonDTO layHoaDonChuaThanhToanCuaBan(int maBan);

public static TaiKhoanDTO dangNhap(String tenDangNhap, String matKhau);

public static boolean xoaTaiKhoan(String tenDangNhap);

2.7. Kết chương 2

Chương 2 của báo cáo tập trung vào việc mô tả mô hình hệ thống và thiết kế giao diện của dự án:

Mô hình chức năng: Chúng ta đã xác định và mô tả các đối tượng quan trọng của hệ thống cùng với các thuộc tính và chức năng tương ứng của từng đối tượng. Điều này bao gồm tài khoản, bàn, loại thức uống, thức uống, và hóa đơn.

Thiết kế giao diện: Chúng ta đã mô tả cách các giao diện người dùng được thiết kế. Các giao diện bao gồm giao diện dăng nhập, giao diện tài khoản, giao diện bàn, giao diện loại thức uống, giao diện thức uống, giao diện bán hàng, giao diện lịch sử hóa đơn, và giao diện thống kê. Mô tả chi tiết của mỗi giao diện giúp hiểu cách người dùng tương tác với hệ thống.

Mô hình kiến trúc hệ thống: Chúng ta đã giới thiệu mô hình kiến trúc tổng quan của hệ thống. Điều này bao gồm các thành phần cốt lõi, cơ sở dữ liệu, và cách chúng tương tác với nhau.

Mô hình cơ sở dữ liệu: Chúng ta đã cung cấp lệnh SQL để chèn dữ liệu ban đầu vào các bảng trong cơ sở dữ liệu. Dữ liệu ban đầu này được sử dụng để khởi tạo cơ sở dữ liệu và cung cấp dữ liệu ví dụ cho phát triển và kiểm thử hệ thống.

Các tiện ích như: tính tổng tiền của hóa đơn, lấy chi tiết hóa đơn chưa thanh toán của bàn, lấy hóa đơn chưa thanh toán, đăng nhập, xóa tài khoản.

# Chương 3. Phát triển hệ thống

## 3.1. Phát triển các giao diện

Giao diện 1:

|  |
| --- |
| public interface IBanDAL {  boolean themBan(BanDTO ban);  boolean suaBan(BanDTO ban);  boolean xoaBan(int maBan);  List<BanDTO> layDanhSachBan();  boolean capNhatTrangThai(int maBan, String trangThai);  BanDTO timBanTheoMa(int maBan);  BanDTO timBanTheoTen(String tenBan);  HoaDonDTO timHoaDonHienTai(int maBan);  List<ChiTietHoaDonDTO> layChiTietHoaDonChuaThanhToanCuaBan(int maBan);  HoaDonDTO layHoaDonChuaThanhToanCuaBan(int maBan);  } |

Giao diện 2:

|  |
| --- |
| public interface IChiTietHoaDonDAL {  boolean themChiTietHoaDon(ChiTietHoaDonDTO chiTietHoaDon);  boolean suaChiTietHoaDon(ChiTietHoaDonDTO chiTietHoaDon);  boolean xoaChiTietHoaDon(int id);  List<ChiTietHoaDonDTO> layDanhSachChiTietHoaDonTheoMaHoaDon(int maHoaDon);  } |

Giao diện 3:

|  |
| --- |
| public interface IHoaDonDAL {  int themHoaDon(HoaDonDTO hoaDon);  boolean suaHoaDon(HoaDonDTO hoaDon);  boolean xoaHoaDon(int maHoaDon);  boolean capNhatTrangThai(int maHoaDon, boolean trangThai);  List<HoaDonDTO> layDanhSachHoaDon();  double tinhTongTienHoaDon(int maHoaDon);  List<OrderStatistics> thongKeHoaDonTheoTenThucUong(String tenThucUong);  } |

Giao diện 4:

|  |
| --- |
| public interface ILoaiDAL {  boolean themLoai(LoaiDTO loai);  boolean suaLoai(LoaiDTO loai);  boolean xoaLoai(int maLoai);  List<LoaiDTO> layDanhSachLoai();  LoaiDTO timLoaiTheoMa(int maLoai);  LoaiDTO timLoaiTheoTen(String tenLoai);  } |

Giao diện 5:

|  |
| --- |
| public interface ITaiKhoanDAL {  boolean themTaiKhoan(TaiKhoanDTO taiKhoan);  boolean suaTaiKhoan(TaiKhoanDTO taiKhoan);  boolean xoaTaiKhoan(String tenDangNhap);  List<TaiKhoanDTO> layDanhSachTaiKhoan();  TaiKhoanDTO timTaiKhoanTheoTenDangNhap(String tenDangNhap);  TaiKhoanDTO dangNhap(String tenDangNhap, String matKhau);  } |

Giao diện 6:

|  |
| --- |
| public interface IThucUongDAL {  boolean themThucUong(ThucUongDTO thucUong);  boolean suaThucUong(ThucUongDTO thucUong);  boolean xoaThucUong(int maThucUong);  List<ThucUongDTO> layDanhSachThucUong();  ThucUongDTO timThucUongTheoMa(int maThucUong);  ThucUongDTO timThucUongTheoTen(String tenThucUong);  } |

## 3.2. Phát triển các lớp, triển khai các giao diện

Class BanDAL:

|  |
| --- |
| package DAL;  import IDAL.IBanDAL;  import DTO.BanDTO;  import DTO.ChiTietHoaDonDTO;  import DTO.HoaDonDTO;  import Utils.JdbcUltils;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class BanDAL implements IBanDAL {  @Override  public boolean themBan(BanDTO ban) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "INSERT INTO Ban (TenBan) VALUES (?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, ban.getTenBan());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean suaBan(BanDTO ban) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE Ban SET TenBan = ? WHERE MaBan = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, ban.getTenBan());  preparedStatement.setInt(2, ban.getMaBan());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaBan(int maBan) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM Ban WHERE MaBan = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maBan);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<BanDTO> layDanhSachBan() {  List<BanDTO> danhSachBan = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachBan;  }  String query = "SELECT \* FROM Ban";  try (PreparedStatement preparedStatement = connection.prepareStatement(query);  ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  BanDTO ban = new BanDTO();  ban.setMaBan(resultSet.getInt("MaBan"));  ban.setTenBan(resultSet.getString("TenBan"));  ban.setTrangThai(resultSet.getString("TrangThai"));  danhSachBan.add(ban);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachBan;  }  @Override  public BanDTO timBanTheoMa(int maBan) {  BanDTO ban = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return ban;  }  String query = "SELECT \* FROM Ban WHERE MaBan = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maBan);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  ban = new BanDTO();  ban.setMaBan(resultSet.getInt("MaBan"));  ban.setTenBan(resultSet.getString("TenBan"));  ban.setTrangThai(resultSet.getString("TrangThai"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return ban;  }  @Override  public BanDTO timBanTheoTen(String tenBan) {  BanDTO ban = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return ban;  }  String query = "SELECT \* FROM Ban WHERE TenBan = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenBan);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  ban = new BanDTO();  ban.setMaBan(resultSet.getInt("MaBan"));  ban.setTenBan(resultSet.getString("TenBan"));  ban.setTrangThai(resultSet.getString("TrangThai"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return ban;  }  @Override  public HoaDonDTO timHoaDonHienTai(int maBan) {  HoaDonDTO hoaDon = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return hoaDon;  }  String query = "SELECT \* FROM HoaDon WHERE MaBan = ? AND TrangThai = 0";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maBan);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  hoaDon = new HoaDonDTO();  hoaDon.setMaHoaDon(resultSet.getInt("MaHoaDon"));  hoaDon.setThoiGianVao(resultSet.getDate("ThoiGianVao"));  hoaDon.setThoiGianRa(resultSet.getDate("ThoiGianRa"));  hoaDon.setMaBan(resultSet.getInt("MaBan"));  hoaDon.setTrangThai(resultSet.getBoolean("TrangThai"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return hoaDon;  }  @Override  public boolean capNhatTrangThai(int maBan, String trangThai) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE Ban SET TrangThai = ? WHERE MaBan = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, trangThai);  preparedStatement.setInt(2, maBan);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<ChiTietHoaDonDTO> layChiTietHoaDonChuaThanhToanCuaBan(int maBan) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return new ArrayList<>();  }  String query = "SELECT HoaDon.MaHoaDon, ChiTietHoaDon.id, ChiTietHoaDon.MaThucUong, ChiTietHoaDon.SoLuong " +  "FROM HoaDon " +  "INNER JOIN ChiTietHoaDon ON HoaDon.MaHoaDon = ChiTietHoaDon.MaHoaDon " +  "INNER JOIN ThucUong ON ChiTietHoaDon.MaThucUong = ThucUong.MaThucUong " +  "WHERE HoaDon.MaBan = ? AND HoaDon.TrangThai = 0";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maBan);  ResultSet resultSet = preparedStatement.executeQuery();  List<ChiTietHoaDonDTO> chiTietHoaDonList = new ArrayList<>();  while (resultSet.next()) {  ChiTietHoaDonDTO chiTietHoaDon = new ChiTietHoaDonDTO();  chiTietHoaDon.setId(resultSet.getInt("id"));  chiTietHoaDon.setMaHoaDon(resultSet.getInt("MaHoaDon"));  chiTietHoaDon.setMaThucUong(resultSet.getInt("MaThucUong"));  chiTietHoaDon.setSoLuong(resultSet.getInt("SoLuong"));  chiTietHoaDonList.add(chiTietHoaDon);  }  return chiTietHoaDonList;  } catch (SQLException e) {  e.printStackTrace();  return new ArrayList<>();  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public HoaDonDTO layHoaDonChuaThanhToanCuaBan(int maBan) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return null;  }  String query = "SELECT \* FROM HoaDon WHERE MaBan = ? AND TrangThai = 0 ORDER BY ThoiGianVao DESC";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maBan);  ResultSet resultSet = preparedStatement.executeQuery();  if (resultSet.next()) {  HoaDonDTO hoaDon = new HoaDonDTO();  hoaDon.setMaHoaDon(resultSet.getInt("MaHoaDon"));  hoaDon.setThoiGianVao(resultSet.getDate("ThoiGianVao"));  hoaDon.setThoiGianRa(resultSet.getDate("ThoiGianRa"));  hoaDon.setMaBan(resultSet.getInt("MaBan"));  hoaDon.setTrangThai(resultSet.getBoolean("TrangThai"));  return hoaDon;  } else {  return null; // Không tìm thấy hóa đơn chưa thanh toán cho bàn này  }  } catch (SQLException e) {  e.printStackTrace();  return null;  } finally {  JdbcUltils.getClose(connection);  }  }  } |

Class ChiTietHoaDonDAL:

|  |
| --- |
| package DAL;  import IDAL.IChiTietHoaDonDAL;  import DTO.ChiTietHoaDonDTO;  import Utils.JdbcUltils;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class ChiTietHoaDonDAL implements IChiTietHoaDonDAL {  @Override  public boolean themChiTietHoaDon(ChiTietHoaDonDTO chiTietHoaDon) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "INSERT INTO ChiTietHoaDon (MaHoaDon, MaThucUong, SoLuong) VALUES (?, ?, ?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, chiTietHoaDon.getMaHoaDon());  preparedStatement.setInt(2, chiTietHoaDon.getMaThucUong());  preparedStatement.setInt(3, chiTietHoaDon.getSoLuong());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean suaChiTietHoaDon(ChiTietHoaDonDTO chiTietHoaDon) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE ChiTietHoaDon SET SoLuong = ? WHERE MaHoaDon = ? AND MaThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, chiTietHoaDon.getSoLuong());  preparedStatement.setInt(2, chiTietHoaDon.getMaHoaDon());  preparedStatement.setInt(3, chiTietHoaDon.getMaThucUong());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaChiTietHoaDon(int id) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM ChiTietHoaDon WHERE id = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, id);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<ChiTietHoaDonDTO> layDanhSachChiTietHoaDonTheoMaHoaDon(int maHoaDon) {  List<ChiTietHoaDonDTO> danhSachChiTietHoaDon = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachChiTietHoaDon;  }  String query = "SELECT \* FROM ChiTietHoaDon WHERE MaHoaDon = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maHoaDon);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  ChiTietHoaDonDTO chiTietHoaDon = new ChiTietHoaDonDTO();  chiTietHoaDon.setId(resultSet.getInt("id"));  chiTietHoaDon.setMaHoaDon(resultSet.getInt("MaHoaDon"));  chiTietHoaDon.setMaThucUong(resultSet.getInt("MaThucUong"));  chiTietHoaDon.setSoLuong(resultSet.getInt("SoLuong"));  danhSachChiTietHoaDon.add(chiTietHoaDon);  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachChiTietHoaDon;  }  } |

Class HoaDonDAL:

|  |
| --- |
| package DAL;  import IDAL.IHoaDonDAL;  import DTO.HoaDonDTO;  import DTO.OrderStatistics;  import Utils.JdbcUltils;  import java.sql.Statement;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Date;  import java.util.ArrayList;  import java.util.List;  public class HoaDonDAL implements IHoaDonDAL {  @Override  public int themHoaDon(HoaDonDTO hoaDon) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return -1; // Trả về -1 nếu không thể kết nối hoặc thêm hóa đơn không thành công.  }  String query = "INSERT INTO HoaDon (MaBan) VALUES (?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {  preparedStatement.setInt(1, hoaDon.getMaBan());  int rowsAffected = preparedStatement.executeUpdate();  if (rowsAffected > 0) {  // Lấy mã hóa đơn vừa thêm  ResultSet generatedKeys = preparedStatement.getGeneratedKeys();  if (generatedKeys.next()) {  return generatedKeys.getInt(1);  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return -1; // Trả về -1 nếu thêm hóa đơn không thành công.  }  @Override  public boolean suaHoaDon(HoaDonDTO hoaDon) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE HoaDon SET ThoiGianRa = GETDATE(), TrangThai = 1 WHERE MaHoaDon = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, hoaDon.getMaHoaDon());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaHoaDon(int maHoaDon) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM HoaDon WHERE MaHoaDon = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maHoaDon);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean capNhatTrangThai(int maHoaDon, boolean trangThai) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE HoaDon SET TrangThai = ? WHERE MaHoaDon = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setBoolean(1, trangThai);  preparedStatement.setInt(2, maHoaDon);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<HoaDonDTO> layDanhSachHoaDon() {  List<HoaDonDTO> danhSachHoaDon = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachHoaDon;  }  String query = "SELECT \* FROM HoaDon";  try (PreparedStatement preparedStatement = connection.prepareStatement(query); ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  HoaDonDTO hoaDon = new HoaDonDTO();  hoaDon.setMaHoaDon(resultSet.getInt("MaHoaDon"));  hoaDon.setThoiGianVao(resultSet.getDate("ThoiGianVao"));  hoaDon.setThoiGianRa(resultSet.getDate("ThoiGianRa"));  hoaDon.setMaBan(resultSet.getInt("MaBan"));  hoaDon.setTrangThai(resultSet.getBoolean("TrangThai"));  danhSachHoaDon.add(hoaDon);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachHoaDon;  }  @Override  public double tinhTongTienHoaDon(int maHoaDon) {  double tongTien = 0;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return tongTien;  }  String query = "SELECT SUM(ThucUong.Gia \* ChiTietHoaDon.SoLuong) AS TongTien "  + "FROM HoaDon "  + "INNER JOIN ChiTietHoaDon ON HoaDon.MaHoaDon = ChiTietHoaDon.MaHoaDon "  + "INNER JOIN ThucUong ON ChiTietHoaDon.MaThucUong = ThucUong.MaThucUong "  + "WHERE HoaDon.MaHoaDon = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maHoaDon);  ResultSet resultSet = preparedStatement.executeQuery();  if (resultSet.next()) {  tongTien = resultSet.getDouble("TongTien");  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return tongTien;  }  public List<OrderStatistics> thongKeHoaDonTheoTenThucUong(String tenThucUong) {  List<OrderStatistics> statistics = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return statistics;  }  String query = "SELECT HoaDon.MaHoaDon, HoaDon.ThoiGianVao, HoaDon.ThoiGianRa, HoaDon.MaBan, "  + "ThucUong.TenThucUong, ChiTietHoaDon.SoLuong, ThucUong.Gia, "  + "(ChiTietHoaDon.SoLuong \* ThucUong.Gia) AS TotalAmount "  + "FROM HoaDon "  + "INNER JOIN ChiTietHoaDon ON HoaDon.MaHoaDon = ChiTietHoaDon.MaHoaDon "  + "INNER JOIN ThucUong ON ChiTietHoaDon.MaThucUong = ThucUong.MaThucUong "  + "WHERE ThucUong.TenThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenThucUong);  ResultSet resultSet = preparedStatement.executeQuery();  while (resultSet.next()) {  OrderStatistics stat = new OrderStatistics();  stat.setOrderID(resultSet.getInt("MaHoaDon"));  stat.setEntryTime(resultSet.getDate("ThoiGianVao"));  stat.setExitTime(resultSet.getDate("ThoiGianRa"));  stat.setTableNumber(resultSet.getInt("MaBan"));  stat.setDrinkName(resultSet.getString("TenThucUong"));  stat.setQuantity(resultSet.getInt("SoLuong"));  stat.setUnitPrice(resultSet.getDouble("Gia"));  stat.setTotalAmount(resultSet.getDouble("TotalAmount"));  statistics.add(stat);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return statistics;  }  } |

Class LoaiDAL:

|  |
| --- |
| package DAL;  import IDAL.ILoaiDAL;  import DTO.LoaiDTO;  import Utils.JdbcUltils;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class LoaiDAL implements ILoaiDAL {  @Override  public boolean themLoai(LoaiDTO loai) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "INSERT INTO Loai (TenLoai) VALUES (?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, loai.getTenLoai());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean suaLoai(LoaiDTO loai) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE Loai SET TenLoai = ? WHERE MaLoai = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, loai.getTenLoai());  preparedStatement.setInt(2, loai.getMaLoai());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaLoai(int maLoai) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM Loai WHERE MaLoai = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maLoai);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<LoaiDTO> layDanhSachLoai() {  List<LoaiDTO> danhSachLoai = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachLoai;  }  String query = "SELECT \* FROM Loai";  try (PreparedStatement preparedStatement = connection.prepareStatement(query);  ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  LoaiDTO loai = new LoaiDTO();  loai.setMaLoai(resultSet.getInt("MaLoai"));  loai.setTenLoai(resultSet.getString("TenLoai"));  danhSachLoai.add(loai);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachLoai;  }  @Override  public LoaiDTO timLoaiTheoMa(int maLoai) {  LoaiDTO loai = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return loai;  }  String query = "SELECT \* FROM Loai WHERE MaLoai = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maLoai);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  loai = new LoaiDTO();  loai.setMaLoai(resultSet.getInt("MaLoai"));  loai.setTenLoai(resultSet.getString("TenLoai"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return loai;  }  @Override  public LoaiDTO timLoaiTheoTen(String tenLoai) {  LoaiDTO loai = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return loai;  }  String query = "SELECT \* FROM Loai WHERE TenLoai = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenLoai);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  loai = new LoaiDTO();  loai.setMaLoai(resultSet.getInt("MaLoai"));  loai.setTenLoai(resultSet.getString("TenLoai"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return loai;  }  } |

Class TaiKhoanDAL:

|  |
| --- |
| package DAL;  import IDAL.ITaiKhoanDAL;  import DTO.TaiKhoanDTO;  import Utils.JdbcUltils;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class TaiKhoanDAL implements ITaiKhoanDAL {  @Override  public boolean themTaiKhoan(TaiKhoanDTO taiKhoan) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "INSERT INTO TaiKhoan (TenDangNhap, TenHienThi, MatKhau, Type) VALUES (?, ?, ?, ?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, taiKhoan.getTenDangNhap());  preparedStatement.setString(2, taiKhoan.getTenHienThi());  preparedStatement.setString(3, taiKhoan.getMatKhau());  preparedStatement.setInt(4, taiKhoan.getType());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean suaTaiKhoan(TaiKhoanDTO taiKhoan) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE TaiKhoan SET TenHienThi = ?, MatKhau = ?, Type = ? WHERE TenDangNhap = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, taiKhoan.getTenHienThi());  preparedStatement.setString(2, taiKhoan.getMatKhau());  preparedStatement.setInt(3, taiKhoan.getType());  preparedStatement.setString(4, taiKhoan.getTenDangNhap());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaTaiKhoan(String tenDangNhap) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM TaiKhoan WHERE TenDangNhap = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenDangNhap);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<TaiKhoanDTO> layDanhSachTaiKhoan() {  List<TaiKhoanDTO> danhSachTaiKhoan = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachTaiKhoan;  }  String query = "SELECT \* FROM TaiKhoan";  try (PreparedStatement preparedStatement = connection.prepareStatement(query);  ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  TaiKhoanDTO taiKhoan = new TaiKhoanDTO();  taiKhoan.setTenDangNhap(resultSet.getString("TenDangNhap"));  taiKhoan.setTenHienThi(resultSet.getString("TenHienThi"));  taiKhoan.setMatKhau(resultSet.getString("MatKhau"));  taiKhoan.setType(resultSet.getInt("Type"));  danhSachTaiKhoan.add(taiKhoan);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachTaiKhoan;  }  @Override  public TaiKhoanDTO timTaiKhoanTheoTenDangNhap(String tenDangNhap) {  TaiKhoanDTO taiKhoan = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return taiKhoan;  }  String query = "SELECT \* FROM TaiKhoan WHERE TenDangNhap = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenDangNhap);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  taiKhoan = new TaiKhoanDTO();  taiKhoan.setTenDangNhap(resultSet.getString("TenDangNhap"));  taiKhoan.setTenHienThi(resultSet.getString("TenHienThi"));  taiKhoan.setMatKhau(resultSet.getString("MatKhau"));  taiKhoan.setType(resultSet.getInt("Type"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return taiKhoan;  }  @Override  public TaiKhoanDTO dangNhap(String tenDangNhap, String matKhau) {  TaiKhoanDTO taiKhoan = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return taiKhoan;  }  String query = "SELECT \* FROM TaiKhoan WHERE TenDangNhap = ? AND MatKhau = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenDangNhap);  preparedStatement.setString(2, matKhau);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  taiKhoan = new TaiKhoanDTO();  taiKhoan.setTenDangNhap(resultSet.getString("TenDangNhap"));  taiKhoan.setTenHienThi(resultSet.getString("TenHienThi"));  taiKhoan.setMatKhau(resultSet.getString("MatKhau"));  taiKhoan.setType(resultSet.getInt("Type"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return taiKhoan;  }  } |

Class ThucUongDAL:

|  |
| --- |
| package DAL;  import IDAL.IThucUongDAL;  import DTO.ThucUongDTO;  import Utils.JdbcUltils;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  public class ThucUongDAL implements IThucUongDAL {  @Override  public boolean themThucUong(ThucUongDTO thucUong) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "INSERT INTO ThucUong (TenThucUong, MaLoai, Gia) VALUES (?, ?, ?)";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, thucUong.getTenThucUong());  preparedStatement.setInt(2, thucUong.getMaLoai());  preparedStatement.setLong(3, thucUong.getGia());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean suaThucUong(ThucUongDTO thucUong) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "UPDATE ThucUong SET TenThucUong = ?, MaLoai = ?, Gia = ? WHERE MaThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, thucUong.getTenThucUong());  preparedStatement.setInt(2, thucUong.getMaLoai());  preparedStatement.setLong(3, thucUong.getGia());  preparedStatement.setInt(4, thucUong.getMaThucUong());  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public boolean xoaThucUong(int maThucUong) {  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return false;  }  String query = "DELETE FROM ThucUong WHERE MaThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maThucUong);  int rowsAffected = preparedStatement.executeUpdate();  return rowsAffected > 0;  } catch (SQLException e) {  e.printStackTrace();  return false;  } finally {  JdbcUltils.getClose(connection);  }  }  @Override  public List<ThucUongDTO> layDanhSachThucUong() {  List<ThucUongDTO> danhSachThucUong = new ArrayList<>();  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return danhSachThucUong;  }  String query = "SELECT \* FROM ThucUong";  try (PreparedStatement preparedStatement = connection.prepareStatement(query);  ResultSet resultSet = preparedStatement.executeQuery()) {  while (resultSet.next()) {  ThucUongDTO thucUong = new ThucUongDTO();  thucUong.setMaThucUong(resultSet.getInt("MaThucUong"));  thucUong.setTenThucUong(resultSet.getString("TenThucUong"));  thucUong.setMaLoai(resultSet.getInt("MaLoai"));  thucUong.setGia(resultSet.getLong("Gia"));  danhSachThucUong.add(thucUong);  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return danhSachThucUong;  }  @Override  public ThucUongDTO timThucUongTheoMa(int maThucUong) {  ThucUongDTO thucUong = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return thucUong;  }  String query = "SELECT \* FROM ThucUong WHERE MaThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setInt(1, maThucUong);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  thucUong = new ThucUongDTO();  thucUong.setMaThucUong(resultSet.getInt("MaThucUong"));  thucUong.setTenThucUong(resultSet.getString("TenThucUong"));  thucUong.setMaLoai(resultSet.getInt("MaLoai"));  thucUong.setGia(resultSet.getLong("Gia"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return thucUong;  }  @Override  public ThucUongDTO timThucUongTheoTen(String tenThucUong) {  ThucUongDTO thucUong = null;  Connection connection = JdbcUltils.getConnect();  if (connection == null) {  return thucUong;  }  String query = "SELECT \* FROM ThucUong WHERE TenThucUong = ?";  try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  preparedStatement.setString(1, tenThucUong);  try (ResultSet resultSet = preparedStatement.executeQuery()) {  if (resultSet.next()) {  thucUong = new ThucUongDTO();  thucUong.setMaThucUong(resultSet.getInt("MaThucUong"));  thucUong.setTenThucUong(resultSet.getString("TenThucUong"));  thucUong.setMaLoai(resultSet.getInt("MaLoai"));  thucUong.setGia(resultSet.getLong("Gia"));  }  }  } catch (SQLException e) {  e.printStackTrace();  } finally {  JdbcUltils.getClose(connection);  }  return thucUong;  }  } |

## 3.3. Phát triển giao diện người dùng

Giao diện bàn:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách bàn |
| 2 | txtMaBan | JTextField | Hiển thị mã bàn |
| 3 | txtTenBan | JTextField | Nhập và hiển thị tên bàn |
| 4 | txtTrangThai | JTextField | Hiển thị trạng thái |
| 5 | btnThem | JButton | Thêm bàn |
| 6 | btnSua | JButton | Sửa bàn |
| 7 | btnXoa | JButton | Xóa bàn |
| 8 | btnClear | JButton | Xóa thông tin trên các component |

Giao chi tiết hóa đơn:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách hóa đơn |
| 2 | jTable2 | JTable | Hiển thị danh sách chi tiết hóa đơn |
| 3 | btnInHoaDon | JButton | In hóa đơn ra excel |

Giao diện đăng nhập:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | txtTenDangNhap | JTextField | Nhập tên đăng nhập |
| 2 | txtMatKhau | JTextField | Nhập mật khẩu |
| 3 | btnDangNhap | JButton | Đăng nhập |
| 4 | btnThoat | JButton | Thoát |

Giao diện hóa đơn:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jPanel1 | JPanel | Hiển thị danh sách bàn dưới dạng button |
| 2 | jTable1 | JTable | Hiển thị chi tiết các thức uống |
| 3 | txtBanHienTai | JTextField | Hiển thị bàn hiện tại đang chọn |
| 4 | cboThucUong | JComboBox | Lựu chọn thức uống |
| 5 | txtGia | JTextField | Hiển thị giá thức uống tương ứng |
| 6 | txtSoLuong | JTextField | Nhập số lượng |
| 7 | txtThanhTien | JTextField | Hiển thị thành tiền |
| 8 | btnThem | JButton | Thêm món ăn vào chi tiết hóa đơn của bàn |
| 9 | btnXoa | JButton | Xóa hơn đơn khỏi chi tiết hóa đơn của bàn |

Giao diện loại:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách loại thức uống |
| 2 | txtMaLoai | JTextField | Hiển thị mã loại |
| 3 | txtTenLoai | JTextField | Nhập và hiển thị tên loại |
| 5 | btnThem | JButton | Thêm |
| 6 | btnSua | JButton | Sửa |
| 7 | btnXoa | JButton | Xóa |
| 8 | btnClear | JButton | Xóa thông tin trên các component |

Giao diện tài khoản:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách tài khoản |
| 2 | txtTenDangNhap | JTextField | Nhập và hiển thị tên đăng nhập |
| 3 | txtTenHienThi | JTextField | Nhập và hiển thị tên hiển thị |
| 5 | txtMatKhau | JButton | Nhập và hiển thị mật khẩu |
| 6 | btnThem | JButton | Thêm |
| 7 | btnSua | JButton | Sửa |
| 8 | btnXoa | JButton | Xóa |
| 9 | btnClear | JButton | Xóa thông tin trên các component |

Giao diện thống kê:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách hóa đơn theo thức uống |
| 2 | cboTenThucUong | JComboBox | Chọn tên thức uống cần thống kê |
| 3 | labelSoHoaDon | JLabel | HIển thị số hóa đơn |
| 4 | labelTongTien | JLabel | Hiển thị tổng tiền |

Giao diện thức uống:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên control** | **Class** | **Chức năng** |
| 1 | jTable1 | JTable | Hiển thị danh sách thức uống |
| 2 | txtMaThucUong | JTextField | Hiển thị mã thức uống |
| 3 | txtTenThucUong | JtextField | Nhập và hiển thị tên thức uống |
| 4 | cboLoai | JcomboBox | Chọn loại thức uống |
| 5 | txtGia | JtextField | Nhập và hiển thị giá |
| 6 | btnThem | JButton | Thêm |
| 7 | btnSua | JButton | Sửa |
| 8 | btnXoa | JButton | Xóa |
| 9 | btnClear | JButton | Xóa thông tin trên các component |

## 3.4. Triển khai cơ sở dữ liệu

Tạo cơ sở dữ liệu:

|  |
| --- |
| CREATE DATABASE QuanLyQuanCafe  GO  USE QuanLyQuanCafe  GO  CREATE TABLE Ban  (  MaBan INT IDENTITY(1,1) PRIMARY KEY,  TenBan NVARCHAR(100) NOT NULL UNIQUE,  TrangThai NVARCHAR(100) NOT NULL DEFAULT N'Trống'  )  GO  CREATE TABLE TaiKhoan  (  TenDangNhap NVARCHAR(100) PRIMARY KEY,  TenHienThi NVARCHAR(100) NOT NULL DEFAULT N'Người dùng',  MatKhau NVARCHAR(100) NOT NULL DEFAULT 0,  Type INT NOT NULL DEFAULT 0 -- 1: admin && 0: staff  )  GO  CREATE TABLE Loai  (  MaLoai INT IDENTITY(1,1) PRIMARY KEY,  TenLoai NVARCHAR(100) NOT NULL UNIQUE  )  GO  CREATE TABLE ThucUong  (  MaThucUong INT IDENTITY(1,1) PRIMARY KEY,  TenThucUong NVARCHAR(100) NOT NULL DEFAULT N'Chưa đặt tên',  MaLoai INT NOT NULL,  Gia bigint NOT NULL DEFAULT 0,  FOREIGN KEY (MaLoai) REFERENCES Loai(MaLoai) on delete cascade  )  GO  CREATE TABLE HoaDon  (  MaHoaDon INT IDENTITY(1,1) PRIMARY KEY,  ThoiGianVao DATE NOT NULL DEFAULT GETDATE(),  ThoiGianRa DATE,  MaBan INT NOT NULL,  TrangThai BIT NOT NULL DEFAULT 0, -- 1: đã thanh toán && 0: chưa thanh toán  FOREIGN KEY (MaBan) REFERENCES Ban(MaBan) on delete cascade  )  GO  CREATE TABLE ChiTietHoaDon  (  id INT IDENTITY PRIMARY KEY,  MaHoaDon INT NOT NULL,  MaThucUong INT NOT NULL,  SoLuong INT NOT NULL DEFAULT 0,  FOREIGN KEY (MaHoaDon) REFERENCES HoaDon(MaHoaDon) on delete cascade,  FOREIGN KEY (MaThucUong) REFERENCES ThucUong(MaThucUong) on delete cascade  )  GO  -- Chèn dữ liệu cho bảng "Ban"  INSERT INTO Ban (TenBan, TrangThai)  VALUES (N'Bàn 1', N'Trống'),  (N'Bàn 2', N'Trống'),  (N'Bàn 3', N'Có người'),  (N'Bàn 4', N'Trống'),  (N'Bàn 5', N'Có người');  -- Chèn dữ liệu cho bảng "TaiKhoan"  INSERT INTO TaiKhoan (TenDangNhap, TenHienThi, MatKhau, Type)  VALUES (N'admin', N'Quản trị viên', N'123', 1),  (N'nhanvien', N'Nhân viên', N'123', 0);  -- Chèn dữ liệu cho bảng "Loai"  INSERT INTO Loai (TenLoai)  VALUES (N'Trà sữa'),  (N'Cà phê'),  (N'Sinh tố'),  (N'Nước uống lạnh');  -- Chèn dữ liệu cho bảng "ThucUong"  INSERT INTO ThucUong (TenThucUong, MaLoai, Gia)  VALUES (N'Capuchino', 2, 35000),  (N'Trà đào', 1, 25000),  (N'Espresso', 2, 32000),  (N'Sinh tố dâu', 3, 30000),  (N'Caramel Macchiato', 2, 38000),  (N'Trà bí đao', 1, 27000),  (N'Cà phê sữa đá', 2, 30000),  (N'Sinh tố bơ', 3, 29000),  (N'Latte', 2, 36000),  (N'Sinh tố xoài', 3, 28000); |

# Kết luận

Trong quá trình nghiên cứu và phát triển dự án, chúng em đã đạt được một số kết quả quan trọng và đã xác định được một số hạn chế cũng như hướng phát triển trong tương lai

## Kết quả đạt được

Xây dựng hệ thống quản lý quán café: Chúng tôi đã phát triển một hệ thống quản lý quán café, bao gồm quản lý tài khoản người dùng, thông tin bàn, loại thức uống, thức uống và hóa đơn. Hệ thống này cho phép quản lý dễ dàng theo dõi và điều hành các hoạt động của quán.

Giao diện người dùng thân thiện: Chúng tôi đã thiết kế giao diện người dùng thân thiện, giúp người dùng sử dụng hệ thống một cách dễ dàng và hiệu quả. Giao diện bao gồm các chức năng quản lý tài khoản, bàn, loại thức uống, thức uống, bán hàng, lịch sử hóa đơn và thống kê.

Cơ sở dữ liệu được khởi tạo: Chúng tôi đã khởi tạo cơ sở dữ liệu với dữ liệu mẫu, bao gồm danh sách bàn, loại thức uống và thức uống. Điều này giúp cho việc phát triển và kiểm thử hệ thống.

## Hạn chế và hướng phát triển của đề tài

Hạn chế về tính năng: Dự án hiện tại chỉ giới thiệu một phạm vi hạn chế của các tính năng. Có thể phát triển thêm tính năng như quản lý nhân viên, tích hợp thanh toán trực tuyến, quản lý kho, và nhiều tính năng khác để làm cho hệ thống phù hợp với nhiều loại doanh nghiệp khác nhau.

Bảo mật và xác thực: Hệ thống cần thêm các biện pháp bảo mật mạnh mẽ và xác thực người dùng để đảm bảo an toàn thông tin và ngăn chặn truy cập trái phép.

Phát triển di động: Để làm cho hệ thống trở nên tiện lợi hơn, có thể phát triển ứng dụng di động để quản lý quán café trên các thiết bị di động.

Tối ưu hóa hiệu suất: Cần tối ưu hóa hiệu suất của cơ sở dữ liệu và hệ thống để đảm bảo khả năng mở rộng dễ dàng khi doanh nghiệp phát triển.

# Danh mục sách tham khảo

1. DeFino, S., & DeFino, D. (2007). "Swing Hacks: Tips and Tools for Building Killer GUIs." O'Reilly Media.
2. Eckel, B. (2006). "Thinking in Java" (4th ed.). Prentice Hall.
3. Horstmann, C. S., & Geary, G. (2014). "Core Java, Volume II--Advanced Features" (10th ed.).
4. Smith, J. (2019). SQL for Beginners. Acme Publishing.