**CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG**

**1. Đặc tả đề tài**

**1.1 Mô tả bài toán**

**1. Tổng quan hệ thống** Hệ thống quản lý cửa hàng điện thoại được thiết kế để hỗ trợ các hoạt động kinh doanh tại một cửa hàng bán lẻ các sản phẩm công nghệ như điện thoại thông minh, máy tính bảng, phụ kiện và đồng hồ thông minh. Được xây dựng trên nền tảng SQL Server, hệ thống cung cấp một giải pháp toàn diện để quản lý thông tin về nhân viên, khách hàng, sản phẩm, nhà cung cấp, đơn hàng, khuyến mãi, tồn kho và thanh toán. Với các tính năng như phân quyền người dùng, tự động hóa quy trình và báo cáo hiệu suất, hệ thống đảm bảo tính bảo mật, hiệu quả và khả năng mở rộng, đáp ứng nhu cầu vận hành của cửa hàng một cách linh hoạt và tối ưu.

**2. Quản lý nhân viên và phân quyền** Hệ thống cho phép quản lý thông tin nhân viên, bao gồm các chi tiết như mã định danh, họ tên, số điện thoại, email, địa chỉ, vị trí công việc, ngày tuyển dụng và mức lương. Mỗi nhân viên được liên kết với một tài khoản người dùng, sử dụng mật khẩu mã hóa an toàn để đăng nhập vào hệ thống. Các vai trò như quản trị viên, quản lý, thu ngân, nhân viên bán hàng và nhân viên cơ bản được xác định rõ ràng, với mỗi vai trò được gán các quyền truy cập cụ thể để thực hiện các tác vụ phù hợp. Ví dụ, quản trị viên có toàn quyền quản lý, trong khi thu ngân chỉ xử lý các giao dịch bán hàng. Hệ thống tự động tạo tài khoản người dùng và gán vai trò tương ứng khi thêm nhân viên mới, đảm bảo kiểm soát chặt chẽ và tăng cường bảo mật.

**3. Quản lý khách hàng và nhà cung cấp** Chức năng quản lý khách hàng cho phép lưu trữ thông tin như mã định danh, họ tên, số điện thoại, email, địa chỉ, ngày sinh, giới tính, loại khách hàng (thường hoặc VIP) và tổng chi tiêu. Hệ thống tự động nâng cấp khách hàng thành VIP khi tổng chi tiêu vượt một ngưỡng nhất định, chẳng hạn 50 triệu đồng, giúp cá nhân hóa trải nghiệm mua sắm. Đối với nhà cung cấp, hệ thống quản lý các thông tin như mã định danh, tên công ty, người liên hệ, số điện thoại, email, địa chỉ và mã số thuế. Chức năng này hỗ trợ theo dõi mối quan hệ với khách hàng và nhà cung cấp, đảm bảo quản lý đơn hàng và nhập kho diễn ra hiệu quả, đồng thời duy trì nguồn cung ổn định.

**4. Quản lý sản phẩm và tồn kho** Hệ thống quản lý thông tin sản phẩm, bao gồm mã định danh, tên, danh mục, thương hiệu, nhà cung cấp, mô tả, thông số kỹ thuật, giá vốn, giá bán, thời gian bảo hành và trạng thái (hoạt động, không hoạt động hoặc ngừng kinh doanh). Sản phẩm được phân loại theo danh mục như điện thoại, phụ kiện, máy tính bảng, đồng hồ thông minh và theo thương hiệu như Apple, Samsung, Xiaomi, giúp dễ dàng tìm kiếm và tư vấn khách hàng. Chức năng quản lý tồn kho theo dõi số lượng sản phẩm có sẵn, số lượng đã đặt và các ngưỡng tồn kho tối thiểu/tối đa. Hệ thống tự động cập nhật số liệu tồn kho khi có đơn hàng mới hoặc hủy đơn, đảm bảo độ chính xác và hỗ trợ cảnh báo khi hàng hóa sắp hết, giúp cửa hàng duy trì cung ứng kịp thời.

**5. Quản lý đơn hàng và thanh toán**

Hệ thống cung cấp một cơ chế quản lý đơn hàng toàn diện, ghi nhận đầy đủ thông tin về các giao dịch mua bán, bao gồm mã đơn hàng, thông tin khách hàng, nhân viên phụ trách, ngày đặt hàng, tổng số tiền, chiết khấu, thuế, phương thức thanh toán và trạng thái đơn hàng (như đang xử lý, hoàn thành, hủy hoặc trả hàng). Mỗi đơn hàng được chi tiết hóa với các mục sản phẩm, số lượng, giá bán tại thời điểm mua, chiết khấu áp dụng và thông tin bảo hành. Hệ thống cũng quản lý các giao dịch thanh toán, lưu trữ chi tiết về phương thức thanh toán (tiền mặt, thẻ, chuyển khoản hoặc trả góp), số tiền, mã tham chiếu và trạng thái thanh toán. Các quy trình tự động được tích hợp để tính toán tổng tiền cho từng mục trong đơn hàng, cập nhật thời gian bảo hành dựa trên chính sách sản phẩm, và điều chỉnh trạng thái thanh toán của đơn hàng dựa trên tổng số tiền đã nhận, đảm bảo tính chính xác và giảm thiểu thao tác thủ công.

**6. Quản lý khuyến mãi**

Hệ thống hỗ trợ quản lý các chương trình khuyến mãi để thu hút khách hàng, bao gồm thông tin như mã khuyến mãi, tên chương trình, mô tả, loại giảm giá (theo phần trăm hoặc số tiền cố định), giá trị giảm, ngày bắt đầu, ngày kết thúc, số tiền đơn hàng tối thiểu để áp dụng, giới hạn số lần sử dụng và trạng thái hoạt động. Hệ thống tự động theo dõi số lần sử dụng khuyến mãi mỗi khi được áp dụng vào đơn hàng và kiểm tra tính hợp lệ dựa trên thời gian hiệu lực và giới hạn sử dụng. Các chức năng hỗ trợ thêm, cập nhật, xóa và truy xuất thông tin khuyến mãi được tích hợp, giúp doanh nghiệp dễ dàng quản lý các chương trình ưu đãi, từ đó tăng cường trải nghiệm mua sắm và tối ưu hóa doanh thu.

**7. Báo cáo và phân tích**

Hệ thống cung cấp các công cụ báo cáo và phân tích để hỗ trợ chủ doanh nghiệp và quản lý đưa ra quyết định kinh doanh hiệu quả. Các báo cáo bao gồm thống kê doanh thu theo ngày, danh sách các sản phẩm bán chạy, đánh giá hiệu suất làm việc của nhân viên và phân tích lợi nhuận theo từng sản phẩm. Một dashboard tổng quan hiển thị các số liệu quan trọng như doanh thu trong ngày, doanh thu tháng, tổng số đơn hàng và số lượng sản phẩm có tồn kho thấp, giúp theo dõi tình hình kinh doanh tức thời. Ngoài ra, hệ thống cung cấp khả năng truy xuất nhanh thông tin về các sản phẩm đang kinh doanh và các chương trình khuyến mãi còn hiệu lực, hỗ trợ nhân viên và quản lý trong việc tư vấn khách hàng và lập kế hoạch kinh doanh.

**8. Phân quyền chi tiết** Hệ thống thiết lập cơ chế phân quyền chặt chẽ để đảm bảo an toàn và phù hợp với vai trò của từng người dùng:

* **Admin**: Có toàn quyền quản lý mọi khía cạnh của hệ thống, từ sản phẩm, đơn hàng, khuyến mãi, tồn kho đến báo cáo và phân quyền người dùng.
* **Manager**: Có quyền truy cập, thêm và chỉnh sửa thông tin liên quan đến khách hàng, đơn hàng, khuyến mãi và tồn kho, đồng thời được phép xem các báo cáo để hỗ trợ quản lý hoạt động cửa hàng.
* **Cashier**: Được cấp quyền thực hiện các tác vụ bán hàng như thêm thông tin khách hàng, tạo đơn hàng, ghi nhận chi tiết đơn hàng và xử lý thanh toán.
* **Salesperson**: Kế thừa quyền của thu ngân, bổ sung khả năng truy cập báo cáo doanh thu và danh sách sản phẩm bán chạy để hỗ trợ tư vấn khách hàng.
* **Staff**: Chỉ được phép xem thông tin cơ bản và thực hiện một số tác vụ giới hạn, đảm bảo hỗ trợ các công việc phụ trợ mà không ảnh hưởng đến dữ liệu quan trọng.

**1.2 Mô tả dữ liệu**

**- Mô tả các tập thực thể**

**Nhân viên** Tập thực thể Nhân viên đại diện cho những người làm việc tại cửa hàng, bao gồm các vai trò như quản lý, thu ngân, nhân viên bán hàng và nhân viên cơ bản. Mỗi nhân viên được xác định bởi mã nhân viên duy nhất, kèm theo thông tin như họ tên, số điện thoại, email, địa chỉ, vị trí công việc, ngày tuyển dụng, mức lương, trạng thái (đang làm việc hoặc không hoạt động), cùng thời gian tạo và cập nhật thông tin. Nhân viên đóng vai trò quan trọng trong việc thực hiện các tác vụ như bán hàng, xử lý đơn hàng, quản lý tồn kho hoặc hỗ trợ khách hàng, tùy thuộc vào quyền hạn được phân công.

**Người dùng** Tập thực thể Người dùng đại diện cho tài khoản đăng nhập của nhân viên trong hệ thống, dùng để truy cập các chức năng được phân quyền. Mỗi người dùng được xác định bởi tên đăng nhập duy nhất, mật khẩu (được mã hóa để đảm bảo an toàn), liên kết với một nhân viên cụ thể, trạng thái tài khoản (hoạt động hoặc không hoạt động), thời gian đăng nhập cuối cùng, và thời gian tạo hoặc cập nhật. Thực thể này giúp quản lý quyền truy cập, đảm bảo nhân viên chỉ thực hiện các thao tác phù hợp với vai trò của họ.

**Vai trò** Tập thực thể Vai trò xác định các cấp độ quyền hạn trong hệ thống, chẳng hạn như Admin, Manager, Cashier, Salesperson hoặc Staff. Mỗi vai trò được xác định bởi tên vai trò duy nhất, mô tả chi tiết về quyền hạn, cùng thời gian tạo và cập nhật. Vai trò đóng vai trò trung tâm trong việc phân quyền, giúp hệ thống kiểm soát chặt chẽ các thao tác mà người dùng có thể thực hiện, đảm bảo an toàn và hiệu quả.

**Khách hàng** Tập thực thể Khách hàng đại diện cho những người mua sắm tại cửa hàng, bao gồm khách hàng thường và khách hàng VIP. Thông tin khách hàng bao gồm mã khách hàng duy nhất, họ tên, số điện thoại (duy nhất), email, địa chỉ, ngày sinh, giới tính (nam, nữ, khác), loại khách hàng, tổng chi tiêu, và thời gian tạo hoặc cập nhật. Thực thể này hỗ trợ theo dõi lịch sử mua sắm, cung cấp ưu đãi cá nhân hóa và nâng cấp trạng thái khách hàng dựa trên chi tiêu.

**Nhà cung cấp** Tập thực thể Nhà cung cấp đại diện cho các công ty hoặc tổ chức cung cấp sản phẩm cho cửa hàng. Mỗi nhà cung cấp được xác định bởi mã nhà cung cấp duy nhất, tên công ty, thông tin người liên hệ, số điện thoại, email, địa chỉ, mã số thuế, trạng thái (hoạt động hoặc không hoạt động), và thời gian tạo hoặc cập nhật. Nhà cung cấp đảm bảo nguồn hàng ổn định, hỗ trợ quản lý nhập kho và kiểm soát chi phí.

**Danh mục sản phẩm** Tập thực thể Danh mục sản phẩm phân loại các sản phẩm được bán tại cửa hàng, như điện thoại, máy tính bảng, phụ kiện hoặc đồng hồ thông minh. Mỗi danh mục được xác định bởi tên danh mục duy nhất, mô tả chi tiết, trạng thái (hoạt động hoặc không hoạt động), và thời gian tạo hoặc cập nhật. Danh mục giúp tổ chức sản phẩm, hỗ trợ tìm kiếm và quản lý hàng hóa hiệu quả.

**Thương hiệu** Tập thực thể Thương hiệu đại diện cho các nhãn hiệu sản phẩm, như Apple, Samsung, Xiaomi. Mỗi thương hiệu bao gồm tên thương hiệu duy nhất, quốc gia xuất xứ, mô tả, URL logo, trạng thái (hoạt động hoặc không hoạt động), và thời gian tạo hoặc cập nhật. Thương hiệu giúp phân loại sản phẩm theo nhãn hiệu, tạo sự nhận diện và hỗ trợ tư vấn khách hàng.

**Sản phẩm** Tập thực thể Sản phẩm đại diện cho các mặt hàng được bán, như điện thoại, phụ kiện, máy tính bảng hoặc đồng hồ thông minh. Mỗi sản phẩm được xác định bởi mã sản phẩm duy nhất, tên, mô tả, thông số kỹ thuật (dạng chuỗi JSON), giá vốn, giá bán, thời gian bảo hành, URL hình ảnh, trạng thái (hoạt động, không hoạt động, ngừng kinh doanh), và thời gian tạo hoặc cập nhật. Sản phẩm là trung tâm của các giao dịch mua bán và quản lý tồn kho.

**Tồn kho** Tập thực thể Tồn kho theo dõi số lượng sản phẩm có sẵn tại cửa hàng. Thông tin bao gồm số lượng sản phẩm có sẵn, số lượng đã đặt, mức tồn kho tối thiểu và tối đa, cùng thời gian cập nhật gần nhất. Thực thể này đảm bảo cửa hàng luôn có đủ hàng hóa để đáp ứng nhu cầu khách hàng, đồng thời cảnh báo khi tồn kho thấp.

**Khuyến mãi** Tập thực thể Khuyến mãi quản lý các chương trình ưu đãi để thu hút khách hàng. Mỗi khuyến mãi bao gồm mã khuyến mãi duy nhất, tên, mô tả, loại giảm giá (phần trăm hoặc số tiền cố định), giá trị giảm, ngày bắt đầu, ngày kết thúc, số tiền đơn hàng tối thiểu, giới hạn sử dụng, số lần đã sử dụng, trạng thái (hoạt động, không hoạt động, hết hạn), và thời gian tạo hoặc cập nhật. Khuyến mãi giúp tăng doanh thu và cải thiện trải nghiệm khách hàng.

**Đơn hàng** Tập thực thể Đơn hàng đại diện cho các giao dịch mua bán giữa khách hàng và cửa hàng. Mỗi đơn hàng bao gồm mã đơn hàng duy nhất, ngày đặt hàng, tổng tiền trước chiết khấu, chiết khấu, thuế, tổng tiền cuối cùng, phương thức thanh toán (tiền mặt, thẻ, chuyển khoản, trả góp), trạng thái thanh toán (chưa thanh toán, thanh toán một phần, đã thanh toán, hoàn tiền), trạng thái đơn hàng (đang xử lý, hoàn thành, hủy, trả hàng), ghi chú, và thời gian tạo hoặc cập nhật. Đơn hàng là trọng tâm của hoạt động bán hàng.

**Chi tiết đơn hàng** Tập thực thể Chi tiết đơn hàng mô tả các mục sản phẩm trong một đơn hàng. Mỗi mục bao gồm số lượng, giá bán tại thời điểm mua, chiết khấu từng mục, tổng tiền mục hàng, ngày bắt đầu và kết thúc bảo hành. Thực thể này cung cấp thông tin chi tiết để quản lý sản phẩm trong đơn hàng và hỗ trợ theo dõi bảo hành.

**Thanh toán** Tập thực thể Thanh toán ghi nhận các giao dịch thanh toán liên quan đến đơn hàng. Mỗi thanh toán bao gồm ngày thanh toán, phương thức thanh toán, số tiền, mã tham chiếu giao dịch, trạng thái (thành công, thất bại, đang chờ), ghi chú, và thời gian tạo. Thực thể này đảm bảo theo dõi và xác nhận các khoản thanh toán chính xác.

**- Mô tả các mối quan hệ**

**Nhân viên và Người dùng** Mỗi nhân viên được liên kết với một tài khoản người dùng duy nhất để đăng nhập vào hệ thống, tạo thành mối quan hệ một-đến-một (1-1). Điều này đảm bảo rằng mỗi nhân viên có một tài khoản riêng để thực hiện các tác vụ theo vai trò được phân quyền, như tạo đơn hàng hoặc quản lý tồn kho.

**Người dùng và Vai trò** Một người dùng có thể được gán một hoặc nhiều vai trò (như Admin, Manager, Cashier), và một vai trò có thể áp dụng cho nhiều người dùng, tạo thành mối quan hệ nhiều-đến-nhiều (N-N). Quan hệ này được quản lý thông qua một thực thể trung gian, cho phép phân quyền linh hoạt và kiểm soát chặt chẽ các thao tác trong hệ thống.

**Nhân viên và Đơn hàng** Một nhân viên có thể xử lý nhiều đơn hàng, nhưng mỗi đơn hàng chỉ được xử lý bởi một nhân viên, hình thành mối quan hệ một-đến-nhiều (1-N). Mối quan hệ này hỗ trợ theo dõi hiệu suất bán hàng của nhân viên và đảm bảo trách nhiệm rõ ràng trong quy trình bán hàng.

**Khách hàng và Đơn hàng** Một khách hàng có thể đặt nhiều đơn hàng, nhưng mỗi đơn hàng chỉ liên kết với một khách hàng, tạo thành mối quan hệ một-đến-nhiều (1-N). Quan hệ này giúp theo dõi lịch sử mua sắm, tổng chi tiêu và cung cấp cơ sở để phân loại khách hàng (thường hoặc VIP).

**Nhà cung cấp và Sản phẩm** Một nhà cung cấp có thể cung cấp nhiều sản phẩm, nhưng mỗi sản phẩm chỉ liên kết với một nhà cung cấp, hình thành mối quan hệ một-đến-nhiều (1-N). Điều này đảm bảo quản lý nguồn gốc sản phẩm và hỗ trợ quy trình nhập kho hiệu quả.

**Danh mục sản phẩm và Sản phẩm** Một danh mục sản phẩm (như điện thoại, phụ kiện) có thể chứa nhiều sản phẩm, nhưng mỗi sản phẩm chỉ thuộc một danh mục, tạo thành mối quan hệ một-đến-nhiều (1-N). Mối quan hệ này giúp tổ chức và tìm kiếm sản phẩm dễ dàng hơn.

**Thương hiệu và Sản phẩm** Một thương hiệu (như Apple, Samsung) có thể bao gồm nhiều sản phẩm, nhưng mỗi sản phẩm chỉ thuộc một thương hiệu, hình thành mối quan hệ một-đến-nhiều (1-N). Quan hệ này hỗ trợ phân loại sản phẩm theo thương hiệu, tăng tính nhận diện và tư vấn khách hàng.

**Sản phẩm và Tồn kho** Mỗi sản phẩm có một bản ghi tồn kho duy nhất để theo dõi số lượng có sẵn và đã đặt, tạo thành mối quan hệ một-đến-một (1-1). Mối quan hệ này đảm bảo quản lý tồn kho chính xác và cảnh báo kịp thời khi số lượng hàng hóa thấp.

**Sản phẩm và Chi tiết đơn hàng** Một sản phẩm có thể xuất hiện trong nhiều chi tiết đơn hàng, nhưng mỗi chi tiết đơn hàng chỉ liên kết với một sản phẩm, hình thành mối quan hệ một-đến-nhiều (1-N). Quan hệ này hỗ trợ theo dõi số lượng, giá bán và doanh thu của sản phẩm trong các giao dịch.

**Đơn hàng và Chi tiết đơn hàng** Một đơn hàng có thể bao gồm nhiều chi tiết đơn hàng, nhưng mỗi chi tiết đơn hàng chỉ thuộc về một đơn hàng, tạo thành mối quan hệ một-đến-nhiều (1-N). Mối quan hệ này cung cấp thông tin chi tiết về sản phẩm, số lượng và bảo hành trong đơn hàng.

**Đơn hàng và Thanh toán** Một đơn hàng có thể có nhiều giao dịch thanh toán, nhưng mỗi giao dịch thanh toán chỉ liên kết với một đơn hàng, hình thành mối quan hệ một-đến-nhiều (1-N). Quan hệ này giúp quản lý các khoản thanh toán và cập nhật trạng thái thanh toán của đơn hàng.

**Đơn hàng và Khuyến mãi** Một đơn hàng có thể áp dụng một chương trình khuyến mãi, và một khuyến mãi có thể được áp dụng cho nhiều đơn hàng, tạo thành mối quan hệ một-đến-nhiều (1-N). Mối quan hệ này hỗ trợ quản lý chiết khấu và theo dõi số lần sử dụng khuyến mãi.

**1.3 Mô tả chức năng**

**Quản lý nhân viên và phân quyền** Hệ thống cung cấp chức năng quản lý thông tin nhân viên, bao gồm việc lưu trữ và cập nhật các chi tiết như họ tên, số điện thoại, email, địa chỉ, vị trí công việc, ngày tuyển dụng và mức lương. Chức năng này cho phép chủ doanh nghiệp theo dõi trạng thái hoạt động của nhân viên và tự động tạo tài khoản người dùng khi thêm nhân viên mới, với mật khẩu được mã hóa để đảm bảo an toàn. Hệ thống hỗ trợ phân quyền chi tiết thông qua các vai trò như Admin, Manager, Cashier, Salesperson và Staff, mỗi vai trò được gán quyền truy cập và thao tác cụ thể. Ví dụ, Admin có toàn quyền quản lý, trong khi thu ngân chỉ được phép xử lý bán hàng và thanh toán. Chức năng này đảm bảo kiểm soát chặt chẽ các hoạt động, giảm thiểu rủi ro bảo mật và nâng cao hiệu quả quản lý nhân sự.

**Quản lý khách hàng** Chức năng quản lý khách hàng cho phép lưu trữ và theo dõi thông tin như họ tên, số điện thoại, email, địa chỉ, ngày sinh, giới tính và tổng chi tiêu. Hệ thống tự động phân loại khách hàng thành thường hoặc VIP dựa trên tổng chi tiêu, với ngưỡng tự động nâng cấp khi đạt mức quy định (ví dụ: 50 triệu đồng). Chức năng này hỗ trợ nhân viên tìm kiếm khách hàng theo số điện thoại, xem lịch sử mua sắm và áp dụng các ưu đãi cá nhân hóa. Khách hàng được hưởng trải nghiệm mua sắm minh bạch với thông tin rõ ràng về sản phẩm, giá cả và khuyến mãi, giúp tăng sự hài lòng và trung thành.

**Quản lý nhà cung cấp** Hệ thống cung cấp chức năng quản lý thông tin nhà cung cấp, bao gồm tên công ty, người liên hệ, số điện thoại, email, địa chỉ và mã số thuế. Chức năng này cho phép chủ doanh nghiệp theo dõi nguồn cung sản phẩm, đảm bảo nguồn hàng ổn định và quản lý chi phí nhập kho hiệu quả. Thông tin nhà cung cấp được liên kết với sản phẩm, giúp dễ dàng truy xuất nguồn gốc hàng hóa và hỗ trợ quy trình nhập hàng, từ đó duy trì chất lượng và số lượng sản phẩm đáp ứng nhu cầu kinh doanh.

**Quản lý sản phẩm và tồn kho** Chức năng quản lý sản phẩm cho phép lưu trữ thông tin chi tiết về các mặt hàng như điện thoại, phụ kiện, máy tính bảng và đồng hồ thông minh, bao gồm tên, danh mục, thương hiệu, thông số kỹ thuật, giá vốn, giá bán, thời gian bảo hành và hình ảnh. Hệ thống hỗ trợ phân loại sản phẩm theo danh mục (như điện thoại, phụ kiện) và thương hiệu (như Apple, Samsung), giúp nhân viên và khách hàng dễ dàng tìm kiếm. Chức năng quản lý tồn kho theo dõi số lượng sản phẩm có sẵn, số lượng đã đặt, cũng như các ngưỡng tồn kho tối thiểu và tối đa. Hệ thống tự động cập nhật tồn kho khi có đơn hàng mới hoặc hủy đơn, đồng thời cảnh báo khi số lượng hàng hóa thấp, đảm bảo cung ứng kịp thời.

**Quản lý đơn hàng và thanh toán** Hệ thống hỗ trợ quản lý đơn hàng bằng cách ghi nhận thông tin về mã đơn hàng, khách hàng, nhân viên phụ trách, ngày đặt hàng, tổng tiền, chiết khấu, thuế, phương thức thanh toán và trạng thái đơn hàng (đang xử lý, hoàn thành, hủy, trả hàng). Mỗi đơn hàng được chi tiết hóa với các mục sản phẩm, số lượng, giá bán, chiết khấu và thông tin bảo hành. Chức năng quản lý thanh toán cho phép ghi lại các giao dịch, bao gồm phương thức thanh toán (tiền mặt, thẻ, chuyển khoản, trả góp), số tiền, mã tham chiếu và trạng thái. Hệ thống tự động tính toán tổng tiền cho từng mục đơn hàng, cập nhật ngày bảo hành và trạng thái thanh toán dựa trên số tiền đã nhận, giảm thiểu lỗi thủ công và đảm bảo tính chính xác.

**Quản lý khuyến mãi** Chức năng quản lý khuyến mãi cho phép tạo, cập nhật và xóa các chương trình ưu đãi, bao gồm thông tin như mã khuyến mãi, tên, mô tả, loại giảm giá (phần trăm hoặc số tiền cố định), giá trị giảm, ngày bắt đầu, ngày kết thúc, số tiền đơn hàng tối thiểu và giới hạn sử dụng. Hệ thống tự động theo dõi số lần sử dụng khuyến mãi và kiểm tra tính hợp lệ dựa trên thời gian hiệu lực hoặc giới hạn sử dụng. Nhân viên có thể dễ dàng áp dụng khuyến mãi vào đơn hàng, giúp tăng doanh thu và cải thiện trải nghiệm mua sắm của khách hàng.

**Báo cáo và phân tích** Hệ thống cung cấp các công cụ báo cáo và phân tích để hỗ trợ ra quyết định kinh doanh. Chức năng báo cáo doanh thu theo ngày tổng hợp số đơn hàng, tổng doanh thu và giá trị trung bình đơn hàng. Báo cáo sản phẩm bán chạy liệt kê các mặt hàng có doanh số cao nhất, trong khi báo cáo hiệu suất nhân viên đánh giá số lượng đơn hàng và doanh thu mà mỗi nhân viên đạt được. Báo cáo lợi nhuận phân tích chênh lệch giữa giá vốn và giá bán để xác định hiệu quả kinh doanh. Ngoài ra, hệ thống cung cấp dashboard tổng quan hiển thị doanh thu hôm nay, doanh thu tháng, số đơn hàng và số lượng sản phẩm tồn kho thấp, giúp chủ doanh nghiệp và quản lý theo dõi tình hình kinh doanh tức thời. Các chức năng truy xuất nhanh thông tin sản phẩm và khuyến mãi đang hoạt động hỗ trợ tư vấn khách hàng hiệu quả.

**Phân quyền chi tiết** Hệ thống tích hợp chức năng phân quyền chặt chẽ để đảm bảo an toàn và phù hợp với vai trò của từng người dùng. Admin có toàn quyền quản lý mọi khía cạnh, từ sản phẩm, đơn hàng, khuyến mãi đến báo cáo. Manager được phép thêm, cập nhật thông tin khách hàng, đơn hàng, khuyến mãi và tồn kho, đồng thời truy cập các báo cáo để quản lý cửa hàng. Thu ngân (Cashier) thực hiện các tác vụ bán hàng như thêm khách hàng, tạo đơn hàng và xử lý thanh toán. Nhân viên bán hàng (Salesperson) kế thừa quyền của thu ngân, bổ sung khả năng xem báo cáo doanh thu và sản phẩm bán chạy để tư vấn khách hàng. Nhân viên cơ bản (Staff) chỉ có quyền xem thông tin cơ bản và thực hiện một số tác vụ giới hạn, đảm bảo không ảnh hưởng đến dữ liệu quan trọng. Chức năng này giúp kiểm soát quyền truy cập, nâng cao bảo mật và hiệu quả vận hành.

**Tự động hóa và tối ưu hóa** Hệ thống tích hợp các chức năng tự động hóa để giảm thiểu thao tác thủ công và tăng độ chính xác. Các quy trình tự động bao gồm tạo mã đơn hàng và mã khách hàng theo định dạng chuẩn, cập nhật tồn kho khi có giao dịch, tính toán chi tiết đơn hàng, cập nhật ngày bảo hành và trạng thái thanh toán. Chức năng tối ưu hóa hiệu suất thông qua việc sử dụng các chỉ mục (index) giúp truy vấn nhanh hơn trên các thông tin thường xuyên tìm kiếm như mã sản phẩm hoặc số điện thoại khách hàng. Các công cụ báo cáo và phân tích cung cấp dữ liệu chi tiết, giúp chủ doanh nghiệp đưa ra quyết định chiến lược và tối ưu hóa quy trình kinh doanh.

**1.4 Mô tả về phần giao diện:**

**Giao diện đăng nhập và phân quyền** Giao diện đăng nhập của hệ thống được thiết kế đơn giản nhưng bảo mật cao, với trường nhập tên đăng nhập và mật khẩu được mã hóa để đảm bảo an toàn. Sau khi đăng nhập, người dùng được chuyển hướng đến trang chính tương ứng với vai trò của họ (Admin, Manager, Cashier, Salesperson hoặc Staff). Giao diện tự động hiển thị các chức năng được phép truy cập dựa trên quyền hạn, ví dụ: Admin thấy toàn bộ menu quản lý, trong khi thu ngân chỉ thấy các chức năng liên quan đến bán hàng và thanh toán. Thanh điều hướng (navigation bar) được bố trí rõ ràng, với các biểu tượng và nhãn dễ hiểu, giúp người dùng nhanh chóng di chuyển giữa các chức năng như quản lý sản phẩm, đơn hàng hoặc báo cáo.

**Giao diện quản lý sản phẩm và tồn kho** Giao diện quản lý sản phẩm được thiết kế để hiển thị danh sách các mặt hàng như điện thoại, máy tính bảng, phụ kiện và đồng hồ thông minh theo dạng bảng hoặc lưới, với các thông tin như mã sản phẩm, tên, thương hiệu, danh mục, giá bán và trạng thái. Người dùng có thể tìm kiếm sản phẩm theo tên, mã hoặc danh mục thông qua thanh tìm kiếm tích hợp, hỗ trợ bộ lọc nâng cao như thương hiệu hoặc mức giá. Giao diện tồn kho hiển thị số lượng sản phẩm có sẵn, số lượng đã đặt, và cảnh báo bằng màu sắc (ví dụ: màu đỏ cho tồn kho thấp). Các nút hành động như thêm, sửa, xóa sản phẩm hoặc cập nhật tồn kho được bố trí trực quan, cho phép chủ doanh nghiệp và quản lý dễ dàng quản lý hàng hóa.

**Giao diện quản lý khách hàng và nhà cung cấp** Giao diện quản lý khách hàng hiển thị danh sách khách hàng với các chi tiết như họ tên, số điện thoại, email, loại khách hàng (thường hoặc VIP) và tổng chi tiêu. Người dùng có thể tìm kiếm khách hàng theo số điện thoại hoặc mã khách hàng, đồng thời thêm hoặc chỉnh sửa thông tin qua các biểu mẫu (form) thân thiện, có kiểm tra dữ liệu đầu vào để tránh lỗi. Giao diện quản lý nhà cung cấp tương tự, cung cấp thông tin về tên công ty, người liên hệ, số điện thoại và địa chỉ, với khả năng tìm kiếm và cập nhật dễ dàng. Các giao diện này được tối ưu để nhân viên nhanh chóng truy xuất thông tin khi tư vấn khách hàng hoặc quản lý nguồn hàng.

**Giao diện quản lý đơn hàng và thanh toán** Giao diện quản lý đơn hàng cho phép nhân viên tạo, xem và cập nhật đơn hàng một cách trực quan. Trang danh sách đơn hàng hiển thị mã đơn hàng, thông tin khách hàng, nhân viên phụ trách, ngày đặt hàng, tổng tiền và trạng thái. Người dùng có thể nhấp vào một đơn hàng để xem chi tiết, bao gồm danh sách sản phẩm, số lượng, giá bán, chiết khấu và thông tin bảo hành. Giao diện tạo đơn hàng cung cấp biểu mẫu để chọn khách hàng, sản phẩm, áp dụng khuyến mãi và chọn phương thức thanh toán (tiền mặt, thẻ, chuyển khoản, trả góp). Giao diện thanh toán hiển thị lịch sử giao dịch với số tiền, mã tham chiếu và trạng thái, hỗ trợ nhân viên xác nhận thanh toán nhanh chóng. Các nút hành động như “Thêm thanh toán” hoặc “Cập nhật trạng thái” được thiết kế nổi bật để dễ thao tác.

**Giao diện quản lý khuyến mãi** Giao diện quản lý khuyến mãi hiển thị danh sách các chương trình ưu đãi với thông tin như mã khuyến mãi, tên, loại giảm giá, giá trị giảm, ngày hiệu lực và số lần sử dụng. Người dùng có thể tìm kiếm khuyến mãi theo mã hoặc trạng thái (đang hoạt động, hết hạn) và thêm, chỉnh sửa hoặc xóa chương trình thông qua các biểu mẫu có gợi ý và kiểm tra hợp lệ (ví dụ: ngày kết thúc không được trước ngày bắt đầu). Giao diện này tích hợp với chức năng tạo đơn hàng, cho phép nhân viên dễ dàng áp dụng khuyến mãi phù hợp khi xử lý giao dịch, mang lại trải nghiệm thuận tiện cho cả nhân viên và khách hàng.

**Giao diện báo cáo và phân tích** Giao diện báo cáo được thiết kế để cung cấp cái nhìn tổng quan và chi tiết về hoạt động kinh doanh. Trang dashboard chính hiển thị các số liệu quan trọng như doanh thu hôm nay, doanh thu tháng, số đơn hàng và số lượng sản phẩm tồn kho thấp, được trình bày dưới dạng biểu đồ hoặc thẻ thông tin (widget) trực quan. Giao diện báo cáo doanh thu theo ngày cho phép chọn khoảng thời gian và hiển thị số liệu về tổng doanh thu, số đơn hàng và giá trị trung bình đơn hàng. Báo cáo sản phẩm bán chạy và hiệu suất nhân viên được trình bày dưới dạng bảng hoặc biểu đồ, với các bộ lọc theo thời gian hoặc nhân viên. Người dùng có thể xuất báo cáo dưới dạng tệp để phân tích thêm, hỗ trợ chủ doanh nghiệp đưa ra quyết định chiến lược.

**Giao diện dành cho khách hàng** Mặc dù hệ thống chủ yếu phục vụ nhân viên và chủ doanh nghiệp, giao diện dành cho khách hàng (nếu tích hợp với ứng dụng hoặc website bán hàng) được thiết kế để hỗ trợ tìm kiếm và mua sắm. Khách hàng có thể xem danh sách sản phẩm với hình ảnh, giá bán, thông số kỹ thuật và các khuyến mãi đang áp dụng. Giao diện hiển thị sản phẩm theo danh mục hoặc thương hiệu, với thanh tìm kiếm và bộ lọc giá cả, giúp khách hàng dễ dàng lựa chọn. Trang chi tiết đơn hàng cho phép khách hàng xem trạng thái đơn hàng, thông tin bảo hành và lịch sử thanh toán, mang lại trải nghiệm mua sắm minh bạch và thuận tiện.

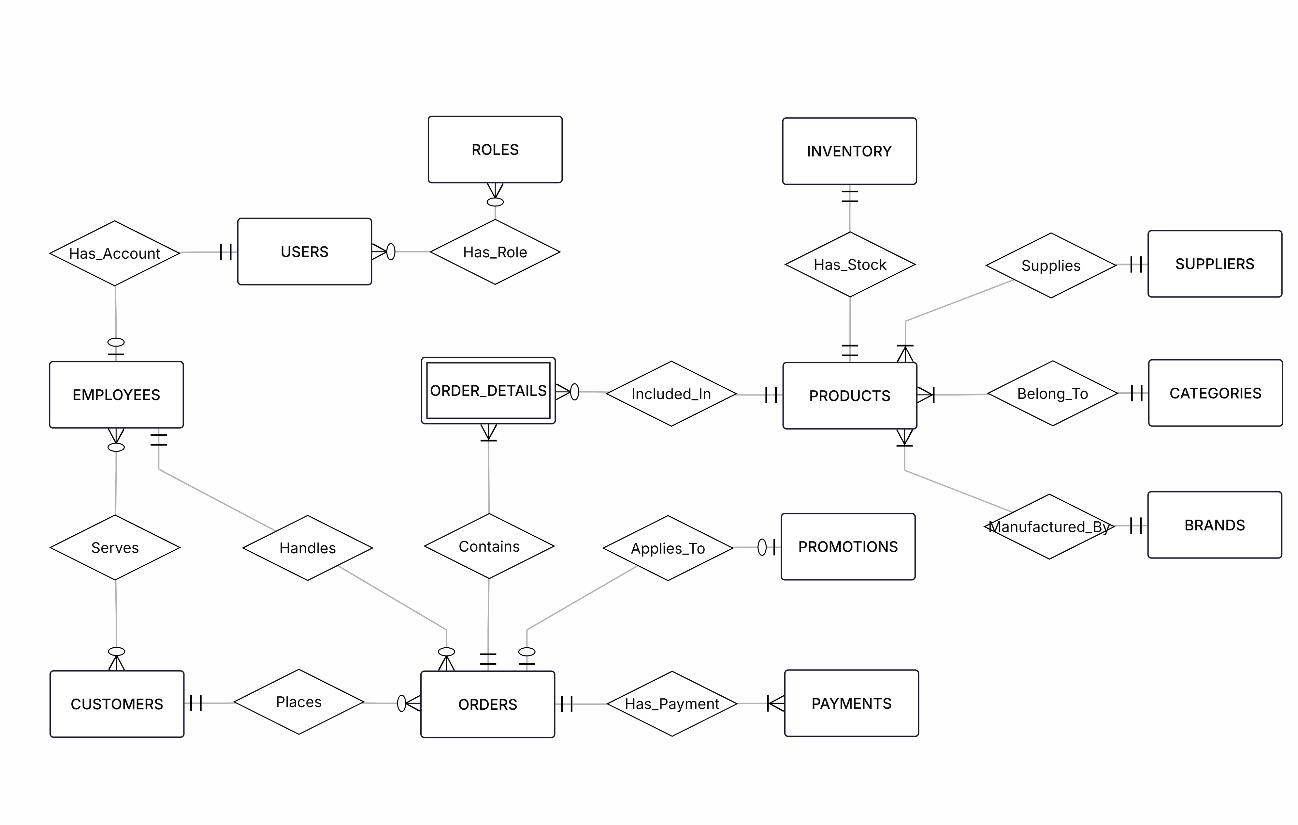
**Tính năng tối ưu hóa giao diện** Giao diện hệ thống được tối ưu hóa để hoạt động mượt mà trên cả máy tính và thiết bị di động, với bố cục responsive đảm bảo hiển thị tốt trên các màn hình khác nhau. Các biểu mẫu nhập liệu có kiểm tra dữ liệu tự động (ví dụ: định dạng số điện thoại, email) để giảm sai sót. Màu sắc và biểu tượng được sử dụng nhất quán, với các yếu tố như nút hành động nổi bật và thông báo trực quan (ví dụ: cảnh báo tồn kho thấp bằng màu đỏ). Hệ thống cũng hỗ trợ ngôn ngữ đa dạng (nếu cần) và cung cấp hướng dẫn sử dụng ngắn gọn trong giao diện, giúp người dùng mới làm quen nhanh chóng.

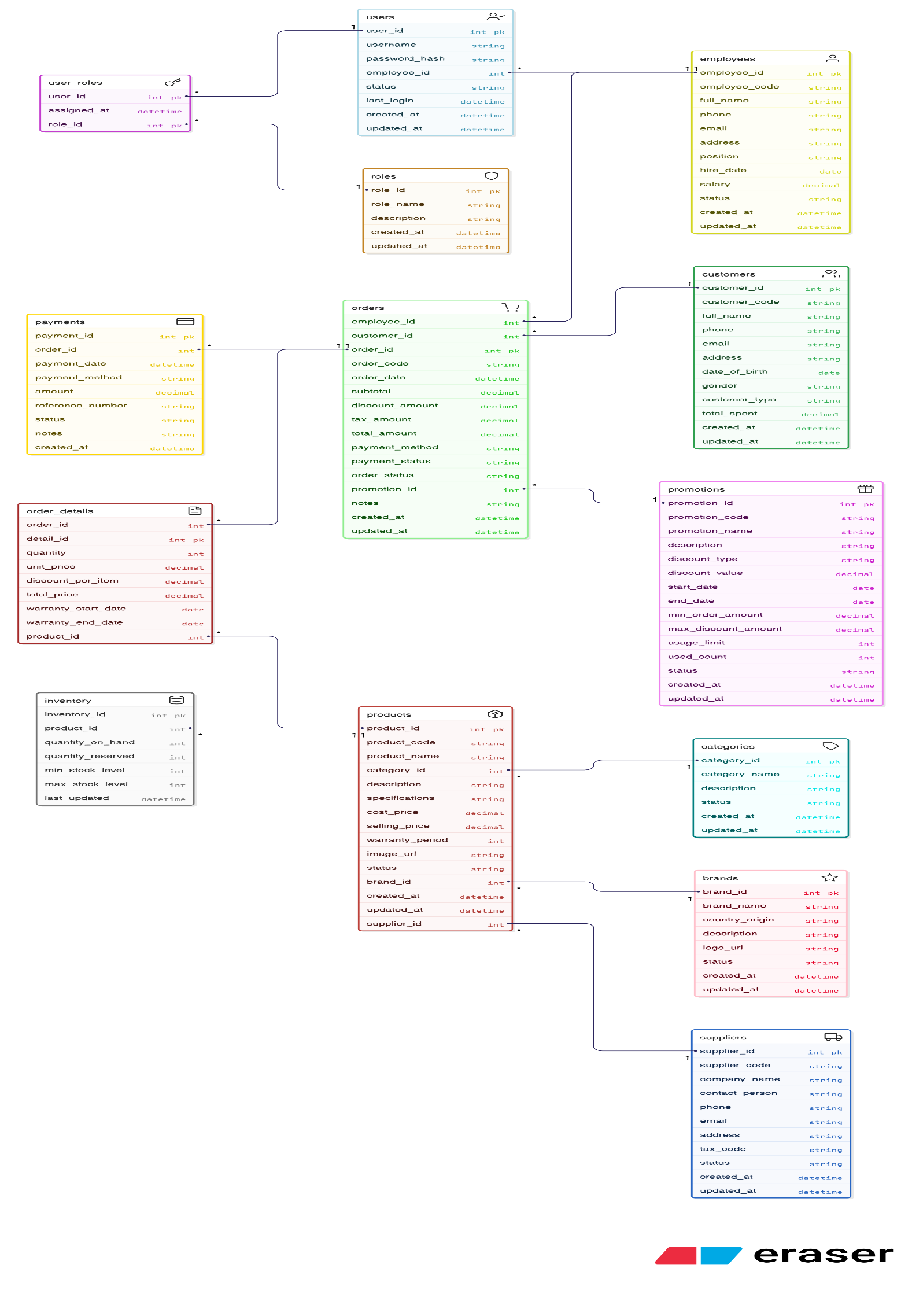
Giao diện của hệ thống quản lý cửa hàng điện thoại được thiết kế để đáp ứng nhu cầu của các đối tượng người dùng khác nhau, từ chủ doanh nghiệp, nhân viên đến khách hàng. Với bố cục trực quan, tính năng tìm kiếm và lọc thông tin mạnh mẽ, cùng các biểu mẫu và báo cáo dễ sử dụng, giao diện đảm bảo trải nghiệm người dùng mượt mà, hỗ trợ tự động hóa quy trình kinh doanh và nâng cao hiệu quả vận hành. Các yếu tố như phân quyền, cảnh báo trực quan và tối ưu hóa responsive giúp hệ thống trở nên an toàn, linh hoạt và phù hợp với môi trường bán lẻ công nghệ.

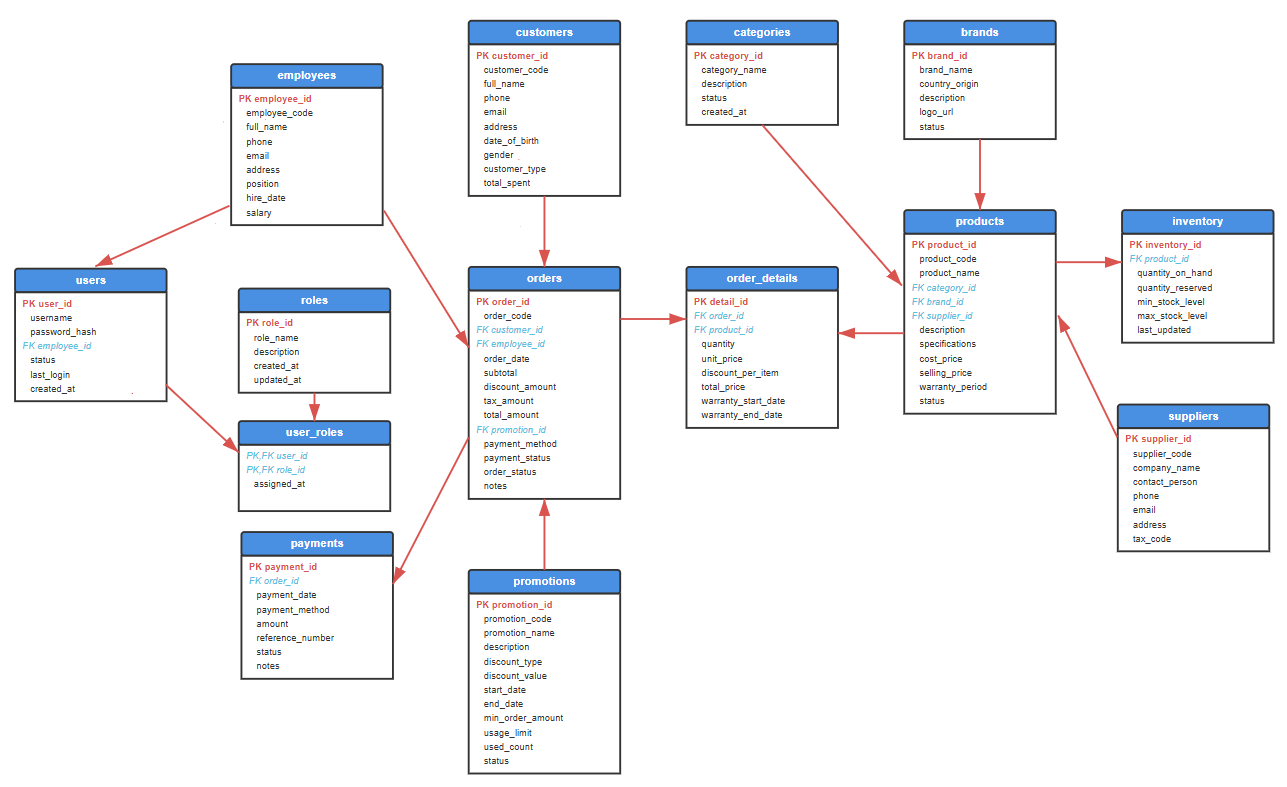
**CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

**1. Thiết kế cơ sở dữ liệu mức quan niệm**

Từ mô tả về dữ liệu cần có ở phần mô tả của bài toán ta hình thành được sơ đồ thực thể kết hợp (ERD).







**employees**(employee\_id, employee\_code, full\_name, phone, email, address, position, hire\_date, salary, status, created\_at, updated\_at)

**users**(user\_id, username, password\_hash, employee\_id, status, last\_login, created\_at, updated\_at)

**roles**(role\_id, role\_name, description, created\_at, updated\_at)

**user\_roles**(user\_id, role\_id, assigned\_at)

**customers**(customer\_id, customer\_code, full\_name, phone, email, address, date\_of\_birth, gender, customer\_type, total\_spent, created\_at, updated\_at)

**categories**(category\_id, category\_name, description, status, created\_at, updated\_at)

**brands**(brand\_id, brand\_name, country\_origin, description, logo\_url, status, created\_at, updated\_at)

**suppliers**(supplier\_id, supplier\_code, company\_name, contact\_person, phone, email, address, tax\_code, status, created\_at, updated\_at)

**products**(product\_id, product\_code, product\_name, category\_id, brand\_id, supplier\_id, description, specifications, cost\_price, selling\_price, warranty\_period, image\_url, status, created\_at, updated\_at)

**inventory**(inventory\_id, product\_id, quantity\_on\_hand, quantity\_reserved, min\_stock\_level, max\_stock\_level, last\_updated)

**promotions**(promotion\_id, promotion\_code, promotion\_name, description, discount\_type, discount\_value, start\_date, end\_date, min\_order\_amount, max\_discount\_amount, usage\_limit, used\_count, status, created\_at, updated\_at)

**orders**(order\_id, order\_code, customer\_id, employee\_id, order\_date, subtotal, discount\_amount, tax\_amount, total\_amount, promotion\_id, payment\_method, payment\_status, order\_status, notes, created\_at, updated\_at)

**order\_details**(detail\_id, order\_id, product\_id, quantity, unit\_price, discount\_per\_item, total\_price, warranty\_start\_date, warranty\_end\_date)

**payments**(payment\_id, order\_id, payment\_date, payment\_method, amount, reference\_number, status, notes, created\_at)

|  |  |  |
| --- | --- | --- |
| **STT** | **BẢNG** | **RÀNG BUỘC** |
| 1 | employees | - **employee\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  - **phone\_number** (UNIQUE, NOT NULL): Số điện thoại phải duy nhất và không được để trống.  - **email** (UNIQUE, NOT NULL): Email phải duy nhất và không được để trống.  - **created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại.  - **status** (CHECK (status IN ('active', 'inactive'))): Trạng thái chỉ chấp nhận 'active' hoặc 'inactive'. |
| 2 | users | - **user\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  - **username** (UNIQUE, NOT NULL): Tên đăng nhập phải duy nhất và không được để trống.  - **password\_hash** (NOT NULL): Mật khẩu mã hóa không được để trống.  - **employee\_id** (FOREIGN KEY REFERENCES employees(employee\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với employees, khi xóa nhân viên thì employee\_id thành NULL, cập nhật thì lan truyền.  - **created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại.  - **last\_login** (DEFAULT NULL): Thời gian đăng nhập cuối cùng có thể NULL. |
| 3 | roles | - **role\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  - **name** (UNIQUE, NOT NULL): Tên vai trò phải duy nhất và không được để trống.  - **description** (NOT NULL): Mô tả vai trò không được để trống.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 4 | user\_roles | - **user\_id, role\_id** (COMPOSITE PRIMARY KEY): Cặp user\_id và role\_id tạo khóa chính.  - **user\_id** (FOREIGN KEY REFERENCES users(user\_id) ON DELETE CASCADE ON UPDATE CASCADE): Liên kết với users, xóa hoặc cập nhật lan truyền.  - **role\_id** (FOREIGN KEY REFERENCES roles(role\_id) ON DELETE CASCADE ON UPDATE CASCADE): Liên kết với roles, xóa hoặc cập nhật lan truyền. |
| 5 | customers | - **customer\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- phone\_number** (UNIQUE, NOT NULL): Số điện thoại phải duy nhất và không được để trống.  **- email** (UNIQUE, NOT NULL): Email phải duy nhất và không được để trống.  **- total\_spent** (DECIMAL(15,2), DEFAULT 0.00): Tổng chi tiêu mặc định là 0.  **- customer\_type** (CHECK (customer\_type IN ('regular', 'vip'))): Loại khách hàng chỉ chấp nhận 'regular' hoặc 'vip'.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 6 | suppliers | **- supplier\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- contact\_phone** (UNIQUE, NOT NULL): Số điện thoại liên hệ phải duy nhất và không được để trống.  **- email** (UNIQUE, NOT NULL): Email phải duy nhất và không được để trống.  **- tax\_id** (UNIQUE, NOT NULL): Mã số thuế phải duy nhất và không được để trống.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại.  **- status** (CHECK (status IN ('active', 'inactive'))): Trạng thái chỉ chấp nhận 'active' hoặc 'inactive'. |
| 7 | categories | **- category\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- name** (UNIQUE, NOT NULL): Tên danh mục phải duy nhất và không được để trống.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại.  **- status** (CHECK (status IN ('active', 'inactive'))): Trạng thái chỉ chấp nhận 'active' hoặc 'inactive'. |
| 8 | brands | **- brand\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- name** (UNIQUE, NOT NULL): Tên thương hiệu phải duy nhất và không được để trống.  **- logo\_url** (NOT NULL): URL logo không được để trống.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại.  **- status** (CHECK (status IN ('active', 'inactive'))): Trạng thái chỉ chấp nhận 'active' hoặc 'inactive'. |
| 9 | products | **- product\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- name** (NOT NULL): Tên sản phẩm không được để trống.  **- category\_id** (FOREIGN KEY REFERENCES categories(category\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với categories, xóa thì category\_id thành NULL, cập nhật lan truyền.  **- brand\_id** (FOREIGN KEY REFERENCES brands(brand\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với brands, xóa thì brand\_id thành NULL, cập nhật lan truyền. **- supplier\_id** (FOREIGN KEY REFERENCES suppliers(supplier\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với suppliers, xóa thì supplier\_id thành NULL, cập nhật lan truyền. - cost\_price, selling\_price (DECIMAL(15,2), NOT NULL, CHECK (selling\_price >= cost\_price)): Giá bán phải lớn hơn hoặc bằng giá vốn.  **- warranty\_period** (INT, CHECK (warranty\_period >= 0)): Thời gian bảo hành không âm.  **- image\_url** (NOT NULL): URL hình ảnh không được để trống.  **- status** (CHECK (status IN ('active', 'inactive', 'discontinued'))): Trạng thái chỉ chấp nhận 'active', 'inactive', hoặc 'discontinued'.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 10 | inventory | **- product\_id** (INT, PRIMARY KEY, FOREIGN KEY REFERENCES products(product\_id) ON DELETE CASCADE ON UPDATE CASCADE): Khóa chính và khóa ngoại liên kết với products, xóa hoặc cập nhật lan truyền.  **- quantity\_in\_stock** (INT, NOT NULL, CHECK (quantity\_in\_stock >= 0)): Số lượng tồn kho không âm.  **- quantity\_ordered** (INT, NOT NULL, CHECK (quantity\_ordered >= 0)): Số lượng đã đặt không âm.  **- min\_stock\_threshold, max\_stock\_threshold** (INT, CHECK (min\_stock\_threshold >= 0 AND max\_stock\_threshold >= min\_stock\_threshold)): Ngưỡng tồn kho tối thiểu và tối đa, tối đa phải lớn hơn hoặc bằng tối thiểu.  **- last\_updated** (DATETIME, NOT NULL, DEFAULT GETDATE()): Thời gian cập nhật cuối cùng không được trống, mặc định là thời gian hiện tại. |
| 11 | promotions | **- promotion\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- code** (NVARCHAR(50), UNIQUE, NOT NULL): Mã khuyến mãi phải duy nhất và không được để trống.  **- name** (NOT NULL): Tên khuyến mãi không được để trống.  **- discount\_type** (CHECK (discount\_type IN ('percentage', 'fixed'))): Loại giảm giá chỉ chấp nhận 'percentage' hoặc 'fixed'.  **- discount\_value** (DECIMAL(15,2), NOT NULL, CHECK (discount\_value >= 0)): Giá trị giảm không âm.  **- start\_date, end\_date** (DATETIME, NOT NULL, CHECK (end\_date > start\_date)): Ngày kết thúc phải sau ngày bắt đầu.  **- min\_order\_amount** (DECIMAL(15,2), NOT NULL, CHECK (min\_order\_amount >= 0)): Số tiền đơn hàng tối thiểu không âm.  **- usage\_limit** (INT, CHECK (usage\_limit >= 0)): Giới hạn sử dụng không âm.  **- usage\_count** (INT, DEFAULT 0, CHECK (usage\_count >= 0)): Số lần đã sử dụng mặc định là 0, không âm.  **- status** (CHECK (status IN ('active', 'inactive', 'expired'))): Trạng thái chỉ chấp nhận 'active', 'inactive', hoặc 'expired'.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 12 | orders | **- order\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- customer\_id** (FOREIGN KEY REFERENCES customers(customer\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với customers, xóa thì customer\_id thành NULL, cập nhật lan truyền.  **- employee\_id** (FOREIGN KEY REFERENCES employees(employee\_id) ON DELETE SET NULL ON UPDATE CASCADE): Liên kết với employees, xóa thì employee\_id thành NULL, cập nhật lan truyền.  **- order\_date** (DATETIME, NOT NULL, DEFAULT GETDATE()): Ngày đặt hàng không được trống, mặc định là thời gian hiện tại.  **- subtotal, discount, tax, total** (DECIMAL(15,2), NOT NULL, CHECK (total >= 0 AND subtotal >= 0 AND discount >= 0 AND tax >= 0 AND total = subtotal - discount + tax)): Tổng tiền phải bằng (subtotal - discount + tax), tất cả giá trị không âm.  **- payment\_method** (CHECK (payment\_method IN ('cash', 'card', 'bank\_transfer', 'installment'))): Phương thức thanh toán chỉ chấp nhận các giá trị định nghĩa.  **- payment\_status** (CHECK (payment\_status IN ('pending', 'partial', 'paid', 'refunded'))): Trạng thái thanh toán chỉ chấp nhận các giá trị định nghĩa.  **- order\_status** (CHECK (order\_status IN ('processing', 'completed', 'cancelled', 'returned'))): Trạng thái đơn hàng chỉ chấp nhận các giá trị định nghĩa.  **- notes** (NVARCHAR(MAX)): Ghi chú có thể NULL.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 13 | order\_details | **- order\_id, product\_id** (COMPOSITE PRIMARY KEY): Cặp order\_id và product\_id tạo khóa chính.  **- order\_id** (FOREIGN KEY REFERENCES orders(order\_id) ON DELETE CASCADE ON UPDATE CASCADE): Liên kết với orders, xóa hoặc cập nhật lan truyền.  **- product\_id** (FOREIGN KEY REFERENCES products(product\_id) ON DELETE CASCADE ON UPDATE CASCADE): Liên kết với products, xóa hoặc cập nhật lan truyền.  **- quantity** (INT, NOT NULL, CHECK (quantity > 0)): Số lượng phải lớn hơn 0.  **- unit\_price** (DECIMAL(15,2), NOT NULL, CHECK (unit\_price >= 0)): Giá đơn vị không âm.  **- discount** (DECIMAL(15,2), DEFAULT 0.00, CHECK (discount >= 0)): Chiết khấu mặc định là 0, không âm.  **- subtotal** (DECIMAL(15,2), NOT NULL, CHECK (subtotal = quantity \* unit\_price - discount)): Tổng tiền chi tiết phải bằng (số lượng \* giá đơn vị - chiết khấu).  **- warranty\_start\_date, warranty\_end\_date** (DATETIME, NOT NULL, CHECK (warranty\_end\_date > warranty\_start\_date)): Ngày kết thúc bảo hành phải sau ngày bắt đầu.  **- created\_at, updated\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo và cập nhật không được trống, mặc định là thời gian hiện tại. |
| 14 | payments | **- payment\_id** (INT, PRIMARY KEY, IDENTITY(1,1)): Khóa chính, tự động tăng.  **- order\_id** (FOREIGN KEY REFERENCES orders(order\_id) ON DELETE CASCADE ON UPDATE CASCADE): Liên kết với orders, xóa hoặc cập nhật lan truyền.  **- amount** (DECIMAL(15,2), NOT NULL, CHECK (amount > 0)): Số tiền thanh toán phải lớn hơn 0.  **- payment\_date** (DATETIME, NOT NULL, DEFAULT GETDATE()): Ngày thanh toán không được trống, mặc định là thời gian hiện tại.  **- payment\_method** (CHECK (payment\_method IN ('cash', 'card', 'bank\_transfer'))): Phương thức thanh toán chỉ chấp nhận các giá trị định nghĩa.  **- reference\_number** (NVARCHAR(50)): Số tham chiếu có thể NULL.  **- status** (CHECK (status IN ('success', 'failed', 'pending'))): Trạng thái chỉ chấp nhận 'success', 'failed', hoặc 'pending'.  **- notes** (NVARCHAR(MAX)): Ghi chú có thể NULL.  **- created\_at** (NOT NULL, DEFAULT GETDATE()): Thời gian tạo không được trống, mặc định là thời gian hiện tại. |

**4. Cài đặt SQL và các ràng buộc**

**Bảng Nhân viên (employees)**

CREATE TABLE employees (

employee\_id INT IDENTITY(1,1) PRIMARY KEY,

employee\_code NVARCHAR(20) UNIQUE NOT NULL,

full\_name NVARCHAR(100) NOT NULL,

phone NVARCHAR(15),

email NVARCHAR(100),

address NVARCHAR(MAX),

position NVARCHAR(50), -- Quản lý, Thu ngân, Bán hàng

hire\_date DATE,

salary DECIMAL(15,2),

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Vai trò (roles)**

CREATE TABLE roles (

role\_id INT IDENTITY(1,1) PRIMARY KEY,

role\_name NVARCHAR(50) NOT NULL UNIQUE, -- Admin, Manager, Cashier, Salesperson

description NVARCHAR(MAX),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Người dùng (users)**

CREATE TABLE users (

user\_id INT IDENTITY(1,1) PRIMARY KEY,

username NVARCHAR(50) UNIQUE NOT NULL,

password\_hash NVARCHAR(256) NOT NULL, -- Lưu mật khẩu đã băm (SHA-256)

employee\_id INT, -- Liên kết với nhân viên

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

last\_login DATETIME,

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE(),

CONSTRAINT FK\_users\_employees FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

GO

**Bảng Phân quyền (user\_roles)**

CREATE TABLE user\_roles (

user\_id INT,

role\_id INT,

assigned\_at DATETIME DEFAULT GETDATE(),

PRIMARY KEY (user\_id, role\_id),

CONSTRAINT FK\_user\_roles\_users FOREIGN KEY (user\_id) REFERENCES users(user\_id),

CONSTRAINT FK\_user\_roles\_roles FOREIGN KEY (role\_id) REFERENCES roles(role\_id)

);

GO

**Bảng Khách hàng (customers)**

CREATE TABLE customers (

customer\_id INT IDENTITY(1,1) PRIMARY KEY,

customer\_code NVARCHAR(20) UNIQUE NOT NULL,

full\_name NVARCHAR(100) NOT NULL,

phone NVARCHAR(15) UNIQUE,

email NVARCHAR(100),

address NVARCHAR(MAX),

date\_of\_birth DATE,

gender NVARCHAR(10) CHECK (gender IN ('male', 'female', 'other')),

customer\_type NVARCHAR(10) DEFAULT 'regular' CHECK (customer\_type IN ('regular', 'vip')),

total\_spent DECIMAL(15,2) DEFAULT 0,

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Nhà cung cấp (suppliers)**

CREATE TABLE suppliers (

supplier\_id INT IDENTITY(1,1) PRIMARY KEY,

supplier\_code NVARCHAR(20) UNIQUE NOT NULL,

company\_name NVARCHAR(200) NOT NULL,

contact\_person NVARCHAR(100),

phone NVARCHAR(15),

email NVARCHAR(100),

address NVARCHAR(MAX),

tax\_code NVARCHAR(20),

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Danh mục sản phẩm (categories)**

CREATE TABLE categories (

category\_id INT IDENTITY(1,1) PRIMARY KEY,

category\_name NVARCHAR(100) NOT NULL,

description NVARCHAR(MAX),

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Thương hiệu (brands)**

CREATE TABLE brands (

brand\_id INT IDENTITY(1,1) PRIMARY KEY,

brand\_name NVARCHAR(100) NOT NULL,

country\_origin NVARCHAR(50),

description NVARCHAR(MAX),

logo\_url NVARCHAR(500),

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Sản phẩm (products)**

CREATE TABLE products (

product\_id INT IDENTITY(1,1) PRIMARY KEY,

product\_code NVARCHAR(50) UNIQUE NOT NULL,

product\_name NVARCHAR(200) NOT NULL,

category\_id INT,

brand\_id INT,

supplier\_id INT,

description NVARCHAR(MAX),

specifications NVARCHAR(MAX), -- Lưu JSON dưới dạng chuỗi

cost\_price DECIMAL(15,2), -- Giá vốn

selling\_price DECIMAL(15,2), -- Giá bán

warranty\_period INT, -- Thời gian bảo hành (tháng)

image\_url NVARCHAR(500),

status NVARCHAR(15) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'discontinued')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE(),

CONSTRAINT FK\_products\_categories FOREIGN KEY (category\_id) REFERENCES categories(category\_id),

CONSTRAINT FK\_products\_brands FOREIGN KEY (brand\_id) REFERENCES brands(brand\_id),

CONSTRAINT FK\_products\_suppliers FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id)

);

GO

**Bảng Tồn kho (inventory)**

CREATE TABLE inventory (

inventory\_id INT IDENTITY(1,1) PRIMARY KEY,

product\_id INT,

quantity\_on\_hand INT DEFAULT 0, -- Số lượng tồn kho

quantity\_reserved INT DEFAULT 0, -- Số lượng đã đặt hàng

min\_stock\_level INT DEFAULT 0, -- Mức tồn kho tối thiểu

max\_stock\_level INT DEFAULT 0, -- Mức tồn kho tối đa

last\_updated DATETIME DEFAULT GETDATE(),

CONSTRAINT FK\_inventory\_products FOREIGN KEY (product\_id) REFERENCES products(product\_id),

CONSTRAINT unique\_product\_inventory UNIQUE (product\_id)

);

GO

**Bảng Khuyến mãi (promotions)**

CREATE TABLE promotions (

promotion\_id INT IDENTITY(1,1) PRIMARY KEY,

promotion\_code NVARCHAR(20) UNIQUE NOT NULL,

promotion\_name NVARCHAR(200) NOT NULL,

description NVARCHAR(MAX),

discount\_type NVARCHAR(20) CHECK (discount\_type IN ('percentage', 'fixed\_amount')),

discount\_value DECIMAL(15,2),

start\_date DATE,

end\_date DATE,

min\_order\_amount DECIMAL(15,2),

max\_discount\_amount DECIMAL(15,2),

usage\_limit INT, -- Số lần sử dụng tối đa

used\_count INT DEFAULT 0, -- Số lần đã sử dụng

status NVARCHAR(10) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'expired')),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE()

);

GO

**Bảng Đơn hàng (orders)**

CREATE TABLE orders (

order\_id INT IDENTITY(1,1) PRIMARY KEY,

order\_code NVARCHAR(20) UNIQUE NOT NULL,

customer\_id INT,

employee\_id INT, -- Nhân viên bán hàng

order\_date DATETIME,

subtotal DECIMAL(15,2), -- Tổng tiền trước chiết khấu

discount\_amount DECIMAL(15,2) DEFAULT 0, -- Số tiền giảm giá

tax\_amount DECIMAL(15,2) DEFAULT 0, -- Thuế VAT

total\_amount DECIMAL(15,2), -- Tổng tiền cuối cùng

promotion\_id INT NULL, -- Khuyến mãi áp dụng

payment\_method NVARCHAR(20) CHECK (payment\_method IN ('cash', 'card', 'transfer', 'installment')),

payment\_status NVARCHAR(20) DEFAULT 'pending' CHECK (payment\_status IN ('pending', 'paid', 'partial', 'refunded')),

order\_status NVARCHAR(20) DEFAULT 'processing' CHECK (order\_status IN ('processing', 'completed', 'cancelled', 'returned')),

notes NVARCHAR(MAX),

created\_at DATETIME DEFAULT GETDATE(),

updated\_at DATETIME DEFAULT GETDATE(),

CONSTRAINT FK\_orders\_customers FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),

CONSTRAINT FK\_orders\_employees FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id),

CONSTRAINT FK\_orders\_promotions FOREIGN KEY (promotion\_id) REFERENCES promotions(promotion\_id)

);

GO

**Bảng Chi tiết đơn hàng (order\_details)**

CREATE TABLE order\_details (

detail\_id INT IDENTITY(1,1) PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT NOT NULL,

unit\_price DECIMAL(15,2), -- Giá bán tại thời điểm đó

discount\_per\_item DECIMAL(15,2) DEFAULT 0,

total\_price DECIMAL(15,2),

warranty\_start\_date DATE,

warranty\_end\_date DATE,

CONSTRAINT FK\_order\_details\_orders FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

CONSTRAINT FK\_order\_details\_products FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

GO

**Bảng Thanh toán (payments)**

CREATE TABLE payments (

payment\_id INT IDENTITY(1,1) PRIMARY KEY,

order\_id INT,

payment\_date DATETIME,

payment\_method NVARCHAR(20) CHECK (payment\_method IN ('cash', 'card', 'transfer', 'installment')),

amount DECIMAL(15,2),

reference\_number NVARCHAR(100), -- Số tham chiếu giao dịch

status NVARCHAR(10) DEFAULT 'successful' CHECK (status IN ('successful', 'failed', 'pending')),

notes NVARCHAR(MAX),

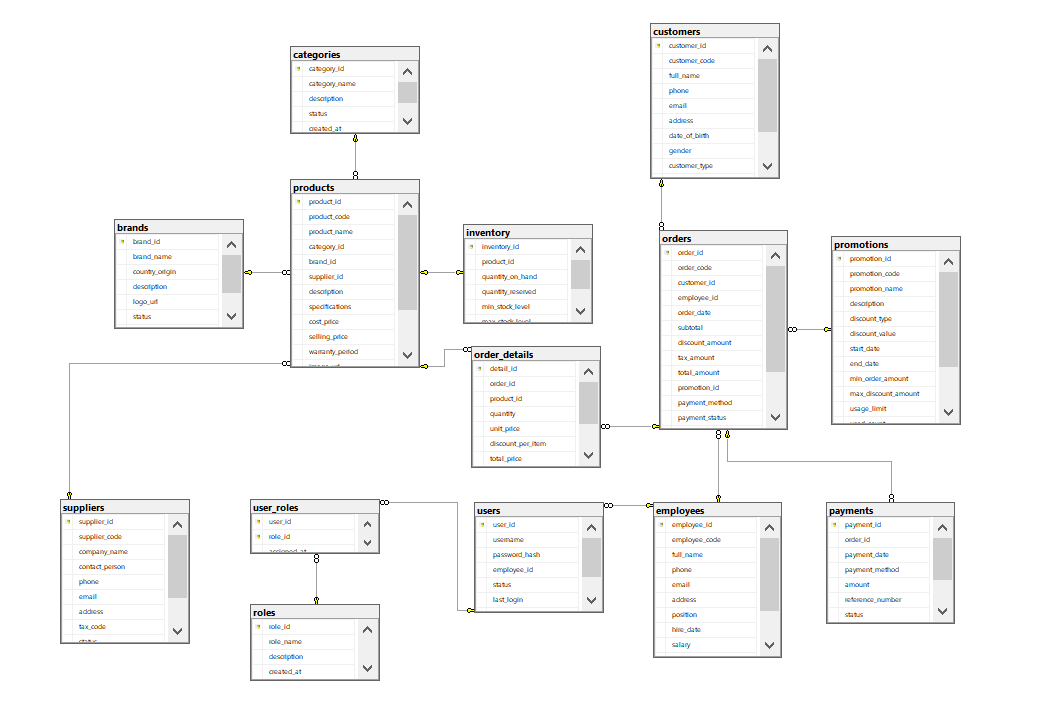
created\_at DATETIME DEFAULT GETDATE(),

CONSTRAINT FK\_payments\_orders FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

GO

**5. Lược đồ quan hệ sau khi cài đặt SQL**



**6. Các view**

**View báo cáo doanh thu theo ngày**

CREATE VIEW daily\_revenue\_report AS

SELECT TOP 100 PERCENT

CAST(order\_date AS DATE) AS report\_date,

COUNT(\*) AS total\_orders,

SUM(total\_amount) AS total\_revenue,

AVG(total\_amount) AS avg\_order\_value,

FROM orders

WHERE order\_status = 'completed'

GROUP BY CAST(order\_date AS DATE)

ORDER BY report\_date DESC;

GO

**View báo cáo hiệu suất nhân viên**

CREATE VIEW employee\_performance

AS

SELECT

e.employee\_id,

e.full\_name,

e.position,

e.created\_at,

o.order\_id,

o.total\_amount,

o.order\_date

FROM employees e

LEFT JOIN orders o

ON e.employee\_id = o.employee\_id

AND o.order\_status = 'completed'

WHERE e.status = 'active';

GO

**View top sản phẩm bán chạy**

CREATE VIEW top\_selling\_products AS

SELECT

p.product\_id,

p.product\_name,

p.product\_code,

b.brand\_name,

c.category\_name,

od.order\_id,

od.quantity,

od.total\_price,

od.unit\_price,

o.order\_date

FROM products p

INNER JOIN order\_details od ON p.product\_id = od.product\_id

INNER JOIN orders o ON od.order\_id = o.order\_id

LEFT JOIN brands b ON p.brand\_id = b.brand\_id

LEFT JOIN categories c ON p.category\_id = c.category\_id

WHERE o.order\_status = 'completed';

GO

**View báo cáo sản phẩm có sẵn**

CREATE VIEW v\_AllActiveProducts

AS

SELECT

p.\*,

c.category\_name,

b.brand\_name,

COALESCE(i.quantity\_on\_hand, 0) AS quantity\_on\_hand

FROM products p

LEFT JOIN categories c ON p.category\_id = c.category\_id

LEFT JOIN brands b ON p.brand\_id = b.brand\_id

LEFT JOIN inventory i ON p.product\_id = i.product\_id

WHERE p.status = 'active';

GO

**View báo cáo nhân viên hoạt động**

CREATE VIEW v\_ActiveEmployees

AS

SELECT

employee\_id,

employee\_code,

full\_name,

position,

status

FROM employees

WHERE status = 'active';

GO

**View báo cáo khuyến mãi còn hoạt động**

CREATE VIEW v\_ActivePromotions

AS

SELECT \*

FROM promotions

WHERE

status = 'active'

AND CAST(GETDATE() AS DATE) BETWEEN start\_date AND end\_date

AND (usage\_limit IS NULL OR used\_count < usage\_limit);

GO

**7. Các trigger**

**Trigger cập nhật thời gian updated\_at**

CREATE TRIGGER tr\_users\_update

ON users

AFTER UPDATE

AS

BEGIN

UPDATE users

SET updated\_at = GETDATE()

FROM inserted

WHERE users.user\_id = inserted.user\_id;

END;

GO

CREATE TRIGGER tr\_roles\_update

ON roles

AFTER UPDATE

AS

BEGIN

UPDATE roles

SET updated\_at = GETDATE()

FROM inserted

WHERE roles.role\_id = inserted.role\_id;

END;

GO

CREATE TRIGGER tr\_employees\_update

ON employees

AFTER UPDATE

AS

BEGIN

UPDATE employees

SET updated\_at = GETDATE()

FROM inserted

WHERE employees.employee\_id = inserted.employee\_id;

END;

GO

CREATE TRIGGER tr\_customers\_update

ON customers

AFTER UPDATE

AS

BEGIN

UPDATE customers

SET updated\_at = GETDATE()

FROM inserted

WHERE customers.customer\_id = inserted.customer\_id;

END;

GO

CREATE TRIGGER tr\_suppliers\_update

ON suppliers

AFTER UPDATE

AS

BEGIN

UPDATE suppliers

SET updated\_at = GETDATE()

FROM inserted

WHERE suppliers.supplier\_id = inserted.supplier\_id;

END;

GO

CREATE TRIGGER tr\_categories\_update

ON categories

AFTER UPDATE

AS

BEGIN

UPDATE categories

SET updated\_at = GETDATE()

FROM inserted

WHERE categories.category\_id = inserted.category\_id;

END;

GO

CREATE TRIGGER tr\_brands\_update

ON brands

AFTER UPDATE

AS

BEGIN

UPDATE brands

SET updated\_at = GETDATE()

FROM inserted

WHERE brands.brand\_id = inserted.brand\_id;

END;

GO

CREATE TRIGGER tr\_products\_update

ON products

AFTER UPDATE

AS

BEGIN

UPDATE products

SET updated\_at = GETDATE()

FROM inserted

WHERE products.product\_id = inserted.product\_id;

END;

GO

CREATE TRIGGER tr\_inventory\_update

ON inventory

AFTER UPDATE

AS

BEGIN

UPDATE inventory

SET last\_updated = GETDATE()

FROM inserted

WHERE inventory.inventory\_id = inserted.inventory\_id;

END;

GO

CREATE TRIGGER tr\_promotions\_update

ON promotions

AFTER UPDATE

AS

BEGIN

UPDATE promotions

SET updated\_at = GETDATE()

FROM inserted

WHERE promotions.promotion\_id = inserted.promotion\_id;

END;

GO

CREATE TRIGGER tr\_orders\_update

ON orders

AFTER UPDATE

AS

BEGIN

UPDATE orders

SET updated\_at = GETDATE()

FROM inserted

WHERE orders.order\_id = inserted.order\_id;

END;

GO

**Trigger cập nhật tồn kho khi có đơn hàng**

CREATE TRIGGER update\_inventory\_after\_order

ON order\_details

AFTER INSERT

AS

BEGIN

UPDATE i

SET i.quantity\_on\_hand = i.quantity\_on\_hand - inserted.quantity,

i.last\_updated = GETDATE()

FROM inventory i

INNER JOIN inserted ON i.product\_id = inserted.product\_id;

END;

GO

**Trigger hoàn trả tồn kho khi hủy đơn hàng**

CREATE TRIGGER restore\_inventory\_on\_cancel

ON orders

AFTER UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE order\_status = 'cancelled')

AND EXISTS (SELECT \* FROM deleted WHERE order\_status != 'cancelled')

BEGIN

UPDATE i

SET i.quantity\_on\_hand = i.quantity\_on\_hand + od.quantity,

i.last\_updated = GETDATE()

FROM inventory i

INNER JOIN order\_details od ON i.product\_id = od.product\_id

INNER JOIN inserted ON od.order\_id = inserted.order\_id

WHERE inserted.order\_status = 'cancelled';

END;

END; GO

**Trigger cập nhật tổng chi tiêu khách hàng**

CREATE TRIGGER update\_customer\_total\_spent

ON orders

AFTER UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE order\_status = 'completed')

AND EXISTS (SELECT \* FROM deleted WHERE order\_status != 'completed')

BEGIN

UPDATE c

SET c.total\_spent = c.total\_spent + inserted.total\_amount

FROM customers c

INNER JOIN inserted ON c.customer\_id = inserted.customer\_id;

END;

IF EXISTS (SELECT \* FROM inserted WHERE order\_status = 'cancelled')

AND EXISTS (SELECT \* FROM deleted WHERE order\_status = 'completed')

BEGIN

UPDATE c

SET c.total\_spent = c.total\_spent - deleted.total\_amount

FROM customers c

INNER JOIN deleted ON c.customer\_id = deleted.customer\_id;

END;

END;

GO

**Trigger cập nhật loại khách hàng**

CREATE TRIGGER update\_customer\_type\_auto

ON customers

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF UPDATE(total\_spent)

BEGIN

UPDATE c

SET c.customer\_type = 'vip'

FROM customers c

INNER JOIN inserted i ON c.customer\_id = i.customer\_id

WHERE

ISNULL(i.customer\_type, '') = 'regular'

AND ISNULL(i.total\_spent, 0) >= 50000000;

END

END;

GO

**Trigger cập nhật số lần sử dụng khuyến mãi**

CREATE TRIGGER update\_promotion\_usage

ON orders

AFTER INSERT

AS

BEGIN

UPDATE p

SET p.used\_count = p.used\_count + 1

FROM promotions p

INNER JOIN inserted ON p.promotion\_id = inserted.promotion\_id

WHERE inserted.promotion\_id IS NOT NULL;

END;

GO

-- Trigger tính tổng tiền chi tiết đơn hàng

CREATE TRIGGER calculate\_order\_detail\_total

ON order\_details

AFTER INSERT

AS

BEGIN

UPDATE od

SET od.total\_price = inserted.quantity \* inserted.unit\_price - inserted.discount\_per\_item

FROM order\_details od

INNER JOIN inserted ON od.detail\_id = inserted.detail\_id;

END;

GO

**Trigger cập nhật ngày bảo hành**

CREATE TRIGGER set\_warranty\_dates

ON order\_details

AFTER INSERT

AS

BEGIN

UPDATE od

SET od.warranty\_start\_date = CAST(GETDATE() AS DATE),

od.warranty\_end\_date = DATEADD(MONTH, p.warranty\_period, CAST(GETDATE() AS DATE))

FROM order\_details od

INNER JOIN inserted ON od.detail\_id = inserted.detail\_id

INNER JOIN products p ON inserted.product\_id = p.product\_id;

END;

GO

**CHƯƠNG 3: THIẾT KẾ CÁC CHỨC NĂNG**

**1. Kết nối cơ sở dữ liệu**

public static class DBConnection

{

public static string Username { get; set; }

public static string Password { get; set; }

public static string GetConnectionString()

{

return $"Data Source=.;Initial Catalog=phone\_store\_db;User ID={Username};Password={Password}";

}

private static string defaultConnStr = "Data Source=.;Initial Catalog=phone\_store\_db;Integrated Security=True;Encrypt=True;TrustServerCertificate=True";

public static SqlConnection GetDefaultConnection()

{

return new SqlConnection(defaultConnStr);

}

public static SqlConnection GetConnection()

{

return new SqlConnection(GetConnectionString());

}

public static bool TestConnection()

{

try

{

using (var conn = GetConnection())

{

conn.Open();

return true;

}

}

catch

{

return false;

}

}

public static bool TestDefaultConnection()

{

try

{

using (var conn = GetDefaultConnection())

{

conn.Open();

return true;

}

}

catch

{

return false;

}

}

}

**4. Các chức năng bán hàng**

**Function tạo order code tự động**

CREATE FUNCTION fn\_GenerateOrderCode()

RETURNS NVARCHAR(50)

AS

BEGIN

DECLARE @prefix NVARCHAR(20) = 'ORD' + FORMAT(GETDATE(), 'yyyyMMdd');

DECLARE @nextNumber INT = (

SELECT COALESCE(MAX(CAST(RIGHT(order\_code, 4) AS INT)), 0) + 1

FROM orders

WHERE order\_code LIKE @prefix + '%'

);

DECLARE @code NVARCHAR(50) = @prefix + RIGHT('0000' + CAST(@nextNumber AS NVARCHAR(4)), 4);

RETURN @code;

END

GO

**Xử lý bên C#:**

public string GenerateOrderCode()

{

const string query = "SELECT dbo.fn\_GenerateOrderCode()";

var result = ExecuteScalar(query);

return result?.ToString() ?? "ORD" + DateTime.Now.ToString("yyyyMMdd") + "0001";

}

**Function tạo customer code tự động**

CREATE FUNCTION fn\_GenerateCustomerCode()

RETURNS NVARCHAR(50)

AS

BEGIN

DECLARE @prefix NVARCHAR(20) = 'CUS' + FORMAT(GETDATE(), 'yyyyMMdd');

DECLARE @nextNumber INT = (

SELECT COALESCE(MAX(CAST(RIGHT(customer\_code, 4) AS INT)), 0) + 1

FROM customers

WHERE customer\_code LIKE @prefix + '%'

);

RETURN @prefix + RIGHT('0000' + CAST(@nextNumber AS NVARCHAR(4)), 4);

END

GO

**Xử lý bên C#:**

public string GenerateCustomerCode()

{

const string query = "SELECT dbo.fn\_GenerateCustomerCode()";

var result = ExecuteScalar(query);

return result?.ToString() ?? "CUS" + DateTime.Now.ToString("yyyyMMdd") + "0001";}

**Procedure lấy số lượng sản phẩm tồn kho**

CREATE PROCEDURE sp\_GetProductStock

@productId INT

AS

BEGIN

SELECT COALESCE(quantity\_on\_hand, 0)

FROM inventory

WHERE product\_id = @productId;

END

GO

**Xử lý bên C#:**

public int GetProductStock(int productId)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@productId", productId)

};

var result = ExecuteScalar("sp\_GetProductStock", parameters, isStoredProcedure: true);

return result != null ? Convert.ToInt32(result) : 0;}

**Procedure lấy sản phẩm theo id**

CREATE PROCEDURE sp\_GetProductById

@productId INT

AS

BEGIN

SELECT

p.\*,

c.category\_name,

b.brand\_name,

COALESCE(i.quantity\_on\_hand, 0) AS quantity\_on\_hand

FROM products p

LEFT JOIN categories c ON p.category\_id = c.category\_id

LEFT JOIN brands b ON p.brand\_id = b.brand\_id

LEFT JOIN inventory i ON p.product\_id = i.product\_id

WHERE p.product\_id = @productId;

END

GO

**Xử lý bên C#:**

public Product GetProductById(int productId)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@productId", productId)

};

var dataTable = ExecuteQuery("sp\_GetProductById", parameters, isStoredProcedure: true);

if (dataTable.Rows.Count > 0)

{

return MapRowToProduct(dataTable.Rows[0]);}

return null;}

**Procedure lấy khách hàng theo số điện thoại**

CREATE PROCEDURE sp\_GetCustomerByPhone

@phone NVARCHAR(20)

AS

BEGIN

SELECT c.\*,

COUNT(o.order\_id) AS total\_orders,

COALESCE(MAX(o.order\_date), c.created\_at) AS last\_order\_date

FROM customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

AND o.order\_status = 'completed'

WHERE c.phone = @phone

GROUP BY

c.customer\_id, c.customer\_code, c.full\_name, c.phone, c.email,

c.address, c.date\_of\_birth, c.gender, c.customer\_type,

c.total\_spent, c.created\_at, c.updated\_at

END

GO

**Xử lý bên C#:**

public Customer GetCustomerByPhone(string phone)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@phone", phone)

};

var dataTable = ExecuteQuery("sp\_GetCustomerByPhone", parameters, isStoredProcedure: true);

if (dataTable.Rows.Count > 0)

{

return MapRowToCustomer(dataTable.Rows[0]);

}

return null;

}

**Procedure lấy chi tiết order bằng order id**

CREATE PROCEDURE sp\_GetOrderDetails

@orderId INT

AS

BEGIN

SELECT

od.\*,

p.product\_name,

p.product\_code

FROM order\_details od

INNER JOIN products p ON od.product\_id = p.product\_id

WHERE od.order\_id = @orderId

ORDER BY od.detail\_id;

END

GO

**Xử lý bên C#:**

public List<OrderDetail> GetOrderDetails(int orderId)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId)

};

var dataTable = ExecuteQuery("sp\_GetOrderDetails", parameters, isStoredProcedure: true);

var details = new List<OrderDetail>();

foreach (DataRow row in dataTable.Rows)

{

details.Add(MapRowToOrderDetail(row));

}

return details;

}

**Procedure lấy order bằng order id**

CREATE PROCEDURE sp\_GetOrderById

@orderId INT

AS

BEGIN

SELECT

o.\*,

c.full\_name AS customer\_name,

e.full\_name AS employee\_name

FROM orders o

LEFT JOIN customers c ON o.customer\_id = c.customer\_id

LEFT JOIN employees e ON o.employee\_id = e.employee\_id

WHERE o.order\_id = @orderId;

END

GO

**Xử lý bên C#:**

public Order GetOrderById(int orderId)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId)

};

var dataTable = ExecuteQuery("sp\_GetOrderById", parameters, isStoredProcedure: true);

if (dataTable.Rows.Count > 0)

{

var order = MapRowToOrder(dataTable.Rows[0]);

order.OrderDetails = GetOrderDetails(orderId);

return order;

}

return null;

}

**Procedure cập nhật trạng thái order**

CREATE PROCEDURE sp\_UpdateOrderStatus

@orderId INT,

@newStatus NVARCHAR(50)

AS

BEGIN

UPDATE orders

SET order\_status = @newStatus

WHERE order\_id = @orderId;

END

GO

**Xử lý bên C#:**

public bool UpdateOrderStatus(int orderId, string newStatus)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId),

new SqlParameter("@newStatus", newStatus)

};

return ExecuteNonQuery("sp\_UpdateOrderStatus", parameters, isStoredProcedure: true) > 0;

}

**Function lấy tất cả các order trong khoảng thời gian đã cho**

CREATE FUNCTION fn\_GetAllOrders

(

@fromDate DATE = NULL,

@toDate DATE = NULL

)

RETURNS TABLE

AS

RETURN

(

SELECT

o.\*,

c.full\_name AS customer\_name,

e.full\_name AS employee\_name,

COUNT(od.detail\_id) AS total\_items

FROM orders o

LEFT JOIN customers c ON o.customer\_id = c.customer\_id

LEFT JOIN employees e ON o.employee\_id = e.employee\_id

LEFT JOIN order\_details od ON o.order\_id = od.order\_id

WHERE

(@fromDate IS NULL OR CAST(o.order\_date AS DATE) >= @fromDate)

AND (@toDate IS NULL OR CAST(o.order\_date AS DATE) <= @toDate)

GROUP BY

o.order\_id, o.order\_code, o.customer\_id, o.employee\_id,

o.order\_date, o.subtotal, o.discount\_amount, o.tax\_amount,

o.total\_amount, o.promotion\_id, o.payment\_method, o.payment\_status,

o.order\_status, o.notes, o.created\_at, o.updated\_at,

c.full\_name, e.full\_name

);

GO

**Xử lý bên C#:**

public List<Order> GetAllOrders(DateTime? fromDate = null, DateTime? toDate = null)

{

var parameters = new List<SqlParameter>

{

new SqlParameter("@fromDate", (object)fromDate ?? DBNull.Value),

new SqlParameter("@toDate", (object)toDate ?? DBNull.Value)

};

string query = "SELECT \* FROM fn\_GetAllOrders(@fromDate, @toDate)";

var dataTable = ExecuteQuery(query, parameters.ToArray());

var orders = new List<Order>();

foreach (DataRow row in dataTable.Rows)

{

orders.Add(MapRowToOrder(row));

}

return orders;

}

**Procedure cập nhật trạng thái thanh toán**

CREATE PROCEDURE sp\_UpdatePaymentStatus

@orderId INT,

@newStatus NVARCHAR(50)

AS

BEGIN

UPDATE orders

SET payment\_status = @newStatus

WHERE order\_id = @orderId;

END

GO

**Xử lý bên C#:**

public bool UpdatePaymentStatus(int orderId, string newStatus)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId),

new SqlParameter("@newStatus", newStatus)

};

return ExecuteNonQuery("sp\_UpdatePaymentStatus", parameters, isStoredProcedure: true) > 0;

}

**Procedure cập nhật trạng thái thanh toán**

CREATE PROCEDURE sp\_UpdateOrderPaymentStatus

@orderId INT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @totalAmount DECIMAL(18,2);

DECLARE @paidAmount DECIMAL(18,2);

DECLARE @paymentStatus NVARCHAR(20);

-- Lấy tổng tiền đơn hàng và tổng số tiền đã thanh toán

SELECT

@totalAmount = o.total\_amount,

@paidAmount = COALESCE(SUM(CASE WHEN p.status = 'successful' THEN p.amount ELSE 0 END), 0)

FROM orders o

LEFT JOIN payments p ON o.order\_id = p.order\_id

WHERE o.order\_id = @orderId

GROUP BY o.total\_amount;

IF @totalAmount IS NULL

BEGIN

THROW 50003, 'Không tìm thấy đơn hàng.', 1;

END

-- Xác định trạng thái thanh toán mới

SET @paymentStatus =

CASE

WHEN @paidAmount = 0 THEN 'pending'

WHEN @paidAmount >= @totalAmount THEN 'paid'

ELSE 'partial'

END;

-- Cập nhật trạng thái đơn hàng

UPDATE orders

SET payment\_status = @paymentStatus

WHERE order\_id = @orderId;

END

GO

**Xử lý bên C#:**

public void UpdateOrderPaymentStatus(int orderId)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId)

};

ExecuteNonQuery("sp\_UpdateOrderPaymentStatus", parameters, isStoredProcedure: true);

}

catch (Exception ex)

{

throw new Exception($"Lỗi cập nhật trạng thái thanh toán: {ex.Message}"); } }

**Procedure lấy khuyến mãi bằng mã khuyến mãi**

CREATE PROCEDURE sp\_GetPromotionByCode

@code NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM promotions

WHERE promotion\_code = @code

AND status = 'active'

AND CAST(GETDATE() AS DATE) BETWEEN start\_date AND end\_date

AND (usage\_limit IS NULL OR used\_count < usage\_limit);

END

GO

**Xử lý bên C#:**

public Promotion GetPromotionByCode(string promotionCode)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@code", promotionCode)

};

var dataTable = ExecuteQuery("sp\_GetPromotionByCode", parameters, isStoredProcedure: true);

if (dataTable.Rows.Count > 0)

{

return MapRowToPromotion(dataTable.Rows[0]);

}

return null;

}

**Procedure lấy tất cả các khuyến mãi**

CREATE PROCEDURE sp\_GetAllPromotions

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM promotions

ORDER BY created\_at DESC;

END

GO

**Xử lý bên C#:**

public List<Promotion> GetAllPromotions()

{

var dataTable = ExecuteQuery("sp\_GetAllPromotions", null, isStoredProcedure: true);

var promotions = new List<Promotion>();

foreach (DataRow row in dataTable.Rows)

{

promotions.Add(MapRowToPromotion(row));

}

return promotions;

}

**Procedure cập nhật trạng thái khuyến mãi**

CREATE PROCEDURE sp\_UpdatePromotionStatus

@promotionId INT,

@status NVARCHAR(20)

AS

BEGIN

SET NOCOUNT ON;

UPDATE promotions

SET status = @status,

updated\_at = GETDATE()

WHERE promotion\_id = @promotionId;

END

GO

**Xử lý bên C#:**

public bool UpdatePromotionStatus(int promotionId, string status)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@status", status),

new SqlParameter("@promotionId", promotionId)

};

ExecuteNonQuery("sp\_UpdatePromotionStatus", parameters, isStoredProcedure: true);

return true;

}

catch

{

return false;

}

}

**Procedure cập nhật khuyến mãi**

CREATE PROCEDURE sp\_UpdatePromotion

@promotionId INT,

@name NVARCHAR(100),

@description NVARCHAR(MAX),

@discountType NVARCHAR(20),

@discountValue DECIMAL(18, 2),

@startDate DATE,

@endDate DATE,

@minOrderAmount DECIMAL(18, 2),

@maxDiscountAmount DECIMAL(18, 2),

@usageLimit INT = NULL,

@status NVARCHAR(20)

AS

BEGIN

SET NOCOUNT ON;

UPDATE promotions

SET promotion\_name = @name,

description = @description,

discount\_type = @discountType,

discount\_value = @discountValue,

start\_date = @startDate,

end\_date = @endDate,

min\_order\_amount = @minOrderAmount,

max\_discount\_amount = @maxDiscountAmount,

usage\_limit = @usageLimit,

status = @status,

updated\_at = GETDATE()

WHERE promotion\_id = @promotionId;

END

GO

**Xử lý bên C#:**

public bool UpdatePromotion(Promotion promotion)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@promotionId", promotion.PromotionId),

new SqlParameter("@name", promotion.PromotionName),

new SqlParameter("@description", promotion.Description ?? ""),

new SqlParameter("@discountType", promotion.DiscountType),

new SqlParameter("@discountValue", promotion.DiscountValue),

new SqlParameter("@startDate", promotion.StartDate.Date),

new SqlParameter("@endDate", promotion.EndDate.Date),

new SqlParameter("@minOrderAmount", promotion.MinOrderAmount),

new SqlParameter("@maxDiscountAmount", promotion.MaxDiscountAmount),

new SqlParameter("@usageLimit", promotion.UsageLimit > 0 ? (object)promotion.UsageLimit : DBNull.Value),

new SqlParameter("@status", promotion.Status ?? "active")

};

ExecuteNonQuery("sp\_UpdatePromotion", parameters, isStoredProcedure: true);

return true;

}

catch

{

return false;

}

}

**Function lấy thông tin thanh toán theo order id**

CREATE FUNCTION fn\_GetPaymentSummaryByOrderId (@orderId INT)

RETURNS TABLE

AS

RETURN

(

SELECT

o.order\_id,

o.order\_code,

o.total\_amount AS total\_order\_amount,

COALESCE(SUM(p.amount), 0) AS total\_paid\_amount,

(o.total\_amount - COALESCE(SUM(p.amount), 0)) AS remaining\_amount,

o.payment\_status,

COUNT(p.payment\_id) AS payment\_count,

MAX(p.payment\_date) AS last\_payment\_date

FROM orders o

LEFT JOIN payments p ON o.order\_id = p.order\_id AND p.status = 'successful'

WHERE o.order\_id = @orderId

GROUP BY o.order\_id, o.order\_code, o.total\_amount, o.payment\_status

);

GO

**Xử lý bên C#:**

public PaymentSummary GetPaymentSummaryByOrderId(int orderId)

{

try

{

string query = "SELECT \* FROM fn\_GetPaymentSummaryByOrderId(@orderId)";

var parameters = new SqlParameter[]

{

new SqlParameter("@orderId", orderId)

};

var dataTable = ExecuteQuery(query, parameters);

if (dataTable.Rows.Count > 0)

{

return MapRowToPaymentSummary(dataTable.Rows[0]);

}

return null;

}

catch (Exception ex)

{

throw new Exception($"Lỗi lấy tóm tắt thanh toán: {ex.Message}");

}

}

**3. Các chức năng báo cáo doanh thu**

**Function lấy báo cáo bán doanh thu theo ngày trong khoảng thời gian đã cho**

CREATE FUNCTION fn\_GetDailySalesReport

(

@fromDate DATE,

@toDate DATE

)

RETURNS TABLE

AS

RETURN

(

SELECT

report\_date,

total\_orders,

total\_revenue,

avg\_order\_value

FROM daily\_revenue\_report

WHERE report\_date BETWEEN @fromDate AND @toDate

AND total\_orders > 0

);

GO

**Xử lý bên C#:**

public List<DailySalesReport> GetDailySalesReport(DateTime fromDate, DateTime toDate)

{

string query = "SELECT \* FROM fn\_GetDailySalesReport(@fromDate, @toDate)";

var parameters = new SqlParameter[]

{

new SqlParameter("@fromDate", fromDate.Date),

new SqlParameter("@toDate", toDate.Date)

};

var dataTable = ExecuteQuery(query, parameters);

var reports = new List<DailySalesReport>();

foreach (DataRow row in dataTable.Rows)

{

reports.Add(new DailySalesReport

{

Date = Convert.ToDateTime(row["report\_date"]),

TotalOrders = Convert.ToInt32(row["total\_orders"]),

TotalRevenue = Convert.ToDecimal(row["total\_revenue"]),

AverageOrderValue = Convert.ToDecimal(row["avg\_order\_value"])

});

}

return reports;

}

**Procedure lấy danh sách sản phẩm bán chạy**

CREATE PROCEDURE sp\_GetTopSellingProducts

@fromDate DATE,

@toDate DATE,

@topCount INT

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP (@topCount)

product\_id,

product\_name,

brand\_name,

SUM(quantity) AS quantity\_sold,

SUM(total\_price) AS total\_revenue,

AVG(unit\_price) AS avg\_price,

COUNT(DISTINCT order\_id) AS total\_orders

FROM top\_selling\_products

WHERE CAST(order\_date AS DATE) BETWEEN @fromDate AND @toDate

GROUP BY product\_id, product\_name, brand\_name

ORDER BY quantity\_sold DESC;

END

GO

**Xử lý bên C#:**

public List<ProductSalesReport> GetTopSellingProducts(DateTime fromDate, DateTime toDate, int topCount = 10)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@topCount", topCount),

new SqlParameter("@fromDate", fromDate.Date),

new SqlParameter("@toDate", toDate.Date)

};

var dataTable = ExecuteQuery("sp\_GetTopSellingProducts", parameters, isStoredProcedure: true);

var reports = new List<ProductSalesReport>();

foreach (DataRow row in dataTable.Rows)

{

reports.Add(new ProductSalesReport

{

ProductId = Convert.ToInt32(row["product\_id"]),

ProductName = row["product\_name"].ToString(),

BrandName = row["brand\_name"].ToString(),

QuantitySold = Convert.ToInt32(row["quantity\_sold"]),

TotalRevenue = Convert.ToDecimal(row["total\_revenue"]),

AveragePrice = Convert.ToDecimal(row["avg\_price"]),

TotalOrders = Convert.ToInt32(row["total\_orders"])

});

}

return reports;

}

**Procedure lấy danh sách năng suất của nhân viên**

CREATE PROCEDURE sp\_GetEmployeePerformanceReport

@fromDate DATE,

@toDate DATE

AS

BEGIN

SET NOCOUNT ON;

SELECT

ep.employee\_id,

ep.full\_name,

ep.position,

COUNT(ep.order\_id) AS total\_orders,

COALESCE(SUM(ep.total\_amount), 0) AS total\_sales,

COALESCE(AVG(ep.total\_amount), 0) AS avg\_order\_value,

COALESCE(MAX(ep.order\_date), ep.created\_at) AS last\_sale\_date

FROM employee\_performance ep

WHERE

ep.order\_date IS NULL -- nhân viên chưa bán

OR CAST(ep.order\_date AS DATE) BETWEEN @fromDate AND @toDate

GROUP BY ep.employee\_id, ep.full\_name, ep.position, ep.created\_at

ORDER BY total\_sales DESC;

END

GO

**Xử lý bên C#:**

public List<EmployeePerformanceReport> GetEmployeePerformanceReport(DateTime fromDate, DateTime toDate)

{

var parameters = new SqlParameter[]

{

new SqlParameter("@fromDate", fromDate.Date),

new SqlParameter("@toDate", toDate.Date)

};

var dataTable = ExecuteQuery("sp\_GetEmployeePerformanceReport", parameters, isStoredProcedure: true);

var reports = new List<EmployeePerformanceReport>();

foreach (DataRow row in dataTable.Rows)

{

reports.Add(new EmployeePerformanceReport

{

EmployeeId = Convert.ToInt32(row["employee\_id"]),

EmployeeName = row["full\_name"].ToString(),

Position = row["position"].ToString(),

TotalOrders = Convert.ToInt32(row["total\_orders"]),

TotalSales = Convert.ToDecimal(row["total\_sales"]),

AverageOrderValue = Convert.ToDecimal(row["avg\_order\_value"]),

LastSaleDate = Convert.ToDateTime(row["last\_sale\_date"])

});

}

return reports;

}

**Procedure lấy dashboard**

CREATE PROCEDURE sp\_GetDashboardStats

AS

BEGIN

SET NOCOUNT ON;

DECLARE @today DATE = CAST(GETDATE() AS DATE);

DECLARE @month\_start DATE = DATEFROMPARTS(YEAR(GETDATE()), MONTH(GETDATE()), 1);

SELECT

-- Today's stats

COALESCE(SUM(CASE WHEN CAST(o.order\_date AS DATE) = @today AND o.order\_status = 'completed' THEN o.total\_amount END), 0) AS today\_revenue,

COALESCE(COUNT(CASE WHEN CAST(o.order\_date AS DATE) = @today AND o.order\_status = 'completed' THEN 1 END), 0) AS today\_orders,

-- Month stats

COALESCE(SUM(CASE WHEN CAST(o.order\_date AS DATE) >= @month\_start AND o.order\_status = 'completed' THEN o.total\_amount END), 0) AS month\_revenue,

COALESCE(COUNT(CASE WHEN CAST(o.order\_date AS DATE) >= @month\_start AND o.order\_status = 'completed' THEN 1 END), 0) AS month\_orders,

-- Other stats

COALESCE(AVG(CASE WHEN o.order\_status = 'completed' THEN o.total\_amount END), 0) AS avg\_order\_value,

(SELECT COUNT(\*) FROM customers) AS total\_customers,

(SELECT COUNT(\*) FROM inventory i WHERE i.quantity\_on\_hand <= i.min\_stock\_level) AS low\_stock\_products

FROM orders o;

END;

GO

**Xử lý bên C#:**

public DashboardStats GetDashboardStats()

{

string procedureName = "sp\_GetDashboardStats";

var dataTable = ExecuteQuery(procedureName, null, isStoredProcedure: true);

if (dataTable.Rows.Count > 0)

{

var row = dataTable.Rows[0];

return new DashboardStats

{

TodayRevenue = Convert.ToDecimal(row["today\_revenue"]),

TodayOrders = Convert.ToInt32(row["today\_orders"]),

MonthRevenue = Convert.ToDecimal(row["month\_revenue"]),

MonthOrders = Convert.ToInt32(row["month\_orders"]),

AverageOrderValue = Convert.ToDecimal(row["avg\_order\_value"]),

TotalCustomers = Convert.ToInt32(row["total\_customers"]),

LowStockProducts = Convert.ToInt32(row["low\_stock\_products"])

};

}

return new DashboardStats();

}

**CHƯƠNG 4: PHÂN QUYỀN**

**1. Tạo Role**

**Tạo role cho admin (admin cửa hàng)**

CREATE ROLE db\_admin;

GO

**Tạo role cho Manager (quản lý cửa hàng)**

CREATE ROLE db\_manager;

GO

**Tạo role cho Cashier (thu ngân)**

CREATE ROLE db\_cashier;

GO

**Tạo role cho Salesperson (nhân viên bán hàng)**

CREATE ROLE db\_salesperson;

GO

**Tạo role cho Staff (nhân viên cơ bản)**

CREATE ROLE db\_staff;

GO

**2. Phân quyền**

**2.1 Phân quyền cho admin role (toàn quyền)**

**Admin có toàn quyền trên tất cả tables**

GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON roles TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON users TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON user\_roles TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON customers TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON suppliers TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON categories TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON brands TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON products TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON inventory TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON promotions TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON orders TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON order\_details TO db\_admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON payments TO db\_admin;

**Admin có quyền thực thi tất cả stored procedures**

GRANT EXECUTE ON sp\_GetDashboardStats TO db\_admin;

GRANT EXECUTE ON sp\_GetCustomerByPhone TO db\_admin;

GRANT EXECUTE ON sp\_GetProductStock TO db\_admin;

GRANT EXECUTE ON sp\_GetProductById TO db\_admin;

GRANT EXECUTE ON sp\_InsertOrder TO db\_admin;

GRANT EXECUTE ON sp\_InsertOrderDetail TO db\_admin;

GRANT EXECUTE ON sp\_GetOrderById TO db\_admin;

GRANT EXECUTE ON sp\_GetOrderDetails TO db\_admin;

GRANT EXECUTE ON sp\_InsertPayment TO db\_admin;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_admin;

GRANT EXECUTE ON sp\_UpdateOrderStatus TO db\_admin;

GRANT EXECUTE ON sp\_UpdatePaymentStatus TO db\_admin;

GRANT EXECUTE ON sp\_GetTopSellingProducts TO db\_admin;

GRANT EXECUTE ON sp\_GetEmployeePerformanceReport TO db\_admin;

GRANT EXECUTE ON sp\_GetPromotionByCode TO db\_admin;

GRANT EXECUTE ON sp\_InsertPromotion TO db\_admin;

GRANT EXECUTE ON sp\_GetAllPromotions TO db\_admin;

GRANT EXECUTE ON sp\_UpdatePromotionStatus TO db\_admin;

GRANT EXECUTE ON sp\_DeletePromotion TO db\_admin;

GRANT EXECUTE ON sp\_UpdatePromotion TO db\_admin;

GRANT EXECUTE ON GetRevenueReport TO db\_admin;

GRANT EXECUTE ON LoginUser TO db\_admin;

GRANT EXECUTE ON sp\_CreateDatabaseUser TO db\_admin;

GRANT EXECUTE ON sp\_UpdateOrderPaymentStatus TO db\_admin;

GRANT EXECUTE ON sp\_DeletePayment TO db\_admin;

GRANT EXECUTE ON CheckUserRoles TO db\_admin;

**Admin có quyền truy cập tất cả views**

GRANT SELECT ON v\_AllActiveProducts TO db\_admin;

GRANT SELECT ON v\_ActiveEmployees TO db\_admin;

GRANT SELECT ON v\_ActivePromotions TO db\_admin;

GRANT SELECT ON daily\_revenue\_report TO db\_admin;

GRANT SELECT ON top\_selling\_products TO db\_admin;

GRANT SELECT ON employee\_performance TO db\_admin;

**Admin có quyền truy cập functions**

GRANT EXECUTE ON fn\_GenerateOrderCode TO db\_admin;

GRANT EXECUTE ON fn\_GenerateCustomerCode TO db\_admin;

GRANT SELECT ON fn\_GetPaymentSummaryByOrderId TO db\_admin;

GRANT SELECT ON fn\_GetAllOrders TO db\_admin;

GRANT SELECT ON fn\_GetDailySalesReport TO db\_admin;

**2.2 Phân quyền cho manager role**

**Manager có quyền đọc hầu hết các bảng**

GRANT SELECT ON employees TO db\_manager;

GRANT SELECT ON customers TO db\_manager;

GRANT SELECT ON suppliers TO db\_manager;

GRANT SELECT ON categories TO db\_manager;

GRANT SELECT ON brands TO db\_manager;

GRANT SELECT ON products TO db\_manager;

GRANT SELECT ON inventory TO db\_manager;

GRANT SELECT ON promotions TO db\_manager;

GRANT SELECT ON orders TO db\_manager;

GRANT SELECT ON order\_details TO db\_manager;

GRANT SELECT ON payments TO db\_manager;

GRANT SELECT ON roles TO db\_manager;

GRANT SELECT ON users TO db\_manager;

GRANT SELECT ON user\_roles TO db\_manager;

**Manager có quyền quản lý khách hàng và đơn hàng**

GRANT INSERT, UPDATE ON customers TO db\_manager;

GRANT INSERT, UPDATE ON orders TO db\_manager;

GRANT INSERT, UPDATE ON order\_details TO db\_manager;

GRANT INSERT, UPDATE ON payments TO db\_manager;

**Manager có quyền quản lý khuyến mãi và kho**

GRANT INSERT, UPDATE ON promotions TO db\_manager;

GRANT UPDATE ON inventory TO db\_manager;

**Manager có quyền thực thi stored procedures cần thiết**

GRANT EXECUTE ON sp\_GetDashboardStats TO db\_manager;

GRANT EXECUTE ON sp\_GetCustomerByPhone TO db\_manager;

GRANT EXECUTE ON sp\_GetProductStock TO db\_manager;

GRANT EXECUTE ON sp\_GetProductById TO db\_manager;

GRANT EXECUTE ON sp\_InsertOrder TO db\_manager;

GRANT EXECUTE ON sp\_InsertOrderDetail TO db\_manager;

GRANT EXECUTE ON sp\_GetOrderById TO db\_manager;

GRANT EXECUTE ON sp\_GetOrderDetails TO db\_manager;

GRANT EXECUTE ON sp\_InsertPayment TO db\_manager;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_manager;

GRANT EXECUTE ON sp\_UpdateOrderStatus TO db\_manager;

GRANT EXECUTE ON sp\_UpdatePaymentStatus TO db\_manager;

GRANT EXECUTE ON sp\_GetTopSellingProducts TO db\_manager;

GRANT EXECUTE ON sp\_GetEmployeePerformanceReport TO db\_manager;

GRANT EXECUTE ON sp\_GetPromotionByCode TO db\_manager;

GRANT EXECUTE ON sp\_GetAllPromotions TO db\_manager;

GRANT EXECUTE ON sp\_InsertPromotion TO db\_manager;

GRANT EXECUTE ON sp\_UpdatePromotionStatus TO db\_manager;

GRANT EXECUTE ON sp\_DeletePromotion TO db\_manager;

GRANT EXECUTE ON sp\_UpdatePromotion TO db\_manager;

GRANT EXECUTE ON sp\_UpdateOrderPaymentStatus TO db\_manager;

GRANT EXECUTE ON sp\_DeletePayment TO db\_manager;

GRANT EXECUTE ON GetRevenueReport TO db\_manager;

GRANT EXECUTE ON CheckUserRoles TO db\_manager;

GRANT EXECUTE ON LoginUser TO db\_manager;

GRANT EXECUTE ON sp\_CreateDatabaseUser TO db\_manager;

**Manager có quyền truy cập tất cả views**

GRANT SELECT ON v\_AllActiveProducts TO db\_manager;

GRANT SELECT ON v\_ActiveEmployees TO db\_manager;

GRANT SELECT ON v\_ActivePromotions TO db\_manager;

GRANT SELECT ON daily\_revenue\_report TO db\_manager;

GRANT SELECT ON top\_selling\_products TO db\_manager;

GRANT SELECT ON employee\_performance TO db\_manager;

GRANT SELECT ON profit\_report TO db\_manager;

**Manager có quyền truy cập functions**

GRANT EXECUTE ON fn\_GenerateOrderCode TO db\_manager;

GRANT EXECUTE ON fn\_GenerateCustomerCode TO db\_manager;

GRANT SELECT ON fn\_GetPaymentSummaryByOrderId TO db\_manager;

GRANT SELECT ON fn\_GetAllOrders TO db\_manager;

GRANT SELECT ON fn\_GetDailySalesReport TO db\_manager;

**2.3 Phân quyền cho cashier (thu ngân)**

**Cashier có quyền SELECT cơ bản cho bán hàng**

GRANT SELECT ON employees TO db\_cashier;

GRANT SELECT ON customers TO db\_cashier;

GRANT SELECT ON suppliers TO db\_cashier;

GRANT SELECT ON categories TO db\_cashier;

GRANT SELECT ON brands TO db\_cashier;

GRANT SELECT ON products TO db\_cashier;

GRANT SELECT ON inventory TO db\_cashier;

GRANT SELECT ON promotions TO db\_cashier;

GRANT SELECT ON orders TO db\_cashier;

GRANT SELECT ON order\_details TO db\_cashier;

GRANT SELECT ON payments TO db\_cashier;

GRANT SELECT ON roles TO db\_cashier;

GRANT SELECT ON users TO db\_cashier;

GRANT SELECT ON user\_roles TO db\_cashier;

**Cashier có quyền INSERT/UPDATE cho bán hàng**

GRANT INSERT ON customers TO db\_cashier;

GRANT INSERT, UPDATE ON orders TO db\_cashier;

GRANT INSERT ON order\_details TO db\_cashier;

GRANT INSERT ON payments TO db\_cashier;

**Cashier có các stored procedures cần thiết cho bán hàng**

GRANT EXECUTE ON sp\_GetCustomerByPhone TO db\_cashier;

GRANT EXECUTE ON sp\_GetProductById TO db\_cashier;

GRANT EXECUTE ON sp\_GetProductStock TO db\_cashier;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_cashier;

GRANT EXECUTE ON sp\_InsertOrder TO db\_cashier;

GRANT EXECUTE ON sp\_InsertOrderDetail TO db\_cashier;

GRANT EXECUTE ON sp\_InsertPayment TO db\_cashier;

GRANT EXECUTE ON sp\_GetOrderById TO db\_cashier;

GRANT EXECUTE ON sp\_GetOrderDetails TO db\_cashier;

GRANT EXECUTE ON sp\_InsertPayment TO db\_cashier;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_cashier;

GRANT EXECUTE ON sp\_UpdateOrderStatus TO db\_cashier;

GRANT EXECUTE ON sp\_UpdatePaymentStatus TO db\_cashier;

GRANT EXECUTE ON sp\_GetPromotionByCode TO db\_cashier;

GRANT EXECUTE ON sp\_GetAllPromotions TO db\_cashier;

GRANT EXECUTE ON CheckUserRoles TO db\_cashier;

GRANT EXECUTE ON LoginUser TO db\_cashier;

GRANT EXECUTE ON sp\_CreateDatabaseUser TO db\_cashier;

**Cashier có các functions cần thiết**

GRANT EXECUTE ON fn\_GenerateOrderCode TO db\_cashier;

GRANT EXECUTE ON fn\_GenerateCustomerCode TO db\_cashier;

GRANT SELECT ON fn\_GetPaymentSummaryByOrderId TO db\_cashier;

GRANT SELECT ON fn\_GetAllOrders TO db\_cashier;

**Cashier có views cần thiết**

GRANT SELECT ON v\_AllActiveProducts TO db\_cashier;

GRANT SELECT ON v\_ActiveEmployees TO db\_cashier;

GRANT SELECT ON v\_ActivePromotions TO db\_cashier;

**2.4 Phân quyền cho salesperson (bán hàng + báo cáo cơ bản)**

**Salesperson kế thừa tất cả quyền của Cashier**

GRANT SELECT ON employees TO db\_salesperson;

GRANT SELECT ON customers TO db\_salesperson;

GRANT SELECT ON suppliers TO db\_salesperson;

GRANT SELECT ON categories TO db\_salesperson;

GRANT SELECT ON brands TO db\_salesperson;

GRANT SELECT ON products TO db\_salesperson;

GRANT SELECT ON inventory TO db\_salesperson;

GRANT SELECT ON promotions TO db\_salesperson;

GRANT SELECT ON orders TO db\_salesperson;

GRANT SELECT ON order\_details TO db\_salesperson;

GRANT SELECT ON payments TO db\_salesperson;

GRANT SELECT ON roles TO db\_salesperson;

GRANT SELECT ON users TO db\_salesperson;

GRANT SELECT ON user\_roles TO db\_salesperson;

GRANT INSERT ON customers TO db\_salesperson;

GRANT INSERT, UPDATE ON orders TO db\_salesperson;

GRANT INSERT ON order\_details TO db\_salesperson;

GRANT INSERT ON payments TO db\_salesperson;

**Salesperson có các stored procedures bán hàng**

GRANT EXECUTE ON sp\_GetCustomerByPhone TO db\_salesperson;

GRANT EXECUTE ON sp\_GetProductById TO db\_salesperson;

GRANT EXECUTE ON sp\_GetProductStock TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertOrder TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertOrderDetail TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertPayment TO db\_salesperson;

GRANT EXECUTE ON sp\_GetOrderById TO db\_salesperson;

GRANT EXECUTE ON sp\_GetOrderDetails TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertPayment TO db\_salesperson;

GRANT EXECUTE ON sp\_InsertCustomer TO db\_salesperson;

GRANT EXECUTE ON sp\_UpdateOrderStatus TO db\_salesperson;

GRANT EXECUTE ON sp\_UpdatePaymentStatus TO db\_salesperson;

GRANT EXECUTE ON sp\_GetPromotionByCode TO db\_salesperson;

GRANT EXECUTE ON sp\_GetAllPromotions TO db\_salesperson;

GRANT EXECUTE ON CheckUserRoles TO db\_salesperson;

GRANT EXECUTE ON LoginUser TO db\_salesperson;

GRANT EXECUTE ON sp\_CreateDatabaseUser TO db\_salesperson;

**Salesperson có các stored procedures báo cáo doanh thu**

GRANT EXECUTE ON sp\_GetDashboardStats TO db\_salesperson;

GRANT EXECUTE ON sp\_GetTopSellingProducts TO db\_salesperson;

**Salesperson có functions**

GRANT EXECUTE ON fn\_GenerateOrderCode TO db\_salesperson;

GRANT EXECUTE ON fn\_GenerateCustomerCode TO db\_salesperson;

GRANT SELECT ON fn\_GetPaymentSummaryByOrderId TO db\_salesperson;

GRANT SELECT ON fn\_GetDailySalesReport TO db\_salesperson;

GRANT SELECT ON fn\_GetAllOrders TO db\_salesperson;

**Salesperson có các views cần thiết**

GRANT SELECT ON v\_AllActiveProducts TO db\_salesperson;

GRANT SELECT ON v\_ActiveEmployees TO db\_salesperson;

GRANT SELECT ON v\_ActivePromotions TO db\_salesperson;

GRANT SELECT ON daily\_revenue\_report TO db\_salesperson;

GRANT SELECT ON top\_selling\_products TO db\_salesperson;

**2.5 Phân quyền cho staff role**

**Staff chỉ có quyền đọc thông tin cơ bản**

GRANT SELECT ON employees TO db\_staff;

GRANT SELECT ON customers TO db\_staff;

GRANT SELECT ON suppliers TO db\_staff;

GRANT SELECT ON categories TO db\_staff;

GRANT SELECT ON brands TO db\_staff;

GRANT SELECT ON products TO db\_staff;

GRANT SELECT ON inventory TO db\_staff;

GRANT SELECT ON promotions TO db\_staff;

GRANT SELECT ON orders TO db\_staff;

GRANT SELECT ON order\_details TO db\_staff;

GRANT SELECT ON payments TO db\_staff;

GRANT SELECT ON roles TO db\_staff;

GRANT SELECT ON users TO db\_staff;

GRANT SELECT ON user\_roles TO db\_staff;

**Staff có quyền thực thi một số stored procedures cơ bản**

GRANT EXECUTE ON sp\_GetCustomerByPhone TO db\_staff;

GRANT EXECUTE ON sp\_GetProductStock TO db\_staff;

GRANT EXECUTE ON sp\_GetProductById TO db\_staff;

GRANT EXECUTE ON sp\_GetOrderById TO db\_staff;

GRANT EXECUTE ON sp\_GetOrderDetails TO db\_staff;

GRANT EXECUTE ON sp\_GetPromotionByCode TO db\_staff;

GRANT EXECUTE ON CheckUserRoles TO db\_staff;

GRANT EXECUTE ON LoginUser TO db\_staff;

GRANT EXECUTE ON sp\_CreateDatabaseUser TO db\_staff;

**Staff có quyền truy cập view sản phẩm**

GRANT SELECT ON v\_AllActiveProducts TO db\_staff;

GRANT SELECT ON v\_ActiveEmployees TO db\_staff;

GRANT SELECT ON v\_ActivePromotions TO db\_staff;

**3. Các chức năng đăng nhập và phân quyền**

**Procedure đăng nhập**

CREATE PROCEDURE LoginUser

@username NVARCHAR(50),

@password NVARCHAR(256),

@result INT OUTPUT, -- 1: Đăng nhập thành công, 0: Thất bại

@user\_id INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @password\_hash NVARCHAR(64) = dbo.HashPassword (@password);

SELECT @user\_id = user\_id

FROM users

WHERE username = @username

AND password\_hash = @password\_hash

AND status = 'active';

IF @user\_id IS NOT NULL

BEGIN

-- Cập nhật thời gian đăng nhập cuối

UPDATE users

SET last\_login = GETDATE()

WHERE user\_id = @user\_id;

SET @result = 1;

END

ELSE

BEGIN

SET @result = 0;

END

END;

GO

**Xử lý bên C#:**

public User LoginUser(string username, string password)

{

try

{

int result = 0, userId = 0;

var parameters = new SqlParameter[]

{

new SqlParameter("@username", username),

new SqlParameter("@password", password),

new SqlParameter("@result", SqlDbType.Int) { Direction = ParameterDirection.Output },

new SqlParameter("@user\_id", SqlDbType.Int) { Direction = ParameterDirection.Output }

};

using (var conn = GetConnection())

{

conn.Open();

using (var cmd = new SqlCommand("LoginUser", conn))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddRange(parameters);

cmd.ExecuteNonQuery();

result = (int)cmd.Parameters["@result"].Value;

userId = cmd.Parameters["@user\_id"].Value != DBNull.Value ?

(int)cmd.Parameters["@user\_id"].Value : 0;

}

}

if (result == 1 && userId > 0)

{

return GetUserById(userId);

}

return null;

}

catch (Exception ex)

{

throw new Exception($"Lỗi đăng nhập: {ex.Message}");

}

}

**Procedure kiểm tra vai trò của người dùng**

CREATE PROCEDURE CheckUserRoles

@user\_id INT

AS

BEGIN

SELECT r.role\_name, r.description

FROM user\_roles ur

INNER JOIN roles r ON ur.role\_id = r.role\_id

WHERE ur.user\_id = @user\_id;

END;

GO

**Xử lý bên C#:**

public List<string> GetUserRoles(int userId)

{

var roles = new List<string>();

using (var conn = GetConnection())

{

conn.Open();

using (var cmd = new SqlCommand("CheckUserRoles", conn))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@user\_id", userId);

using (var reader = cmd.ExecuteReader())

{

while (reader.Read())

{

roles.Add(reader["role\_name"].ToString());

}

}

}

}

return roles;

}

**Function hash password**

CREATE FUNCTION dbo.HashPassword (@password NVARCHAR(256))

RETURNS NVARCHAR(64)

AS

BEGIN

DECLARE @hash NVARCHAR(64);

-- Always cast to VARCHAR to avoid NVARCHAR vs VARCHAR mismatches

SET @hash = CONVERT(NVARCHAR(64), HASHBYTES('SHA2\_256', CAST(@password AS VARCHAR(256))), 2);

RETURN @hash;

END;

GO

**Procedure tạo user và phân quyền**

CREATE PROCEDURE sp\_CreateDatabaseUser

@username NVARCHAR(50),

@password NVARCHAR(256),

@role\_name NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Tạo login

DECLARE @sql NVARCHAR(MAX);

SET @sql = 'CREATE LOGIN [' + @username + '] WITH PASSWORD = ''' + @password + '''';

EXEC sp\_executesql @sql;

-- Tạo user trong database

SET @sql = 'CREATE USER [' + @username + '] FOR LOGIN [' + @username + ']';

EXEC sp\_executesql @sql;

-- Phân quyền role

DECLARE @database\_role NVARCHAR(50);

SET @database\_role = CASE @role\_name

WHEN 'Admin' THEN 'db\_admin'

WHEN 'Manager' THEN 'db\_manager'

WHEN 'Cashier' THEN 'db\_cashier'

WHEN 'Salesperson' THEN 'db\_salesperson'

WHEN 'Staff' THEN 'db\_staff'

ELSE NULL

END;

IF @database\_role IS NOT NULL

BEGIN

SET @sql = 'ALTER ROLE ' + @database\_role + ' ADD MEMBER [' + @username + ']';

EXEC sp\_executesql @sql;

END

SELECT 1 as Result, 'User created and permissions assigned successfully' as Message;

END TRY

BEGIN CATCH

SELECT 0 as Result, ERROR\_MESSAGE() as Message;

END CATCH

END;

GO

**Trigger tự động tạo tài khoản người dùng khi thêm nhân viên**

CREATE TRIGGER tr\_employees\_insert

ON employees

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

SET XACT\_ABORT OFF; -- Cho phép xử lý lỗi tốt hơn

DECLARE @employee\_id INT, @employee\_code NVARCHAR(20), @username NVARCHAR(50);

DECLARE @default\_password NVARCHAR(50) = 'default123';

DECLARE @position NVARCHAR(50);

DECLARE @role\_name NVARCHAR(50);

DECLARE employee\_cursor CURSOR FOR

SELECT employee\_id, employee\_code, position FROM inserted;

OPEN employee\_cursor;

FETCH NEXT FROM employee\_cursor INTO @employee\_id, @employee\_code, @position;

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN TRY

-- Tạo username từ employee\_code

SET @username = 'user\_' + @employee\_code;

-- Xác định role name từ position

SET @role\_name = CASE

WHEN @position = N'Admin' THEN 'Admin'

WHEN @position = N'Quản lý' THEN 'Manager'

WHEN @position = N'Thu ngân' THEN 'Cashier'

WHEN @position = N'Bán hàng' THEN 'Salesperson'

ELSE 'Staff'

END;

-- Thêm vào bảng users (dùng mật khẩu đã hash để lưu riêng cho ứng dụng)

INSERT INTO users (username, password\_hash, employee\_id, status)

VALUES (

@username,

dbo.HashPassword(@default\_password),

@employee\_id,

'active'

);

-- Gán vai trò (ghi vào bảng user\_roles)

DECLARE @role\_id INT;

SELECT @role\_id = role\_id FROM roles WHERE role\_name = @role\_name;

IF @role\_id IS NOT NULL

BEGIN

INSERT INTO user\_roles (user\_id, role\_id)

SELECT user\_id, @role\_id FROM users WHERE employee\_id = @employee\_id;

END

-- Gọi stored procedure để tạo SQL login/user (với xử lý lỗi)

BEGIN TRY

EXEC sp\_CreateDatabaseUser

@username = @username,

@password = @default\_password,

@role\_name = @role\_name;

END TRY

BEGIN CATCH

-- Nếu tạo database user thất bại, chỉ log lỗi nhưng không làm fail toàn bộ transaction

PRINT 'Warning: Failed to create database user for ' + @username + ': ' + ERROR\_MESSAGE();

END CATCH

END TRY

BEGIN CATCH

-- Nếu có lỗi nghiêm trọng, log và tiếp tục với employee tiếp theo

PRINT 'Error processing employee ' + @employee\_code + ': ' + ERROR\_MESSAGE();

END CATCH

FETCH NEXT FROM employee\_cursor INTO @employee\_id, @employee\_code, @position;

END;

CLOSE employee\_cursor;

DEALLOCATE employee\_cursor;

END;

GO

**CHƯƠNG 5: TRANSACTION**

**Transaction thêm order**

CREATE PROCEDURE sp\_InsertOrder

@order\_code NVARCHAR(50),

@customer\_id INT = NULL,

@employee\_id INT = NULL,

@order\_date DATETIME,

@subtotal DECIMAL(18,2),

@discount\_amount DECIMAL(18,2),

@tax\_amount DECIMAL(18,2),

@total\_amount DECIMAL(18,2),

@promotion\_id INT = NULL,

@payment\_method NVARCHAR(50),

@payment\_status NVARCHAR(50),

@order\_status NVARCHAR(50),

@notes NVARCHAR(MAX),

@order\_id INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO orders (

order\_code, customer\_id, employee\_id, order\_date,

subtotal, discount\_amount, tax\_amount, total\_amount,

promotion\_id, payment\_method, payment\_status, order\_status, notes

)

VALUES (

@order\_code, @customer\_id, @employee\_id, @order\_date,

@subtotal, @discount\_amount, @tax\_amount, @total\_amount,

@promotion\_id, @payment\_method, @payment\_status, @order\_status, @notes

);

SET @order\_id = SCOPE\_IDENTITY();

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

THROW;

END CATCH

END

**Transaction thêm order\_details**

CREATE PROCEDURE sp\_InsertOrderDetail

@order\_id INT,

@product\_id INT,

@quantity INT,

@unit\_price DECIMAL(18,2),

@discount\_per\_item DECIMAL(18,2),

@total\_price DECIMAL(18,2)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO order\_details (

order\_id, product\_id, quantity, unit\_price, discount\_per\_item, total\_price

)

VALUES (

@order\_id, @product\_id, @quantity, @unit\_price, @discount\_per\_item, @total\_price

);

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

THROW;

END CATCH

END

GO

**Xử lý bên C#:**

public int InsertOrder(Order order, List<OrderDetail> orderDetails)

{

using (var conn = GetConnection())

{

conn.Open();

using (var transaction = conn.BeginTransaction())

{

try

{

// 1. Insert Order using stored procedure

using (var cmd = new SqlCommand("sp\_InsertOrder", conn, transaction))

{

cmd.CommandType = CommandType.StoredProcedure;

var orderIdParam = new SqlParameter("@order\_id", SqlDbType.Int)

{

Direction = ParameterDirection.Output

};

cmd.Parameters.AddRange(new SqlParameter[]

{

new SqlParameter("@order\_code", order.OrderCode),

new SqlParameter("@customer\_id", (object)order.CustomerId ?? DBNull.Value),

new SqlParameter("@employee\_id", (object)order.EmployeeId ?? DBNull.Value),

new SqlParameter("@order\_date", order.OrderDate),

new SqlParameter("@subtotal", order.Subtotal),

new SqlParameter("@discount\_amount", order.DiscountAmount),

new SqlParameter("@tax\_amount", order.TaxAmount),

new SqlParameter("@total\_amount", order.TotalAmount),

new SqlParameter("@promotion\_id", (object)order.PromotionId ?? DBNull.Value),

new SqlParameter("@payment\_method", order.PaymentMethod),

new SqlParameter("@payment\_status", order.PaymentStatus),

new SqlParameter("@order\_status", order.OrderStatus),

new SqlParameter("@notes", order.Notes ?? ""),

orderIdParam

});

cmd.ExecuteNonQuery();

int orderId = Convert.ToInt32(orderIdParam.Value);

// 2. Insert Order Details

foreach (var detail in orderDetails)

{

using (var detailCmd = new SqlCommand("sp\_InsertOrderDetail", conn, transaction))

{

detailCmd.CommandType = CommandType.StoredProcedure;

detailCmd.Parameters.AddRange(new SqlParameter[]

{

new SqlParameter("@order\_id", orderId),

new SqlParameter("@product\_id", detail.ProductId),

new SqlParameter("@quantity", detail.Quantity),

new SqlParameter("@unit\_price", detail.UnitPrice),

new SqlParameter("@discount\_per\_item", detail.DiscountPerItem),

new SqlParameter("@total\_price", detail.TotalPrice)

});

detailCmd.ExecuteNonQuery();

}

}

transaction.Commit();

return orderId;

}

}

catch

{

transaction.Rollback();

throw;

}

}

}

}

**Transaction thêm thanh toán**

CREATE PROCEDURE sp\_InsertPayment

@order\_id INT,

@payment\_date DATETIME,

@payment\_method NVARCHAR(50),

@amount DECIMAL(18,2),

@reference\_number NVARCHAR(100),

@status NVARCHAR(50),

@notes NVARCHAR(MAX),

@payment\_id INT OUTPUT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO payments (

order\_id, payment\_date, payment\_method, amount,

reference\_number, status, notes

)

VALUES (

@order\_id, @payment\_date, @payment\_method, @amount,

@reference\_number, @status, @notes

);

SET @payment\_id = SCOPE\_IDENTITY();

-- Cập nhật trạng thái thanh toán của đơn hàng

DECLARE @total\_paid DECIMAL(18,2) = (

SELECT SUM(amount)

FROM payments

WHERE order\_id = @order\_id AND status = 'successful'

);

DECLARE @order\_total DECIMAL(18,2) = (

SELECT total\_amount FROM orders WHERE order\_id = @order\_id

);

UPDATE orders

SET payment\_status =

CASE

WHEN @total\_paid >= @order\_total THEN 'paid'

WHEN @total\_paid > 0 THEN 'partial'

ELSE 'unpaid'

END

WHERE order\_id = @order\_id;

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

THROW;

END CATCH

END

GO

**Xử lý bên C#:**

public int InsertPayment(Payment payment)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@order\_id", payment.OrderId),

new SqlParameter("@payment\_date", payment.PaymentDate),

new SqlParameter("@payment\_method", payment.PaymentMethod),

new SqlParameter("@amount", payment.Amount),

new SqlParameter("@reference\_number", payment.ReferenceNumber ?? ""),

new SqlParameter("@status", payment.Status),

new SqlParameter("@notes", payment.Notes ?? ""),

new SqlParameter

{

ParameterName = "@payment\_id",

SqlDbType = SqlDbType.Int,

Direction = ParameterDirection.Output

}

};

ExecuteNonQuery("sp\_InsertPayment", parameters, isStoredProcedure: true);

return Convert.ToInt32(parameters.First(p => p.ParameterName == "@payment\_id").Value);

}

catch (Exception ex)

{

throw new Exception($"Lỗi thêm thanh toán: {ex.Message}");

}

}

**Transaction xóa thanh toán**

CREATE PROCEDURE sp\_DeletePayment

@paymentId INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

-- Lấy order\_id từ payment

DECLARE @orderId INT;

SELECT @orderId = order\_id

FROM payments

WHERE payment\_id = @paymentId;

IF @orderId IS NULL

BEGIN

THROW 50001, 'Không tìm thấy thanh toán để xóa.', 1;

END

-- Xóa payment

DELETE FROM payments

WHERE payment\_id = @paymentId;

-- Gọi cập nhật trạng thái thanh toán

EXEC sp\_UpdateOrderPaymentStatus @orderId;

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

DECLARE @msg NVARCHAR(4000) = ERROR\_MESSAGE();

THROW 50002, @msg, 1;

END CATCH

END

GO

**Xử lý bên C#:**

public bool DeletePayment(int paymentId)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@paymentId", paymentId)

};

ExecuteNonQuery("sp\_DeletePayment", parameters, isStoredProcedure: true);

return true;

}

catch (Exception ex)

{

throw new Exception($"Lỗi xóa thanh toán: {ex.Message}");

} }

**Transaction thêm khách hàng**

CREATE PROCEDURE sp\_InsertCustomer

@customer\_code NVARCHAR(50),

@full\_name NVARCHAR(100),

@phone NVARCHAR(20),

@email NVARCHAR(100),

@address NVARCHAR(255),

@date\_of\_birth DATE = NULL,

@gender NVARCHAR(10),

@customer\_type NVARCHAR(50)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO customers (

customer\_code, full\_name, phone, email, address,

date\_of\_birth, gender, customer\_type

)

VALUES (

@customer\_code, @full\_name, @phone, @email, @address,

@date\_of\_birth, @gender, @customer\_type

);

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

THROW;

END CATCH

END

GO

**Xử lý bên C#:**

public bool InsertCustomer(Customer customer)

{

customer.CustomerCode = GenerateCustomerCode();

var parameters = new SqlParameter[]

{

new SqlParameter("@customer\_code", customer.CustomerCode),

new SqlParameter("@full\_name", customer.FullName),

new SqlParameter("@phone", customer.Phone ?? ""),

new SqlParameter("@email", customer.Email ?? ""),

new SqlParameter("@address", customer.Address ?? ""),

new SqlParameter("@date\_of\_birth", (object)customer.DateOfBirth ?? DBNull.Value),

new SqlParameter("@gender", customer.Gender ?? ""),

new SqlParameter("@customer\_type", customer.CustomerType ?? "regular")

};

return ExecuteNonQuery("sp\_InsertCustomer", parameters, isStoredProcedure: true) > 0;

}

**Transaction thêm khuyến mãi**

CREATE PROCEDURE sp\_InsertPromotion

@code NVARCHAR(50),

@name NVARCHAR(100),

@description NVARCHAR(MAX),

@discountType NVARCHAR(20),

@discountValue DECIMAL(18, 2),

@startDate DATE,

@endDate DATE,

@minOrderAmount DECIMAL(18, 2),

@maxDiscountAmount DECIMAL(18, 2),

@usageLimit INT = NULL,

@status NVARCHAR(20)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

INSERT INTO promotions (

promotion\_code, promotion\_name, description, discount\_type,

discount\_value, start\_date, end\_date, min\_order\_amount,

max\_discount\_amount, usage\_limit, status

)

VALUES (

@code, @name, @description, @discountType, @discountValue,

@startDate, @endDate, @minOrderAmount, @maxDiscountAmount, @usageLimit, @status

);

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

DECLARE @err NVARCHAR(4000) = ERROR\_MESSAGE();

THROW 50001, @err, 1;

END CATCH

END

GO

**Xử lý bên C#:**

public bool InsertPromotion(Promotion promotion)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@code", promotion.PromotionCode),

new SqlParameter("@name", promotion.PromotionName),

new SqlParameter("@description", promotion.Description ?? ""),

new SqlParameter("@discountType", promotion.DiscountType),

new SqlParameter("@discountValue", promotion.DiscountValue),

new SqlParameter("@startDate", promotion.StartDate.Date),

new SqlParameter("@endDate", promotion.EndDate.Date),

new SqlParameter("@minOrderAmount", promotion.MinOrderAmount),

new SqlParameter("@maxDiscountAmount", promotion.MaxDiscountAmount),

new SqlParameter("@usageLimit", promotion.UsageLimit > 0 ? (object)promotion.UsageLimit : DBNull.Value),

new SqlParameter("@status", promotion.Status ?? "active")

};

ExecuteNonQuery("sp\_InsertPromotion", parameters, isStoredProcedure: true);

return true;

}

catch

{

return false;

}

}

**Transaction xóa khuyến mãi**

CREATE PROCEDURE sp\_DeletePromotion

@promotionId INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

DELETE FROM promotions

WHERE promotion\_id = @promotionId;

COMMIT;

END TRY

BEGIN CATCH

ROLLBACK;

THROW;

END CATCH

END

GO

**Xử lý bên C#:**

public bool DeletePromotion(int promotionId)

{

try

{

var parameters = new SqlParameter[]

{

new SqlParameter("@promotionId", promotionId)

};

ExecuteNonQuery("sp\_DeletePromotion", parameters, isStoredProcedure: true);

return true;

}

catch

{

return false;

}

}

**CHƯƠNG 6: THIẾT KẾ HỆ THỐNG VÀ GIAO DIỆN**

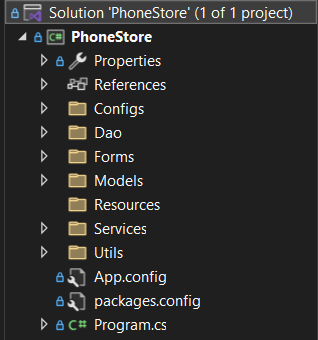
**1. Môi trường cài đặt và công nghệ sử dụng**

- Hệ cơ sở dữ liệu Microsoft SQL Server Management Studio 20.

- Chương trình Winform được viết bằng Microsoft Visual Studio 2022 - trên nền tảng ASP.NET Web Application (.NET Framework) C#.

**2. Cấu trúc thiết kế hệ thống**

**Tổng quan cấu trúc dự án**

****

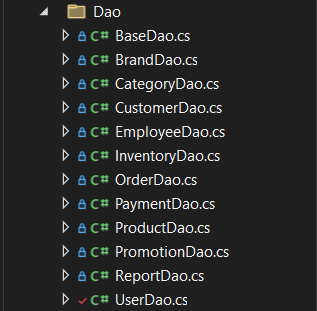
Dự án **PhoneStore** được xây dựng dưới dạng một ứng dụng **Windows Forms (WinForms)** trong ngôn ngữ C#. Cấu trúc thư mục của hệ thống được tổ chức khoa học, theo phong cách **3 tầng (3-tier architecture)**: giao diện người dùng (UI), tầng nghiệp vụ (Business Logic) và tầng truy xuất dữ liệu (Data Access). Điều này giúp hệ thống dễ dàng mở rộng, bảo trì và kiểm soát các luồng xử lý dữ liệu một cách rõ ràng.

**Thư mục Configs**

****

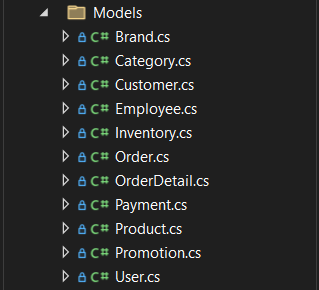
**Configs** là nơi lưu trữ các tệp cấu hình tùy chỉnh như cấu hình kết nối cơ sở dữ liệu, các thông tin tĩnh như địa chỉ email hỗ trợ, đường dẫn lưu trữ tệp, API key,...

**Thư mục DAO (Data Access Object)**

****

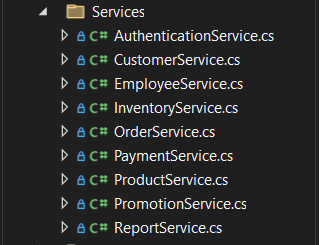
Thư mục Dao chứa các lớp chuyên phụ trách **tương tác với cơ sở dữ liệu**. Mỗi lớp DAO thường tương ứng với một bảng hoặc thực thể trong cơ sở dữ liệu như ProductDao, CustomerDao,… Các thao tác như thêm, sửa, xóa, tìm kiếm đều được định nghĩa tại đây. Đây là tầng **gần nhất với database**.

**Thư mục Models**

****

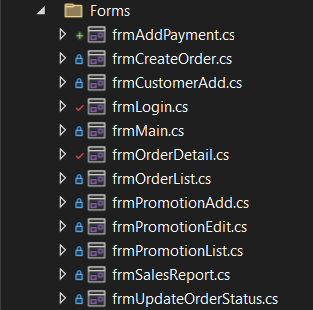
Models định nghĩa các **class mô hình dữ liệu (data model/entity)** đại diện cho các bảng trong cơ sở dữ liệu như User, Order, Product,… Những class này thường chỉ chứa thuộc tính (property) tương ứng với các cột trong bảng dữ liệu, không chứa logic xử lý.

**Thư mục Services**

****

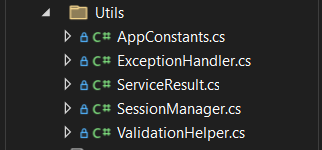
Tầng Services đóng vai trò **xử lý nghiệp vụ** của hệ thống. Đây là nơi trung gian giữa tầng DAO và Forms. Ví dụ: OrderService có thể xử lý việc tính tổng tiền đơn hàng, kiểm tra tồn kho trước khi gọi OrderDao để lưu đơn hàng xuống database. Cách tổ chức này giúp mã nguồn tách biệt giữa thao tác dữ liệu và logic kinh doanh.

**Thư mục Forms**

****

Forms là nơi chứa **giao diện người dùng** – các màn hình WinForms của ứng dụng. Mỗi Form có thể tương ứng với một chức năng như: đăng nhập (LoginForm), quản lý sản phẩm (ProductForm), thống kê đơn hàng,… Từ các Form này, người dùng tương tác với hệ thống và gọi đến các phương thức trong Service để xử lý.

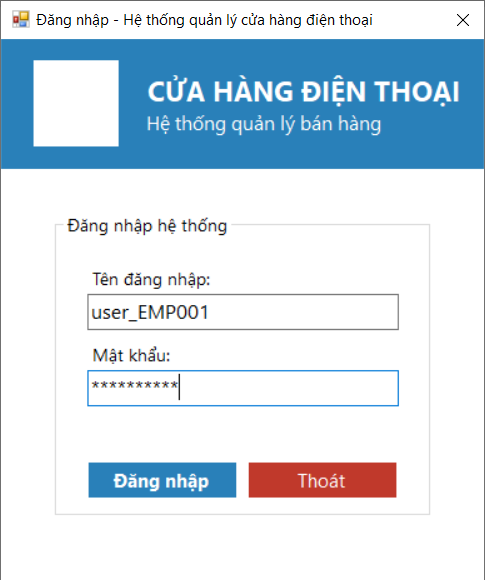
**Thư mục Resources và Utils**

****

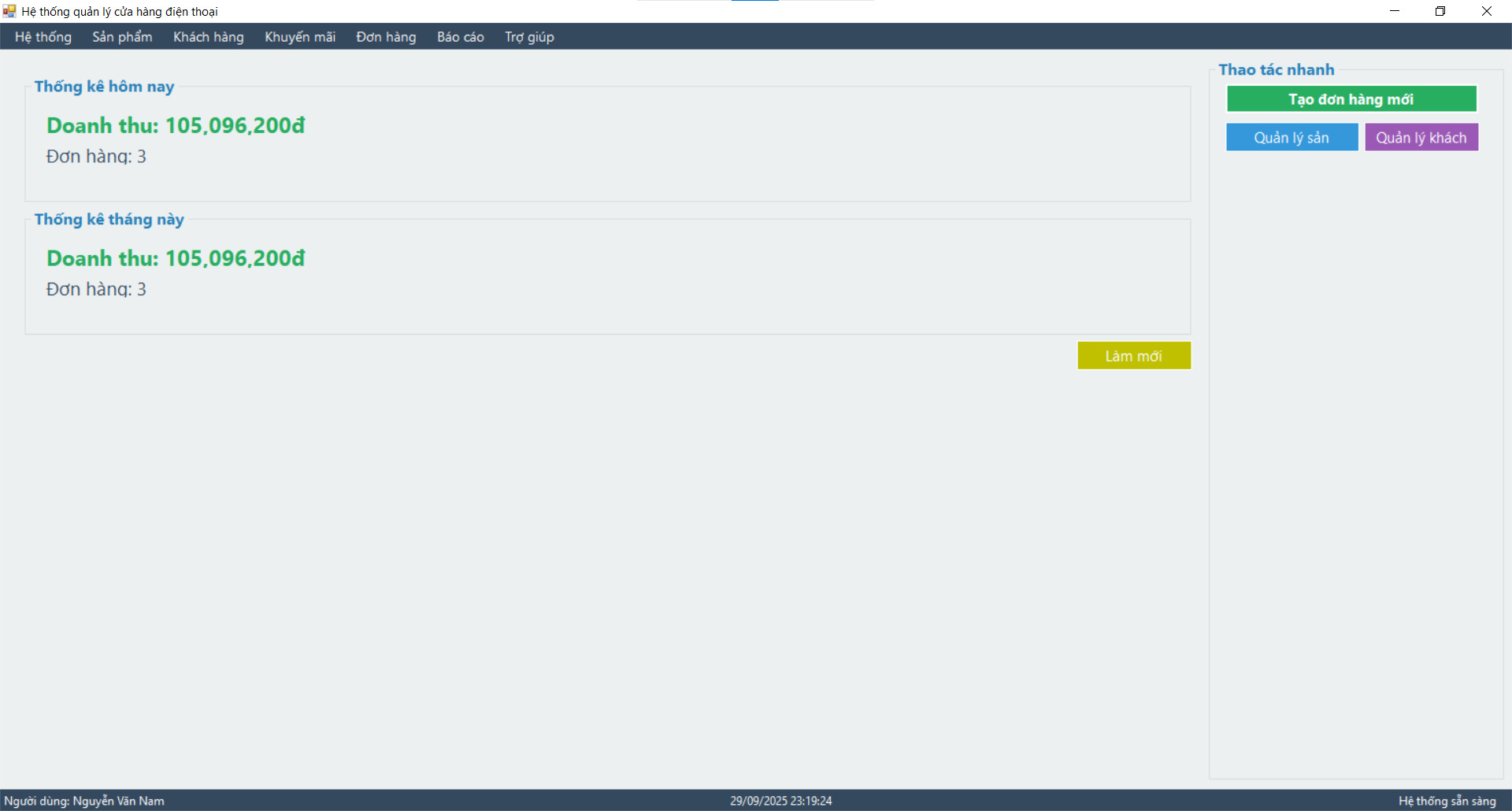
Thư mục Resources lưu trữ các **tài nguyên dùng trong giao diện** như ảnh logo, icon, hoặc file ngôn ngữ. Trong khi đó, Utils là nơi chứa các **tiện ích dùng chung** trong toàn bộ hệ thống như StringHelper, Validator, FileHelper,... giúp tái sử dụng và tránh lặp mã.

Cấu trúc dự án PhoneStore phản ánh một cách tổ chức **rõ ràng, dễ hiểu và hiệu quả** theo mô hình 3 tầng. Việc tách biệt các lớp giao diện, xử lý nghiệp vụ và truy xuất dữ liệu giúp nâng cao khả năng bảo trì, kiểm thử và mở rộng trong tương lai.

**3. Giao diện hệ thống**

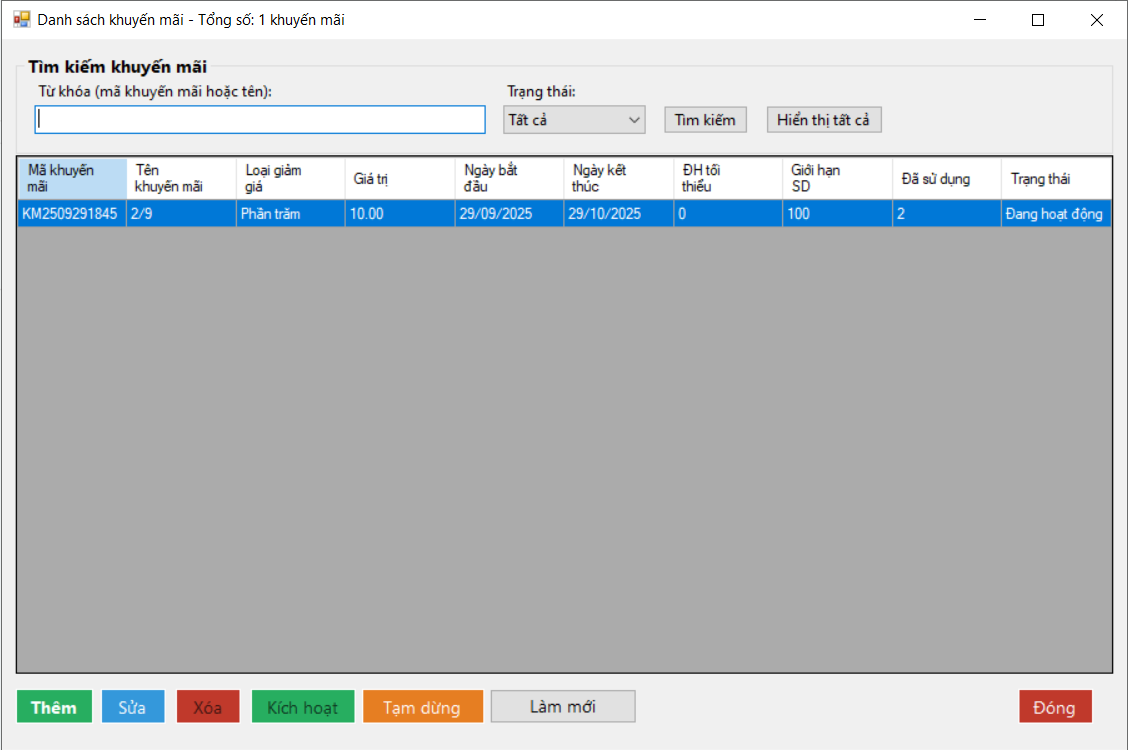
****

*From login*

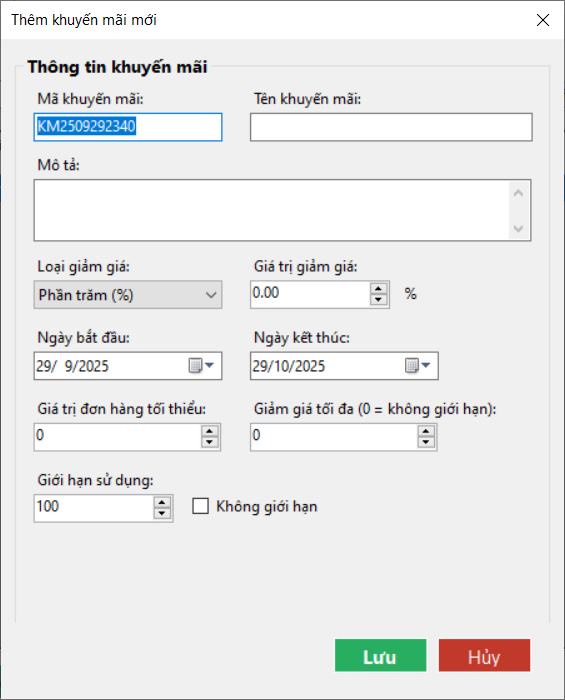


*From main*

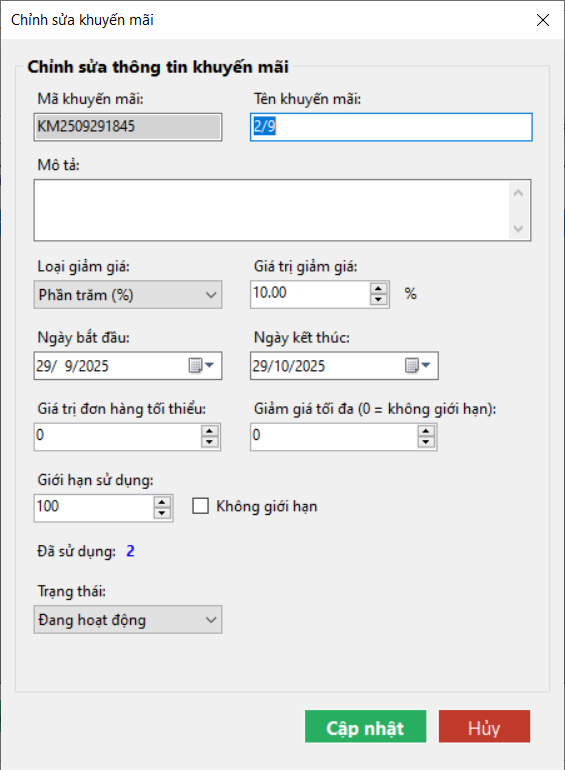
**Quản lý khuyến mãi**



*From danh sách khuyến mãi*

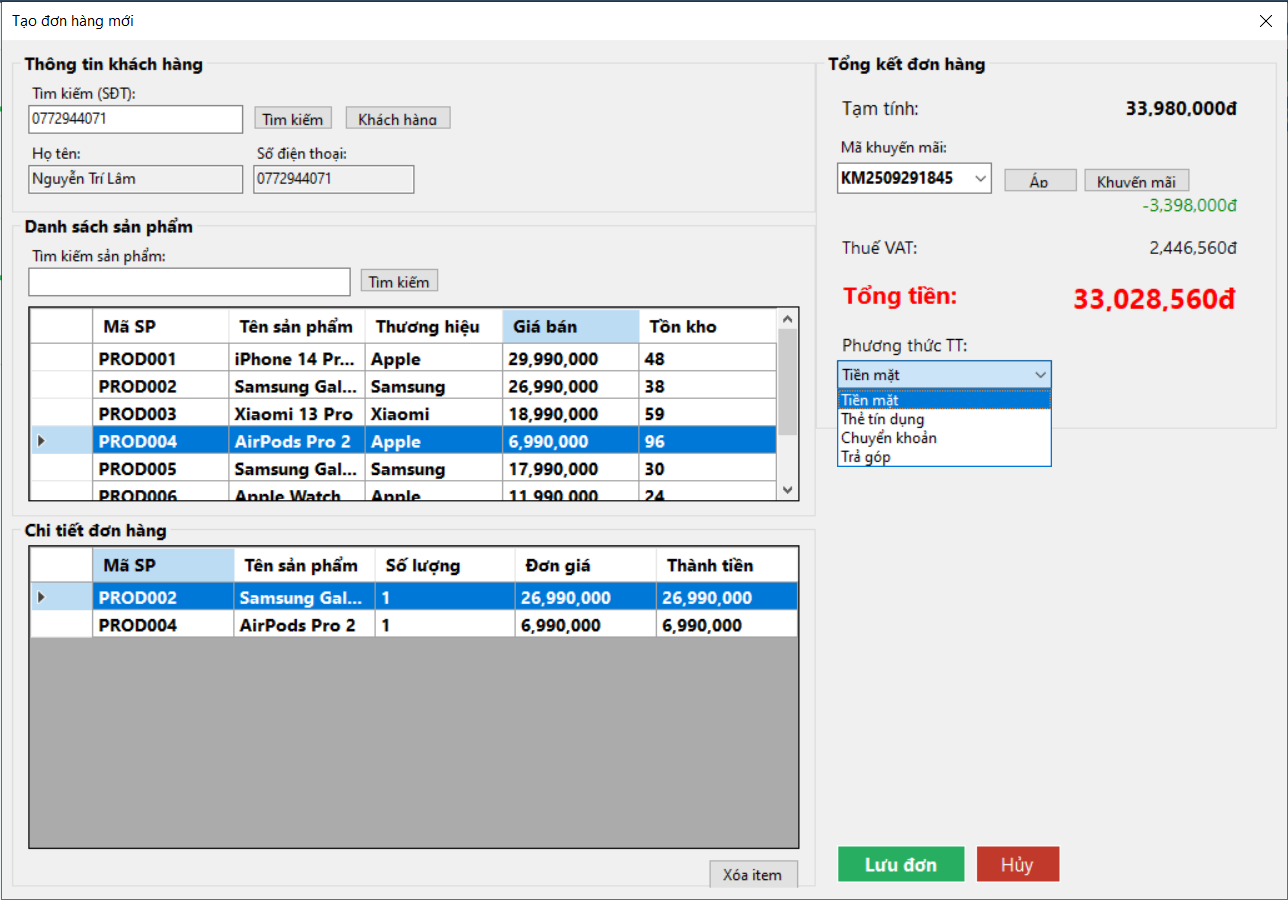


*From thêm khuyến mãi*

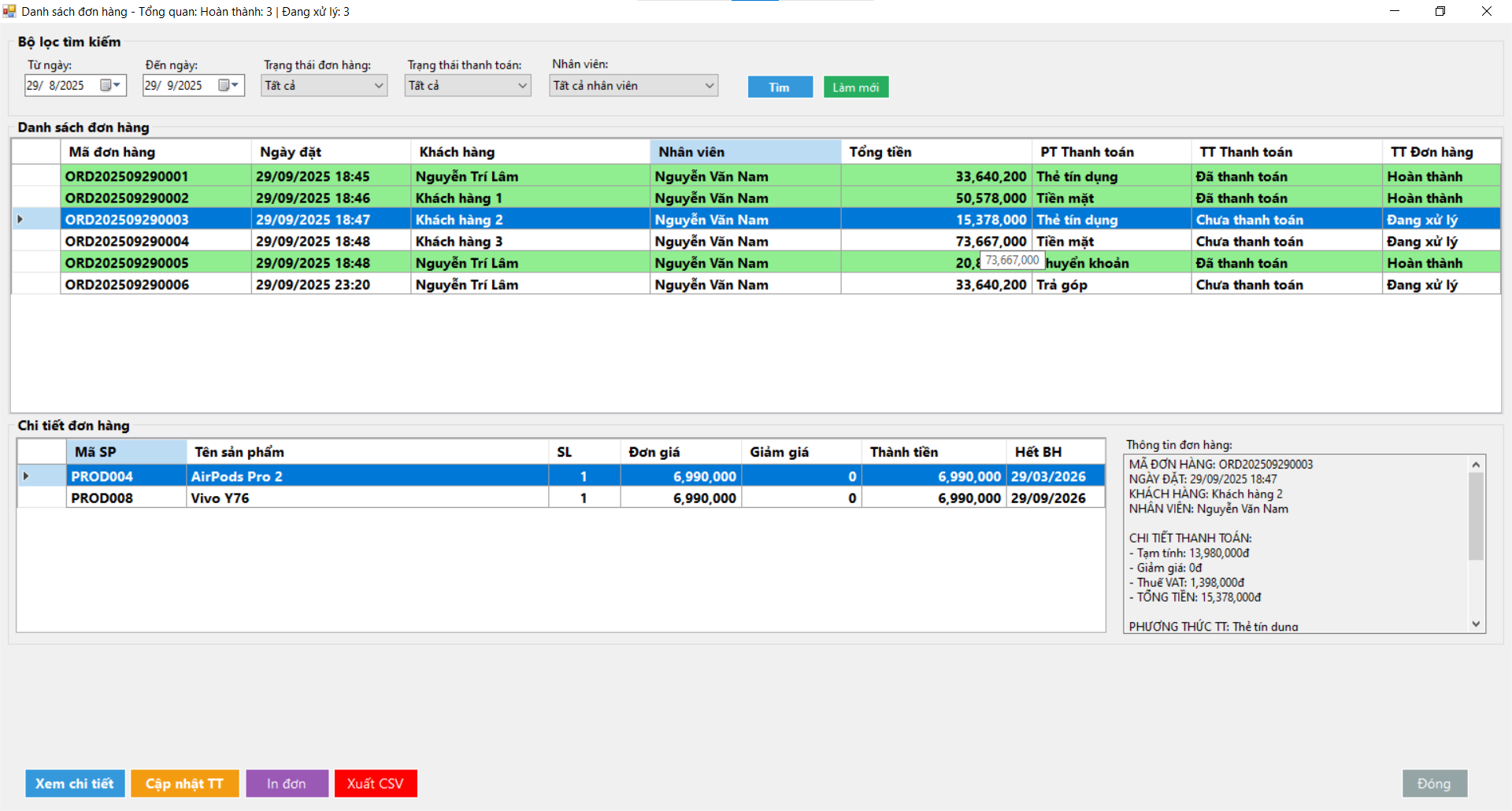


*From cập nhật khuyến mãi*

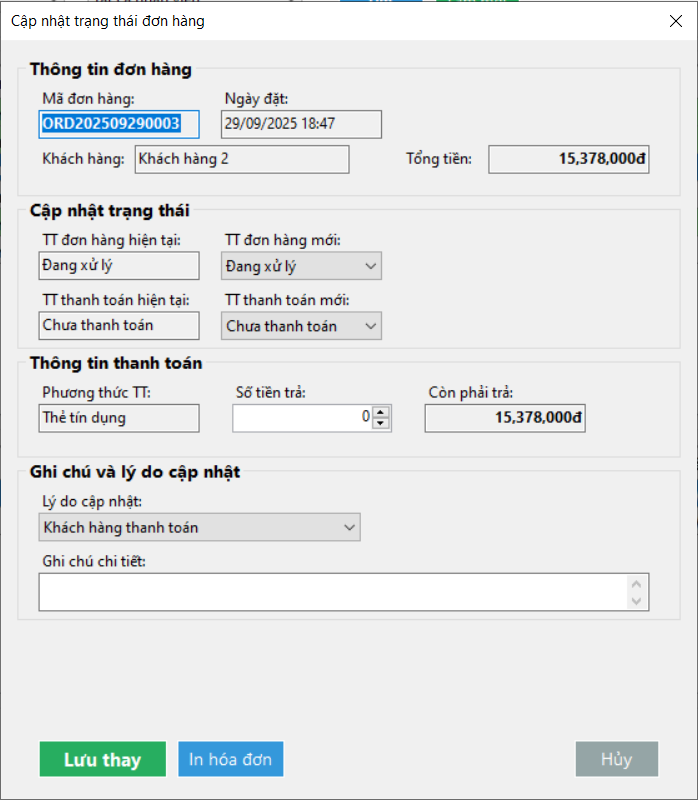
**Quản lý đơn hàng và thanh toán**



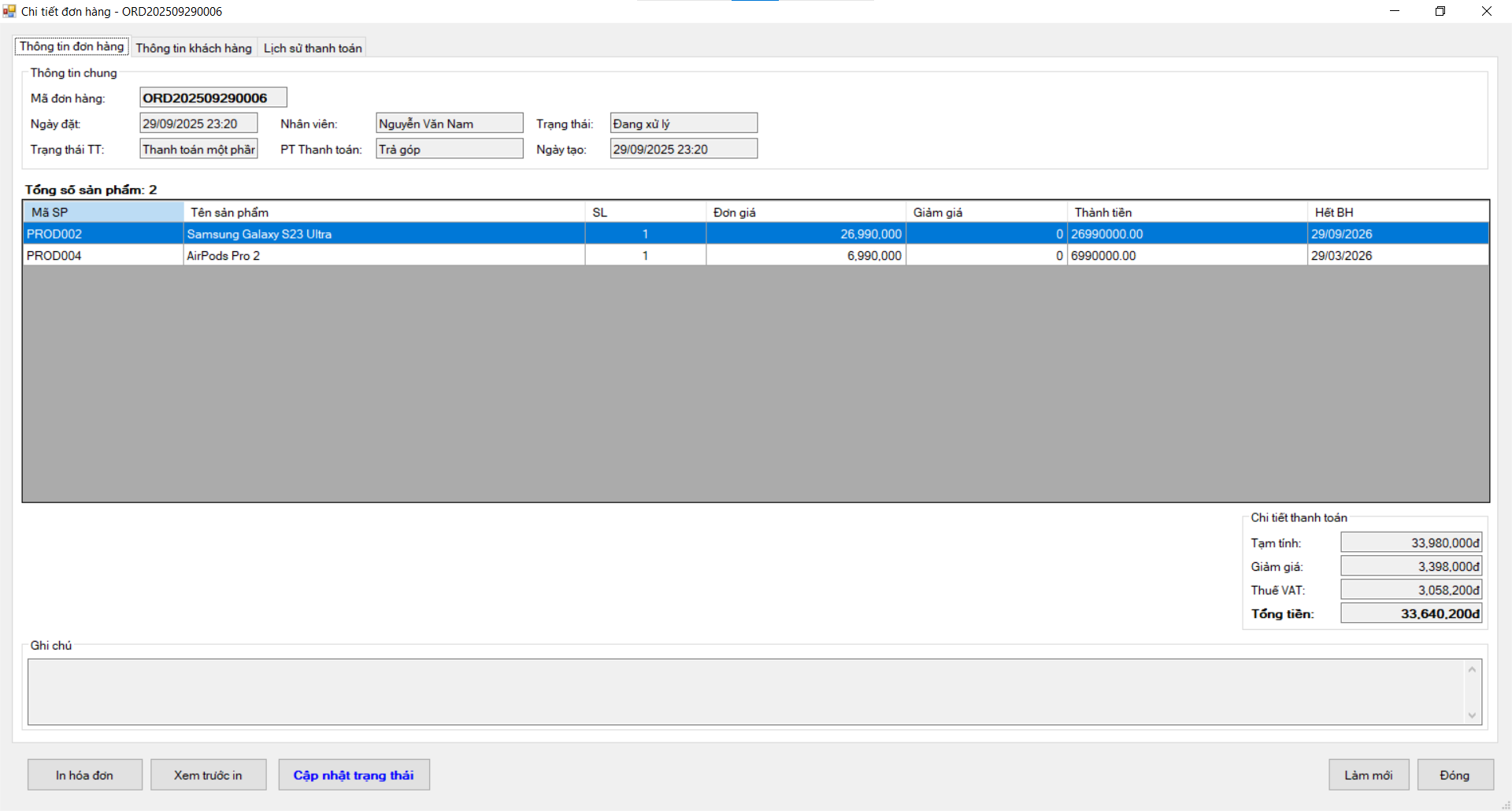
*From tạo mới đơn hàng*

****

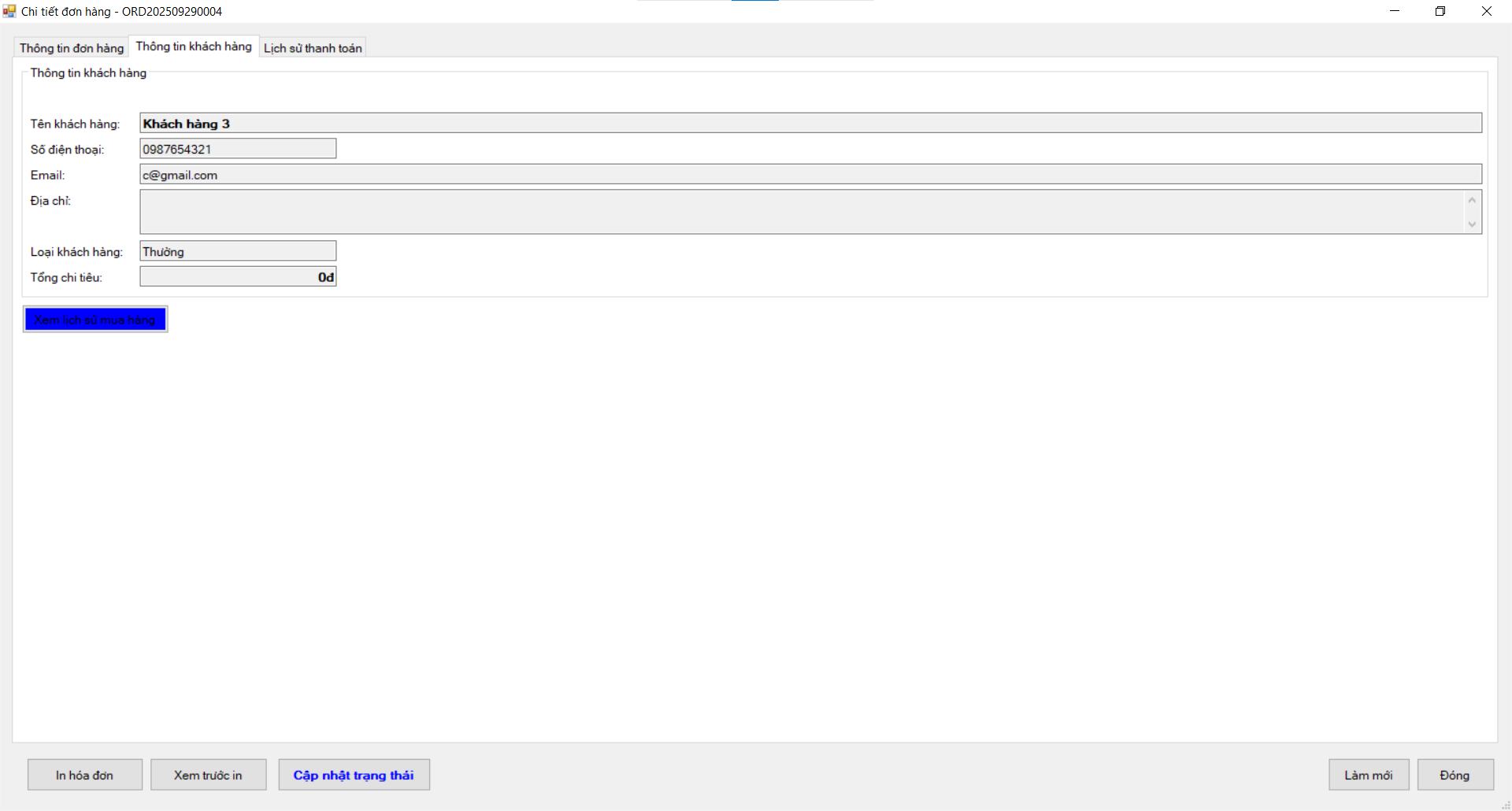
*From danh sách đơn hàng*

****

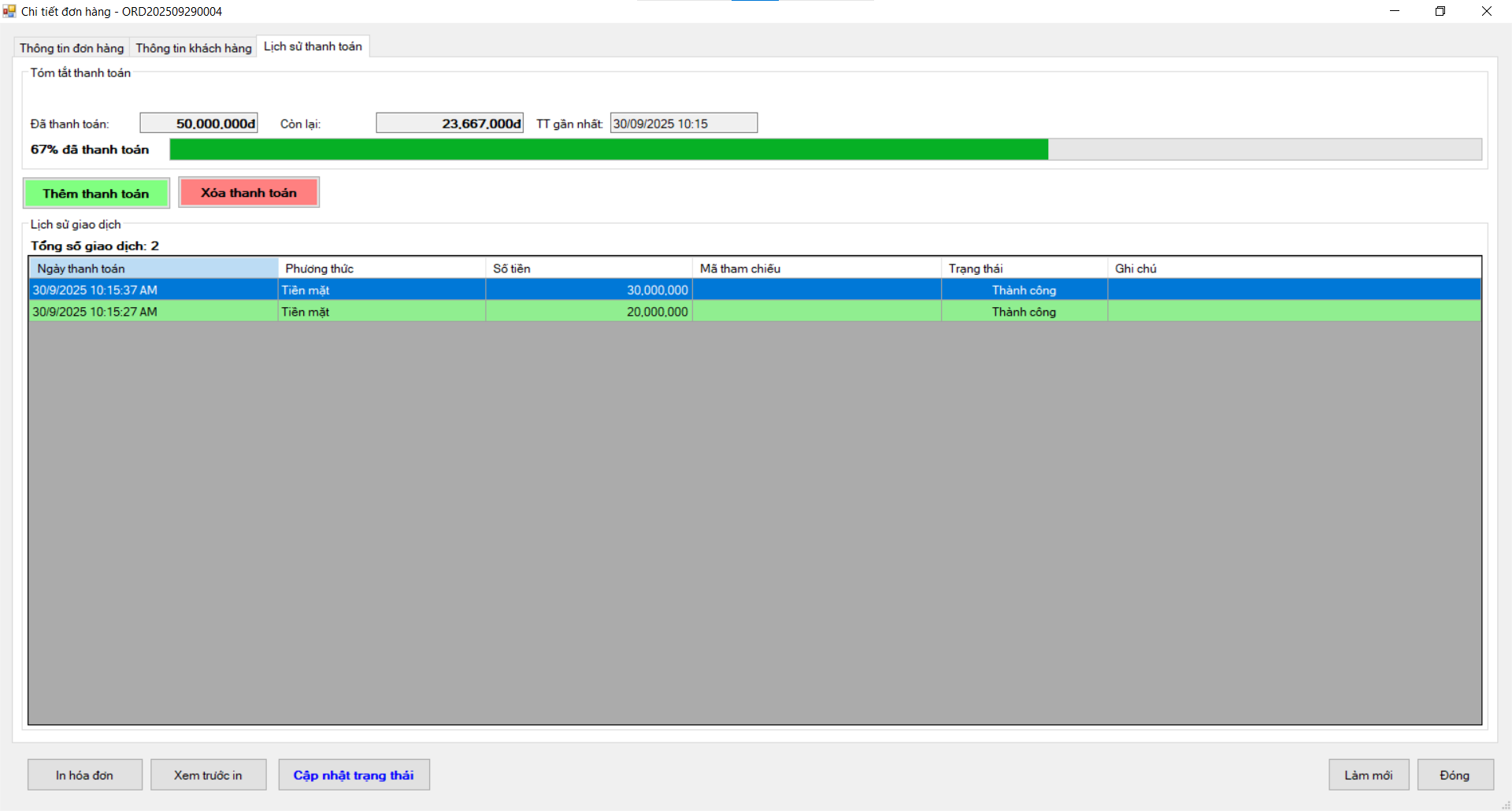
*From cập nhật đơn hàng*

****

*From chi tiết đơn hàng (trang thông tin đơn hàng)*

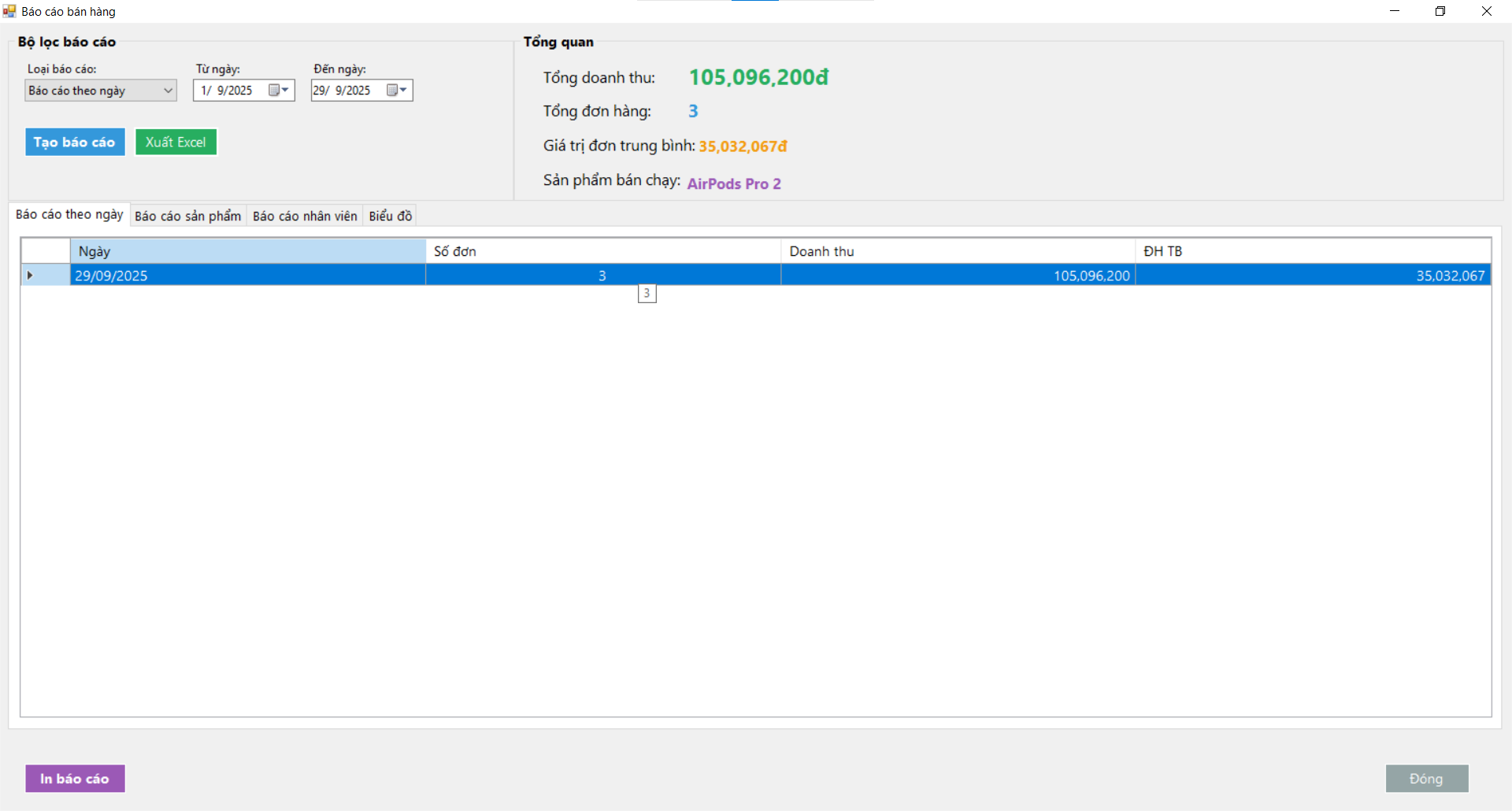
****

*From chi tiết đơn hàng (trang thông tin khách hàng)*

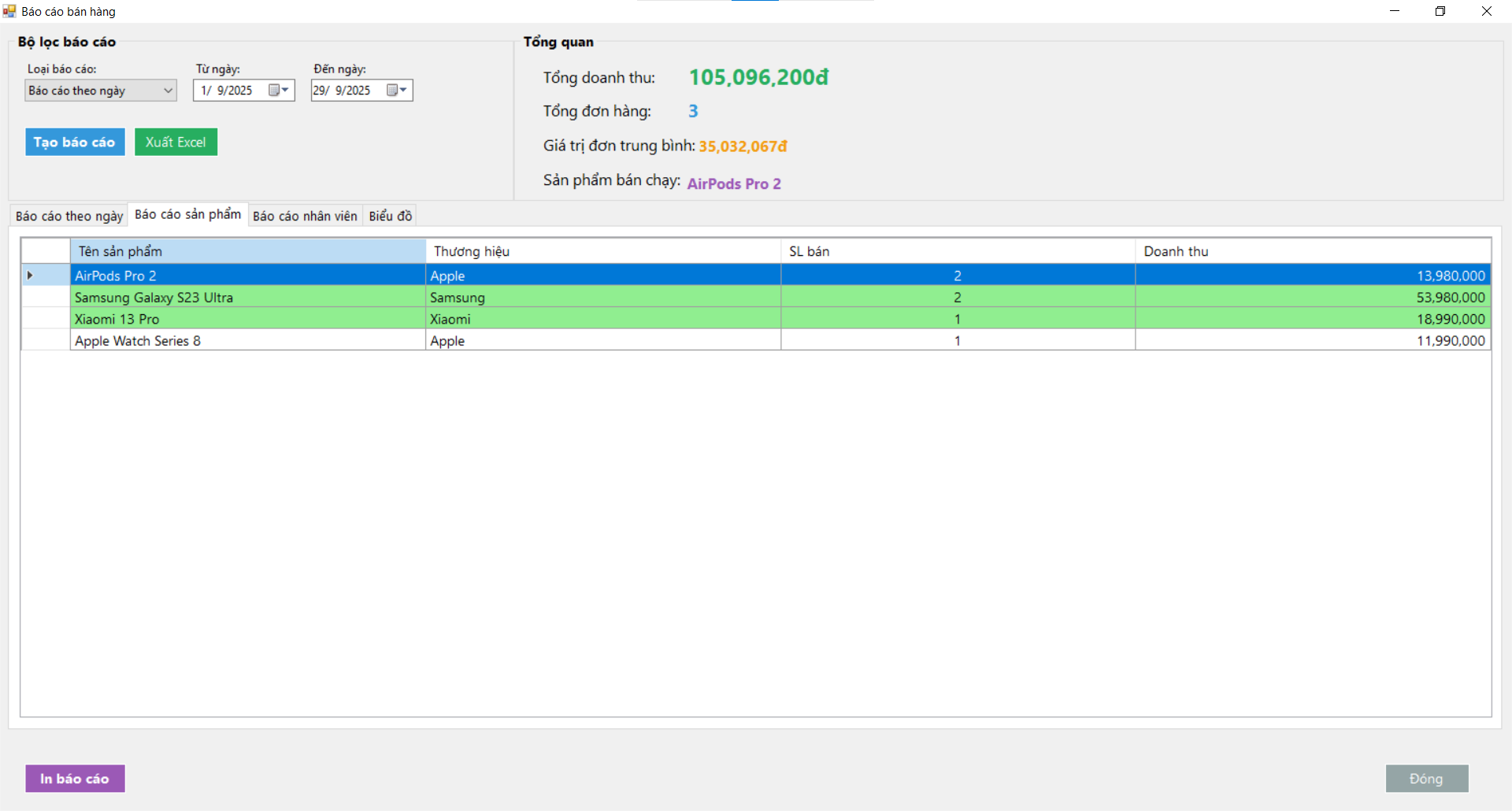
****

*From chi tiết đơn hàng (trang lịch sử thanh toán)*

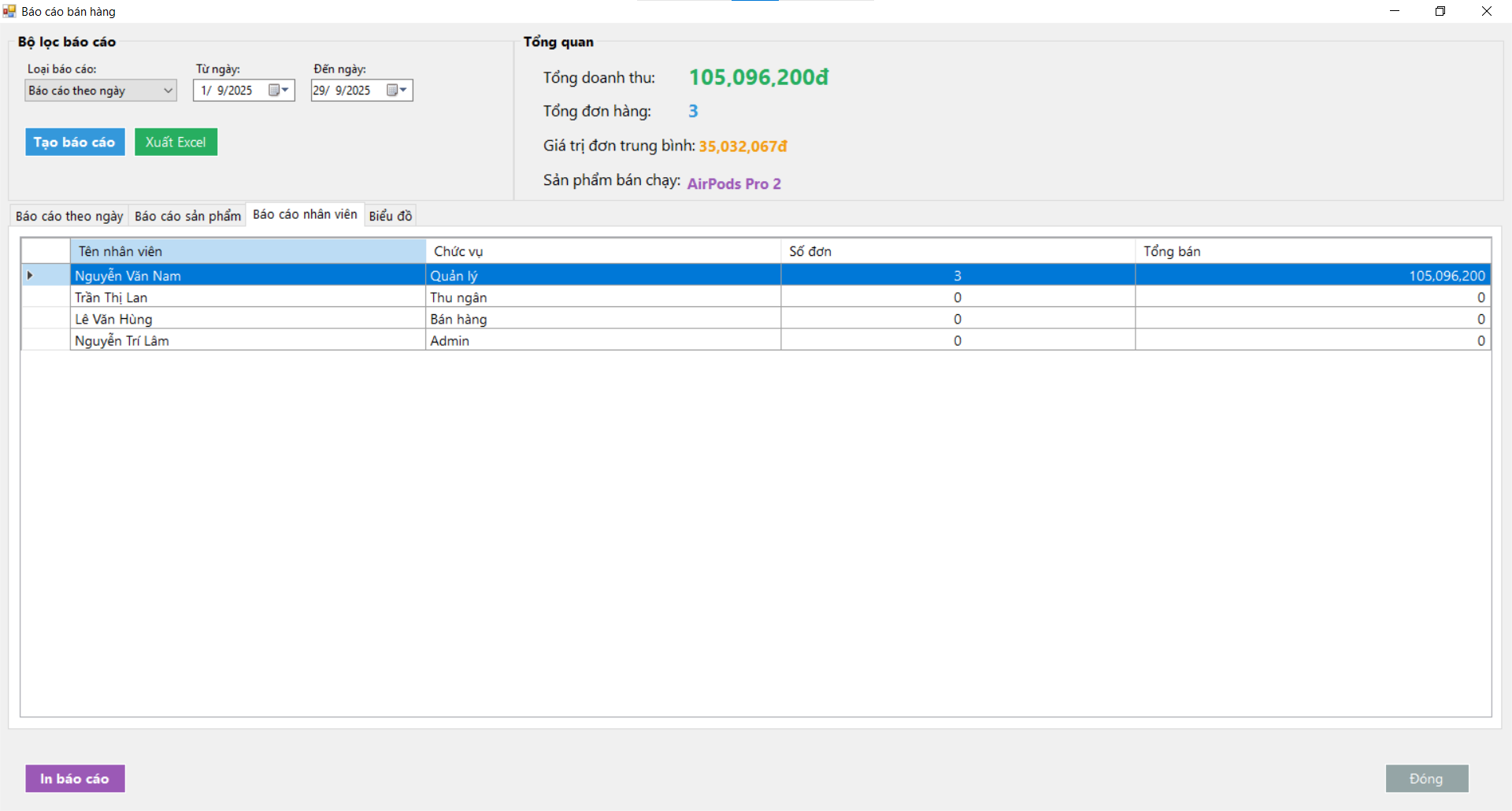
**Quản lý báo cáo doanh thu**

****

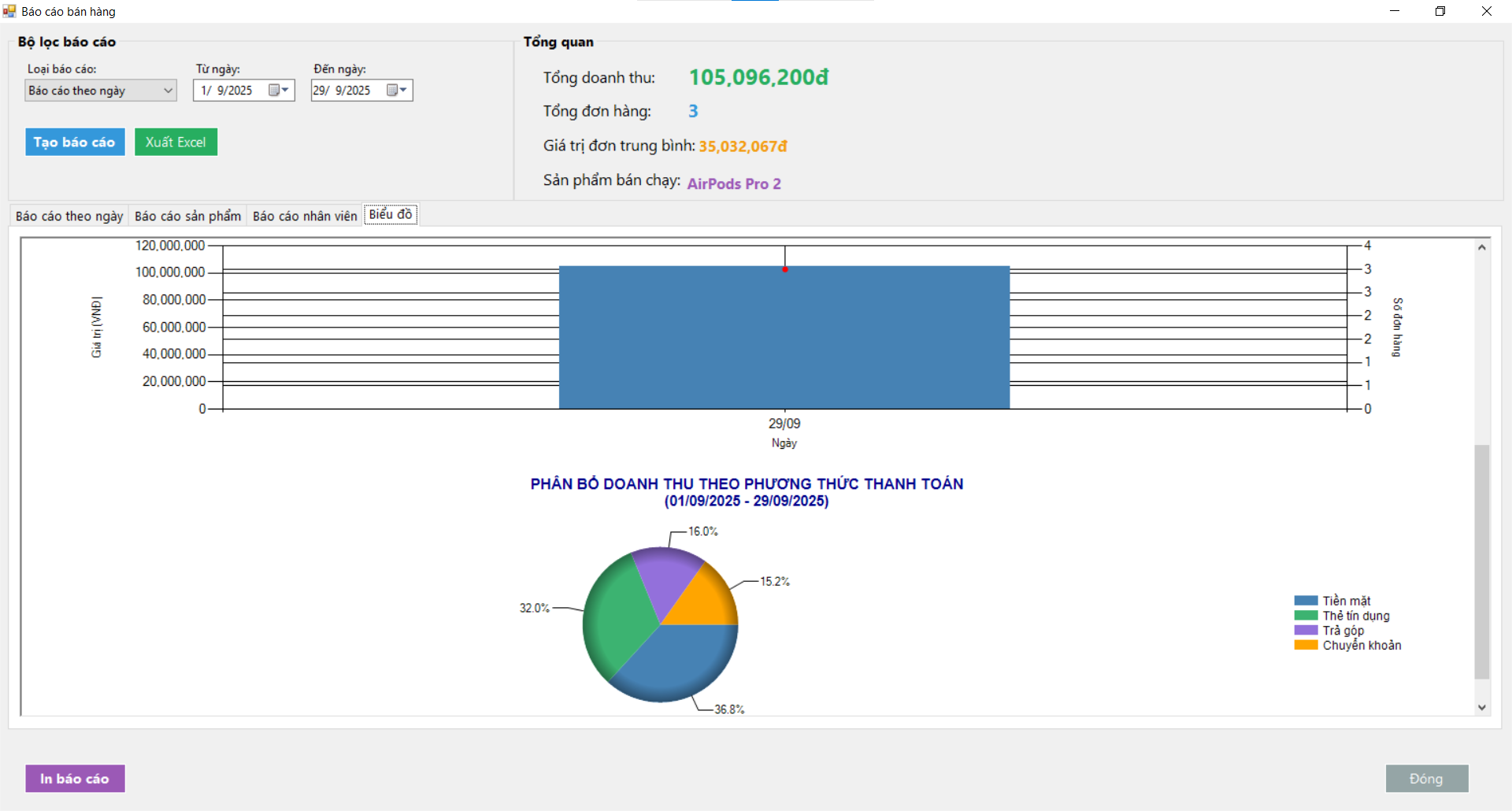
*From báo cáo (trang báo cáo theo ngày)*

****

*From báo cáo (trang báo cáo sản phẩm bán chạy)*

****

*From báo cáo (trang báo cáo hiệu xuất nhân viên)*

****

*0From báo cáo (trang biểu đồ)*