**TRƯỜNG ĐẠI HỌC AN GIANG  
KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN**

**LẬP TRÌNH QUẢN LÝ**

**PHẦN MỀM QUẢN LÝ CỬA HÀNG ĐIỆN TỬ**

**HỌ TÊN SINH VIÊN**

DTH225650 - Huỳnh Quốc Huy

**HỌ TÊN GIẢNG VIÊN**

Nguyễn Thị Mỹ Truyền

Nguyễn Hoài Nam

**AN GIANG, THÁNG 5 NĂM 2025**

**TRƯỜNG ĐẠI HỌC AN GIANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN LẬP TRÌNH QUẢN LÝ**

**PHẦN MỀM QUẢN LÝ CỬA HÀNG ĐIỆN TỬ**

**HỌ TÊN TÁC GIẢ**

DTH225650 - Huỳnh Quốc Huy

**HỌ TÊN GIẢNG VIÊN HƯỚNG DẪN**

Nguyễn Thị Mỹ Truyền

Nguyễn Hoài Nam

**AN GIANG, THÁNG 5 NĂM 2025**

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

**Giảng viên hướng dẫn**

( Ký và ghi rõ họ tên)

Nội dung nhận xét:

* **Đồng ý** hay **Không đồng ý**  cho sinh viên báo cáo TTCK; Nếu không đồng ý cần ghi rõ lý do.
* Kết quả được so với yêu cầu;
* Ý kiến khác (nếu có)

**MỤC LỤC**

[**1.** **GIỚI THIỆU** 5](#_Toc198530686)

[**1.1.** **GIỚI THIỆU ĐỀ TÀI** 5](#_Toc198530687)

[**1.1.1.** **Đặt vấn đề** 5](#_Toc198530688)

[**1.1.2.** **Phạm vi đề tài** 5](#_Toc198530689)

[**1.1.3.** **Các công cụ sử dụng** 6](#_Toc198530690)

[**1.2.** **CƠ SỞ LÝ THUYẾT** 7](#_Toc198530691)

[**1.2.1.** **.NET** 7](#_Toc198530692)

[**1.2.1.1.** **Giới thiệu về nền tảng .NET** 7](#_Toc198530693)

[**1.2.1.2.** **Vì sao sử dụng WinForms .NET Core thay vì WinForms .NET Framework?** 7](#_Toc198530694)

[**1.2.2.** **Entity Framework Core** 8](#_Toc198530695)

[**1.2.2.1.** **Giới thiệu Entity Framework Core** 8](#_Toc198530696)

[**1.2.2.2.** **Ưu điểm của EF Core trong phát triển theo mô hình 3-Layer** 9](#_Toc198530697)

[**1.2.3.** **Microsoft SQL Server** 9](#_Toc198530698)

[**1.2.3.1.** **Giới thiệu tổng quan** 9](#_Toc198530699)

[**1.2.3.2.** **Vai trò dự án** 10](#_Toc198530700)

[**2.** **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 10](#_Toc198530701)

[**2.1.** **SƠ ĐỒ USE CASE** 10](#_Toc198530702)

[**2.1.1.** **Mô tả tổng quan về sơ đồ Use Case** 10](#_Toc198530703)

[**2.2.** **SƠ ĐỒ CLASS** 12](#_Toc198530704)

[**2.2.1.** **Thiết kế lớp theo hướng đối tượng** 12](#_Toc198530705)

[**2.2.2.** **Mối quan hệ giữa các lớp** 13](#_Toc198530706)

[**2.3.** **SƠ ĐỒ QUAN HỆ DATABASE** 13](#_Toc198530707)

[**2.3.1.** **Mô tả sơ đồ ER (Entity Relationship)** 13](#_Toc198530708)

[**2.3.2.** **Quy tắc thiết kế quan hệ** 14](#_Toc198530709)

[**2.4.** **MÔ TẢ CÁC BẢNG TRONG DATABASE** 14](#_Toc198530710)

[**2.4.1.** **Bảng Categories** 14](#_Toc198530711)

[**2.4.2.** **Bảng Manufacturers** 14](#_Toc198530712)

[**2.4.3.** **Bảng Customers** 14](#_Toc198530713)

[**2.4.4.** **Bảng Employees** 14](#_Toc198530714)

[**2.4.5.** **Bảng Products** 14](#_Toc198530715)

[**2.4.6.** **Bảng Orders** 15](#_Toc198530716)

[**2.4.7.** **Bảng Order\_Details** 15](#_Toc198530717)

[**3.** **THIẾT KẾ** 15](#_Toc198530718)

[**3.1.** **THIẾT KẾ GIAO DIỆN** 15](#_Toc198530719)

[**3.1.1.** **Form Main** 15](#_Toc198530720)

[**3.1.2.** **Form Login** 16](#_Toc198530721)

[**3.1.3.** **Form Flash** 16](#_Toc198530722)

[**3.1.4.** **Form Aboutbox** 17](#_Toc198530723)

[**3.1.5.** **Form Categories** 17](#_Toc198530724)

[**3.1.6.** **Form Manufactures** 18](#_Toc198530725)

[**3.1.7.** **Form Customers** 18](#_Toc198530726)

[**3.1.8.** **Form Employees** 18](#_Toc198530727)

[**3.1.9.** **Form Products** 19](#_Toc198530728)

[**3.1.10.** **Form Orders** 19](#_Toc198530729)

[**3.1.11.** **Form Order Detais** 20](#_Toc198530730)

[**3.1.12.** **Form Print Order** 21](#_Toc198530731)

[**3.1.13.** **Form Sale** 22](#_Toc198530732)

[**3.1.14.** **Form Confirm** 23](#_Toc198530733)

[**3.1.15.** **Form Product Statistic** 24](#_Toc198530734)

[**3.1.16.** **Form Revenue Statistic** 24](#_Toc198530735)

[**3.1.17.** **Form Change Password** 24](#_Toc198530736)

[**3.2.** **THIẾT KẾ DỮ LIỆU** 25](#_Toc198530737)

[**3.2.1.** **Kiến trúc mô hình 3 lớp (Three-layer Architecture)** 25](#_Toc198530738)

[**3.2.2.** **Mô hình Entity trong Entity Framework Core** 25](#_Toc198530739)

[**3.2.2.1.** **Context** 26](#_Toc198530740)

[**3.2.2.2.** **Categories** 27](#_Toc198530741)

[**3.2.2.3.** **Manufactueres** 28](#_Toc198530742)

[**3.2.2.4.** **Customers** 29](#_Toc198530743)

[**3.2.2.5.** **Employees** 30](#_Toc198530744)

[**3.2.2.6.** **Products** 31](#_Toc198530745)

[**3.2.2.7.** **Orders** 32](#_Toc198530746)

[**3.2.2.8.** **OrderDetails** 33](#_Toc198530747)

[**3.3.** **THIẾT KẾ XỬ LÝ** 34](#_Toc198530748)

[**3.3.1.** **Luồng xử lý đăng nhập** 34](#_Toc198530749)

[**3.3.2.** **Luồng xử lý quản lý sản phẩm** 34](#_Toc198530750)

[**3.3.3.** **Luồng xử lý đặt hàng (Tạo hóa đơn)** 35](#_Toc198530751)

[**3.3.4.** **Luồng xử lý thống kê hóa đơn** 35](#_Toc198530752)

[**3.3.5.** **Luồng xử lý quản lý khách hàng** 36](#_Toc198530753)

[**3.3.6.** **Luồng xử lý quản lý nhân viên** 36](#_Toc198530754)

[**4.** **TỔNG KẾT** 36](#_Toc198530755)

[**4.1.** **KẾT QUẢ ĐẠT ĐƯỢC** 36](#_Toc198530756)

[**4.1.1.** **Hiểu rõ kiến trúc và quy trình phát triển phần mềm quản lý** 36](#_Toc198530757)

[**4.1.2.** **Xây dựng hoàn chỉnh phần mềm quản lý bán hàng điện tử** 37](#_Toc198530758)

[**4.1.3.** **Nâng cao kỹ năng chuyên môn và làm việc thực tế** 37](#_Toc198530759)

[**4.2.** **HƯỚNG PHÁT TRIỂN** 37](#_Toc198530760)

[**4.2.1.** **Phát triển thành phần Web hoặc Mobile** 37](#_Toc198530761)

[**4.2.2.** **Mở rộng chức năng nâng cao** 37](#_Toc198530762)

[**4.1.3.** **Kết nối đa người dùng** 38](#_Toc198530763)

[**4.3.** **KẾT LUẬN CHUNG** 38](#_Toc198530764)

1. **GIỚI THIỆU**
   1. **GIỚI THIỆU ĐỀ TÀI**
      1. **Đặt vấn đề**

Trong bối cảnh hiện nay, công nghệ thông tin ngày càng phát triển mạnh mẽ và đóng vai trò quan trọng trong hầu hết các lĩnh vực của đời sống, trong đó có lĩnh vực thương mại và quản lý bán hàng. Đặc biệt, đối với các cửa hàng bán lẻ như cửa hàng đồ điện tử, nhu cầu về việc tin học hóa quy trình quản lý là một yêu cầu tất yếu nhằm nâng cao hiệu quả hoạt động kinh doanh, tiết kiệm chi phí, giảm thiểu sai sót và tăng khả năng cạnh tranh trên thị trường.

Trên thực tế, nhiều cửa hàng nhỏ và vừa hiện nay vẫn quản lý hoạt động kinh doanh thông qua các phương pháp thủ công như ghi chép bằng sổ sách hoặc sử dụng các phần mềm phổ thông như Excel. Những cách làm này thường tiềm ẩn nhiều rủi ro như mất dữ liệu, sai lệch số liệu, khó khăn trong việc truy xuất thông tin, và không đảm bảo tính đồng bộ trong quản lý kho hàng, đơn hàng, khách hàng và doanh thu. Điều này dẫn đến việc giảm năng suất lao động và ảnh hưởng trực tiếp đến hiệu quả kinh doanh.

Xuất phát từ nhu cầu thực tiễn đó, đề tài “Phần mềm quản lý cửa hàng đồ điện tử” được thực hiện nhằm xây dựng một hệ thống phần mềm hỗ trợ quản lý toàn diện cho các cửa hàng kinh doanh mặt hàng điện tử như: điện thoại, ipad, máy tính, laptop, RAM, ổ cứng, và các linh kiện điện tử khác,... Phần mềm được thiết kế theo mô hình ba lớp (3-Layer Architecture), kết hợp cùng các công nghệ hiện đại và công cụ hỗ trợ phát triển phần mềm nhằm tạo ra một giải pháp có tính ứng dụng cao, dễ sử dụng, dễ bảo trì và mở rộng trong tương lai.

Mục tiêu chính của đề tài là xây dựng một phần mềm quản lý thân thiện với người dùng, có đầy đủ các chức năng thiết yếu như: quản lý sản phẩm, quản lý danh mục sản phẩm, quản lý nhân viên, khách hàng, đơn hàng, chi tiết đơn hàng, thống kê doanh thu,... đồng thời hỗ trợ in ấn các báo cáo thông qua công cụ ReportViewer. Qua đó, phần mềm không chỉ giúp nhân viên và chủ cửa hàng thực hiện công việc nhanh chóng, chính xác mà còn giúp nâng cao hiệu quả hoạt động kinh doanh tổng thể của cửa hàng.

* + 1. **Phạm vi đề tài**

Đề tài “Phần mềm quản lý cửa hàng đồ điện tử” tập trung phát triển và hoàn thiện một hệ thống phần mềm quản lý ứng dụng trong môi trường WinForms với các chức năng cơ bản và phổ biến trong mô hình quản lý bán hàng điện tử. Cụ thể, phạm vi của đề tài bao gồm:

* Quản lý sản phẩm: Bao gồm các chức năng thêm, sửa, xóa, tìm kiếm và hiển thị danh sách sản phẩm cùng các thông tin chi tiết như mã sản phẩm, tên sản phẩm, mô tả, giá bán, số lượng tồn, hình ảnh, và danh mục sản phẩm.
* Quản lý danh mục sản phẩm: Cho phép phân loại các sản phẩm thành từng danh mục cụ thể để thuận tiện trong quản lý và tìm kiếm.
* Quản lý nhân viên: Lưu trữ và quản lý thông tin nhân viên làm việc tại cửa hàng.
* Quản lý khách hàng: Lưu trữ thông tin khách hàng để phục vụ cho việc lập hóa đơn và quản lý lịch sử mua hàng.
* Quản lý đơn hàng và chi tiết đơn hàng: Hỗ trợ lập hóa đơn bán hàng, tính tổng tiền, theo dõi tình trạng đơn hàng và các sản phẩm đã bán.
* Thống kê – báo cáo: Thực hiện thống kê doanh thu, sản phẩm bán hóa đơn bằng công cụ ReportViewer.

Lưu trữ dữ liệu bằng SQL Server: Toàn bộ dữ liệu được lưu trữ và xử lý thông qua hệ quản trị cơ sở dữ liệu SQL Server, đảm bảo tính toàn vẹn và an toàn của thông tin.

Kiến trúc 3 lớp (3-Layer Architecture): Phần mềm được tổ chức theo ba lớp gồm Presentation Layer (Giao diện người dùng), Business Logic Layer (Xử lý nghiệp vụ), và Data Access Layer (Truy xuất dữ liệu), và lớp hỗ trợ vận chuyển dữ liệu Data Transfer Object.

Đề tài không đi sâu vào các tính năng nâng cao như tích hợp bán hàng trực tuyến, thanh toán qua ví điện tử, mã vạch sản phẩm hay các công nghệ Web/Mobile. Tuy nhiên, phần mềm được xây dựng trên nền tảng có khả năng mở rộng, hoàn toàn có thể tích hợp thêm các chức năng này trong tương lai.

* + 1. **Các công cụ sử dụng**

Trong quá trình thiết kế và phát triển phần mềm, sinh viên Huỳnh Quốc Huy sử dụng một số công cụ, ngôn ngữ lập trình và thư viện hỗ trợ như sau:

* **Visual Studio 2022**: Là môi trường phát triển tích hợp (IDE) chính được sử dụng để xây dựng giao diện người dùng, viết mã chương trình và quản lý toàn bộ dự án.
* **.NET 8.0 WinForms**: Nền tảng chính để xây dựng ứng dụng Windows Forms với giao diện đồ họa thân thiện, hỗ trợ thao tác bằng chuột và bàn phím.
* **SQL Server 2022**: Hệ quản trị cơ sở dữ liệu quan hệ dùng để lưu trữ, truy vấn và xử lý dữ liệu một cách hiệu quả.
* Entity Framework Core 8 (Microsoft.EntityFrameworkCore): Là ORM (Object-Relational Mapping) được sử dụng để tạo mô hình dữ liệu, ánh xạ giữa các bảng trong CSDL và các lớp đối tượng trong chương trình, hỗ trợ truy vấn dữ liệu thông qua LINQ.
* **ReportViewerCore.WinForms**: Công cụ hiển thị báo cáo cho WinForms, giúp lập và xuất các báo cáo như hóa đơn, doanh thu, danh sách sản phẩm.
* **ClosedXML**: Thư viện mã nguồn mở hỗ trợ thao tác với tệp Excel (XLSX), giúp xuất dữ liệu báo cáo hoặc bảng tính ra định dạng Excel một cách dễ dàng.
* Microsoft.Extensions.DependencyInjection.Abstractions: Hỗ trợ kỹ thuật Dependency Injection (DI) để nâng cao khả năng mở rộng, kiểm thử và tái sử dụng mã nguồn.
* **SlugGenerator**: Thư viện hỗ trợ chuyển đổi tên sản phẩm/danh mục sang dạng “slug” dễ đọc và thân thiện với URL hoặc mã định danh.

Việc kết hợp các công cụ và thư viện hiện đại này không chỉ giúp quá trình phát triển phần mềm trở nên chuyên nghiệp và có tổ chức hơn, mà còn đảm bảo phần mềm có thể hoạt động ổn định, dễ bảo trì và nâng cấp về sau.

* 1. **CƠ SỞ LÝ THUYẾT**
     1. **.NET**
        1. **Giới thiệu về nền tảng .NET**

.NET là một nền tảng phát triển phần mềm do Microsoft phát triển, hỗ trợ xây dựng nhiều loại ứng dụng từ desktop, web đến di động và các hệ thống phân tán. Nền tảng này gồm có hai nhánh chính là .NET Framework (ra mắt từ năm 2002) và .NET Core (ra đời năm 2016, và đến năm 2020 thì hợp nhất thành .NET 5+ và nay là .NET 8).

Trong đó, .NET Framework là phiên bản gắn chặt với hệ điều hành Windows, còn .NET Core/.NET 5+ là nền tảng mã nguồn mở, đa nền tảng, hỗ trợ Windows, Linux và macOS, được thiết kế lại với hiệu suất cao, nhẹ, linh hoạt và phù hợp hơn cho các kiến trúc hiện đại.

Việc chuyển từ WinForms .NET Framework cũ sang WinForms .NET Core / .NET 8 mang lại nhiều lợi ích đáng kể trong phát triển phần mềm hiện đại, nhất là khi áp dụng mô hình nhiều lớp (3-layer architecture).

* + - 1. **Vì sao sử dụng WinForms .NET Core thay vì WinForms .NET Framework?**

Trong đồ án này, tác giả Huỳnh Quốc Huy lựa chọn WinForms .NET 8 (thuộc nhánh .NET Core) thay vì WinForms trên .NET Framework truyền thống, vì các lý do quan trọng sau:

Hiệu suất cao và tối ưu hóa tốt hơn: .NET Core (và đặc biệt từ .NET 5 trở lên) được tái kiến trúc với runtime hiệu suất cao hơn, thời gian khởi động nhanh, quản lý bộ nhớ tốt hơn, phù hợp với những phần mềm yêu cầu xử lý dữ liệu nhiều như phần mềm quản lý cửa hàng.

Hỗ trợ Entity Framework Core tốt hơn: EF Core là ORM thế hệ mới tối ưu cho .NET Core, cung cấp LINQ mạnh mẽ, hỗ trợ async, caching, tracking... Nếu dùng .NET Framework, chỉ có thể dùng EF6, vốn không còn phát triển mạnh. EF Core tương thích hoàn toàn và dễ tích hợp với WinForms .NET Core.

Cấu trúc nhẹ và linh hoạt: WinForms .NET Core cho phép tách nhỏ các thành phần cần thiết (runtime trimming, single-file deployment), giúp đóng gói phần mềm nhẹ hơn, dễ triển khai mà không cần cài đặt thêm trên máy khách.

Hỗ trợ Dependency Injection (DI) mặc định: Khả năng tích hợp DI giúp xây dựng kiến trúc 3 lớp chặt chẽ hơn, giúp dễ dàng test các lớp logic nghiệp vụ, tách rời các phụ thuộc, và tuân theo nguyên lý SOLID.

Đặc biệt là tương lai phát triển lâu dài: Microsoft đã định hướng chuyển toàn bộ hệ sinh thái sang nhánh .NET mới (bắt đầu từ .NET 5), do đó các tính năng mới, công cụ và thư viện đều ưu tiên hỗ trợ trên .NET Core/.NET 8+. Chọn WinForms .NET Core giúp đảm bảo khả năng nâng cấp và mở rộng phần mềm về sau.

Mở rộng tương lai: Nếu sau này muốn tích hợp Web API, Blazor, hoặc chuyển sang nền tảng Web hoặc đa nền tảng thì .NET Core hỗ trợ tốt hơn rất nhiều so với .NET Framework.

Vì thế việc sử dụng WinForms .NET 8 thay vì .NET Framework cũ là sự lựa chọn mang tính chiến lược ngay từ khi tôi khởi tạo dự án dù nó khác với hướng xây dựng cũ của giáo án cũ, giúp tận dụng các tính năng mới, tăng hiệu suất và đảm bảo phần mềm có thể thích ứng tốt với xu hướng công nghệ trong tương lai.

* + 1. **Entity Framework Core**
       1. **Giới thiệu Entity Framework Core**

Entity Framework Core (EF Core) là một thư viện ORM (Object-Relational Mapping) hiện đại do Microsoft phát triển, cho phép lập trình viên thao tác với cơ sở dữ liệu thông qua các đối tượng C# thay vì viết trực tiếp các câu lệnh SQL thuần. EF Core là phiên bản cải tiến hoàn toàn của Entity Framework cổ điển, được thiết kế riêng cho nền tảng .NET Core và .NET 5+.

EF Core hoạt động như một lớp trung gian giữa tầng logic nghiệp vụ và hệ quản trị cơ sở dữ liệu (SQL Server, MySQL, PostgreSQL, v.v.). EF Core hỗ trợ:

* Mapping giữa bảng dữ liệu và lớp đối tượng (entity).
* Tạo database từ mô hình (Code First) hoặc tạo mô hình từ database (Database First).
* Truy vấn dữ liệu bằng LINQ (Language Integrated Query).

Giao dịch, kết nối, kiểm soát dữ liệu theo hướng object.

* + - 1. **Ưu điểm của EF Core trong phát triển theo mô hình 3-Layer**

EF Core là công cụ lý tưởng khi áp dụng mô hình 3-Layer Architecture vì những lý do sau:

Tách biệt rõ ràng giữa các tầng: EF Core hoạt động ở tầng truy xuất dữ liệu (DAL), có thể tích hợp trực tiếp với các repository, giúp phân tách rõ logic nghiệp vụ và giao diện.

Code First hỗ trợ thiết kế từ mô hình: Dễ dàng tạo bảng từ class hoặc dùng Migration để cập nhật cấu trúc CSDL, giúp dễ bảo trì.

Hỗ trợ Dependency Injection tốt: Cho phép inject DbContext thông qua constructor, dễ dàng cấu hình trong DI Container của .NET Core.

Truy vấn mạnh mẽ bằng LINQ: Cho phép viết truy vấn dữ liệu như C# thuần, tránh lỗi cú pháp SQL, tăng năng suất và giảm sai sót.

Khả năng mở rộng cao: Có thể thêm các thư viện như AutoMapper để chuyển đổi giữa Entity và DTO, phục vụ cho tầng Business Logic và Presentation Layer.

Hỗ trợ theo dõi thay đổi (Change Tracking): Dễ dàng quản lý trạng thái entity (Thêm, Sửa, Xóa), giúp đơn giản hóa thao tác với dữ liệu.

Tích hợp tốt với ReportViewer và các công cụ xuất báo cáo: Có thể dễ dàng đưa dữ liệu từ DbContext ra dataset hoặc collection phục vụ cho xuất báo cáo qua RDLC hoặc ClosedXML hỗ trợ xuất Excel.

Nhờ EF Core, việc xây dựng mô hình nhiều lớp trở nên gọn gàng, dễ bảo trì, dễ test và có khả năng mở rộng cao. Đây là thành phần cốt lõi giúp phần mềm đạt được tính hiện đại, linh hoạt và tối ưu trong quản lý dữ liệu.

* + 1. **Microsoft SQL Server**
       1. **Giới thiệu tổng quan**

Microsoft SQL Server là hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ do Microsoft phát triển. SQL Server cung cấp một nền tảng ổn định, bảo mật và có khả năng xử lý lượng dữ liệu lớn, được sử dụng rộng rãi trong nhiều hệ thống phần mềm từ nhỏ đến lớn.

SQL Server cung cấp các công cụ quản lý CSDL trực quan như SQL Server Management Studio (SSMS), hỗ trợ ngôn ngữ T-SQL để truy vấn và thao tác dữ liệu, cùng nhiều tính năng mạnh như backup/restore, trigger, stored procedure, và tích hợp tốt với các sản phẩm Microsoft khác.

* + - 1. **Vai trò dự án**

Trong đồ án này, SQL Server được sử dụng làm nền tảng lưu trữ và quản lý toàn bộ dữ liệu của phần mềm. Cụ thể:

* Lưu trữ thông tin sản phẩm, đơn hàng, khách hàng, nhân viên, danh mục, hóa đơn,...
* Tích hợp chặt chẽ với Entity Framework Core: giúp dễ dàng tạo bảng, ánh xạ đối tượng và xử lý dữ liệu bằng code.
* Tính ổn định và bảo mật cao, đảm bảo dữ liệu không bị mất mát hay sai lệch trong quá trình sử dụng.
* Khả năng mở rộng tốt, có thể nâng cấp quy mô dữ liệu lên hàng triệu bản ghi mà không ảnh hưởng đến hiệu năng nếu tối ưu đúng cách.

1. **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**
   1. **SƠ ĐỒ USE CASE**
      1. **Mô tả tổng quan về sơ đồ Use Case**

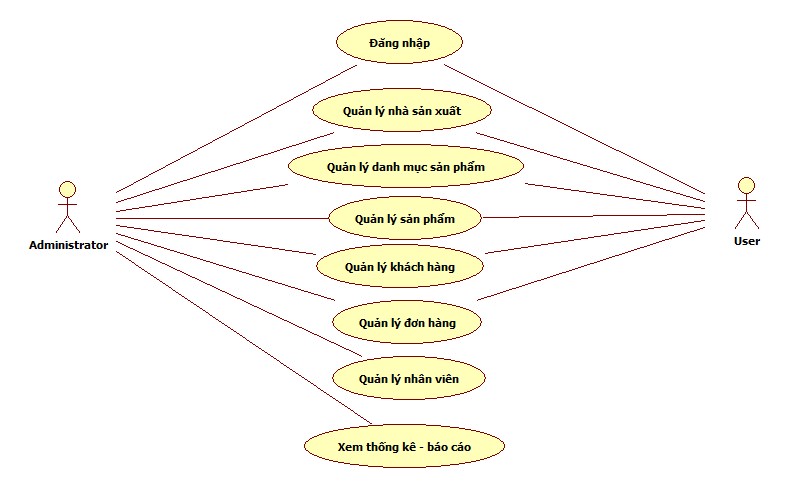
Sơ đồ Use Case thể hiện các chức năng mà hệ thống cần cung cấp cho người sử dụng thông qua các tác nhân (actors). Trong hệ thống quản lý cửa hàng điện tử này, có hai tác nhân chính:

Administrator (Admin): người có toàn quyền truy cập và quản lý mọi thành phần trong hệ thống.

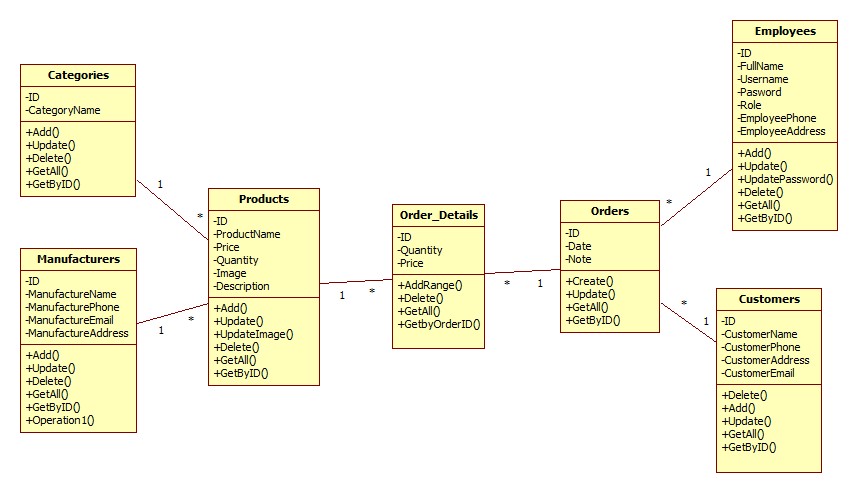
Nhân viên (Employee/Member): người sử dụng hệ thống nhưng bị giới hạn một số quyền nhất định (không quản lý nhân viên, không xem báo cáo).

Mỗi tác nhân sẽ tương tác với các chức năng khác nhau:

* Quản lý danh mục (category)
* Quản lý nhà sản xuất (manufacturer)
* Quản lý khách hàng (customer)
* Quản lý nhân viên (chỉ Admin)
* Quản lý sản phẩm
* Quản lý đơn hàng và chi tiết đơn hàng
* Xem báo cáo (chỉ Admin)
  + 1. **Chi tiết các Use Case chính**



* Use Case: Quản lý danh mục sản phẩm
  + Tác nhân: Admin, Nhân viên
  + Mô tả: Thêm, sửa, xóa loại sản phẩm
* Use Case: Quản lý sản phẩm
  + Tác nhân: Admin, Nhân viên
  + Mô tả: Cho phép thêm, sửa, xóa, tìm kiếm và xem thông tin sản phẩm.
* Use Case: Quản lý đơn hàng
  + Tác nhân: Admin, Nhân viên
  + Mô tả: Tạo đơn hàng mới, cập nhật, xem chi tiết đơn hàng, tính toán tổng tiền.
* Use Case: Quản lý nhân viên
  + Tác nhân: Chỉ Admin
  + Mô tả: Thêm, sửa, xóa nhân viên, phân quyền vai trò.
* Use Case: Quản lý nhà sản xuất
  + Tác nhân: Admin, Nhân viên
  + Mô tả: Thêm, sửa, xóa nhà sản xuất.
* Use Case: Xem báo cáo
  + Tác nhân: Chỉ Admin
  + Mô tả: Thống kê sản phẩm, doanh thu theo ngày/tháng.
* Use Case: Đăng nhập
  + Tác nhân: Admin, Nhân viên
  + Mô tả: Xác thực người dùng vào hệ thống, phân quyền sau khi đăng nhập.
  1. **SƠ ĐỒ CLASS**
     1. **Thiết kế lớp theo hướng đối tượng**



Hệ thống được thiết kế theo hướng hướng đối tượng với các class chính tương ứng với các bảng trong cơ sở dữ liệu. Mỗi class đại diện cho một đối tượng và chứa các thuộc tính (properties) và hành vi (methods) tương ứng.

Cụ thể hơn với đồ án đang xây dựng Huỳnh Quốc Huy thiết kế class như sau:

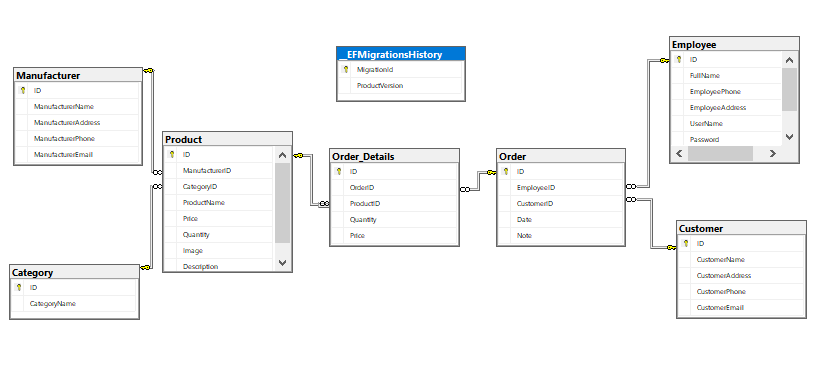
* Manufacturers:
  + Thuộc tính: ID,ManufactuerName,ManufactuerAddress,ManufactuerPhone,ManufactuerEmail
  + Phương thức: Add(), Update(), Delete(), GetAll(), FindById(), GetAll().
* Categories:
  + Thuộc tính: ID, CategoryName.
  + Phương thức: Add(), Update(), Delete(), GetAll(), FindById(), GetAll().
* Products
  + Thuộc tính: ID, ProductName, Price, Quantity, Image, Description, CategoryID, ManufacturerID
  + Phương thức: Add(), Update(), Delete(), GetAll(), FindById(),GetAll().
* Employees:
  + Thuộc tính: ID, FullName, EmployeePhone, EmployeeAddress, UserName, Password, Role.
  + Phương thức: Add(), Update(), Delete(), GetAll(), FindById(), GetAll().
* Orders
  + Thuộc tính: ID, EmployeeID, CustomerID, Date, Note
  + Phương thức: CreateOrder(), UpdateOrder(), GetOrderDetails(), GetAll().
* Order Details:
  + Thuộc tính: ID, OrderID, ProductID, Quantity, Price
  + Phương thức: Insert(), AddRange(), Update(), GetAll().
    1. **Mối quan hệ giữa các lớp**

Các lớp sẽ có mối quan hệ như sau:

* Lớp Order có quan hệ 1-n với Order\_Details
* Lớp Product có quan hệ n-1 với Manufacturer và Category
* Lớp Employee quản lý các đơn hàng (1-n)
* Lớp Customer là người đặt đơn hàng (1-n)

Các class có thể sử dụng mô hình DTO (Data Transfer Object) để trao đổi dữ liệu giữa các tầng (UI – BLL – DAL)

* 1. **SƠ ĐỒ QUAN HỆ DATABASE**

****

* + 1. **Mô tả sơ đồ ER (Entity Relationship)**

Sơ đồ quan hệ thể hiện rõ cách các bảng dữ liệu liên kết với nhau thông qua các khóa chính và khóa ngoại:

Product liên kết với Manufacturer và Category thông qua ManufacturerID và CategoryID.

Order\_Details là bảng phụ giữa Order và Product, có các khóa ngoại OrderID và ProductID.

Order liên kết với Customer và Employee thông qua CustomerID và EmployeeID.

* + 1. **Quy tắc thiết kế quan hệ**

Quy tắc thiết kế quan hệ được thiết lập như sau:

* Tất cả các mối quan hệ đều là quan hệ 1-n hoặc n-1.
* Các bảng đều có khóa chính tự động tăng (ID).
* Dữ liệu được chuẩn hóa để tránh trùng lặp thông tin.
* Sử dụng khóa ngoại để đảm bảo toàn vẹn dữ liệu (referential integrity).
  1. **MÔ TẢ CÁC BẢNG TRONG DATABASE**
     1. **Bảng Categories**

ID: khóa chính, định danh danh mục.

CategoryName: tên danh mục (vd: Laptop, Điện thoại…).

* + 1. **Bảng Manufacturers**

ID: khóa chính.

ManufacturerName: tên nhà sản xuất.

ManufacturerAddress, ManufacturerPhone, ManufacturerEmail: thông tin liên lạc.

* + 1. **Bảng Customers**

ID: khóa chính.

CustomerName, CustomerAddress, CustomerPhone, CustomerEmail: thông tin khách hàng.

* + 1. **Bảng Employees**

ID: khóa chính.

FullName, EmployeePhone, EmployeeAddress: thông tin cá nhân.

UserName, Password, Role: thông tin đăng nhập và phân quyền (Admin / Nhân viên).

* + 1. **Bảng Products**

ID: khóa chính.

ManufacturerID: khóa ngoại đến Manufacturers.

CategoryID: khóa ngoại đến Categories.

ProductName, Price, Quantity, Image, Description: thông tin sản phẩm.

* + 1. **Bảng Orders**

ID: khóa chính.

EmployeeID, CustomerID: khóa ngoại đến Employees và Customers.

Date, Note: ngày đặt hàng và ghi chú.

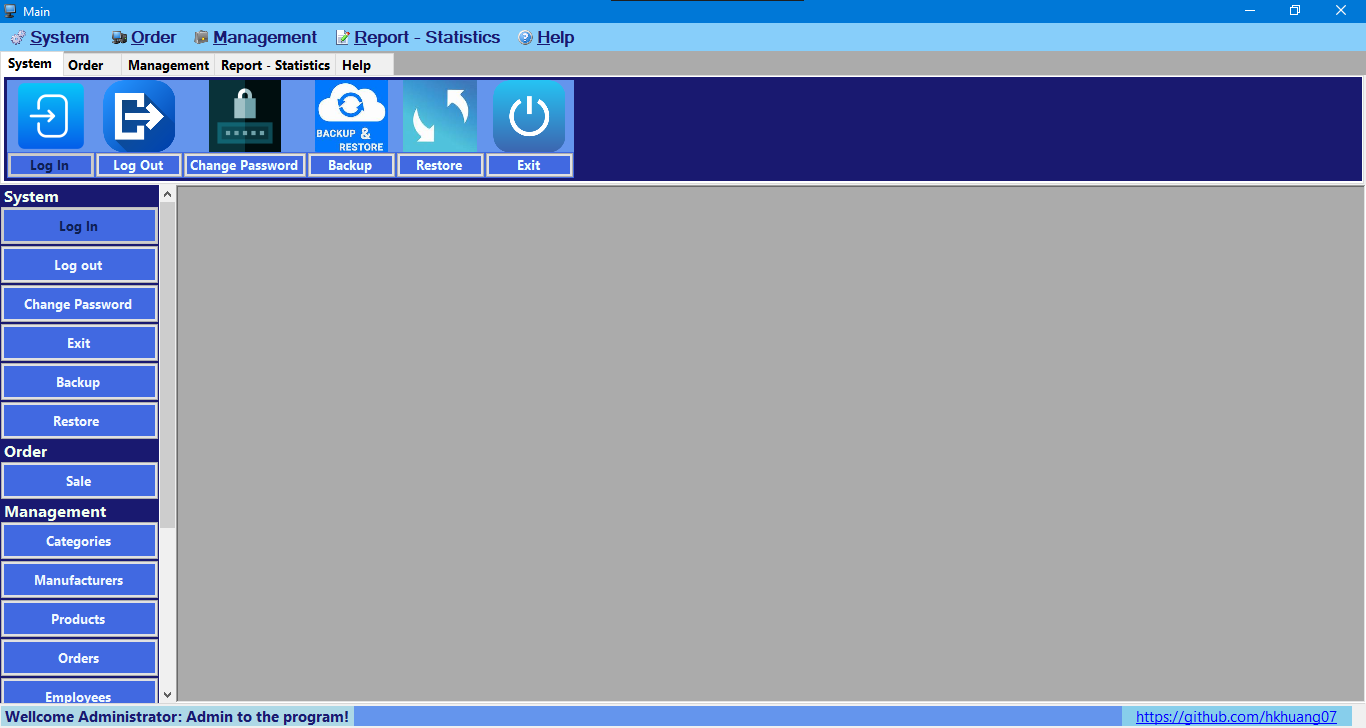
* + 1. **Bảng Order\_Details**

ID: khóa chính.

OrderID, ProductID: khóa ngoại.

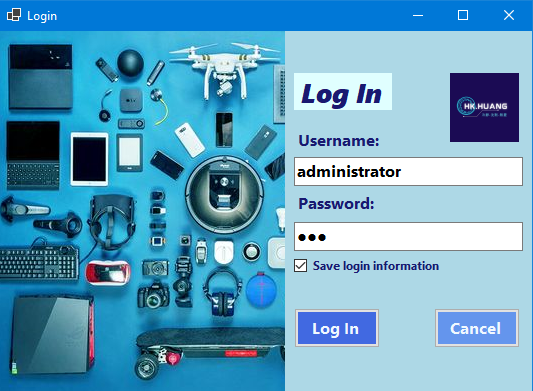
Quantity, Price: số lượng và giá tại thời điểm đặt hàng.

1. **THIẾT KẾ**
   1. **THIẾT KẾ GIAO DIỆN**
      1. **Form Main**

****

Form main được thiết kế gồm 3 vùng tùy chọn chính : menu, các tab tùy chọn, và siderbar. Trong đó 3 vùng này thực hiện các chức năng tương tự nhau. Các tùy chọn sẽ bao gồm:

* System: Log in, Log out, Change password, Data: Backup, Restore, Exit.
* Order: fomr Sale thực hiện chức năng nghiệp vụ order sản phẩm
* Management: Quản lý các bảng: categories, manufactures, products, employees, customers, orders…
* Report-Statistics: thống kê sản phẩm, thống kê doanh thu.
* Help: Thông tin phần mềm, mở web help…
  + 1. **Form Login**

****

Form Login hỗ trợ tính năng lưu thông tin, enter tại textbox Password để đăng nhập giúp nâng cao trải nghiệm người dùng. Button Log in để đăng nhập, cancel để đóng form.

* + 1. **Form Flash**

****

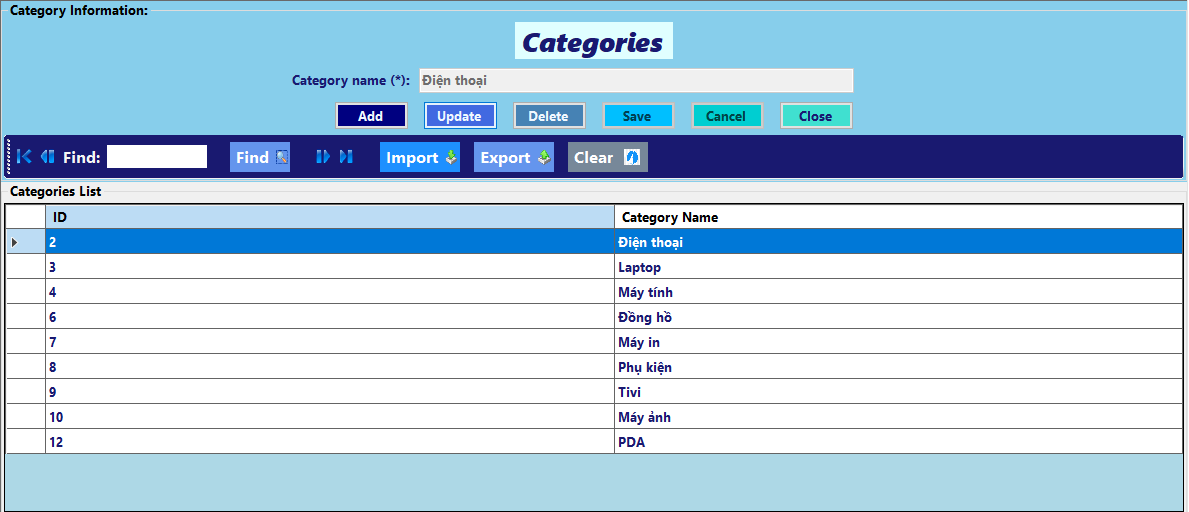
Form hiển thị trước khi người dùng vào giao diện chính phần mềm có hỗ trợ hiển thị Processbar và hình ảnh tượng trưng

* + 1. **Form Aboutbox**

****

Form hiển thị thông tin phần mềm được cấu hình trong project. Hiển thị các thông tin như tên phần mềm, bản quyền, phiên bản…

* + 1. **Form Categories**

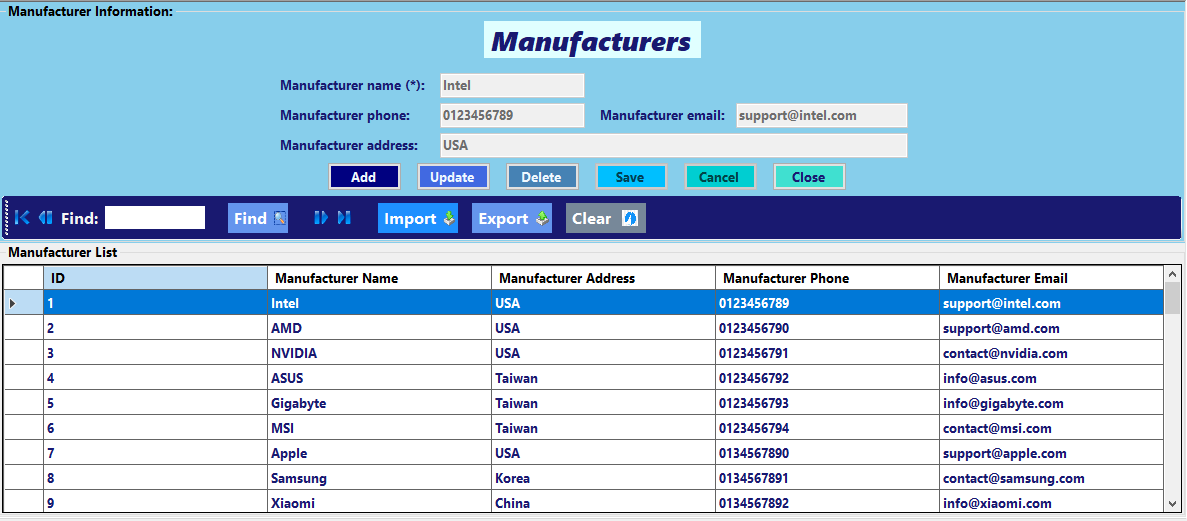
****

Form quản lý của bảng Categories hỗ trợ các chức năng thêm, sửa, xóa các bảng ghi trong form.

* Button Add để xóa trắng các trường, đánh dấu là thêm.
* Button Update để đánh dấu là sửa và lấy ID của bảng ghi hiện tại.
* Button Save thì tùy vào chọn Add hay Update sẽ xử lý Thêm hoặc Cập nhật dữ liệu.
* Button Cancel để hủy các hành động bằng cách load lại trang. Button Close để đóng form.

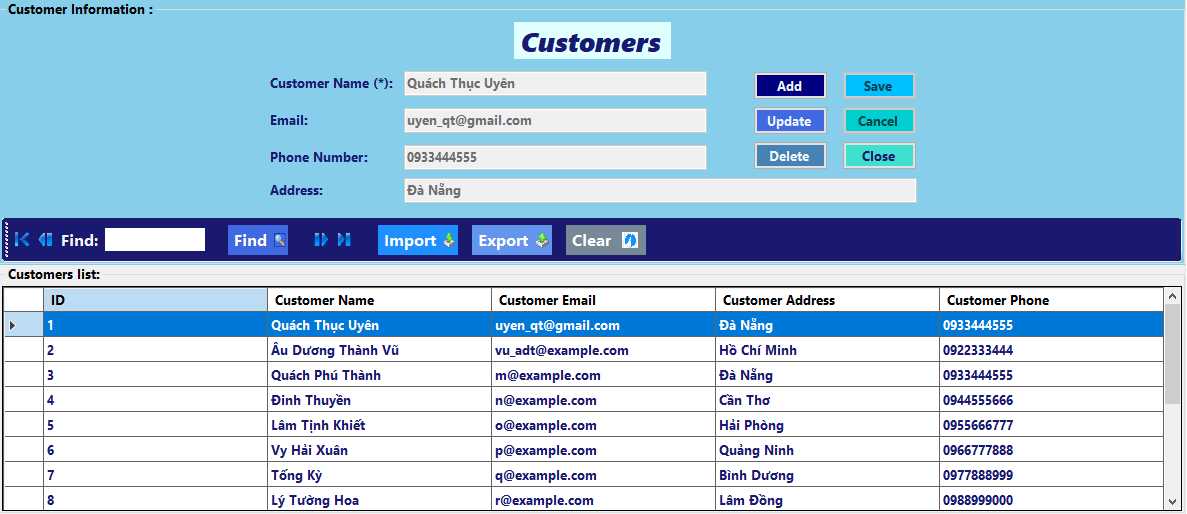
Ngoài ra form còn hỗ trợ tính năng thanh điều hướng được binding với datagridView và các controls, tại đây có thể tìm kiếm các record, hoặc import/export data dạng excel.

* + 1. **Form Manufactures**

****

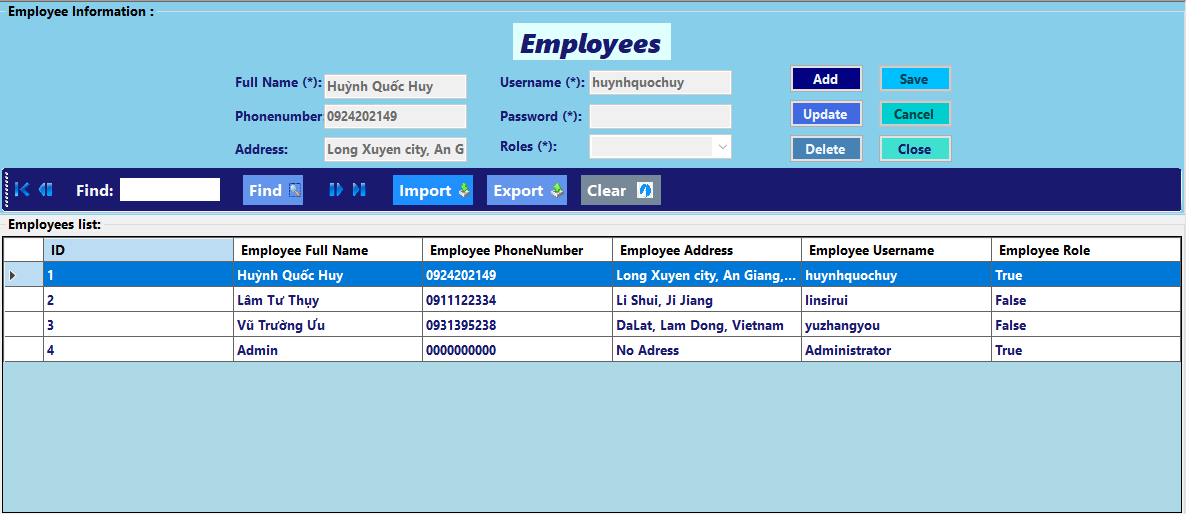
Form Manufactures cũng có các chức năng tương tự như form Categories.

* + 1. **Form Customers**

****

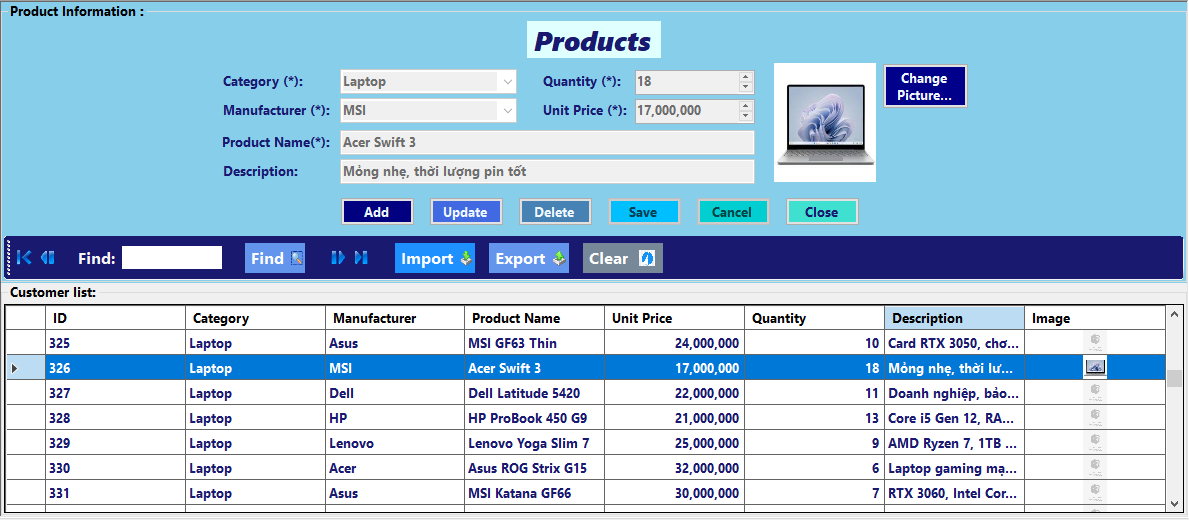
Form Customers cũng có các chức năng tương tự như form Categories.

* + 1. **Form Employees**

****

Form Employee cũng có các chức năng tương tự như form Categories.

* + 1. **Form Products**

****

Form Product cũng có các chức năng tương tự như form Categories. Được hỗ trợ binding và hiển thị hình ảnh sản phẩm giúp việc quản lý trực quan hơn.

* + 1. **Form Orders**

****

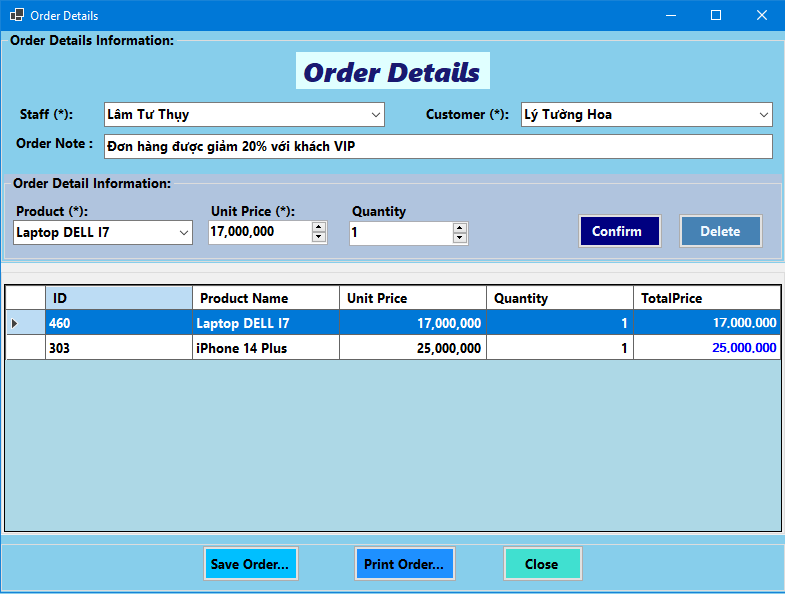
Form Orders chứa thanh điều hướng binding với datagridview và bổ sung chức năng export và tìm kiếm Order.

Bên cạnh đó là button Create order để mở ra form Order Details bên dưới. Button Print order để xuất hóa đơn dạng PDF tại dòng đang chọn trên datagridView.

* Button Update để mở form Order Details để cập nhật thêm nếu khách hàng thay đổi hóa đơn trước đó.
* Button Delete để xóa hóa đơn, nút Close để đóng form.

Ngoài ra khi double click vào cột Details tại 1 dòng nào đó thì sẽ hiển thị form OrderDetails của hóa đơn đó điều này cũng nhắm giúp nâng cao trải nghiệm của người dùng.

* + 1. **Form Order Detais**

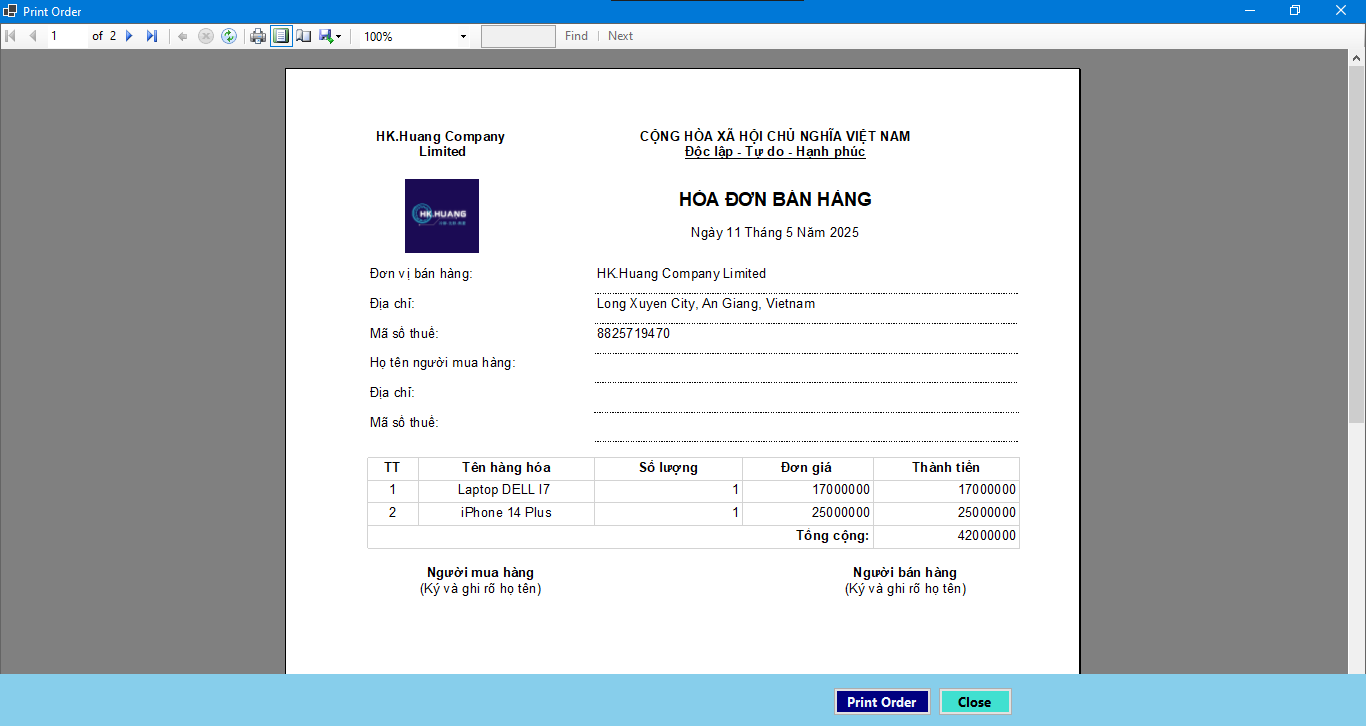
****

Form này hiển thị khi người dùng bấm Create order tại form order hoặc bấm update order đang chọn cũng như double click vào dòng đang chọn.

Form này cho phép người dùng chỉnh sửa chi tiết hóa đơn. Button Confirm để thêm sản phẩm đang chọn trên combobox vào danh sách chi tiết hóa đơn. Delete để xóa chi tiết đó khỏi hóa đơn.

Save Order để cập nhật hóa đơn cũ hoặc thêm hóa đơn mới tùy vào tùy chọn ở form Order. Close để đóng form

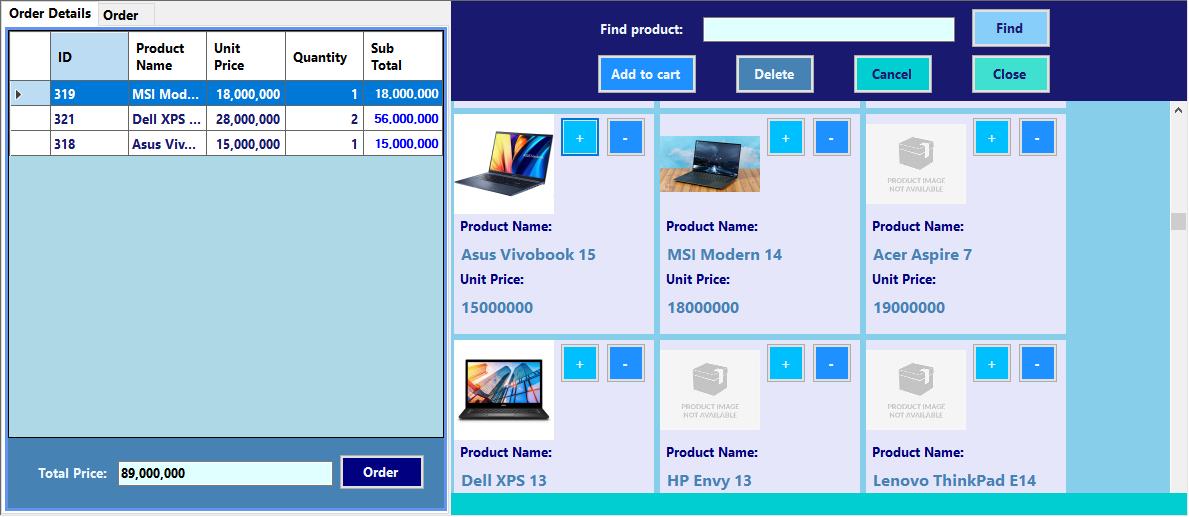
* + 1. **Form Print Order**

****

Form này hiển thị khi người dùng chọn Print order trong form Order để xuất hóa đơn cho khách hàng tại dòng đang chọ trên datagridview. Nó cũng tương tự khi người dùng chọn Print trong form Sale (nghiệp vụ bán hàng).

Form hỗ trợ xuất hóa đơn theo thiết kế report dạng PDF.

* + 1. **Form Sale**

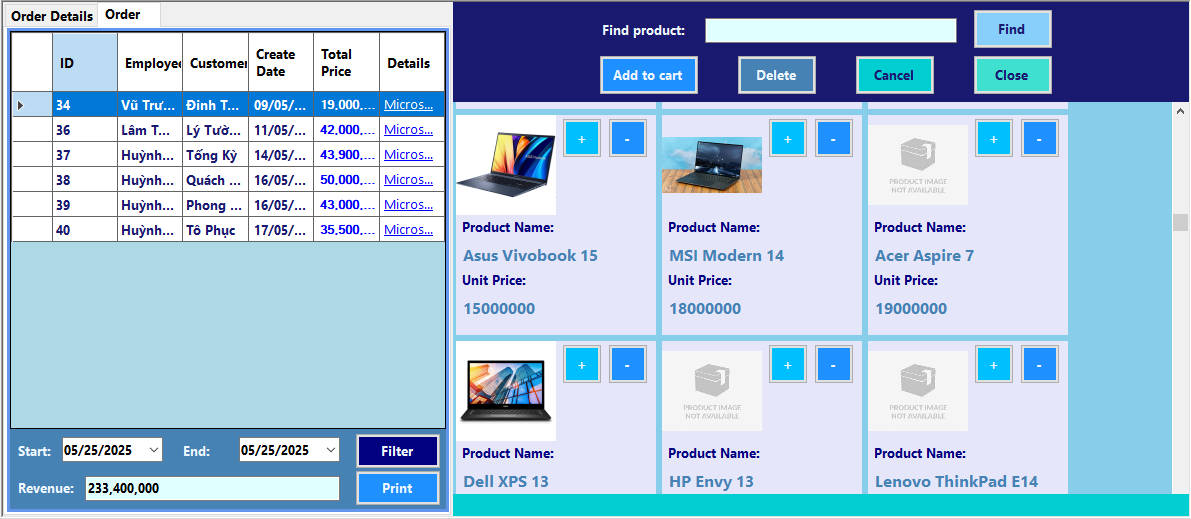
****

Form này thực hiện chức năng nghiệp vụ chính của phần mềm là bán hàng. Form có lưới hiển thị các sản phẩm.

Bổ sung thêm các tính năng tìm kiếm. Button add to card để thêm sản phẩm vào danh sách chi tiết hóa đơn.

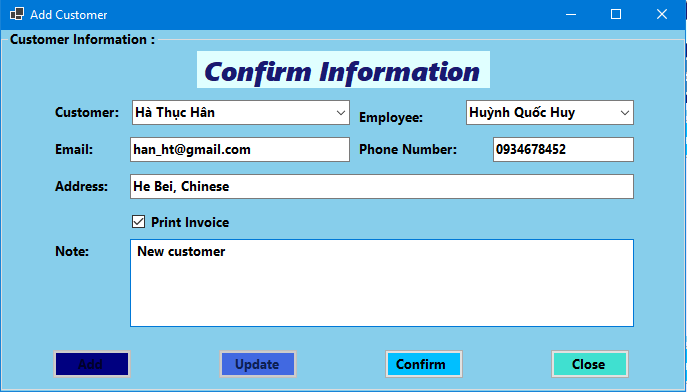
Button Delete để giảm số lượng sản phẩm. Hoặc xóa khỏi danh sách chi tiết nếu số lượng <= 0.

Button cộng/trừ trên UserControl để tăng giảm số lượng hoặc xóa sản phẩm ra khỏi danh sách chi tiết hóa đơn.

****

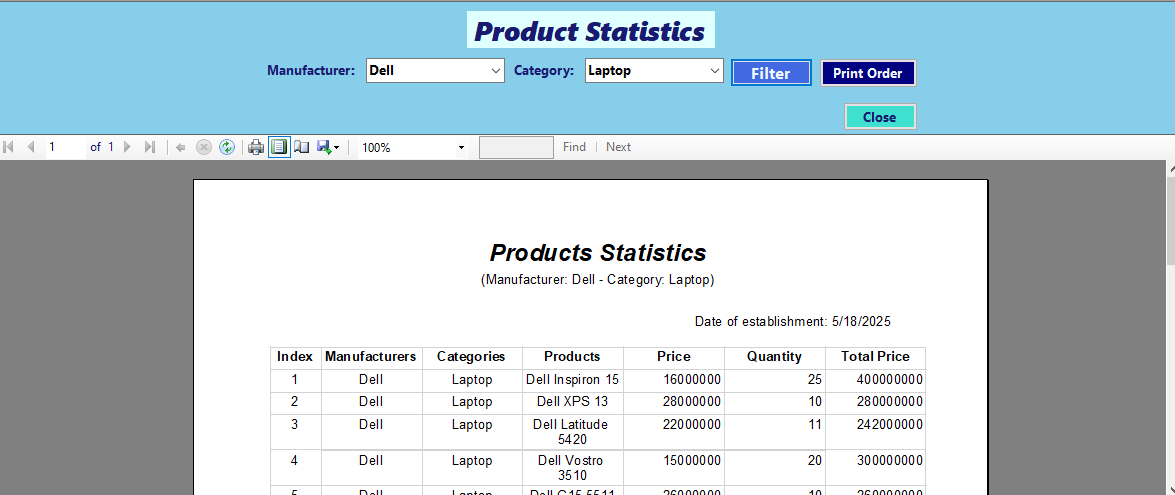
Button Order thực hiện chức năng thêm order mới. Trước hết form sẽ mở fomr Confimr bên dưới nhằm lấy thông tin nhân viên, người dùng để tạo hóa đơn. Khi form Confirm đóng sẽ tạo hóa đơn mới với khách hàng trên từ form Confirm sau đó xuất hóa đơn theo tùy chọn của người dùng.

* + 1. **Form Confirm**

****

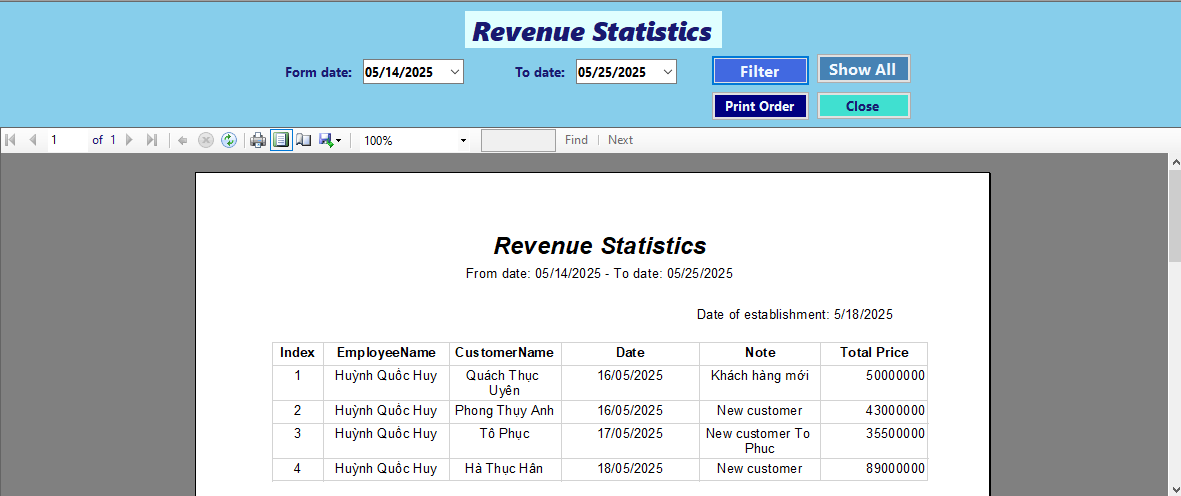
Form này có mục đích lấy thông tin để lập order.

* Button Add đánh dấu Add nếu khách hàng mới, xóa trắng các trường.
* Button Update để cập nhật nếu khác hàng cũ. Confirm để xác nhận và đóng form trả về form Sale hoàn tất order. Tại đây sẽ hiển thị form Print Order nếu người dùng check Print Invoice.
  + 1. **Form Product Statistic**

****

Form thống kê các sản phẩm được filter theo manufacture theo category. Nếu hai combobox trắng trường thì load toàn bộ

* + 1. **Form Revenue Statistic**

****

Form thống kê doanh thu được filter ngày bắt đầu ngày kết thúc

Mặc định thì sẽ load toàn bộ thời gian.

* + 1. **Form Change Password**

****

Form có chức năng giúp người dùng đổi mật khẩu.

* 1. **THIẾT KẾ DỮ LIỆU**
     1. **Kiến trúc mô hình 3 lớp (Three-layer Architecture)**

Trong hệ thống quản lý bán hàng, việc phân tách mã nguồn theo mô hình 3 lớp giúp cho phần mềm dễ bảo trì, mở rộng và kiểm thử. Các lớp được chia như sau:

**Presentation Layer (Giao diện người dùng):** Là nơi người dùng tương tác với hệ thống, gồm các Form, Controls trong WinForms. Dữ liệu được hiển thị từ tầng trung gian là DTO và được gửi đi thông qua các phương thức Service.

**Business Logic Layer (BLL - Xử lý nghiệp vụ):** Chứa các class/service xử lý logic nghiệp vụ như kiểm tra tính hợp lệ của dữ liệu, gọi repository để lấy dữ liệu từ DB, ánh xạ qua lại giữa Entity và DTO. Lớp này sử dụng AutoMapper để chuyển đổi giữa các tầng.

**Data Access Layer (DAL - Truy cập dữ liệu):** Chứa các lớp tương tác trực tiếp với database thông qua Entity Framework Core. Mỗi entity tương ứng với một bảng trong database.

* + 1. **Mô hình Entity trong Entity Framework Core**
       1. **Context**
    - **Lớp ElectronicsStoreContext**

Lớp này đại diện cho lớp trung tâm giao tiếp với CSDL, quản lý entity và ánh xạ entity <-> bảng.

Khi EF Core thực hiện lệnh Add-Migration, nó sẽ đọc thông tin từ DbSet để sinh ra file migration.

Khi thực hiện Update-Database, EF Core sẽ dùng ElectronicsStoreContext để kết nối tới database và thực thi SQL tương ứng từ migration đã tạo.

public class ElectronicsStoreContext : DbContext

{

public DbSet<Categories> Category { get; set; }

public DbSet<Manufacturers> Manufacturer { get; set; }

public DbSet<Products> Product { get; set; }

public DbSet<Employees> Employee { get; set; }

public DbSet<Customers> Customer { get; set; }

public DbSet<Orders> Order { get; set; }

public DbSet<Order\_Details> Order\_Details { get; set; }

public ElectronicsStoreContext() { }

public ElectronicsStoreContext(DbContextOptions<ElectronicsStoreContext> options)

: base(options)

{

}

// Nếu muốn giữ OnConfiguring cho chạy thực tế

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Data Source=.;Database=ElectronsStore;Integrated Security=True;MultipleActiveResultSets=True;TrustServerCertificate=True");

}

}

}

Lệnh khởi tạo CSDL sử dụng Entity Framework Core Migration:

**Add-Migration KhoiTaoCSDL**

**Update-Database**

Add-Migration KhoiTaoCSDL: tạo file migration từ các entity đã định nghĩa.

Update-Database: áp dụng migration, tạo cơ sở dữ liệu thực tế. Điều này cho phép xây dựng CSDL đồng bộ với code, dễ bảo trì, hỗ trợ rollback khi cần.

* + - **Lớp ElectronicsStoreContextFactory**

Lớp này triển khai interface IDesignTimeDbContextFactory, nghĩa là nó được gọi tự động bởi EF Core trong thời điểm "design-time", như khi bạn chạy lệnh: Add-Migration, sau đó chạy lệnh: Update-Database

Mục đích: Giúp EF Core biết tạo ra một đối tượng DbContext như thế nào nếu chương trình chính không cung cấp được đối tượng DbContext.

Điều này cực kỳ quan trọng khi bạn chạy các lệnh migration trong dự án WinForms hoặc Console app, vì chúng không sử dụng DI như ASP.NET Core.

public class ElectronicsStoreContextFactory : IDesignTimeDbContextFactory<ElectronicsStoreContext>

{

public ElectronicsStoreContext CreateDbContext(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<ElectronicsStoreContext>();

optionsBuilder.UseSqlServer(

"Data Source=.;Database=ElectronsStore;Integrated Security=True;MultipleActiveResultSets=True;TrustServerCertificate=True"

);

return new ElectronicsStoreContext(optionsBuilder.Options);

}

}

* + - 1. **Categories**

Mô tả thực thể : lớp Categories đại diện cho bảng Categories trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - CategoryName: tên danh mục.
    - Products: danh sách các sản phẩm liên quan (quan hệ 1-nhiều).

Mối quan hệ**:** Một Category có nhiều Products (1 - N).

Repository và xử lýGiao diện ICategoryRepository định nghĩa các thao tác: GetAll, GetById, Add, Update, Delete.Lớp CategoryRepository cài đặt các thao tác này, tương tác với ElectronicsStoreContext.

public class Categories

{

public int ID { get; set; }

public string CategoryName { get; set; } = string.Empty;

public virtual ObservableCollectionListSource<Products> Products { get; } = new();

}

public class CategoryRepository : ICategoryRepository

{

private readonly ElectronicsStoreContext \_context;

public CategoryRepository()

{

\_context = new ElectronicsStoreContext();

}  
 //Các phương thức khác của class…

}

List<Categories> GetAll();

Categories? GetById(int id);

void Add(Categories category);

void Update(Categories category);

void Delete(Categories category);

* + - 1. **Manufactueres**

Mô tả thực thể: lớp Manufacturers đại diện cho bảng Manufacturers trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - ManufacturerName: tên nhà sản xuất.
    - ManufacturerAddress: địa chỉ nhà sản xuất (tùy chọn).
    - ManufacturerPhone: số điện thoại nhà sản xuất (tùy chọn).
    - ManufacturerEmail: email nhà sản xuất (tùy chọn).
    - Product: danh sách các sản phẩm do nhà sản xuất này cung cấp (quan hệ 1 - N).

Mối quan hệ: một Manufacturer có thể có nhiều Products (1 - N).

Repository và xử lý. Giao diện IManufacturerRepository định nghĩa các thao tác:GetAll(),GetById(int id), Add(Manufacturers manufacturer), Update(Manufacturers manufacturer), Delete(Manufacturers manufacturer) .Lớp ManufacturerRepository cài đặt các phương thức trên, thực hiện các thao tác truy vấn, thêm, sửa, xóa từ cơ sở dữ liệu thông qua ElectronicsStoreContext.

public class Manufacturers

{

public int ID { get; set; }

public string ManufacturerName { get; set; }

public string? ManufacturerAddress { get; set; }

public string? ManufacturerPhone { get; set; }

public string? ManufacturerEmail { get; set; }

public virtual ObservableCollectionListSource<Products> Product { get; } = new();

public class ManufacturerRepository: IManufacturerRepository

{

private readonly ElectronicsStoreContext \_context;

public ManufacturerRepository()

{

\_context = new ElectronicsStoreContext();

}

//Các phương thức khác của class…

}

public interface IManufacturerRepository

{

List<Manufacturers> GetAll();

Manufacturers? GetById(int id);

void Add(Manufacturers category);

void Update(Manufacturers category);

void Delete(Manufacturers manufacturer);

}

* + - 1. **Customers**

Mô tả thực thể: Lớp Customers đại diện cho bảng Customers trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - CustomerName: tên khách hàng.
    - CustomerAddress, CustomerPhone, CustomerEmail: thông tin liên lạc.

Order: danh sách các đơn hàng liên quan (quan hệ 1-nhiều).

Mối quan hệ: Một Customer có thể có nhiều Orders (1 - N). Repository và xử lý: ICustomerRepository định nghĩa các thao tác: GetAll, GetById, Add, Update, Delete. CustomerRepository cài đặt các thao tác trên và tương tác với ElectronicsStoreContext.

public class Customers

{

public int ID { get; set; }

public string CustomerName { get; set; }

public string? CustomerAddress { get; set; }

public string? CustomerPhone { get; set; }

public string? CustomerEmail { get; set; }

public virtual ObservableCollectionListSource<Orders> Order { get; } = new();

}

public class CustomerRepository : ICustomerRepository

{

private readonly ElectronicsStoreContext \_context;

public CustomerRepository()

{

\_context = new ElectronicsStoreContext();

}

//Các phương thức khác của class…

}

public interface ICustomerRepository

{

List<Customers> GetAll();

Customers? GetById(int id);

void Add(Customers customer);

void Update(Customers customer);

void Delete(Customers customer);

}

* + - 1. **Employees**

Mô tả thực thể: Lớp Employees đại diện cho bảng Employees trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - FullName, EmployeePhone, EmployeeAddress: thông tin cá nhân.
    - UserName, Password, Role: tài khoản đăng nhập và quyền.

Order: danh sách đơn hàng đã xử lý (quan hệ 1-nhiều).

Mối quan hệ: Một Employee có thể xử lý nhiều Orders (1 - N). Repository và xử lý: IEmployeeRepository định nghĩa: GetAll, GetById, GetByUserName, Add, Update, UpdatePassword, Delete. EmployeeRepository cài đặt các thao tác này, làm việc với ElectronicsStoreContext.

public class Employees

{

public int ID { get; set; }

public string FullName { get; set; }

public string? EmployeePhone { get; set; }

public string? EmployeeAddress { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

public bool Role { get; set; }

public virtual ObservableCollectionListSource<Orders> Order { get; } = new();

}

public class EmployeeRepository: IEmployeeRepository

{

private readonly ElectronicsStoreContext \_context;

public EmployeeRepository()

{

\_context = new ElectronicsStoreContext();

}

}

public interface IEmployeeRepository

{

List<Employees> GetAll();

Employees? GetById(int id);

Employees? GetbyUserName(string userName);

void Add(Employees employee);

void Update(Employees employee);

void UpdatePassword(int id, string hashedPassword);

void Delete(Employees employee);

}

* + - 1. **Products**

Mô tả thực thể: lớp Products đại diện cho bảng Products trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - ManufacturerID, CategoryID: liên kết đến nhà sản xuất và danh mục.
    - ProductName, Price, Quantity, Image, Description: thông tin sản phẩm.

Quan hệ: Nhiều sản phẩm thuộc một Category (N - 1). Nhiều sản phẩm thuộc một Manufacturer (N - 1). Một sản phẩm có thể có nhiều Order\_Details (1 - N).

Repository và xử lý: IProductRepository định nghĩa: GetAll, GetById, Add, Update, UpdateImage, Delete, GetAllWithCategoryManufacturer. ProductRepository triển khai các thao tác trên thông qua ElectronicsStoreContext.

public class Products

{

public int ID { get; set; }

public int ManufacturerID { get; set; }

public int CategoryID { get; set; }

public string ProductName { get; set; }

public int Price { get; set; }

public int Quantity { get; set; }

public string? Image { get; set; }

public string? Description { get; set; }

public virtual ObservableCollectionListSource<Order\_Details> Order\_Details { get; } = new();

public virtual Categories Category { get; set; } = null!;

public virtual Manufacturers Manufacturer { get; set; } = null!;

}

public class ProductRepository : IProductRepository

{

private readonly ElectronicsStoreContext \_context;

public ProductRepository()

{

\_context = new ElectronicsStoreContext();

}

}  
public interface IProductRepository

{

List<Products> GetAll();

Products? GetById(int id);

void Add(Products product);

void Update(Products product);

void UpdateImage(int productId, string imageFileName);

void Delete(Products product);

List<Products> GetAllWithCategoryManufacturer();

}

* + - 1. **Orders**

Mô tả thực thể: lớp Orders đại diện cho bảng Orders trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - EmployeeID, CustomerID: liên kết đến nhân viên và khách hàng.
    - Date, Note: ngày đặt và ghi chú đơn hàng.
    - ViewDetails: danh sách các chi tiết hóa đơn (1 - N).

Quan hệ: Một Order thuộc một Customer (N - 1). Một Order thuộc một Employee (N - 1).

Repository và xử lý: IOrderRepository định nghĩa: GetAll, GetAllWithDetails, GetById, Add, Insert, Update, Delete. OrderRepository cài đặt các phương thức và làm việc với ElectronicsStoreContext.

public class Orders

{

public int ID { get; set; }

public int EmployeeID { get; set; }

public int CustomerID { get; set; }

public DateTime Date { get; set; }

public string? Note { get; set; }

public virtual ObservableCollectionListSource<Order\_Details> ViewDetails { get; } = new();

public virtual Customers Customer { get; set; } = null!;

public virtual Employees Employee { get; set; } = null!;

}

public class OrderRepository : IOrderRepository

{

private readonly ElectronicsStoreContext \_context;

public OrderRepository()

{

\_context = new ElectronicsStoreContext();

}

}

public interface IOrderRepository

{

List<Orders> GetAllWithDetails();

List<Orders> GetAll();

Orders? GetById(int id);

void Add(Orders entity);

int Insert(Orders order);

void Update(Orders order);

void Delete(Orders order);

}

* + - 1. **OrderDetails**

Mô tả thực thể: lớp Order\_Details đại diện cho bảng Order\_Details trong cơ sở dữ liệu.

Thuộc tính chính:

* + - ID: khóa chính, tự tăng.
    - OrderID, ProductID: liên kết đến đơn hàng và sản phẩm.
    - Quantity, Price: số lượng và đơn giá tại thời điểm đặt hàng.

Quan hệ: Một Order\_Detail thuộc một Order (N - 1). Một Order\_Detail thuộc một Product (N - 1).

Repository và xử lý: IOrderDetailsRepository định nghĩa: GetByOrderID, DeleteByOrderID, AddRange, Insert. OrderDetailsRepository triển khai các phương thức này tương tác với ElectronicsStoreContext.

public class Order\_Details

{

public int ID { get; set; }

public int OrderID { get; set; }

public int ProductID { get; set; }

public short Quantity { get; set; }

public int Price { get; set; }

public virtual Orders Order { get; set; } = null!;

public virtual Products Product { get; set; } = null!;

}

public class OrderDetailsRepository : IOrderDetailsRepository

{

private readonly ElectronicsStoreContext \_context;

public OrderDetailsRepository()

{

\_context = new ElectronicsStoreContext();

}

}

public interface IOrderDetailsRepository

{

List<Order\_Details> GetByOrderID(int orderId);

void DeleteByOrderID(int orderId);

void AddRange(List<Order\_Details> details);

void Insert(Order\_Details detail);

}

* 1. **THIẾT KẾ XỬ LÝ**
     1. **Luồng xử lý đăng nhập**

Mục đích: xác thực người dùng (nhân viên) khi đăng nhập vào hệ thống.

Luồng xử lý:

1. Người dùng nhập UserName và Password tại form đăng nhập.
2. Form gửi thông tin cho EmployeeService (BLL).
3. EmployeeService gọi EmployeeRepository.GetbyUserName(string userName) để lấy thông tin nhân viên.
4. Nếu tồn tại UserName, kiểm tra mật khẩu bằng cách so sánh với mật khẩu đã mã hóa (dùng BCrypt).
5. Nếu đúng, trả về đối tượng Employees, cho phép đăng nhập; nếu sai, trả về thông báo lỗi.

Sơ đồ luồng xử lý:

* + 1. **Luồng xử lý quản lý sản phẩm**

Chức năng bao gồm: Thêm, sửa, xóa, hiển thị danh sách sản phẩm, bao gồm cả hình ảnh, nhà sản xuất, danh mục.

1. Thêm sản phẩm:

Người dùng nhập thông tin sản phẩm và chọn ảnh.

Giao diện gọi ProductService.Add(ProductsDTO productDto) để kiểm tra dữ liệu.

Nếu hợp lệ, ProductService ánh xạ sang entity Products và gọi ProductRepository.Add(product). Lưu ảnh nếu có, và hiển thị sản phẩm mới.

1. Sửa sản phẩm:

Người dùng chọn sản phẩm cần sửa. Dữ liệu được load lên form, cho phép chỉnh sửa. Gọi ProductService.Update(productDto) để xử lý cập nhật. Nếu có ảnh mới, gọi UpdateImage(productId, fileName).

1. Xóa sản phẩm:

Giao diện xác nhận xóa sản phẩm.

Gọi ProductService.Delete(productId) → ProductRepository.Delete.

* + 1. **Luồng xử lý đặt hàng (Tạo hóa đơn)**

Quy trình đặt hàng:

1. Tại giao diện frmOrderDetails, người dùng chọn khách hàng cũ hoặc thêm khách hàng mới, chọn nhân viên và thêm các sản phẩm.
2. Nhấn nút “Lưu hóa đơn”:
   * + Gọi OrderService.Insert(OrderDTO orderDto):
     + Kiểm tra hợp lệ (sản phẩm còn hàng? số lượng hợp lệ?)
     + Tạo Orders → OrderRepository.Insert(order) → nhận lại OrderID.
     + Gọi OrderDetailsService.AddRange(List<Order\_Details>) để thêm chi tiết hóa đơn.
     + Cập nhật lại số lượng tồn kho sản phẩm.
     1. **Luồng xử lý thống kê hóa đơn**
3. Người dùng chọn khoảng thời gian cần thống kê tại frmReport.
4. Giao diện gọi OrderService.GetAllWithDetails() hoặc theo điều kiện lọc.
5. Dữ liệu trả về danh sách đơn hàng kèm theo chi tiết.
6. Gửi dữ liệu vào ReportViewer, sử dụng mẫu rdlc để hiển thị thống kê doanh thu, đơn hàng, sản phẩm đã bán.
   * 1. **Luồng xử lý quản lý khách hàng**
7. Thêm khách hàng mới: nhập thông tin tại form → CustomerService.Add() → CustomerRepository.Add().
8. Cập nhật thông tin: tải dữ liệu lên form → sửa → Update().
9. Xóa khách hàng: Delete() (nếu chưa liên quan hóa đơn).
   * 1. **Luồng xử lý quản lý nhân viên**
10. Thêm nhân viên mới: nhập thông tin tại form → CustomerService.Add() → CustomerRepository.Add().
11. Cập nhật thông tin: tải dữ liệu lên form → sửa → Update().
12. Xóa nhân viên: Delete() (nếu chưa liên quan hóa đơn).
13. Cập nhật mật khẩu: gọi UpdatePassword(id, hashedPassword) tại EmployeeRepository.
14. **TỔNG KẾT**
    1. **KẾT QUẢ ĐẠT ĐƯỢC**

Sau quá trình nghiên cứu, thiết kế và xây dựng hệ thống quản lý cửa hàng điện tử, nhóm (hoặc cá nhân) chúng em đã hoàn thành được nhiều nội dung quan trọng cả về mặt lý thuyết lẫn thực hành, cụ thể như sau:

* + 1. **Hiểu rõ kiến trúc và quy trình phát triển phần mềm quản lý**

Trong quá trình thực hiện đồ án, em đã có cơ hội nghiên cứu và áp dụng mô hình kiến trúc phần mềm 3 lớp (Three-Tier Architecture) bao gồm: Presentation Layer (UI), Business Logic Layer (BLL) và Data Access Layer (DAL). Kiến trúc này giúp tách biệt rõ ràng các vai trò trong phần mềm, hỗ trợ việc bảo trì, mở rộng và tái sử dụng mã nguồn trong tương lai.

Bên cạnh đó, việc sử dụng công cụ ORM là Entity Framework Core đã giúp giảm bớt công việc thao tác trực tiếp với cơ sở dữ liệu, hỗ trợ mạnh trong việc ánh xạ các bảng sang class mô hình (entities) một cách tự động và rõ ràng.

* + 1. **Xây dựng hoàn chỉnh phần mềm quản lý bán hàng điện tử**

Phần mềm được thiết kế và cài đặt với đầy đủ các chức năng cơ bản và cần thiết cho một cửa hàng điện tử, bao gồm:

* Quản lý danh mục sản phẩm: sản phẩm, loại sản phẩm, nhà sản xuất;
* Quản lý khách hàng, nhân viên;
* Quản lý đơn hàng và chi tiết đơn hàng;
* Báo cáo thống kê doanh thu, sản phẩm bán;
* Chức năng đăng nhập, phân quyền nhân viên theo vai trò (admin/nhân viên).

Giao diện phần mềm được xây dựng bằng WinForms (.NET 8), thân thiện với người dùng, dễ sử dụng và có tính trực quan cao. Mọi thao tác đều được kiểm tra, xác thực dữ liệu nhằm đảm bảo tính chính xác trong quá trình nhập liệu.

* + 1. **Nâng cao kỹ năng chuyên môn và làm việc thực tế**

Thông qua đồ án này, chúng em đã nâng cao kỹ năng lập trình với C#, .NET Core, SQL Server, cũng như rèn luyện kỹ năng thiết kế cơ sở dữ liệu chuẩn hóa, tổ chức project theo mô hình thực tế. Ngoài ra, việc làm quen với AutoMapper, DTO, và thao tác báo cáo bằng RDLC Report Designer giúp mở rộng tư duy và khả năng phát triển ứng dụng thực tế theo hướng chuyên nghiệp. Kỹ năng làm việc độc lập, tư duy hệ thống, tổ chức công việc và kiểm thử phần mềm cũng được cải thiện rõ rệt sau thời gian thực hiện đồ án.

* 1. **HƯỚNG PHÁT TRIỂN**

Mặc dù phần mềm đã hoàn thiện ở mức cơ bản, tuy nhiên để đáp ứng tốt hơn các yêu cầu thực tế tại các cửa hàng điện tử hoặc mở rộng quy mô doanh nghiệp, hệ thống vẫn còn nhiều tiềm năng để phát triển thêm. Một số hướng phát triển tiêu biểu bao gồm:

* + 1. **Phát triển thành phần Web hoặc Mobile**

Trong bối cảnh chuyển đổi số ngày càng mạnh mẽ, việc mở rộng hệ thống sang nền tảng Web (ASP.NET Core Web App hoặc Web API) hoặc ứng dụng Mobile (Xamarin/Maui) sẽ giúp khách hàng có thể đặt hàng online, nhân viên có thể quản lý từ xa, từ đó nâng cao trải nghiệm và hiệu quả vận hành.

* + 1. **Mở rộng chức năng nâng cao**

Trong các phiên bản tiếp theo, phần mềm có thể được bổ sung nhiều chức năng nâng cao như:

* Tích hợp QR code/barcode để hỗ trợ bán hàng nhanh;
* Gửi email xác nhận đơn hàng hoặc thông báo khuyến mãi cho khách hàng;
* Xây dựng chức năng Log, ghi nhật ký.
* Xử lý trạng thái hóa đơn. Như đang sẵn sàng, hoàn tất, đã hủy…
* Thống kê thông minh như: biểu đồ doanh thu, so sánh theo kỳ, gợi ý hàng tồn, hàng bán chạy.
  + 1. **Kết nối đa người dùng**

Hiện tại phần mềm chủ yếu hoạt động dưới môi trường đơn máy hoặc mạng LAN đơn giản. Trong tương lai, phần mềm có thể được triển khai client-server nhiều người dùng đồng thời, có cơ chế phân quyền chi tiết hơn như: chỉ cho phép nhân viên bán hàng xem, nhưng không sửa đơn hàng cũ; admin mới có quyền thống kê, xuất báo cáo, v.v.

Ngoài ra, có thể tích hợp tính năng ghi log thao tác, phục vụ cho việc theo dõi lịch sử và bảo mật hệ thống.

* 1. **KẾT LUẬN CHUNG**

Đồ án đã giúp em có cái nhìn thực tế hơn về quy trình xây dựng phần mềm quản lý trong môi trường doanh nghiệp. Với kiến thức tích lũy được từ trường học, kết hợp với khả năng tự nghiên cứu, chúng em đã xây dựng được một hệ thống quản lý cửa hàng điện tử có tính ứng dụng cao, đúng chuẩn kỹ thuật, và có tiềm năng phát triển trong tương lai.

Em chân thành cảm ơn quý thầy cô đã tận tình hướng dẫn trong suốt quá trình thực hiện đồ án. Đồng thời, chúng em cũng mong muốn nhận được các góp ý từ thầy cô và hội đồng để có thể hoàn thiện sản phẩm ngày một tốt hơn.