**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# ĐỒ ÁN TỐT NGHIỆP

## Hệ thống quản lý hóa đơn và tồn kho

**NGÔ PHÚ THÁI**
thai.np187187@sis.hust.edu.vn

**Ngành Công nghệ thông tin và truyền thông**

**Giảng viên hướng dẫn:** TS. Đỗ Quốc Huy _____

Chữ kí GVHD

**Khoa:** Khoa học máy tính

**Trường:** Công nghệ Thông tin và Truyền thông

**HÀ NỘI, 01/2024**

# LỜI CẢM ƠN

Lời đầu tiên, tôi xin được gửi lời chào trân trọng và lời chúc sức khỏe đến thầy cô, và bạn bè đã đồng hành với tôi trong chạng đường dài học tập và nghiên cứu. 5 năm qua, tôi đã có những trải nghiệm thú vị và những kỉ niệm đẹp trong cuộc sống sinh viên. Cho dù có những khó khăn, thử thách, tôi vẫn luôn cố gắng hết mình để vượt qua và hoàn thành tốt nhiệm vụ được giao.

Trường Đại học Bách Khoa Hà Nội, ngôi trường mà tôi không bao giờ nghĩ mình có thể một ngày nhập học. Từ những buổi hướng nghiệp, những buổi giới thiệu về các ngành của trường, tôi đã định hình trong mình những ước muốn, dự định tương lai trong sự nghiệp. Bắt đầu là những lớp học vỡ lòng ngành Công nghệ thông tin, đến những bài giảng lý thuyết chuyên sâu trong thiết kế hệ thống, tôi không nghĩ rằng sức mạnh đến từ thế giới số 1 và 0 này đã đang và sẽ định hình thế giới như thế nào. Tôi đã học được cách tự học, cách tự nghiên cứu và cách giải quyết vấn đề mà chính tôi không nghĩ mình làm được hồi còn là học sinh.

Tôi cũng đã có cơ hội được tiếp xúc với những người bạn tài năng, đồng hành cùng tôi trên giảng đường và ngoài sự nghiệp học tập. Được làm việc nhóm, học cách trao đổi và khám phá những điều mới mẻ trong bản thân. Tôi cũng muốn gửi lời cảm ơn chân thành đến gia đình, những người đã luôn đồng hành cùng tôi trong suốt quãng đường dài học tập. Gửi lời cảm ơn đến thầy cô, những người đã truyền đạt cho tôi những kiến thức quý báu trong suốt quãng đường học tập.

Đặc biệt trong suốt thời gian này, tôi đã nhận được sự giúp đỡ nhiệt tình và tận tâm của thầy Đỗ Quốc Huy. Thầy đã hướng dẫn tôi trong suốt quá trình thực hiện đồ án tốt nghiệp. Thầy gợi ý cho tôi những chi tiết cần thiết, những điều cần phải chú ý để đạt được kết quả tốt nhất. Và cũng trong thời gian quý báu này mà tôi học được thêm nhiều kiến thức mới, cũng như tổng kết lại và lần đầu tiên hoàn thiện một sản phẩm thực tế.

Một lần nữa, cảm ơn mọi người vì đã luôn đồng hành cạnh bên. Tôi biết mình không thể thành công một mình và rất biết ơn sự hiện diện quý giá này của tất cả mọi người.

Tôi xin chân thành cảm ơn!

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Đối với các cửa hàng vừa và nhỏ, việc vận hành và duy trì hoạt động kinh doanh là một trong những vấn đề quan trọng để cửa hàng có thể tồn tại và phát triển. Khi lưu lượng hàng hóa và khách hàng tăng lên, việc quản lý cửa hàng trở nên phức tạp hơn. Các giấy tờ chi chép thủ công không còn đáp ứng đủ nhu cầu của người quản lí nữa. Các mặt hàng trong cửa hàng không được quản lý một cách khoa học, dẫn đến việc hàng hóa bị thất thoát, hết hạn sử dụng, v.v. Và khi cửa hàng mở rộng và cần thêm nhân viên, trở ngại lớn nhất với họ là việc nhân viên mới mất bao lâu để quen việc, ghi nhớ được vị trí của hàng hóa cũng như giá cả và số lượng của chúng.

Khi thời đại công nghệ số đang phát triển, để giải quyết vấn đề hóc búa trên, các hệ thống quản lý cửa hàng cũng dần phổ biến hơn. Nhưng các giải pháp hiện tại vẫn còn nhiều hạn chế và thiếu sự đa dạng. Vì kích cỡ quy mô kinh doanh của các cá nhân quản lý cửa hàng vừa và nhỏ rất đặc thù, việc áp dụng các quy trình rườm rà phức tạp của các hệ thống đại lí lớn, cộng thêm rào cản công nghệ cũng như nhân lực để có thể triển khai các hệ thống đó là điều không thể. Vì vậy, tôi đã đề xuất một giải pháp mới, một hệ thống quản lý hóa đơn, hàng hóa, tồn kho cho các cửa hàng vừa và nhỏ, với các tính năng đơn giản, dễ sử dụng, dễ triển khai, và đặc biệt là giá thành thấp. Tận dụng từ các thiết bị thường trực như máy tính, điện thoại thông minh, việc điều hành kinh doanh trở nên dễ dàng và thuận tiện hơn bao giờ hết.

Đây là một cơ hội tốt để tôi có thể áp dụng những kiến thức đã học được trong suốt quá trình học tập và nghiên cứu. Từ việc quan sát các vấn đề thực tế và xu hướng công nghệ số mới, tôi đã thấy được sự ứng dụng của chúng trong việc phục vụ con người là vô cùng lớn. Biết bao thời gian và công sức chúng ta có thể tiết kiệm được nếu biết tận dụng những cơ hội và tiềm năng mà công nghệ số mang lại. Nguồn cảm hứng này đã thúc đẩy tôi tìm hiểu và phát triển các giải pháp công nghệ số cho các vấn đề thực tế, và tôi tin rằng đây là một xu hướng tất yếu trong tương lai.

Sinh viên thực hiện

*(Ký và ghi rõ họ tên)*

# MỤC LỤC

# DANH MỤC HÌNH VẼ

# DANH MỤC BẢNG BIỂU

# DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

| **Thuật ngữ** | **Ý nghĩa** |
|---|---|
| API | Giao diện lập trình ứng dụng (Application Programming Interface) |
| Backend | Phía máy chủ, thường là phần xử lý logic, xử lý dữ liệu, ... |
| Client | Máy khách trong mô hình Client-Server, thường gửi các yêu cầu đến Server và nhận kết quả từ Server |
| Framework | Bộ khung phát triển ứng dụng, là một tập các thư viện, công cụ, hàm, lớp, ... giúp cho việc phát triển ứng dụng nhanh hơn |
| Frontend | Phía máy khách, thường là phần giao diện người dùng |
| MVVM | Mô hình thiết kế ứng dụng người dùng cuối (Model-View-ViewModel) |
| NoSQL | Hệ quản trị cơ sở dữ liệu phi quan hệ (Not Only SQL) |
| OOP | Lập trình hướng đối tượng (Object Oriented Programming) |
| Server | Máy chủ trong mô hình Client-Server, thường ám chỉ máy tính có cấu hình mạnh, có nhiệm vụ xử lý các yêu cầu từ phía Client |

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Kinh doanh có thể được coi là một trong những hoạt động quan trọng nhất của con người. Từ những năm đầu của thời kỳ công nghiệp, chúng đã trở thành một trong những ngành nghề phát triển mạnh nhất, và đóng góp rất lớn vào sự phát triển của xã hội. Với sự phát triển của khoa học công nghệ, kinh doanh cũng dần được cải tiến, đưa vào sử dụng các công nghệ mới để tối ưu hóa quy trình và tăng hiệu quả. Từ việc sử dụng các máy móc để thay thế lao động thủ công, đến việc sử dụng các phần mềm để quản lý và vận hành, công nghệ đã đóng một vai trò rất quan trọng trong việc phát triển kinh doanh.

Quy mô hoạt động kinh doanh có thể được chia thành ba loại: lớn, vừa và nhỏ. Các doanh nghiệp lớn thường có quy mô hoạt động rộng lớn, với nhiều chi nhánh, nhiều nhân viên, và nhiều khách hàng. Các doanh nghiệp vừa và nhỏ thường có quy mô hoạt động nhỏ hơn, với ít nhân viên và ít khách hàng hơn. Tuy nhiên, các doanh nghiệp vừa và nhỏ lại chiếm phần lớn số lượng so với các doanh nghiệp lớn. Vì vậy, việc quản lý và vận hành các doanh nghiệp vừa và nhỏ là một vấn đề rất quan trọng, và cần được đầu tư nghiên cứu và phát triển.

Bản thân tôi đang sống trong một hộ kinh doanh nhỏ trong lĩnh vực bán buôn bán lẻ. Tôi đã nhận thấy rằng, việc quản lý và vận hành một cửa hàng nhỏ là một vấn đề rất khó khăn. Với một cửa hàng nhỏ, việc quản lý hàng hóa, hóa đơn, và tồn kho là một việc rất quan trọng, nhưng lại có rất nhiều trở ngại.

Không phải mặt hàng nào cũng được bán hết, biến động thị trường cũng tác động rất nhiều vào giá cả của chúng và độ ưa chuộng của khách hàng. Phải có giải pháp để người quản lý có thể theo dõi và cập nhật giá cả của hàng hóa một cách nhanh chóng và thuận tiện. Sự quyết định vào giá cả của hàng hóa không nên dựa trên cảm tính, mà phải dựa trên các con số thống kê và phân tích. Từ những hóa đơn bán hàng, người quản lý có thể phân tích được các mặt hàng bán chạy. Từ đó, họ có thể đưa ra các quyết định về việc nhập thêm hàng hóa, hay giảm giá các mặt hàng đó để tăng doanh số bán hàng.

Và khi nhắc đến số lượng của hàng hóa, điều không tránh khỏi là việc chúng có thể bị hết hạn sử dụng, hay hỏng hóc trong quá trình bảo quản, lưu trữ khiến số lượng tồn kho không còn chính xác nữa. Việc theo dõi số lượng tồn kho của hàng hóa là một việc rất quan trọng, nhưng lại rất khó khăn. Người quản lý phải thường xuyên kiểm tra số lượng tồn kho của hàng hóa, và cập nhật chúng vào các

giấy tờ chi chép. Điều này dẫn đến việc mất nhiều thời gian, và có thể dẫn đến sai sót. Ngoài ra, việc theo dõi số lượng tồn kho cũng là một thước đo để người quản lý nắm bắt được dòng tiền trong cửa hàng. Nếu số lượng tồn kho quá nhiều, người quản lý có thể đưa ra quyết định giảm giá để bán hết hàng hóa, và ngược lại.

Giấy tờ, chúng là phương tiện ghi chép của con người lâu đời nhất. Từ khi con người bắt đầu sử dụng chúng, giấy tờ đã trở thành một phần không thể thiếu trong cuộc sống. Trong kinh doanh, từ việc ghi chép các hóa đơn bán hàng, đến việc ghi chép các thông tin cá nhân của khách hàng khiến chúng là một công cụ rất tất yếu. Song chúng lại có những hạn chế của nó. Giấy tờ có thể bị thất lạc, bị hư hỏng, và có thể bị mất bởi thời gian. Vì vậy, việc lưu trữ giấy tờ là một vấn đề rất quan trọng. Giấy tờ có thể rất nhiều, nếu không được lưu trữ một cách khoa học, việc không thể truy xuất được thông tin trong giấy tờ đó là không thể. Ngoài ra, việc lưu trữ chúng cũng là một việc tốn kém. Giấy tờ cần được lưu trữ trong một kho lưu trữ, và chỉ có thể truy cập tại một chỗ, kèm với cần sự quản lý và bảo quản nghiêm ngặt. Điều này dẫn đến việc tốn kém về chi phí, và không thể truy cập được thông tin bất cứ lúc nào.

Từ những trở ngại trên, nếu hộ kinh doanh đó quyết định mở rộng quy mô, thì họ sẽ cần thêm nhân lực. Và khi đó, trở ngại lớn nhất với họ là việc nhân viên mới mất bao lâu để quen việc, ghi nhớ được vị trí của hàng hóa cũng như giá cả và số lượng của chúng. Điều này dẫn đến việc họ phải đầu tư thêm nhiều thời gian và công sức để đào tạo nhân viên mới, và có thể sẽ mất nhiều thời gian hơn để nhân viên mới có thể làm việc hiệu quả nếu không có một quy chuẩn trong quá trình hoạt động kinh doanh.

Vì quy mô hoạt động nhỏ, việc sử dụng các phần mềm quản lý kinh doanh trong các đại lí lớn là không khả thi. Các phần mềm đó thường có quy trình phức tạp, và cần nhiều người để vận hành. Vì vậy, việc sử dụng các phần mềm đó sẽ làm tăng chi phí, và không đảm bảo hiệu quả. Đặc biệt là rào cản công nghệ cũng là một trong những trở ngại lớn nhất nếu hộ kinh doanh đó gồm những người lớn tuổi. Họ không thể sử dụng các thiết bị công nghệ mới một cách hiệu quả, và việc sử dụng các phần mềm quản lý kinh doanh sẽ làm cho họ cảm thấy khó khăn và không thích thú.

## 1.2 Mục tiêu và phạm vi đề tài

Khi được hỏi về nhu cầu của người dùng, họ đều muốn giảm thiểu công việc quản lý thủ công bằng giấy tờ. Và khi sống trong thời đại công nghệ số 4.0 này, họ cũng mong muốn được sử dụng các thiết bị công nghệ hỗ trợ trong hoạt động kinh doanh. Dẫn tới, họ cũng đâu đó biết một hai chữ về phần mềm quản lí kinh doanh.

Tuy nhiên, bởi vì quy mô của họ không lớn, kèm thêm nhiều người không được đào tạo chuyên sâu về công nghệ hay không trải qua các trường lớp khác nhau, họ khó có thể sử dụng các phần mềm quản lí kinh doanh hiện tại. Họ cũng không có nhiều thời gian để tìm hiểu và học cách sử dụng các phần mềm đó. Và vì vậy, họ vẫn tiếp tục sử dụng các phương pháp quản lý thủ công bằng giấy tờ.

Để tìm giải pháp cho các trở ngại trên, họ đều mong muốn một phần mềm quản lý kinh doanh có các tính năng đơn giản, dễ sử dụng, dễ triển khai, tiết kiệm chi phí. Tốt hơn nữa, họ muốn dễ dàng sử dụng ngay trên các thiết bị công nghệ thông minh như điện thoại, máy tính bảng, v.v. Điều này sẽ giúp họ tiết kiệm được thời gian và công sức, và có thể tập trung vào các hoạt động kinh doanh chính. Đồng thời, họ có thể chia sẻ được thông tin hoạt động kinh doanh với các thành viên. Điều này giúp ích rất nhiều trong việc vận hàng cũng như giúp nhân viên mới có thể bắt kịp được nhịp độ của công việc. Khả năng truy cập từ mọi nơi mọi lúc cũng khiến họ mang được công việc theo bên mình, và có thể giải quyết các vấn đề nhanh chóng.

Khi đi tìm hiểu về các phần mềm quản lý hóa đơn, tồn kho hiện có. Tôi nhận thấy được rằng, các phần mềm đó đều có quy trình phức tạp, phù hợp hơn cho các chuỗi đại lí lớn. Và đặc biệt, vì đối tượng khách hàng của các phần mềm đó là các tổ chức có lợi nhuận lớn có thể đầu tư cho chi phí vận hành cùng với đội ngũ nhân viên có chuyên môn cao, nên giá thành của các phần mềm đó cũng rất cao. Không những thế, các phần mềm đó được tích hợp quá nhiều chức năng nâng cao mà hộ kinh doanh vừa và nhỏ không cần thiết dùng tới. Với quy mô của các phần mềm này, chúng đòi hỏi khả năng xử lí của thiết bị chạy chúng cũng rất cao, khiến nền tảng của thiết bị hay giới hạn ở máy tính. Điều này chứng tỏ thị trường này còn thiếu sự đa dạng nhắm tới các kiểu hình kinh doanh khác nhau.

Dựa trên các phân tích và đánh giá ở trên, một ý tưởng đã nảy sáng lên. Tại sao không phát triển một ứng dụng quản lí hóa đơn, tồn kho dành cho các hộ kinh doanh vừa và nhỏ? Ứng dụng này sẽ có các tính năng đơn giản dễ sử dụng, giao diện thân thiện với nhiều lứa tuổi. Đặc biệt, ứng dụng này có thể sử dụng được ngay trên điện thoại thông minh, thứ mà hầu như ai cũng có thể lôi từ trong túi ra sử dụng. Và chỉ cần có kết nối mạng, họ có thể truy cập vào ứng dụng bất cứ lúc nào, bất cứ nơi đâu. Với một hệ thống lưu trữ dữ liệu trên đám mây, họ không cần phải lo lắng về việc lưu trữ và bảo quản dữ liệu nữa. Ngoài ra, ứng dụng sẽ chỉ phân quyền hạn thành nhân viên và quản lí để họ có thể dễ dàng tin tưởng sử dụng mà không lo đến yếu tố bảo mật thông tin. Cuối cùng, ứng dụng này được phát triển dành riêng cho các hộ kinh doanh vừa và nhỏ, nên giá thành của nó sẽ rất thấp, phù hợp với túi tiền của họ.

## 1.3 Định hướng giải pháp

Khi thiết kế giải pháp cho các nhu cầu của các hộ kinh doanh vừa và nhỏ. Tôi để ý thấy nếu người dùng muốn hoàn toàn trọn vẹn điều hành hệ thống kèm với chi phí thấp thì mô hình Server-Client là một giải pháp tốt. Với mô hình này, người dùng sẽ có một máy chủ để lưu trữ dữ liệu, và các máy khách để truy cập vào dữ liệu đó. Họ có thể tận dụng các thiết bị cũ để làm máy chủ, và đối với máy khách, thì một ứng dụng di động là đủ. Điều này cũng giúp họ có thể dễ dàng mở rộng hệ thống khi cần thiết. Hơn nữa, dữ liệu của họ sẽ hoàn toàn họ kiểm soát và không lo lắng bên thứ ba nào cả.

Suy nghĩ về hệ thống đang đề xuất, để quá trình phát triển được tốt và triển khai được nhanh chóng, việc lựa chọn các công nghệ phù hợp là rất quan trọng. Tiêu chí các công nghệ này sẽ quyết định chất lượng và tương lai của hệ thống, do đó cần phải lựa chọn một cách cẩn thận. Đầu tiên, cần lựa chọn một ngôn ngữ lập trình để phát triển ứng dụng và phát triển hệ thống dữ liệu. Ngôn ngữ lập trình sẽ quyết định cấu trúc của ứng dụng, và cũng quyết định được các công nghệ khác có thể được sử dụng hay không. Tiếp theo, tôi cần lựa chọn một hệ quản trị cơ sở dữ liệu để lưu trữ dữ liệu của ứng dụng. Sau đây là các tiêu chí mà tôi đã đặt ra để lựa chọn các công nghệ phù hợp:

- **Đơn giản:** Các công nghệ được lựa chọn cần phải đơn giản, dễ sử dụng, và dễ triển khai. Điều này giúp quá trình phát triển và triển khai ứng dụng được trơn chu.
- **Hiệu quả:** Các công nghệ được lựa chọn cần phải hiệu quả, và có thể đáp ứng được lưu lượng lớn dữ liệu. Điều đó sẽ giúp hệ thống hoạt động được tối ưu nhất với mức độ sử dụng tài nguyên thấp nhất. Dẫn tới chi phí vận hành hệ thống sẽ thấp nhất.
- **Mở rộng:** Các công nghệ được lựa chọn cần phải có khả năng mở rộng. Dễ dàng đề xuất và cải tiến các tính năng mới, và có thể đáp ứng được nhu cầu của người dùng.
- **Đa nền tảng:** Các công nghệ được lựa chọn cần phải có khả năng đa nền tảng. Nếu được, hệ thống dùng chung một công nghệ cho cả máy chủ và máy khách sẽ giảm thời gian phát triển đi rất nhiều. Mà độ tương thích giữa chúng cũng sẽ tốt hơn.
- **Hệ sinh thái:** Các công nghệ được lựa chọn cần phải có một hệ sinh thái phát triển tốt. Điều này giúp tôi có thể phát triển một ứng dụng có thể dễ dàng mở rộng.

- **Cộng đồng:** Các công nghệ được lựa chọn cần phải có một cộng đồng phát triển tốt. Một cộng đồng phát triển tốt sẽ giúp tôi có thể tìm kiếm được nhiều tài liệu hơn, và có thể nhận được sự giúp đỡ từ các thành viên trong cộng đồng.

Theo kinh nghiệm bản thân tôi tích lũy được, hệ sinh thái .NET (C#) của Microsoft là một lựa chọn tốt. Hệ sinh thái này có framework là ASP.NET Core, một framework chuyên về phát triển ứng dụng web đã được chứng minh là hiệu quả và mở rộng. Ngoài ra, với hệ sinh thái này, tôi có thể phát triển một ứng dụng di động trên nền tảng Android và iOS, một ứng dụng máy tính trên nền tảng Windows, Linux, bằng framework .NET MAUI. Tính đa nền tảng, hiệu quả, và mở rộng của hệ sinh thái này đã được chứng minh qua nhiều năm phát triển. Ngoài ra, hệ sinh thái này cũng có một cộng đồng phát triển rất lớn, và có rất nhiều tài liệu hướng dẫn.

Khi lựa chọn hệ cơ sở dữ liệu, thì khi làm việc với cấu trúc dữ liệu không ngừng thay đổi thì NoSQL là một lựa chọn tốt. Với NoSQL, tôi có thể lưu trữ dữ liệu dưới dạng tài liệu, và có thể thay đổi cấu trúc dữ liệu một cách dễ dàng. Điều này giúp tôi có thể phát triển ứng dụng một cách nhanh chóng, và có thể mở rộng ứng dụng một cách dễ dàng. Cụ thể, MongoDB là một hệ cơ sở dữ liệu NoSQL tôi đã lựa chọn. MongoDB có bộ tài liệu rất tường minh cùng với nhiều diễn đàn giải đáp. Ngoài ra, MongoDB cũng có một hệ quản trị cơ sở dữ liệu trực quan, giúp tôi có thể quản lý dữ liệu một cách dễ dàng.

Cuối cùng, khi đóng gói sản phẩm lại cho người dùng sử dụng, việc hệ thống đề xuất này phải có khả năng hoạt động hầu hết trên mọi môi trường là rất quan trọng. Vì vậy, tôi đã lựa chọn Docker để đóng gói sản phẩm. Docker là một công nghệ ảo hóa, giúp tôi có thể đóng gói ứng dụng và triển khai dễ dàng, tiện lợi. Ngoài ra, Docker cũng có một hệ sinh thái phát triển tốt, và có một cộng đồng phát triển lớn, được tin cậy và được sử dụng rộng rãi.

## 1.4 Bố cục đồ án

Trong báo cáo đồ án tốt nghiệp này, bố cục sẽ được chia thành 6 chương:

**Chương 2: Khảo sát và phân tích yêu cầu.** Chương này sẽ trình bày về các phương pháp khảo sát và phân tích yêu cầu của người dùng. Từ các kết quả thu được từ việc khảo sát kĩ lưỡng chức năng của hệ thống hiện có, tôi đã so sánh với yêu cầu người dùng và chỉ ra rõ những điểm mạnh và điểm yếu của hệ thống hiện có. Từ đó, tôi đưa ra được các yêu cầu cần thiết cho hệ thống mới.

**Chương 3: Công nghệ sử dụng.** Chương này sẽ giới thiệu về các công nghệ được sử dụng và sự áp dụng của chúng trong thế giới nói chung và đồ án này nói riêng.

**Chương 4: Thiết kế, triển khai và đánh giá hệ thống.** Chương này sẽ trình bày về sự quan trọng của việc lựa chọn kiến trúc hệ thống. Nó quyết định quy trình hệ thống tương tác các thành phần với nhau như thế nào cho hiệu quả. Sau đó, chương này sẽ chứa các thông tin về quá trình triển khai hệ thống, và đánh giá hiệu năng của hệ thống.

**Chương 5: Các giải pháp và đóng góp nổi bật.** Chương này trình bày tất cả nội dung đóng góp mà tôi thấy tâm đắc nhất trong suốt thời gian thực hiện đồ án. Cũng như các giải pháp mà tôi đã đưa ra để giải quyết các vấn đề khó khăn trong quá trình phát triển ứng dụng.

**Chương 6: Kết luận và hướng phát triển.** Chương này sẽ bao gồm các kết quả đạt được của sản phẩm so với các hệ thống hiện có. Cùng với đó là những cái tôi đã làm được hay chưa làm được và những bài học kinh nghiệm mà tôi đã rút ra được trong quá trình thực hiện đồ án. Cuối cùng, tôi sẽ đưa ra một số hướng phát triển cho sản phẩm trong tương lai.

# CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

## 2.1 Khảo sát hiện trạng

Để lên ý tưởng, khảo sát nhu cầu, và tìm giải pháp cho bài toán, bằng cách áp dụng các phương pháp như phỏng vấn người dùng, tìm hiểu và nghiên cứu các hệ thống hiện có. Ta có thể chỉ ra được những cái thiếu sót, đặc điểm đặc biệt và đưa ra được các chức năng phù hợp.

### 2.1.1 Phỏng vấn người dùng

Các chủ cửa hàng nhỏ thường phải đối mặt với những thách thức trong việc quản lý hàng tồn kho và hóa đơn một cách hiệu quả và chính xác. Ví dụ trong các quầy thuốc, các hóa đơn, giấy ghi chép nhập xuất hàng theo thời gian trở lên càng nhiều. Hơn nữa, thống kê doanh thu trở lên lặp lại nhàm chán, tốn sức. Đặc biệt hơn, rất khó để biết chính xác được sản phẩm gì còn hay hết. Việc họ cần một ứng dụng điện thoại cầm tay để theo dõi lượng hàng tồn, và ghi chép lại doanh số bán hàng, hóa đơn sẽ giúp tiết kiệm thời gian và giảm thiểu tổn thất sai sót do việc thủ công bàn giấy.

### 2.1.2 Khảo sát các hệ thống hiện có

Trên thị trường hiện nay đã có ứng dụng giúp các doanh nghiệp quản lý tồn kho, hóa đơn và khách hàng như là KiotViet. Họ cung cấp đầy đủ chức năng nhưng đối tượng khách hàng nhắm đến quá rộng và ứng dụng của họ được thêm các chức năng mà các cửa hàng nhỏ hầu như không cần thiết.

- **Quản lý và bán hàng**: Họ tách ra làm riêng 2 ứng dụng riêng lẻ khiến việc đổi ứng dụng để sử dụng khiến thêm thao tác thừa tốn thời gian.
- **Quảng cáo tiện ích**: Ứng dụng của họ hay có các quảng cáo về các dịch vụ tiện ích khác, lấy đi nhiều khoảng trống trên màn hình, gây khó chịu.
- **Chức năng sàn thương mại điện tử**: Họ cung cấp chức năng bán hàng trực tuyến, nhưng đối tượng khách hàng của họ là các doanh nghiệp lớn, không phù hợp với các cửa hàng nhỏ.
- **Dịch vụ theo tháng**: Để sử dụng ứng dụng, các cửa hàng phải bỏ ra một khoản tiền hàng tháng để có thể sử dụng và bị giới hạn chức năng, giới hạn số tài khoản người dùng theo gói đăng kí.
- **Dữ liệu không hoàn toàn kiểm soát**: Các dữ liệu khách hàng, hàng hóa đều nằm trên hệ thống của họ, khiến việc nếu các chủ cửa hàng muốn hoàn toàn nắm giữ dữ liệu và trích xuất ra các phần mềm khác bị khó khăn, phụ thuộc.

### 2.1.3 Mục tiêu và phạm vi của đề tài

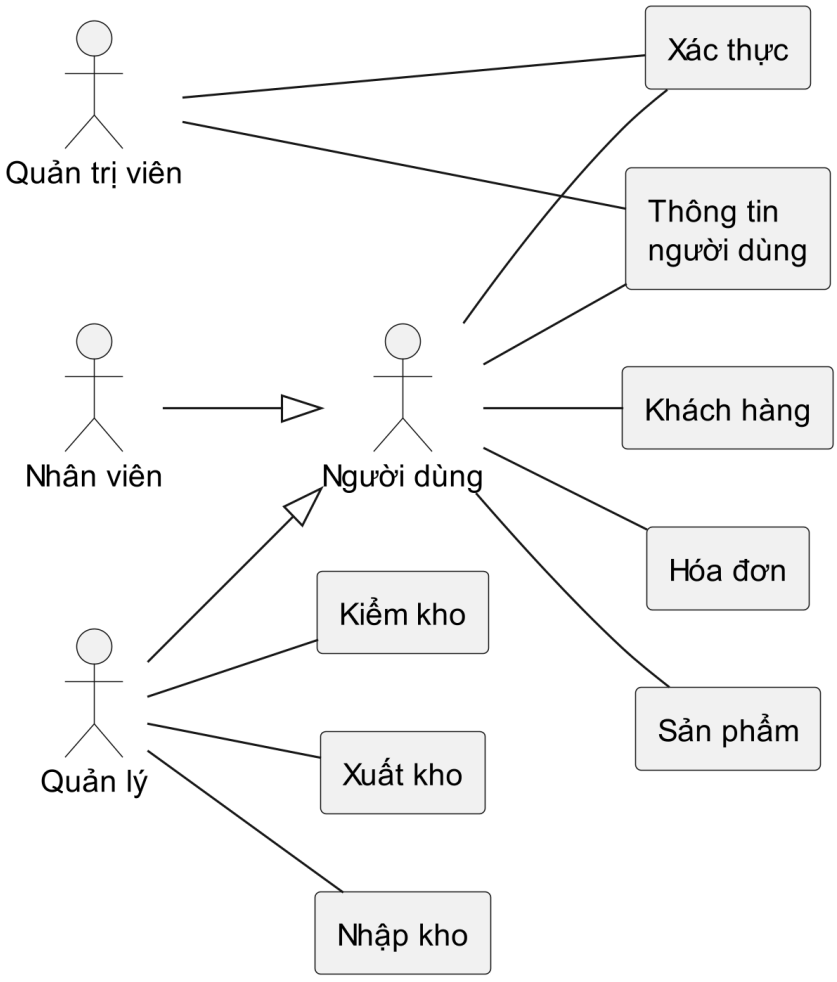
Hệ thống đề xuất sẽ giải quyết các vấn đề như trên và đáp ứng được chính xác đối tượng khách hàng mà đề tài đề ra. Hệ thống sẽ được xây dựng với các mục tiêu chính như sau:

- **Giao diện đơn giản**: Loại bỏ các chức năng không cần thiết, tập trung vào các chức năng chính, giúp người dùng dễ dàng sử dụng cho dù họ không có nhiều kinh nghiệm về công nghệ.
- **Hỗ trợ đa nền tảng**: Hệ thống được sử dụng bằng ứng dụng điện thoại chạy được trên Android, iOS.
- **Self-hosting**: Hệ thống được cài đặt trên máy chủ của người dùng, giúp người dùng hoàn toàn nắm giữ dữ liệu và có thể trích xuất ra các phần mềm khác. Ngoài ra, hệ thống cũng có thể được cài đặt trên các dịch vụ đám mây như AWS, Google Cloud, Azure, ... để giảm thiểu chi phí cho người dùng mà không bị giới hạn chức năng.
- **Lưu trữ đám mây**: Dữ liệu được lưu trữ trên đám mây, giúp người dùng có thể truy cập từ bất cứ đâu, bất cứ khi nào và tránh mất dữ liệu khi thiết bị bị hỏng.
- **Bảo mật dữ liệu**: Dữ liệu hoàn toàn được nằm trong máy chủ người dùng, hệ thống sẽ không lưu trữ bất cứ dữ liệu nào trên máy chủ của nhà phát triển.
- **Tính mở rộng**: Hệ thống có thể được thiết kế để hỗ trợ thêm các chức năng mới tùy theo nhu cầu của người dùng.

## 2.2 Tổng quan chức năng

### 2.2.1 Biểu đồ use case tổng quát

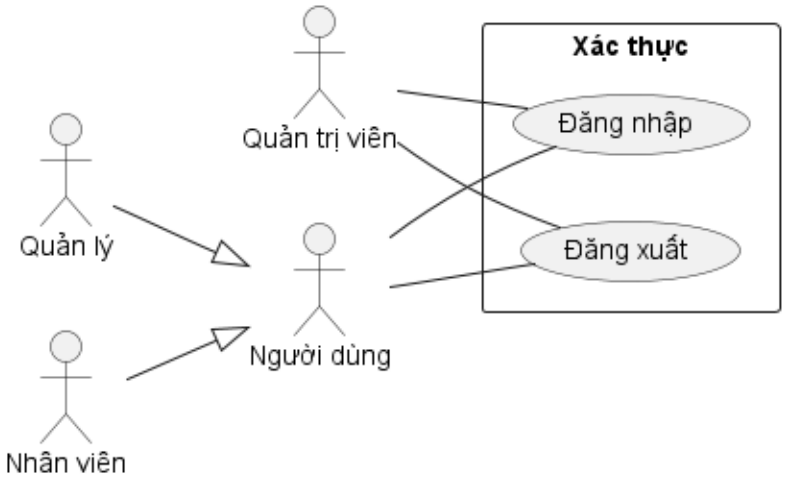
Hệ thống bao gồm các tác nhân được liệt kệ trong hình 2.1: Quản trị viên, Nhân viên, Quản lý. Phần lớn các use case đều xoay quanh các tác nhân Nhân viên và Quản lý. Quản trị viên chỉ có vai trò quản lý tài khoản người dùng, phân quyền truy cập hệ thống. Hệ thống gồm các use case tương ứng với từng thực thể chính: Sản phẩm (2.2.3), Phiếu nhập kho (2.2.4), xuất kho (2.2.5), kiểm kho (2.2.6), Hóa đơn (2.2.8), Khách hàng (2.2.7), Thông tin người dùng (2.2.9).




**Hình 2.1:** Biểu đồ use case tổng quát

### 2.2.2 Biểu đồ use case phân rã Xác thực người dùng

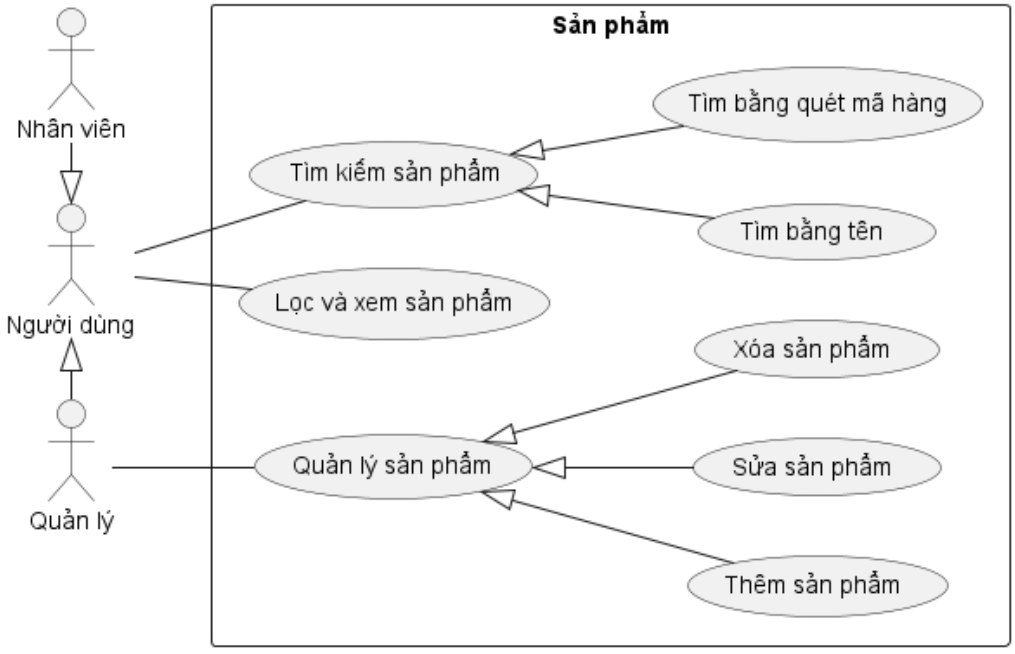
Như được mô tả trong hình 2.2. Use case này bao gồm hai use case con: Đăng nhập và Đăng xuất. Đăng nhập được sử dụng khi người dùng muốn đăng nhập vào hệ thống. Đăng xuất được sử dụng khi người dùng muốn đăng xuất khỏi hệ thống.




**Hình 2.2:** Biểu đồ use case phân rã Xác thực người dùng

### 2.2.3 Biểu đồ use case phân rã Sản phẩm

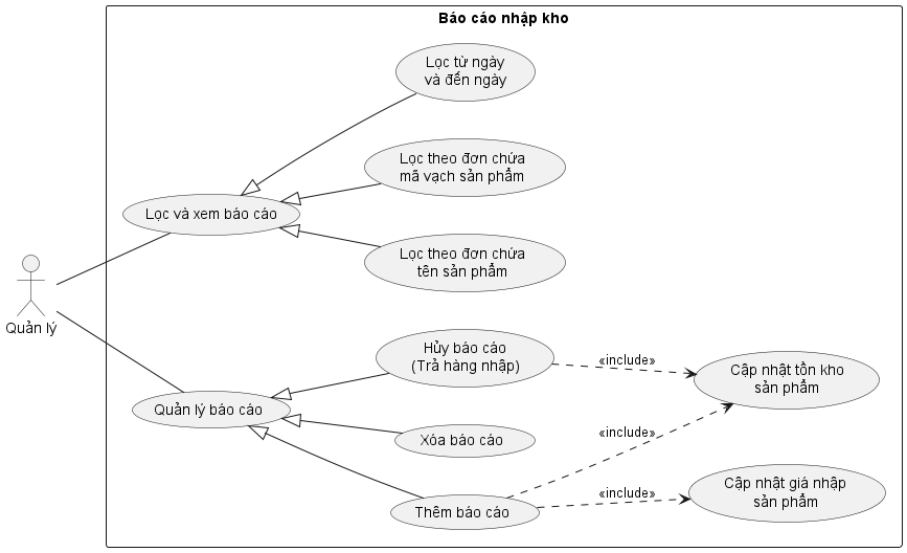
Như được mô tả trong hình 2.3. Use case này bao gồm các use case con: Lọc và xem sản phẩm, Quản lý sản phẩm, Tìm kiếm sản phẩm. Lọc và xem sản phẩm được sử dụng khi người dùng muốn lọc và xem thông tin sản phẩm. Quản lý sản phẩm được sử dụng khi người dùng muốn thêm, sửa, xóa sản phẩm. Tìm kiếm sản phẩm được sử dụng khi người dùng muốn tìm kiếm sản phẩm để thêm vào đơn từ.




**Hình 2.3:** Biểu đồ use case phân rã Sản phẩm

### 2.2.4 Biểu đồ use case phân rã Nhập kho

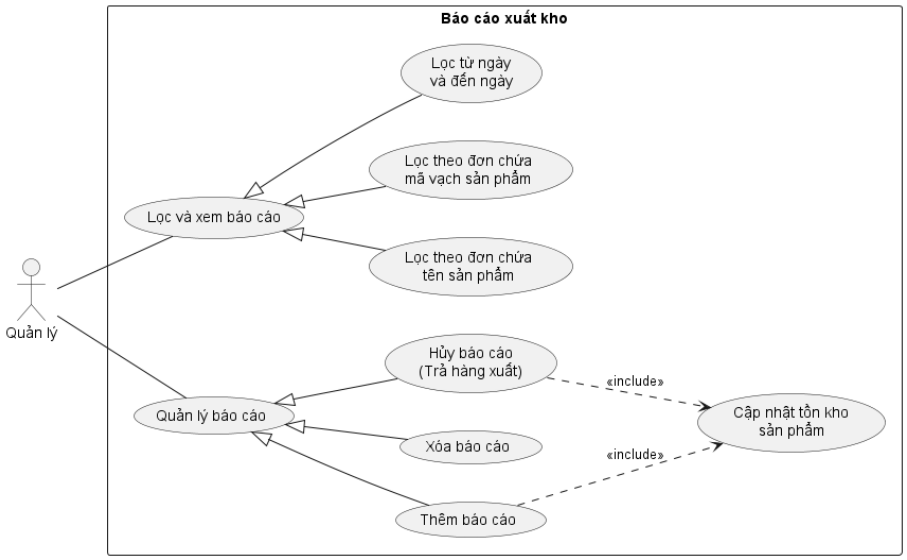
Như được mô tả trong hình 2.4. Use case này bao gồm các use case con: Lọc và xem phiếu nhập kho, Quản lý phiếu nhập kho. Lọc và xem phiếu nhập kho được sử dụng khi người dùng muốn lọc và xem thông tin phiếu nhập kho. Quản lý phiếu nhập kho được sử dụng khi người dùng muốn thêm, hủy, xóa phiếu nhập kho.




**Hình 2.4:** Biểu đồ use case phân rã Nhập kho

### 2.2.5 Biểu đồ use case phân rã Xuất kho

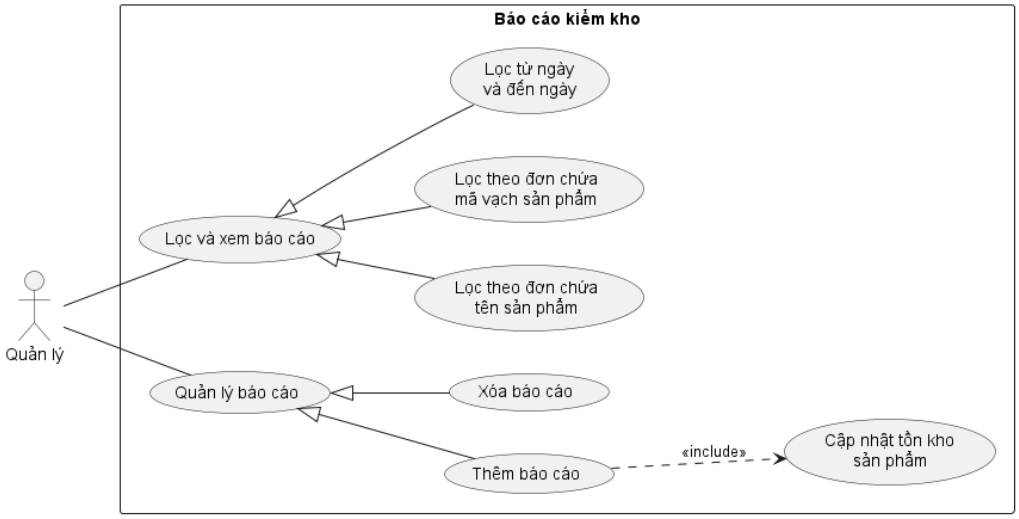
Như được mô tả trong hình 2.5. Use case này bao gồm các use case con: Lọc và xem phiếu xuất kho, Quản lý phiếu xuất kho. Lọc và xem phiếu xuất kho được sử dụng khi người dùng muốn lọc và xem thông tin phiếu xuất kho. Quản lý phiếu xuất kho được sử dụng khi người dùng muốn thêm, hủy, xóa phiếu xuất kho.




**Hình 2.5:** Biểu đồ use case phân rã Xuất kho

### 2.2.6 Biểu đồ use case phân rã Kiểm kho

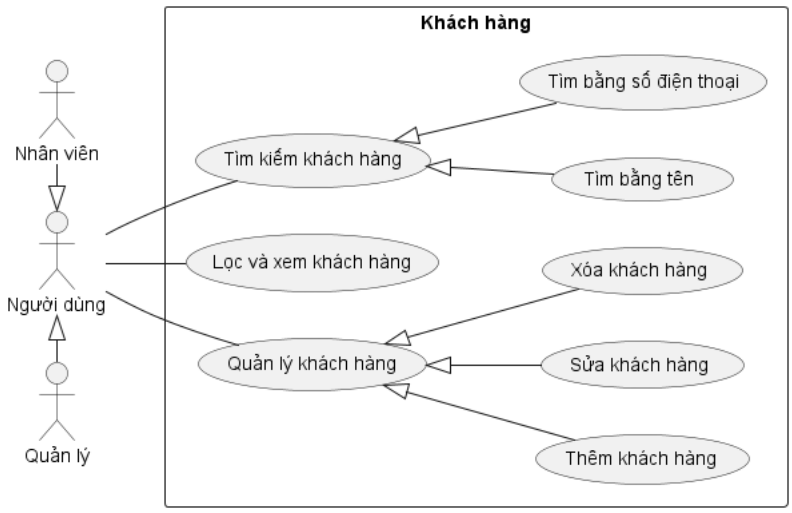
Như được mô tả trong hình 2.6. Use case này bao gồm các use case con: Lọc và xem phiếu kiểm kho, Quản lý phiếu kiểm kho. Lọc và xem phiếu kiểm kho được sử dụng khi người dùng muốn lọc và xem thông tin phiếu kiểm kho. Quản lý phiếu kiểm kho được sử dụng khi người dùng muốn thêm, xóa phiếu kiểm kho.




**Hình 2.6:** Biểu đồ use case phân rã Kiểm kho

### 2.2.7 Biểu đồ use case phân rã Khách hàng

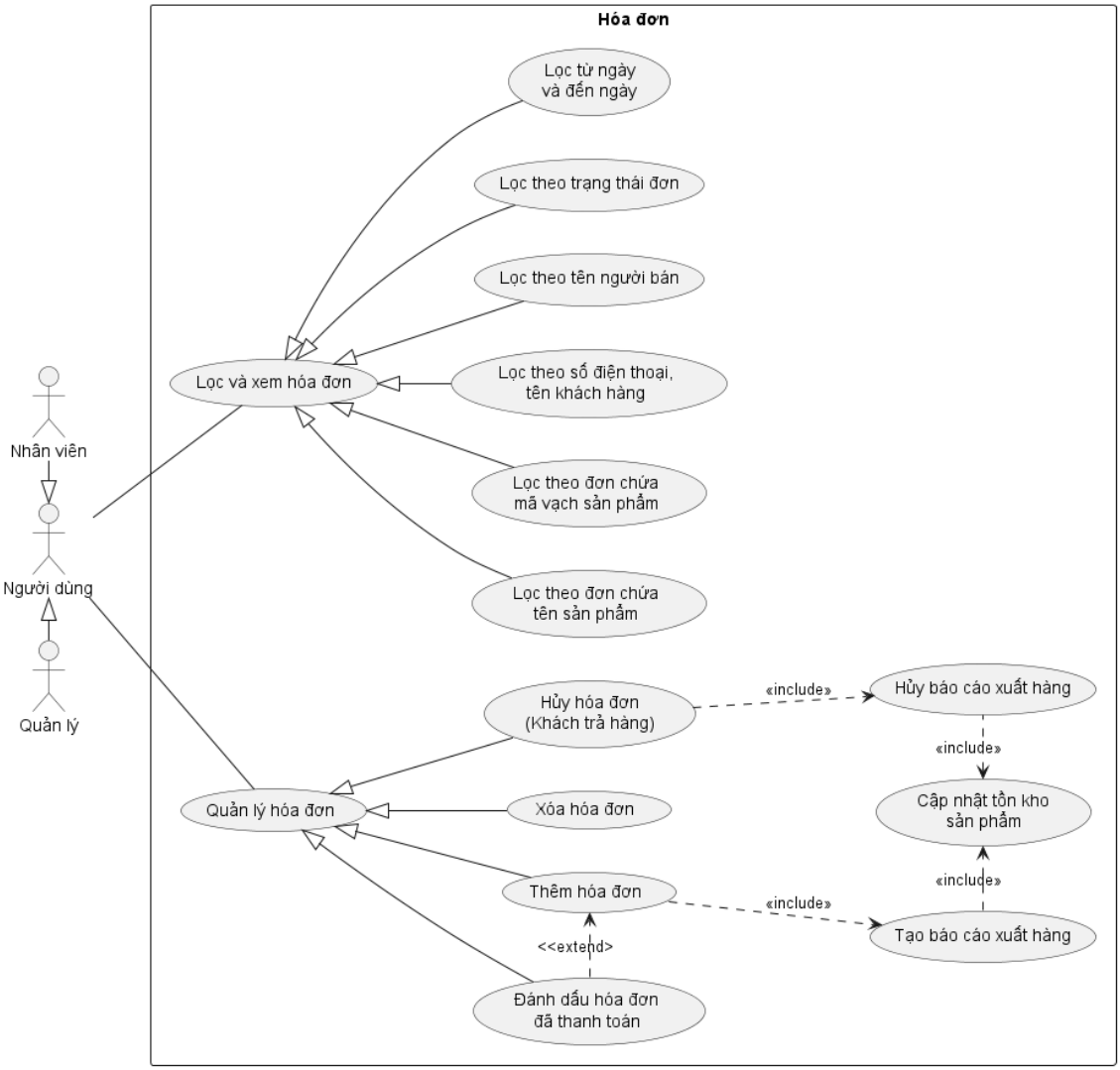
Như được mô tả trong hình 2.7. Use case này bao gồm các use case con: Lọc và xem khách hàng, Quản lý khách hàng, Tìm kiếm khách hàng. Lọc và xem khách hàng được sử dụng khi người dùng muốn lọc và xem thông tin khách hàng. Quản lý khách hàng được sử dụng khi người dùng muốn thêm, sửa, xóa khách hàng. Tìm kiếm khách hàng được sử dụng khi người dùng muốn tìm kiếm khách hàng để thêm vào hóa đơn.




**Hình 2.7:** Biểu đồ use case phân rã Khách hàng

### 2.2.8 Biểu đồ use case phân rã Hóa đơn

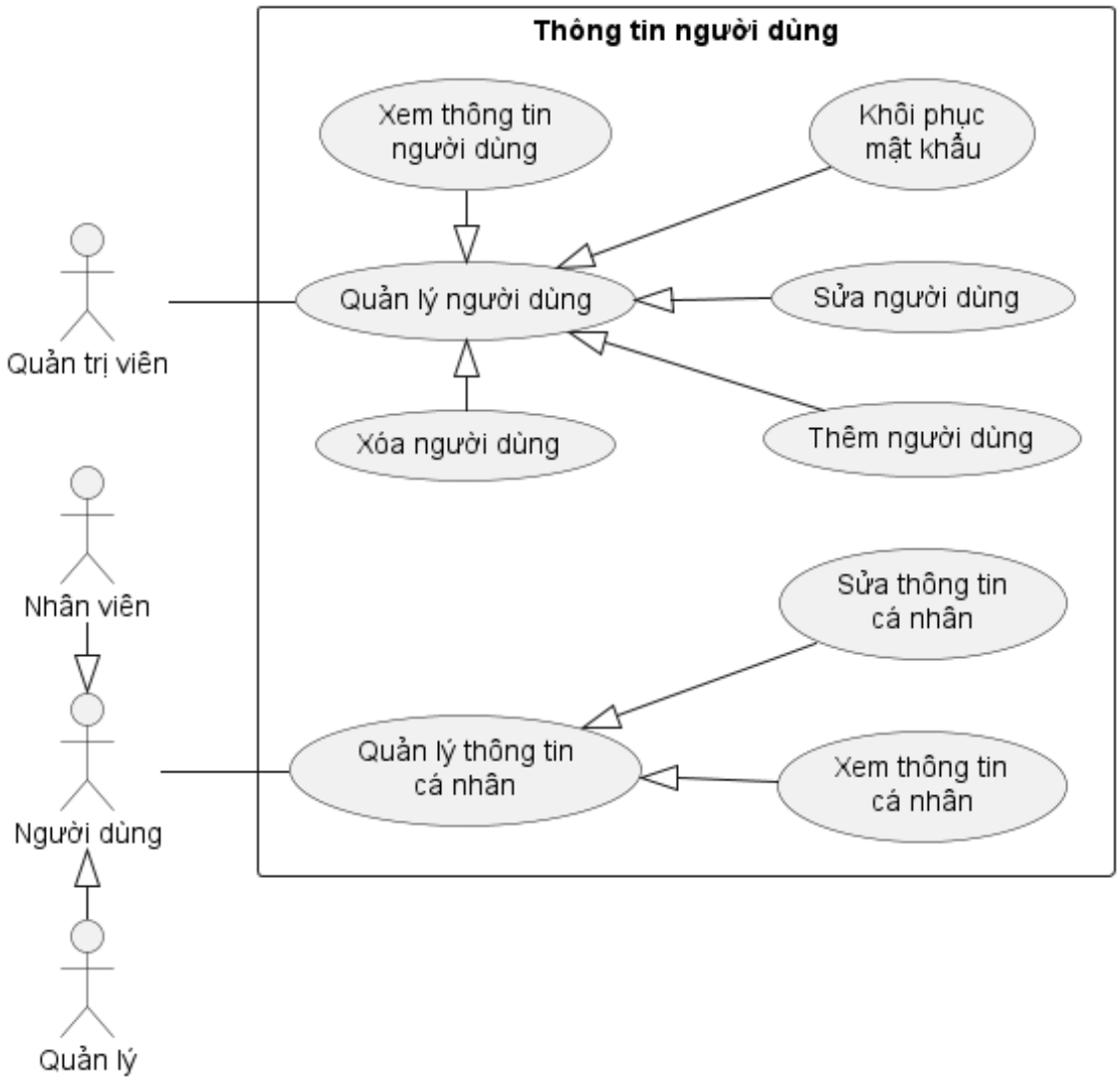
Như được mô tả trong hình 2.8. Use case này bao gồm các use case con: Lọc và xem hóa đơn, Quản lý hóa đơn. Lọc và xem hóa đơn được sử dụng khi người dùng muốn lọc và xem thông tin hóa đơn. Quản lý hóa đơn được sử dụng khi người dùng muốn thêm, xóa hóa đơn.




**Hình 2.8:** Biểu đồ use case phân rã Hóa đơn

### 2.2.9 Biểu đồ use case phân rã Thông tin người dùng

Như được mô tả trong hình 2.9. Use case này bao gồm các use case con: Quản lý thông tin người dùng, Quản lý thông tin cá nhân. Quản lý thông tin người dùng được sử dụng khi quản trị viên muốn thêm, sửa, xóa truy cập của người dùng hệ thống. Quản lý thông tin cá nhân được sử dụng khi người dùng muốn thay đổi thông tin cá nhân của mình.




**Hình 2.9:** Biểu đồ use case phân rã Thông tin người dùng

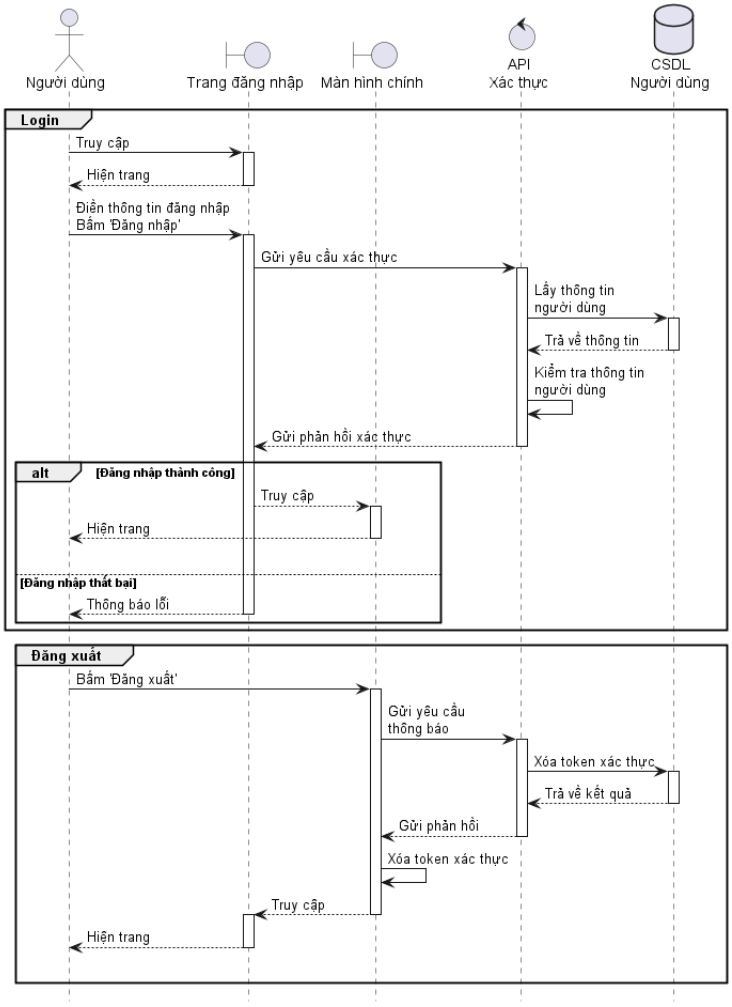
## 2.3 Đặc tả chức năng

### 2.3.1 Đặc tả use case Xác thực người dùng

Bảng 2.1 mô tả chi tiết use case Xác thực người dùng. Hình 2.10 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC00 | | |
| **Tên Use case** | Xác thực người dùng | | |
| **Tác nhân** | Quản trị viên, Nhân viên, Quản lý | | |
| **Mô tả** | Người dùng muốn đăng nhập, đăng xuất | | |
| **Tiền điều kiện** | Có tài khoản trong hệ thống | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Nhập tên, mật khẩu | Hiện trang chính |
| | **2** | Chọn nút đăng xuất | Hiện trang đăng nhập |
| **Luồng ngoại lệ** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Nhập tên, mật khẩu | Thông báo sai thông tin |
| **Hậu điều kiện** | Đăng nhập hoặc đăng xuất thành công | | |

**Bảng 2.1:** Đặc tả use case Xác thực người dùng



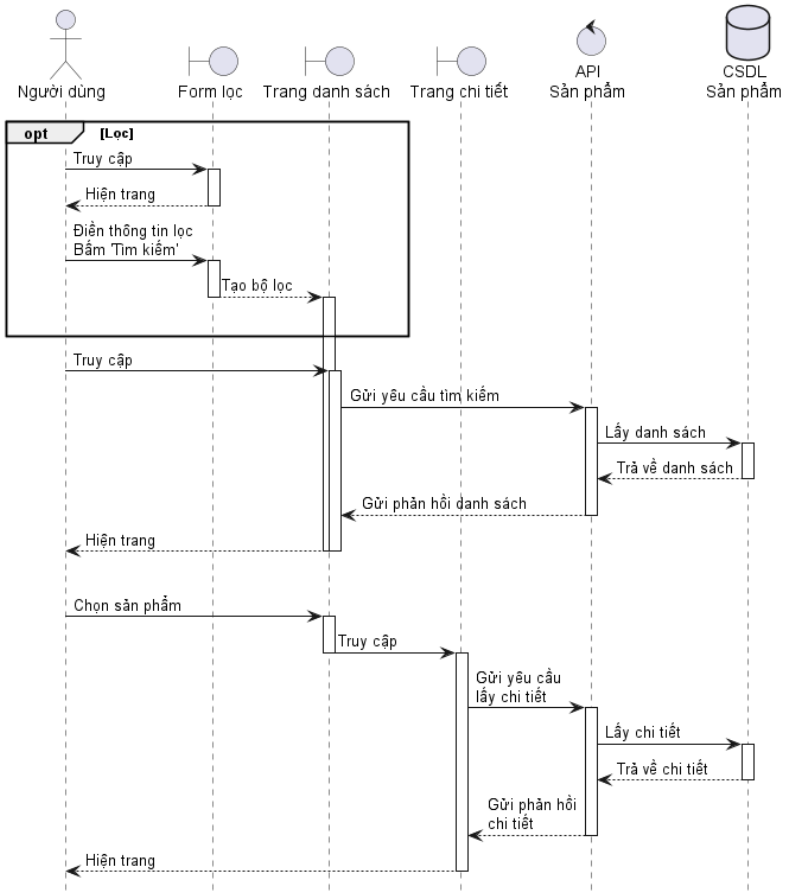

**Hình 2.10:** Biểu đồ tuần tự use case Xác thực người dùng

### 2.3.2 Đặc tả use case Lọc và xem sản phẩm

Bảng 2.2 mô tả chi tiết use case Lọc và xem sản phẩm. Hình 2.11 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC10 | | |
| **Tên Use case** | Lọc và xem sản phẩm | | |
| **Tác nhân** | Nhân viên, Quản lý | | |
| **Mô tả** | Người dùng muốn lọc bằng tên, mã vạch, sắp xếp và xem thông tin | | |
| **Tiền điều kiện** | Người dùng là Nhân viên hoặc Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút hình kính lúp | Hiện form lọc |
| | | Điền dữ liệu<br>Có thể chọn nút hình mã vạch để quét mã vạch | Điền mã vạch vào trường dữ liệu |
| | | Chọn nút 'Tìm kiếm' | Hiện màn danh sách |
| | | Chọn ô thông tin sản phẩm | Hiện màn thông tin |
| **Hậu điều kiện** | Lọc và Hiện chi tiết thông tin | | |

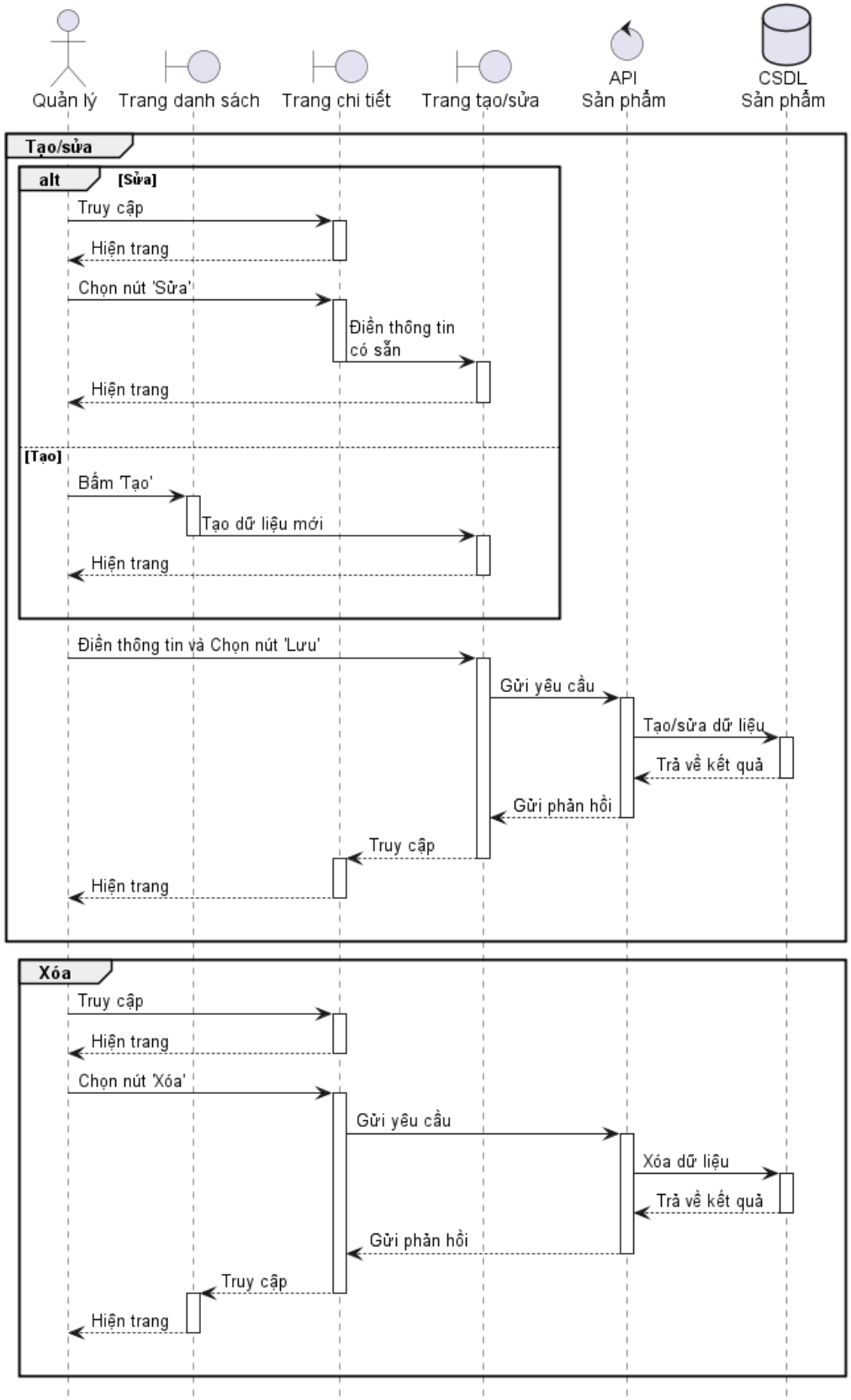
**Bảng 2.2:** Đặc tả use case Lọc và xem sản phẩm




**Hình 2.11:** Biểu đồ tuần tự use case Lọc và xem sản phẩm

### 2.3.3 Đặc tả use case Quản lý sản phẩm

Bảng 2.3 mô tả chi tiết use case Quản lý sản phẩm. Hình 2.12 mô tả biểu đồ tuần tự của use case này.




**Hình 2.12:** Biểu đồ tuần tự use case Quản lý sản phẩm

| **Mã Use case** | UC11 | | |
|---|---|---|---|
| **Tên Use case** | Quản lý sản phẩm | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn thêm, sửa, xóa sản phẩm | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn danh sách |
| | **2** | Chọn ô thông tin sản phẩm | Hiện màn thông tin |
| | | Chọn nút ’ ⋮ ’ → ’Sửa’ | Hiện form sửa |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn thông tin |
| | **3** | Chọn nút ’ ⋮ ’ → ’Xóa’ | Hiện màn danh sách |
| **Hậu điều kiện** | Thêm, sửa, xóa thông tin thành công | | |

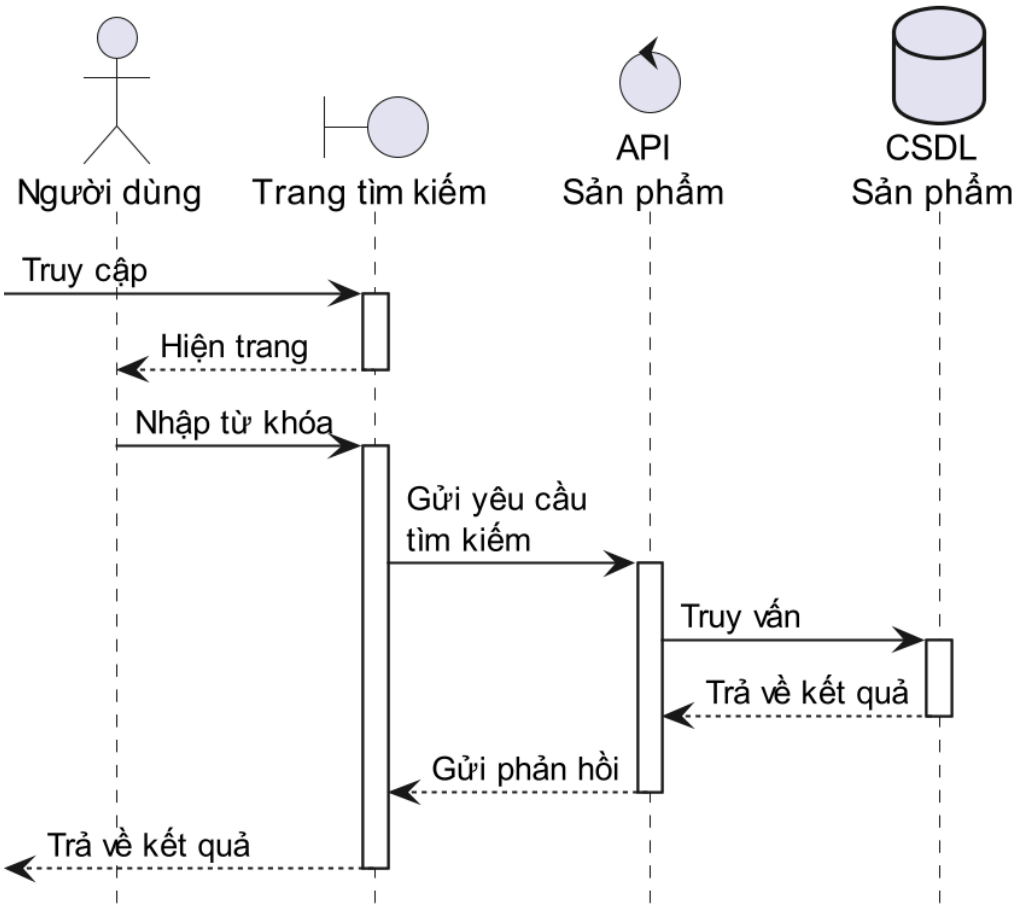
**Bảng 2.3:** Đặc tả use case Quản lý sản phẩm

### 2.3.4 Đặc tả use case Tìm kiếm sản phẩm

Bảng 2.4 mô tả chi tiết use case Tìm kiếm sản phẩm. Hình 2.13 mô tả biểu đồ tuần tự của use case này.

| **Mã Use case** | UC12 | | |
|---|---|---|---|
| **Tên Use case** | Tìm kiếm sản phẩm | | |
| **Tác nhân** | Nhân viên, Quản lý | | |
| **Mô tả** | Người dùng muốn tìm kiếm sản phẩm để thêm vào đơn từ | | |
| **Tiền điều kiện** | Người dùng là Nhân viên hoặc Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút ’+ Thêm’ | Hiện màn tìm kiếm |
| | | Điền tên hoặc mã vạch<br>Có thể chọn nút hình mã vạch để quét mã vạch | Điền mã vạch vào trường dữ liệu |
| | | Chọn nút ’Tìm kiếm’ | Hiện chuỗi danh sách |
| | | Chọn ô thông tin sản phẩm | Quay về màn đơn từ |
| **Hậu điều kiện** | Tìm kiếm và thêm sản phẩm vào đơn từ | | |

**Bảng 2.4:** Đặc tả use case Tìm kiếm sản phẩm



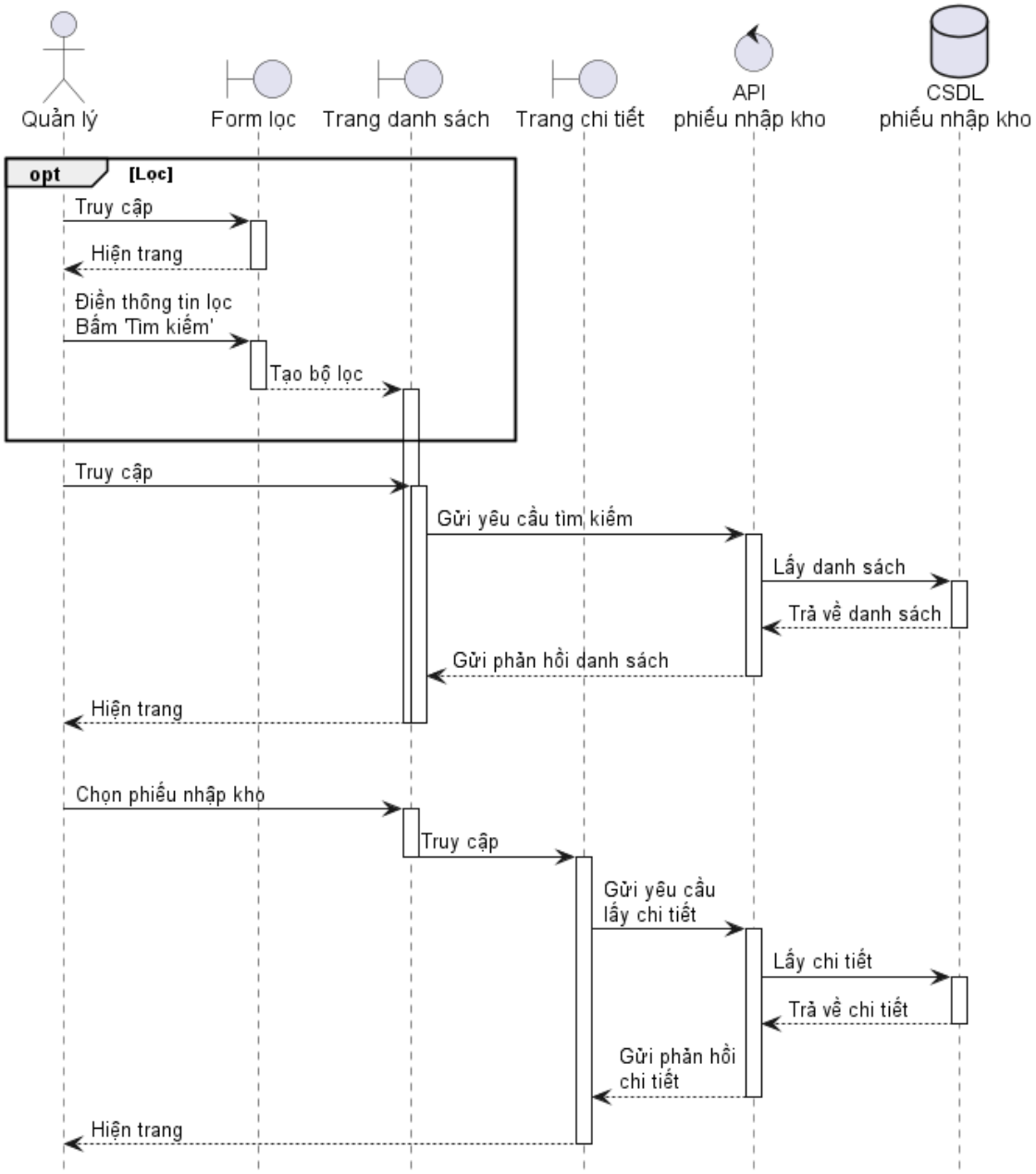

**Hình 2.13:** Biểu đồ tuần tự use case Tìm kiếm sản phẩm

### 2.3.5 Đặc tả use case Lọc và xem phiếu nhập kho

Bảng 2.5 mô tả chi tiết use case Lọc và xem phiếu nhập kho. Hình 2.14 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC20 | | |
| **Tên Use case** | Lọc và xem phiếu nhập kho | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn lọc bằng tên, mã vạch, ngày tháng, sắp xếp và xem thông tin | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút hình kính lúp | Hiện form lọc |
| | | Điền dữ liệu<br>Có thể chọn nút hình<br>mã vạch để quét mã vạch | Điền mã vạch vào<br>trường dữ liệu |
| | | Chọn nút 'Tìm kiếm' | Hiện màn danh sách |
| | | Chọn ô thông tin phiếu đơn | Hiện màn thông tin |
| **Hậu điều kiện** | Lọc và Hiện chi tiết thông tin | | |

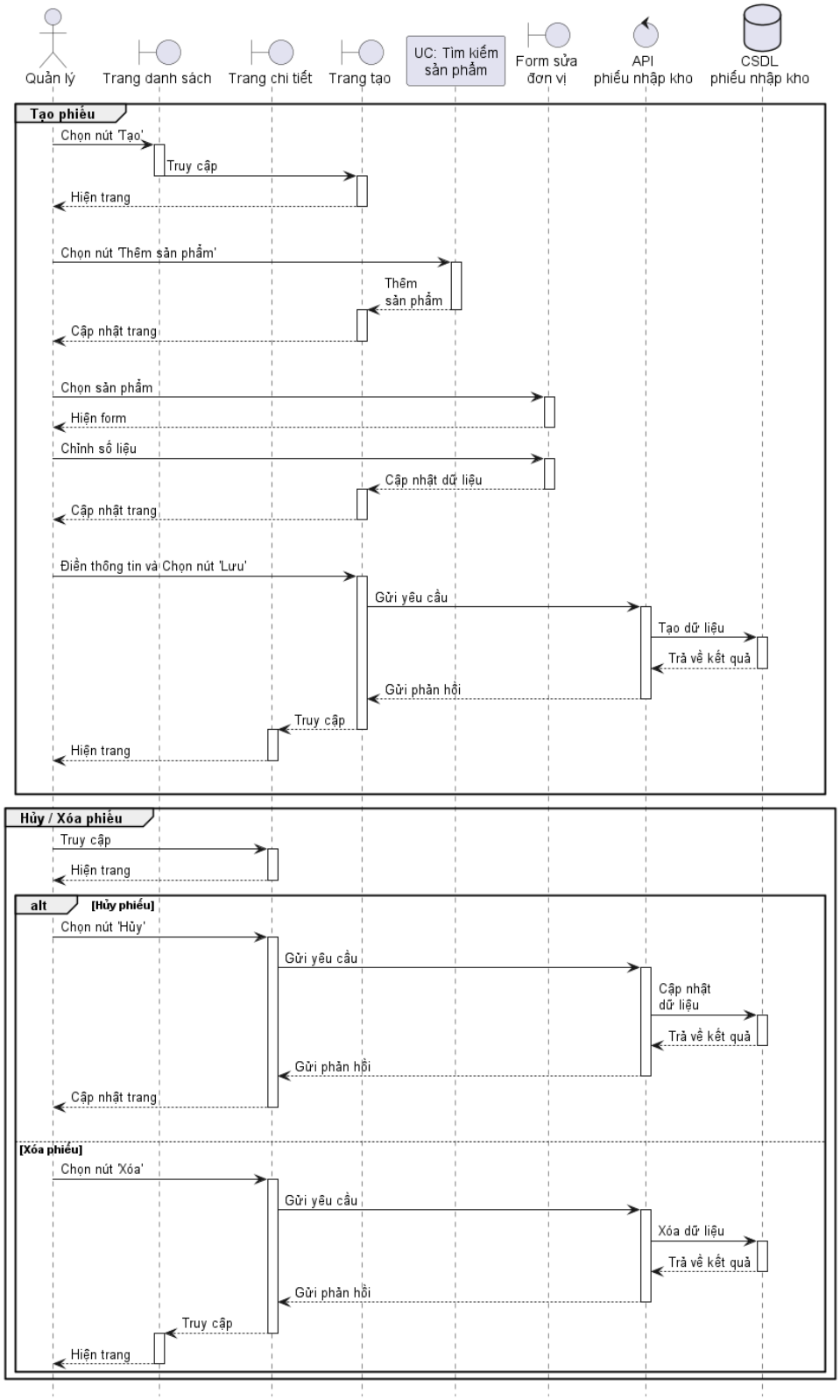
**Bảng 2.5:** Đặc tả use case Lọc và xem phiếu nhập kho




**Hình 2.14:** Biểu đồ tuần tự use case Lọc và xem phiếu nhập kho

### 2.3.6 Đặc tả use case Quản lý phiếu nhập kho

Bảng 2.6 mô tả chi tiết use case Quản lý phiếu nhập kho. Hình 2.15 mô tả biểu đồ tuần tự của use case này.




**Hình 2.15:** Biểu đồ tuần tự use case Quản lý phiếu nhập kho

| **Mã Use case** | UC21 | | |
|---|---|---|---|
| **Tên Use case** | Quản lý phiếu nhập kho | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn thêm, hủy, xóa phiếu nhập kho | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Chọn nút ’+ Thêm’ để thêm sản phẩm vào phiếu | Hiện màn tìm kiếm sản phẩm |
| | | Sau khi chọn sản phẩm, chọn ô thông tin sản phẩm | Hiện form chỉnh số lượng, giá nhập |
| | | Có thể quẹt trái để xóa | Xóa mục sản phẩm |
| | | Có thể quẹt phải để xem chi tiết sản phẩm | Hiện màn thông tin sản phẩm |
| | | Có thể chọn nút ’Xóa’ | Xóa hết mục sản phẩm |
| | | Chọn nút ’Lưu’ | Hiện màn danh sách |
| | **2** | Chọn ô thông tin phiếu đơn | Hiện màn thông tin |
| | | Chọn nút ’⋮’ $\rightarrow$ ’Hủy’ | Cập nhật thông tin |
| | **3** | Chọn nút ’⋮’ $\rightarrow$ ’Xóa’ | Hiện màn danh sách |
| **Luồng ngoại lệ** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Sau khi chọn sản phẩm | Thông báo đã tồn tại sản phẩm trong đơn |
| **Hậu điều kiện** | Thêm, hủy, xóa phiếu đơn thành công | | |

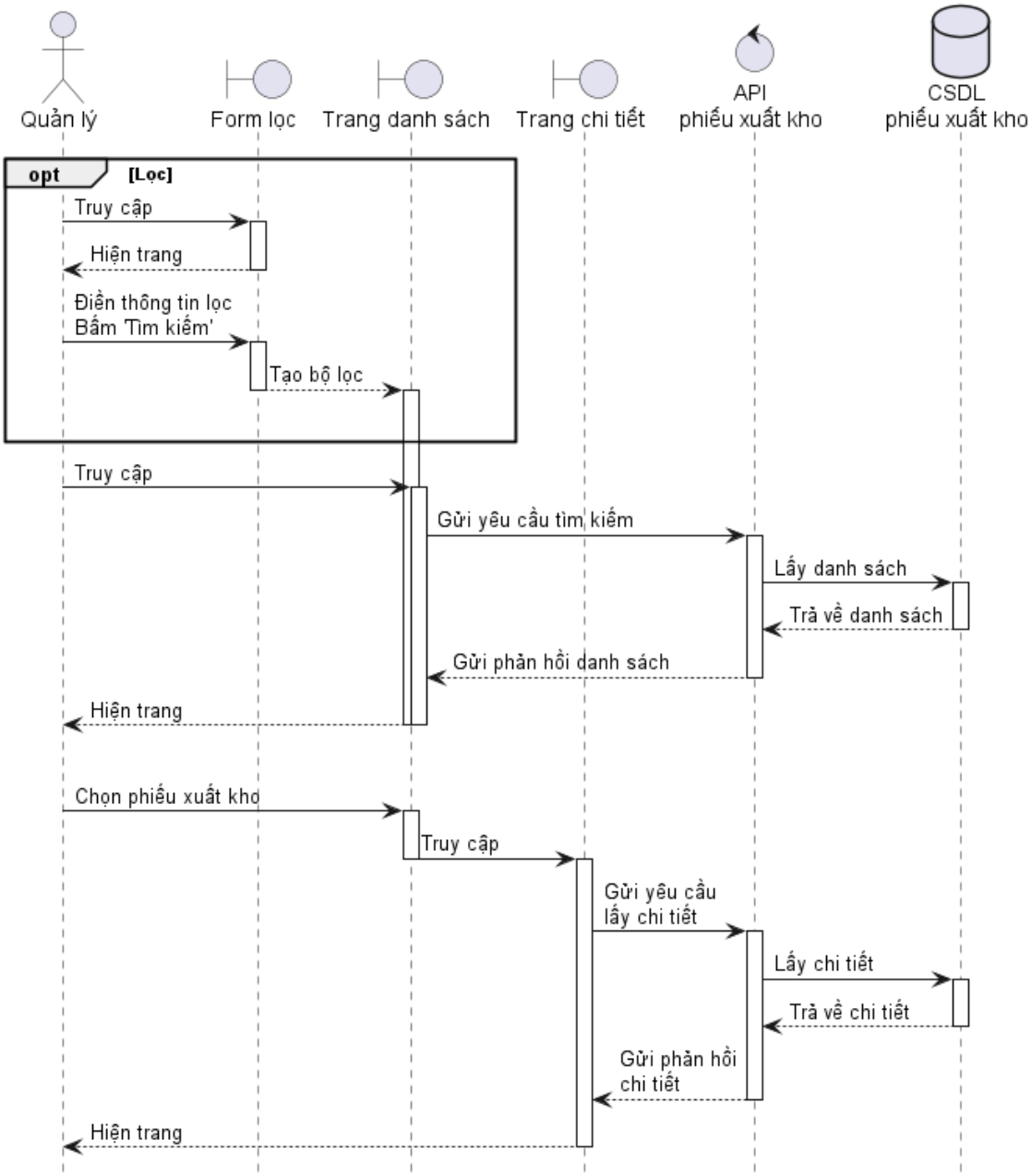
**Bảng 2.6:** Đặc tả use case Quản lý phiếu nhập kho

### 2.3.7 Đặc tả use case Lọc và xem phiếu xuất kho

Bảng 2.7, Hình 2.16 mô tả chi tiết use case Lọc và xem phiếu xuất kho.

| **Mã Use case** | UC30 | | |
|---|---|---|---|
| **Tên Use case** | Lọc và xem phiếu xuất kho | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn lọc bằng tên, mã vạch, ngày tháng, sắp xếp và xem thông tin | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút hình kính lúp | Hiện form lọc |
| | | Điền dữ liệu<br>Có thể chọn nút hình mã vạch để quét mã vạch | Điền mã vạch vào trường dữ liệu |
| | | Chọn nút ’Tìm kiếm’ | Hiện màn danh sách |
| | | Chọn ô thông tin phiếu đơn | Hiện màn thông tin |
| **Hậu điều kiện** | Lọc và Hiện chi tiết thông tin | | |

**Bảng 2.7:** Đặc tả use case Lọc và xem phiếu xuất kho



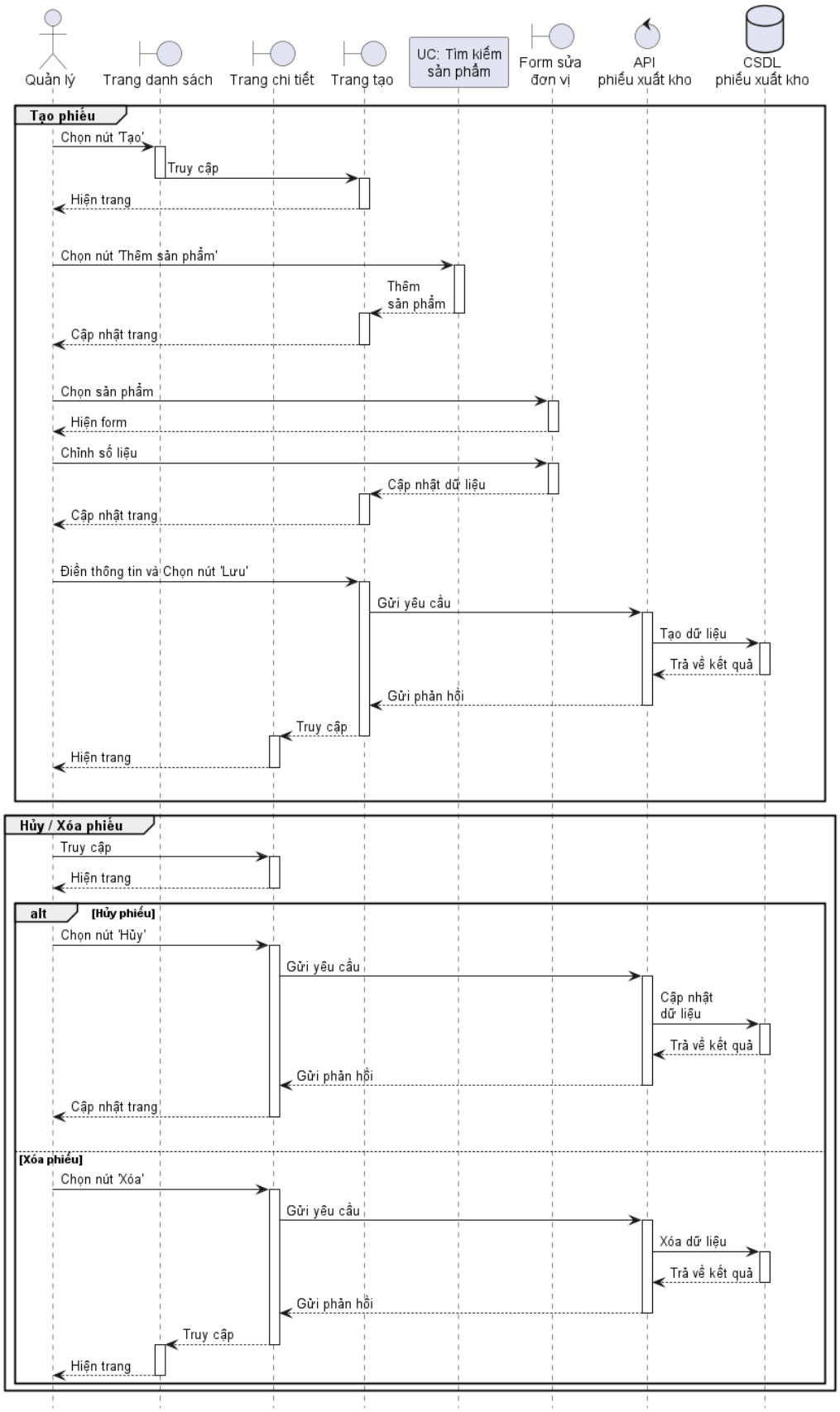

**Hình 2.16:** Biểu đồ tuần tự use case Lọc và xem phiếu xuất kho

### 2.3.8 Đặc tả use case Quản lý phiếu xuất kho

Bảng 2.8 mô tả chi tiết use case Quản lý phiếu xuất kho. Hình 2.17 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC31 | | |
| **Tên Use case** | Quản lý phiếu xuất kho | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn thêm, hủy, xóa phiếu xuất kho | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Chọn nút ’+ Thêm’ để thêm sản phẩm vào phiếu | Hiện màn tìm kiếm sản phẩm |
| | | Sau khi chọn sản phẩm, chọn ô thông tin sản phẩm | Hiện form chỉnh số lượng |
| | | Có thể quẹt trái để xóa | Xóa mục sản phẩm |
| | | Có thể quẹt phải để xem chi tiết sản phẩm | Hiện màn thông tin sản phẩm |
| | | Có thể chọn nút ’Xóa’ | Xóa hết mục sản phẩm |
| | | Chọn nút ’Lưu’ | Hiện màn danh sách |
| | **2** | Chọn ô thông tin phiếu đơn | Hiện màn thông tin |
| | | Chọn nút ’⋮’ $\rightarrow$ ’Hủy’ | Cập nhật thông tin |
| | **3** | Chọn nút ’⋮’ $\rightarrow$ ’Xóa’ | Hiện màn danh sách |
| **Luồng ngoại lệ** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút ’+ Thêm’ để thêm sản phẩm vào đơn | Hiện màn tìm kiếm sản phẩm |
| | | Sau khi chọn sản phẩm | Thông báo đã tồn tại sản phẩm trong đơn |
| | | | Thông báo hết hàng sản phẩm trong kho |
| **Hậu điều kiện** | Thêm, hủy, xóa phiếu đơn thành công | | |

**Bảng 2.8:** Đặc tả use case Quản lý phiếu xuất kho



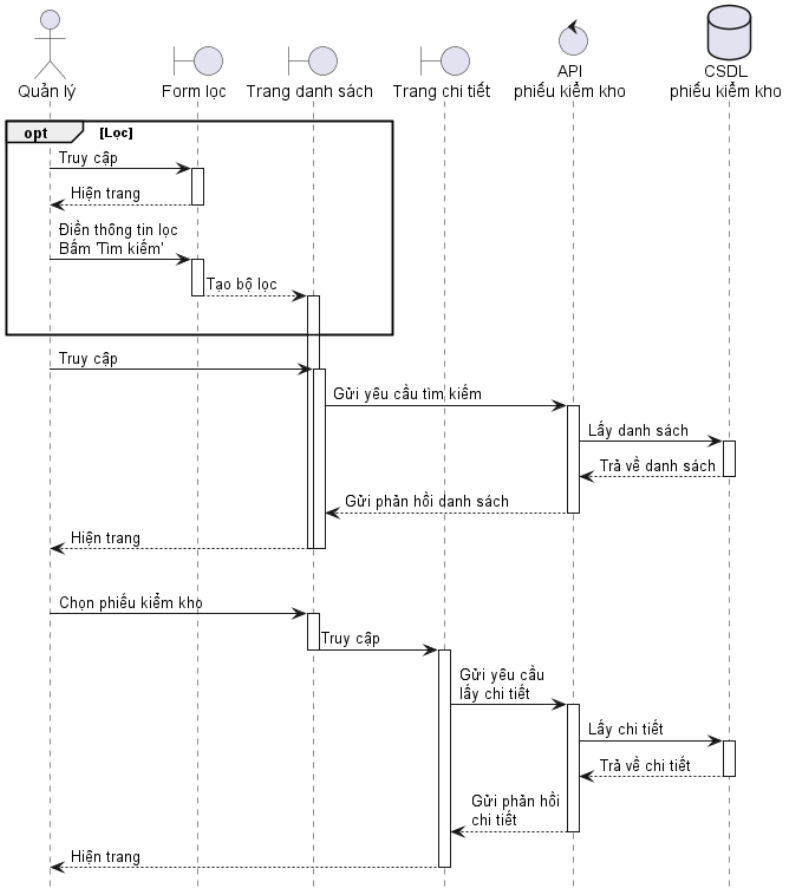

**Hình 2.17:** Biểu đồ tuần tự use case Quản lý phiếu xuất kho

### 2.3.9 Đặc tả use case Lọc và xem phiếu kiểm kho

Bảng 2.9 mô tả chi tiết use case Lọc và xem phiếu kiểm kho. Hình 2.18 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC40 | | |
| **Tên Use case** | Lọc và xem phiếu kiểm kho | | |
| **Tác nhân** | Quản lý | | |
| **Mô tả** | Người dùng muốn lọc bằng tên, mã vạch, ngày tháng, sắp xếp và xem thông tin | | |
| **Tiền điều kiện** | Người dùng là Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút hình kính lúp | Hiện form lọc |
| | | Điền dữ liệu<br>Có thể chọn nút hình<br>mã vạch để quét mã vạch | Điền mã vạch vào<br>trường dữ liệu |
| | | Chọn nút 'Tìm kiếm' | Hiện màn danh sách |
| | | Chọn ô thông tin phiếu đơn | Hiện màn thông tin |
| **Hậu điều kiện** | Lọc và Hiện chi tiết thông tin | | |

**Bảng 2.9:** Đặc tả use case Lọc và xem phiếu kiểm kho



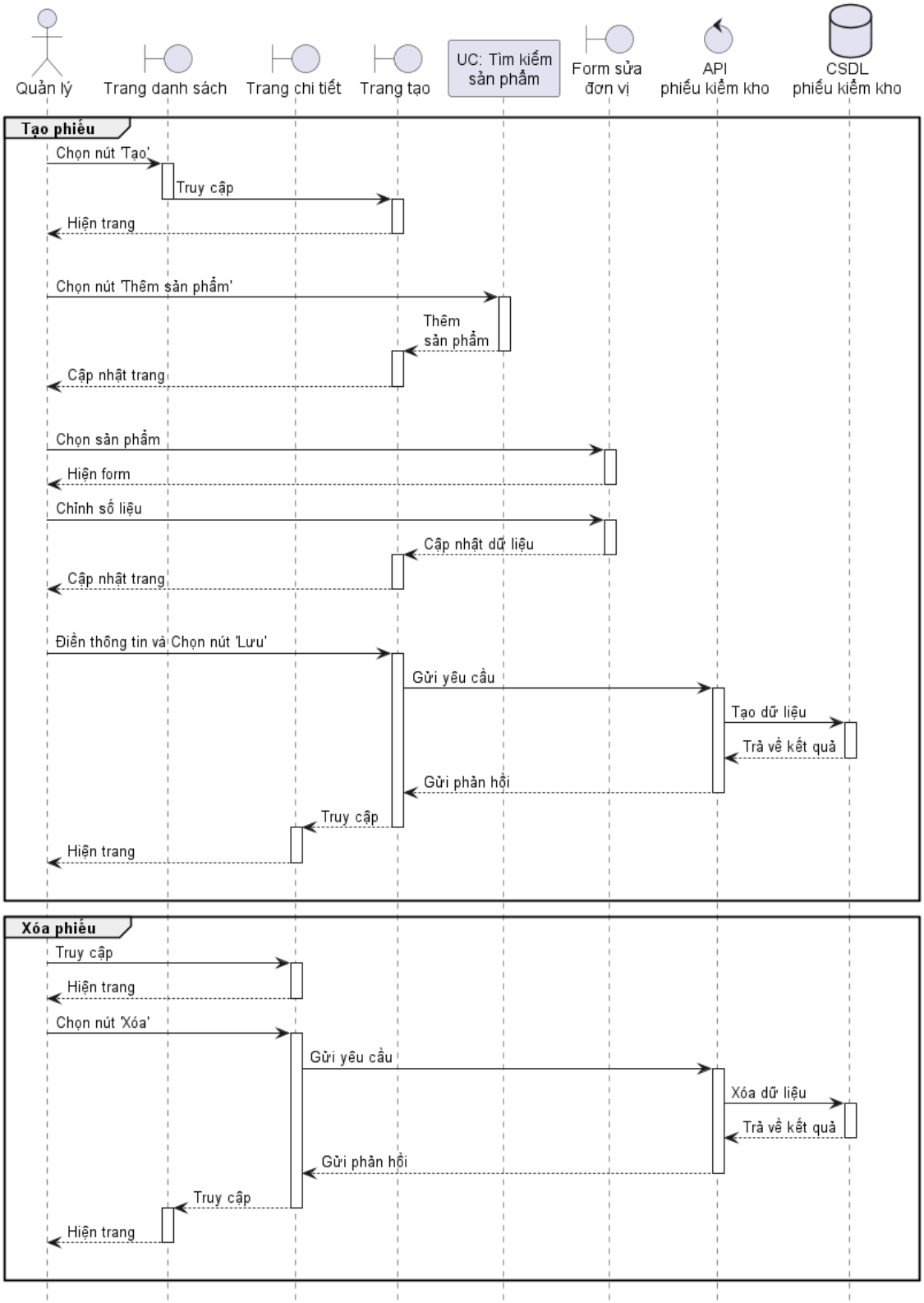

**Hình 2.18:** Biểu đồ tuần tự use case Lọc và xem phiếu kiểm kho

### 2.3.10 Đặc tả use case Quản lý phiếu kiểm kho

Bảng 2.10 mô tả chi tiết use case Quản lý phiếu kiểm kho. Hình 2.19 mô tả biểu đồ tuần tự của use case này.

<table>
<tr><td>Mã Use case</td><td colspan="3">UC41</td></tr>
<tr><td>Tên Use case</td><td colspan="3">Quản lý phiếu kiểm kho</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Quản lý</td></tr>
<tr><td>Mô tả</td><td colspan="3">Người dùng muốn thêm, xóa phiếu kiểm kho</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng là Quản lý</td></tr>
<tr><td rowspan="10">Luồng sự kiện</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="7">1</td><td>Chọn nút tạo mới ’ + ’</td><td>Hiện form tạo</td></tr>
<tr><td>Chọn nút ’+ Thêm’ để thêm sản phẩm vào phiếu</td><td>Hiện màn tìm kiếm sản phẩm</td></tr>
<tr><td>Sau khi chọn sản phẩm, chọn ô thông tin sản phẩm</td><td>Hiện form chỉnh số lượng chênh lệch</td></tr>
<tr><td>Có thể quẹt trái để xóa</td><td>Xóa mục sản phẩm</td></tr>
<tr><td>Có thể quẹt phải để xem chi tiết sản phẩm</td><td>Hiện màn thông tin sản phẩm</td></tr>
<tr><td>Có thể chọn nút ’Xóa’</td><td>Xóa hết mục sản phẩm</td></tr>
<tr><td>Chọn nút ’Lưu’</td><td>Hiện màn danh sách</td></tr>
<tr><td rowspan="2">2</td><td>Chọn ô thông tin phiếu đơn</td><td>Hiện màn thông tin</td></tr>
<tr><td>Chọn nút ’ ⋮ ’ $\rightarrow$ ’Xóa’</td><td>Hiện màn danh sách</td></tr>
<tr><td rowspan="3">Luồng ngoại lệ</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="2">1</td><td>Chọn nút ’+ Thêm’ để thêm sản phẩm vào đơn</td><td>Hiện màn tìm kiếm sản phẩm</td></tr>
<tr><td>Sau khi chọn sản phẩm</td><td>Thông báo đã tồn tại sản phẩm trong đơn</td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Thêm, xóa phiếu đơn thành công</td></tr>
</table>

**Bảng 2.10:** Đặc tả use case Quản lý phiếu kiểm kho



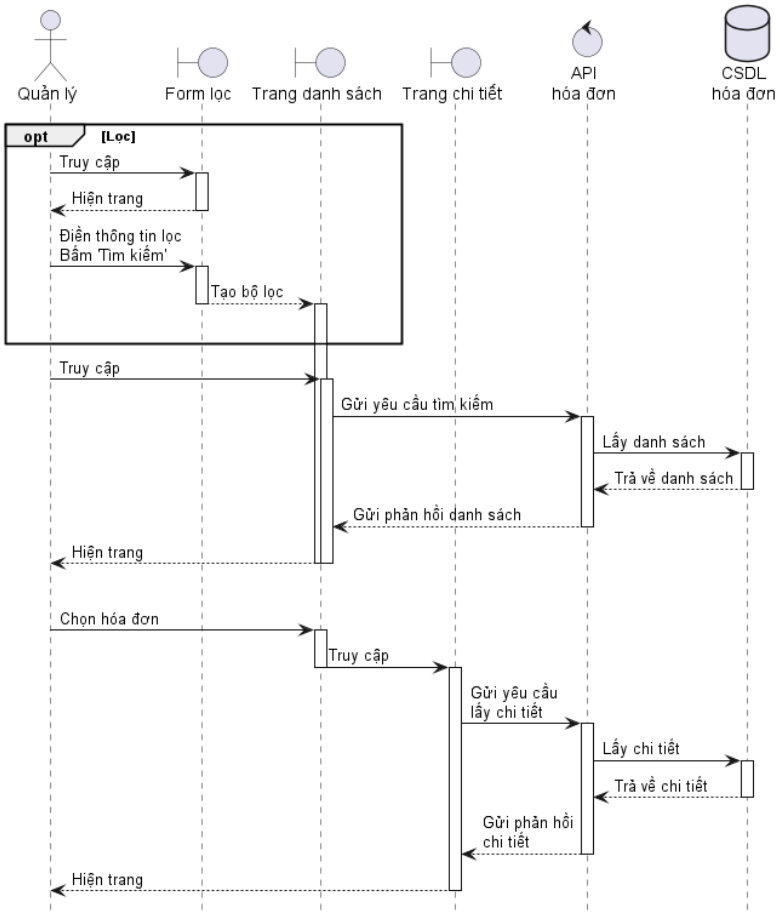

**Hình 2.19:** Biểu đồ tuần tự use case Quản lý phiếu kiểm kho

### 2.3.11 Đặc tả use case Lọc và xem hóa đơn

Bảng 2.11 mô tả chi tiết use case Lọc và xem hóa đơn. Hình 2.20 mô tả biểu đồ tuần tự của use case này.

<table>
<tr><td>Mã Use case</td><td colspan="3">UC50</td></tr>
<tr><td>Tên Use case</td><td colspan="3">Lọc và xem hóa đơn</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Nhân viên, Quản lý</td></tr>
<tr><td>Mô tả</td><td colspan="3">Người dùng muốn lọc bằng tên, mã vạch, trạng thái ngày tháng, sắp xếp và xem thông tin</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng là Nhân viên hoặc Quản lý</td></tr>
<tr><td rowspan="5">Luồng sự kiện</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="4">1</td><td>Chọn nút hình kính lúp</td><td>Hiện form lọc</td></tr>
<tr><td>Điền dữ liệu<br>Có thể chọn nút hình mã vạch để quét mã vạch</td><td>Điền mã vạch vào trường dữ liệu</td></tr>
<tr><td>Chọn nút ’Tìm kiếm’</td><td>Hiện màn danh sách</td></tr>
<tr><td>Chọn ô thông tin hóa đơn</td><td>Hiện màn thông tin</td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Lọc và Hiện chi tiết thông tin</td></tr>
</table>

**Bảng 2.11:** Đặc tả use case Lọc và xem hóa đơn



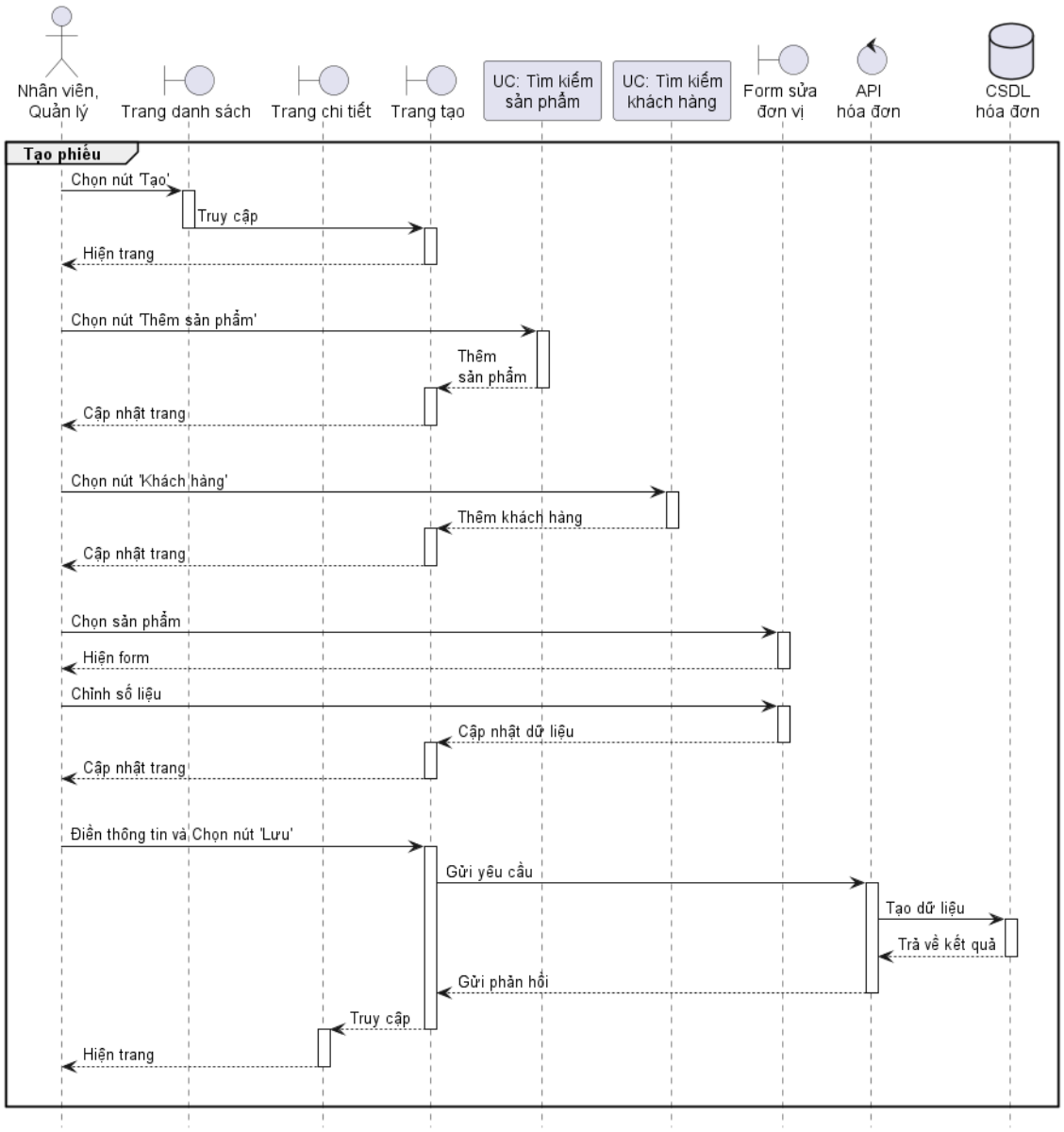

**Hình 2.20:** Biểu đồ tuần tự use case Lọc và xem hóa đơn

### 2.3.12 Đặc tả use case Quản lý hóa đơn

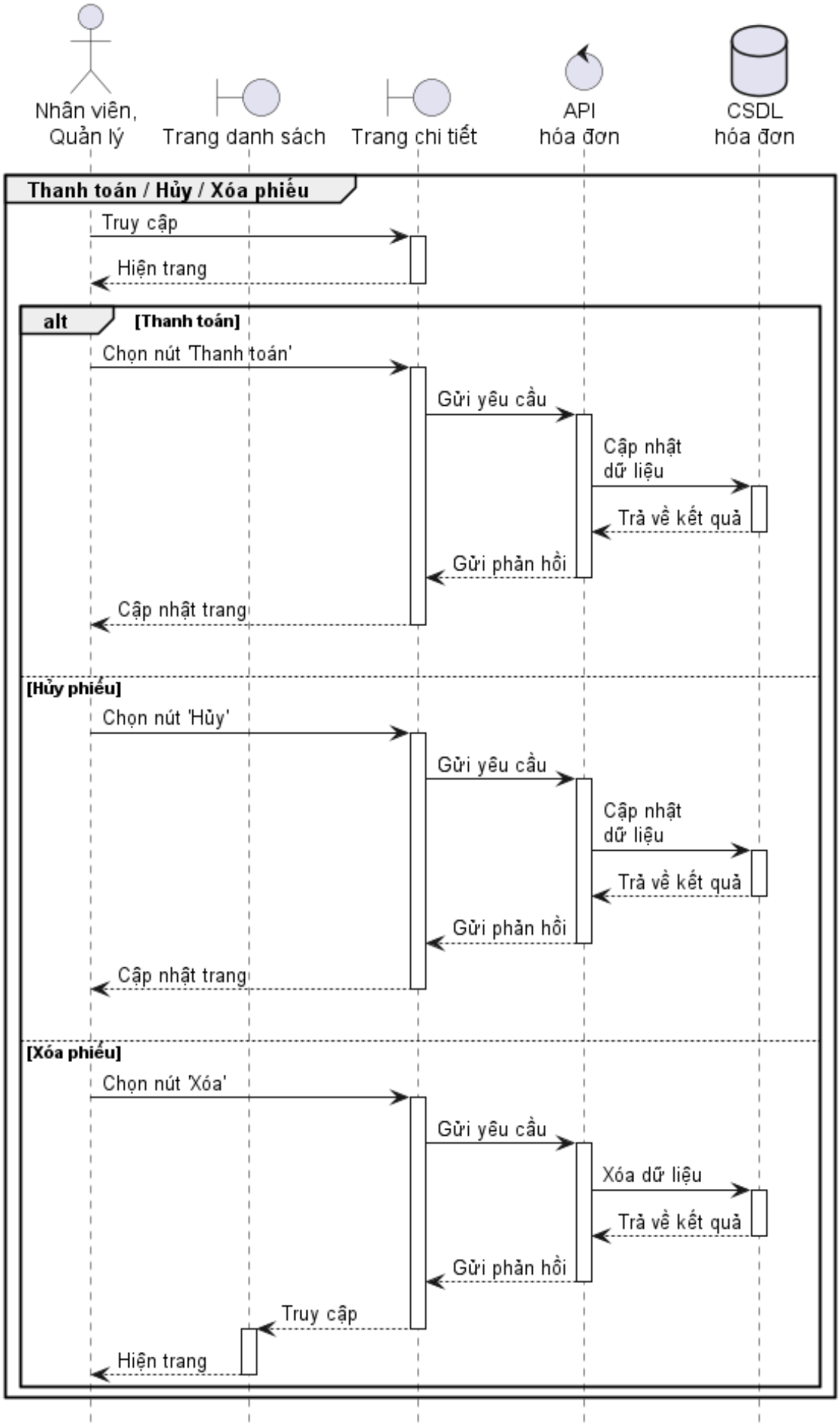
Bảng 2.12 mô tả chi tiết use case Quản lý hóa đơn. Hình 2.21 và 2.22 mô tả biểu đồ tuần tự của use case này.

<table>
<tr><td>Mã Use case</td><td colspan="3">UC51</td></tr>
<tr><td>Tên Use case</td><td colspan="3">Quản lý hóa đơn</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Nhân viên, Quản lý</td></tr>
<tr><td>Mô tả</td><td colspan="3">Người dùng muốn thêm, thanh toán, hủy, xóa hóa đơn</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng là Nhân viên hoặc Quản lý</td></tr>
<tr><td rowspan="12">Luồng sự kiện</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="8">1</td><td>Chọn nút tạo mới ’ + ’</td><td>Hiện form tạo</td></tr>
<tr><td>Chọn nút ’+ Thêm’ để thêm sản phẩm vào đơn</td><td>Hiện màn tìm kiếm sản phẩm</td></tr>
<tr><td>Sau khi chọn sản phẩm, chọn ô thông tin sản phẩm</td><td>Hiện form chỉnh số lượng sản phẩm</td></tr>
<tr><td>Có thể quẹt trái để xóa</td><td>Xóa mục sản phẩm</td></tr>
<tr><td>Có thể quẹt phải để xem chi tiết sản phẩm</td><td>Hiện màn thông tin sản phẩm</td></tr>
<tr><td>Có thể chọn nút ’Xóa’</td><td>Xóa hết mục sản phẩm</td></tr>
<tr><td>Chọn thanh ’Khách hàng’ để thêm vào hóa đơn</td><td>Hiện màn tìm kiếm khách hàng</td></tr>
<tr><td>Điền tổng tiền sẽ bán<br>Có thể tích ô để đánh dấu thanh toán ngay<br>Chọn nút ’Lưu’</td><td><br><br>Hiện màn danh sách</td></tr>
<tr><td rowspan="2">2</td><td>Chọn ô thông tin hóa đơn</td><td>Hiện màn thông tin</td></tr>
<tr><td>Chọn nút ’ ⋮ ’ → ’Hủy’</td><td>Cập nhật thông tin</td></tr>
<tr><td>3</td><td>Chọn nút<br>’ ⋮ ’ → ’Thanh toán’</td><td>Cập nhật thông tin</td></tr>
<tr><td>4</td><td>Chọn nút ’ ⋮ ’ → ’Xóa’</td><td>Hiện màn danh sách</td></tr>
<tr><td rowspan="4">Luồng ngoại lệ</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="3">1</td><td>Chọn nút ’+ Thêm’ để thêm sản phẩm vào đơn</td><td>Hiện màn tìm kiếm sản phẩm</td></tr>
<tr><td rowspan="2">Sau khi chọn sản phẩm</td><td>Thông báo đã tồn tại sản phẩm trong đơn</td></tr>
<tr><td>Thông báo hết hàng sản phẩm trong kho</td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Thêm, thanh toán, hủy, xóa hóa đơn thành công</td></tr>
</table>

**Bảng 2.12:** Đặc tả use case Quản lý hóa đơn




**Hình 2.21:** Biểu đồ tuần tự use case Quản lý hóa đơn (phần 1)



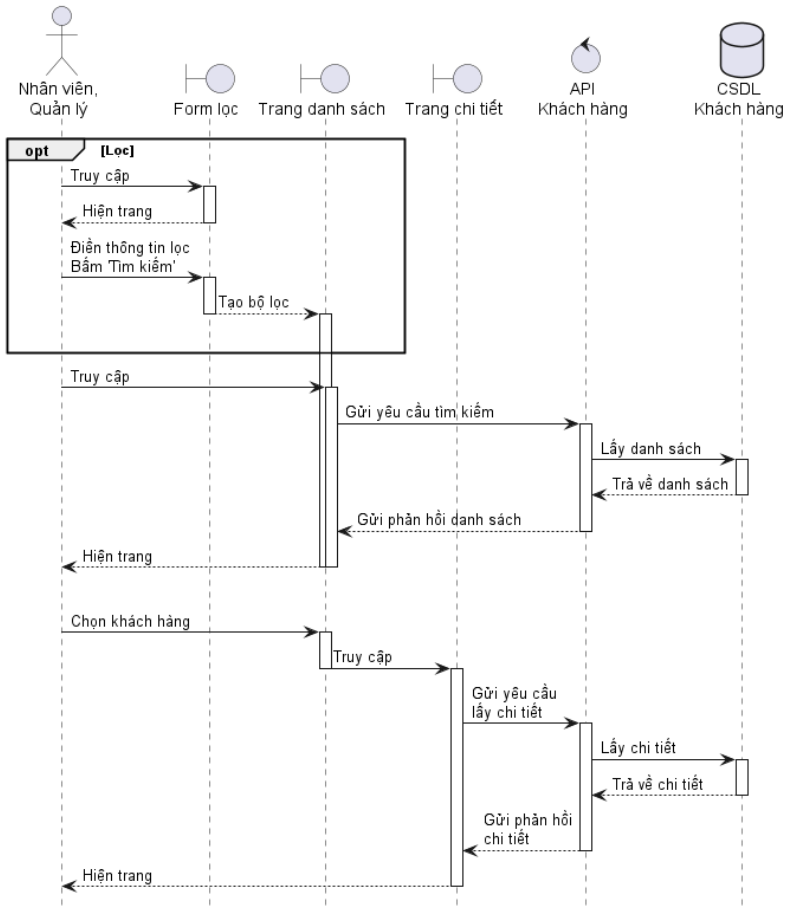

**Hình 2.22:** Biểu đồ tuần tự use case Quản lý hóa đơn (phần 2)

### 2.3.13 Đặc tả use case Lọc và xem khách hàng

Bảng 2.13 mô tả chi tiết use case Lọc và xem khách hàng. Hình 2.23 mô tả biểu đồ tuần tự của use case này.

<table>
<tr><td>Mã Use case</td><td colspan="3">UC60</td></tr>
<tr><td>Tên Use case</td><td colspan="3">Lọc và xem khách hàng</td></tr>
<tr><td>Tác nhân</td><td colspan="3">Nhân viên, Quản lý</td></tr>
<tr><td>Mô tả</td><td colspan="3">Người dùng muốn lọc bằng tên, số điện thoại, sắp xếp<br>và xem chi tiết thông tin khách hàng</td></tr>
<tr><td>Tiền điều kiện</td><td colspan="3">Người dùng là Nhân viên hoặc Quản lý</td></tr>
<tr><td rowspan="4">Luồng sự kiện</td><td>STT</td><td>Hành động</td><td>Hệ thống phản hồi</td></tr>
<tr><td rowspan="3">1</td><td>Chọn nút hình kính lúp</td><td>Hiện form lọc</td></tr>
<tr><td>Điền dữ liệu<br>Chọn nút ’Tìm kiếm’</td><td>Hiện màn danh sách</td></tr>
<tr><td>Chọn ô thông tin khách hàng</td><td>Hiện màn thông tin</td></tr>
<tr><td>Hậu điều kiện</td><td colspan="3">Lọc và Hiện chi tiết thông tin</td></tr>
</table>

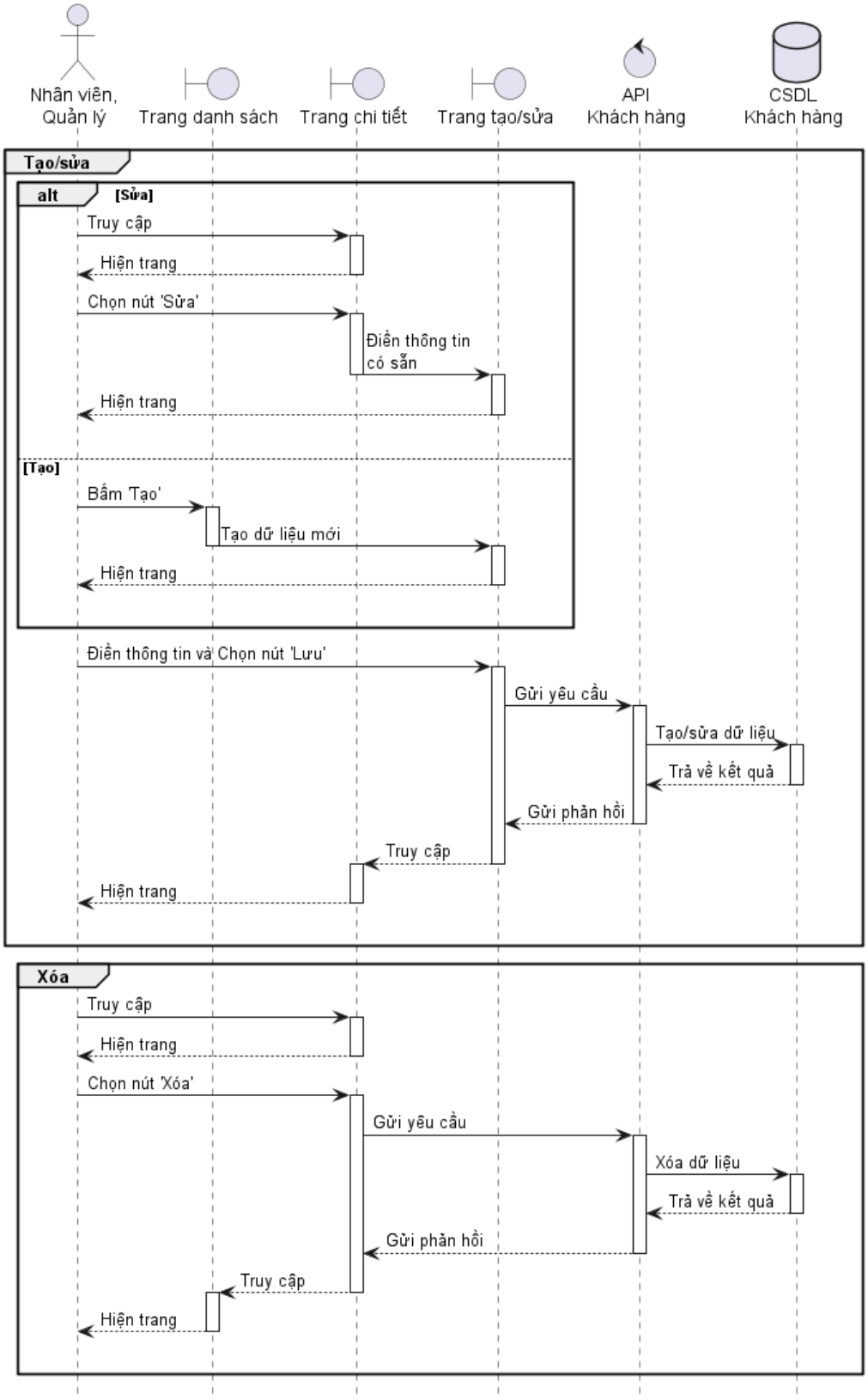
**Bảng 2.13:** Đặc tả use case Lọc và xem khách hàng




**Hình 2.23:** Biểu đồ tuần tự use case Lọc và xem khách hàng

### 2.3.14 Đặc tả use case Quản lý khách hàng

Bảng 2.14 mô tả chi tiết use case Quản lý khách hàng. Hình 2.24 mô tả biểu đồ tuần tự của use case này.




**Hình 2.24:** Biểu đồ tuần tự use case Quản lý khách hàng

| **Mã Use case** | UC61 | | |
|---|---|---|---|
| **Tên Use case** | Quản lý khách hàng | | |
| **Tác nhân** | Nhân viên, Quản lý | | |
| **Mô tả** | Người dùng muốn thêm, sửa, xóa khách hàng | | |
| **Tiền điều kiện** | Người dùng là Nhân viên hoặc Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn danh sách |
| | **2** | Chọn khách hàng | Hiện màn thông tin |
| | | Chọn nút ’ ⋮ ’ → ’Sửa’ | Hiện form sửa |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn thông tin |
| | **3** | Chọn nút ’ ⋮ ’ → ’Xóa’ | Hiện màn danh sách |
| **Hậu điều kiện** | Thêm, sửa, xóa thông tin thành công | | |

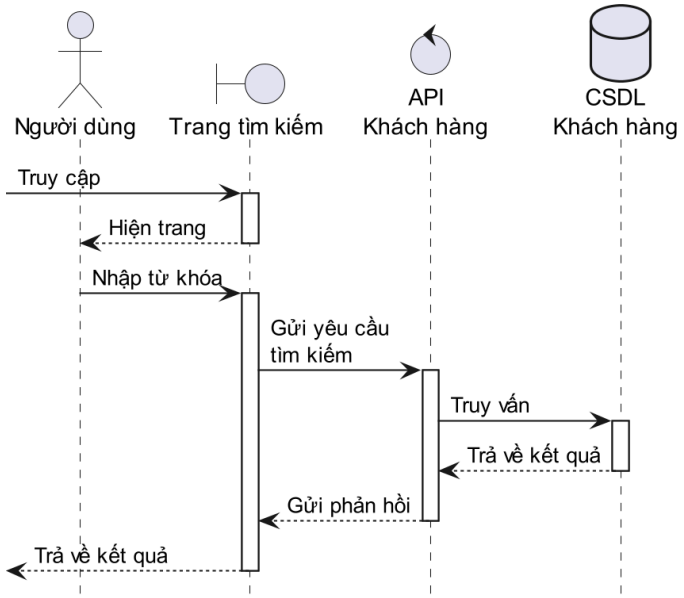
**Bảng 2.14:** Đặc tả use case Quản lý khách hàng

### 2.3.15 Đặc tả use case Tìm kiếm khách hàng

Bảng 2.15 mô tả chi tiết use case Tìm kiếm khách hàng. Hình 2.25 mô tả biểu đồ tuần tự của use case này.

| **Mã Use case** | UC62 | | |
|---|---|---|---|
| **Tên Use case** | Tìm kiếm khách hàng | | |
| **Tác nhân** | Nhân viên, Quản lý | | |
| **Mô tả** | Người dùng muốn tìm kiếm khách hàng<br>để thêm vào đơn từ | | |
| **Tiền điều kiện** | Người dùng là Nhân viên hoặc Quản lý | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Điền tên hoặc số điện thoại<br>khách hàng<br>Chọn ô thông tin sản phẩm | Hiện danh sách chuỗi<br>khách hàng<br>Quay về màn đơn từ |
| **Hậu điều kiện** | Tìm kiếm và thêm khách hàng vào đơn từ | | |

**Bảng 2.15:** Đặc tả use case Tìm kiếm khách hàng



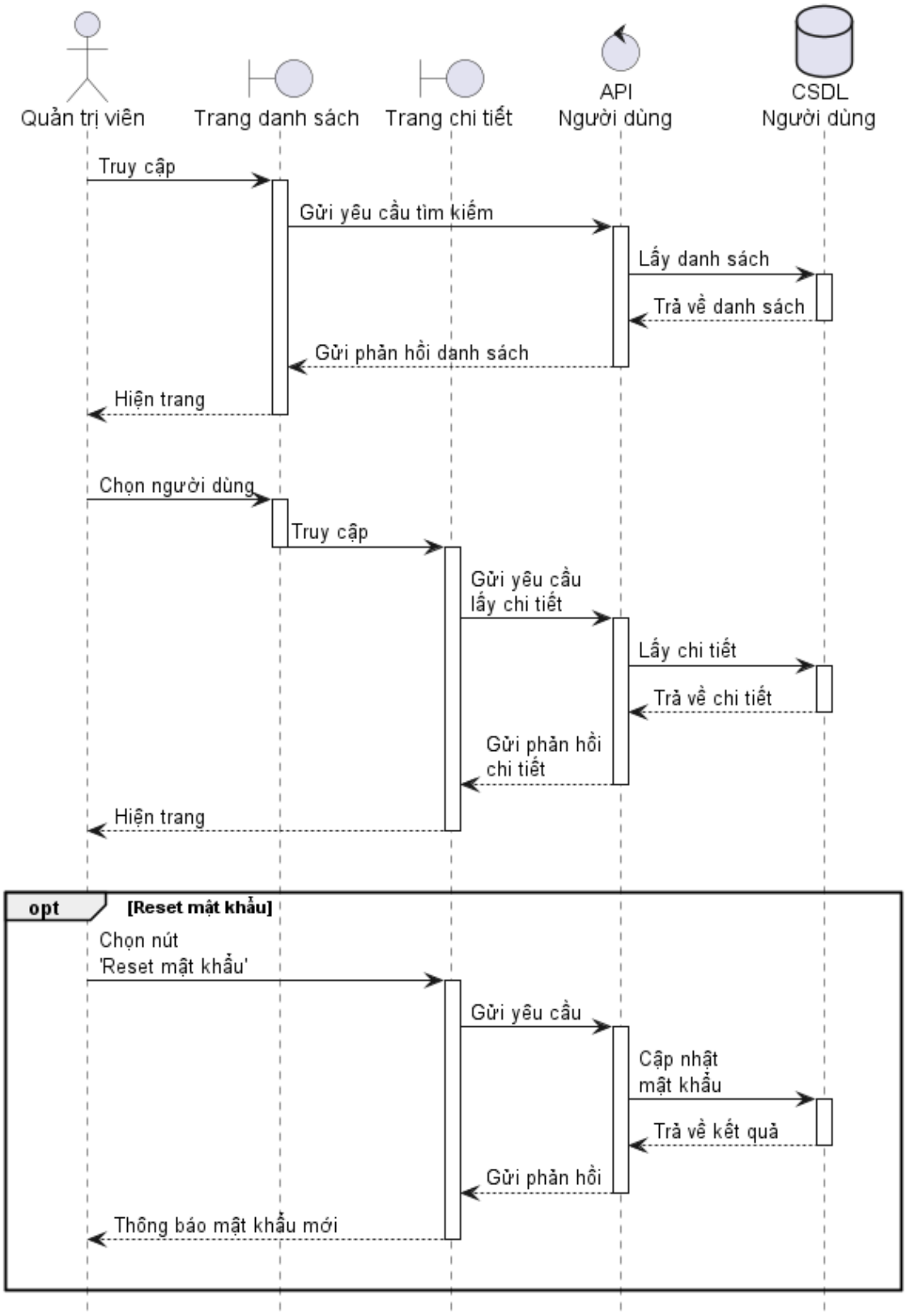

**Hình 2.25:** Biểu đồ tuần tự use case Tìm kiếm khách hàng

### 2.3.16 Đặc tả use case Quản lý người dùng

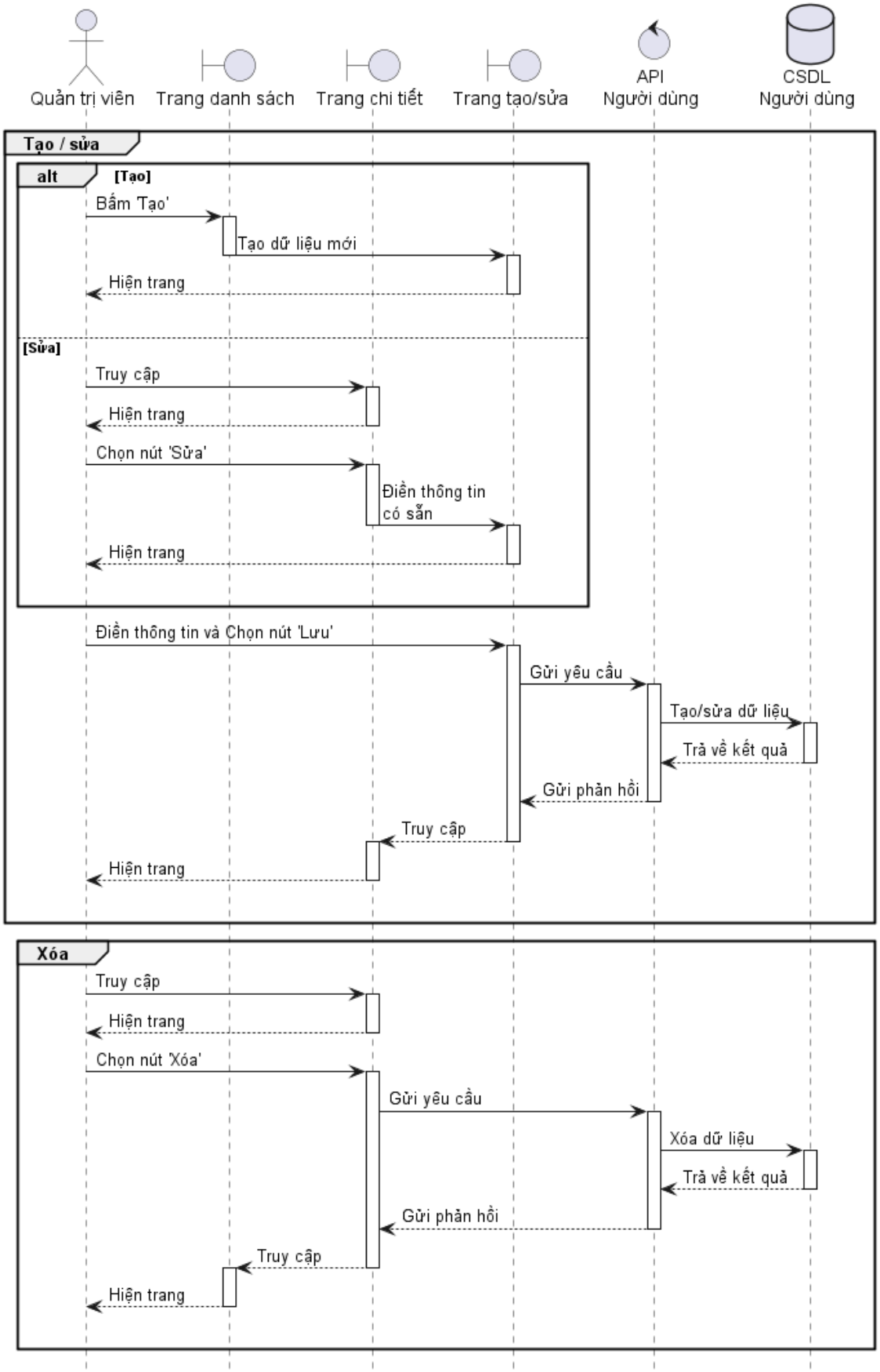
Bảng 2.16 mô tả chi tiết use case Quản lý người dùng. Hình 2.26 và 2.27 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC70 | | |
| **Tên Use case** | Quản lý người dùng | | |
| **Tác nhân** | Quản trị viên | | |
| **Mô tả** | Người dùng muốn xem chi tiết, thêm, sửa,<br>thay đổi mật khẩu, xóa người dùng | | |
| **Tiền điều kiện** | Người dùng là Quản trị viên | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn danh sách |
| | **2** | Chọn ô người dùng | Hiện màn thông tin |
| | | Chọn nút ’Reset mật khẩu’ | Thông báo mật khẩu mới |
| | **3** | Chọn nút ’ ⋮ ’ $\rightarrow$ ’Sửa’ | Hiện form sửa |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn thông tin |
| | **4** | Chọn nút ’ ⋮ ’ $\rightarrow$ ’Xóa’ | Hiện màn danh sách |
| **Luồng ngoại lệ** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn nút tạo mới ’ + ’ | Hiện form tạo |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Thông báo tên đăng<br>nhập đã tồn tại |
| **Hậu điều kiện** | Thêm, sửa, thay đổi mật khẩu, xóa người dùng thành công | | |

**Bảng 2.16:** Đặc tả use case Quản lý người dùng




**Hình 2.26:** Biểu đồ tuần tự use case Quản lý người dùng (phần 1)



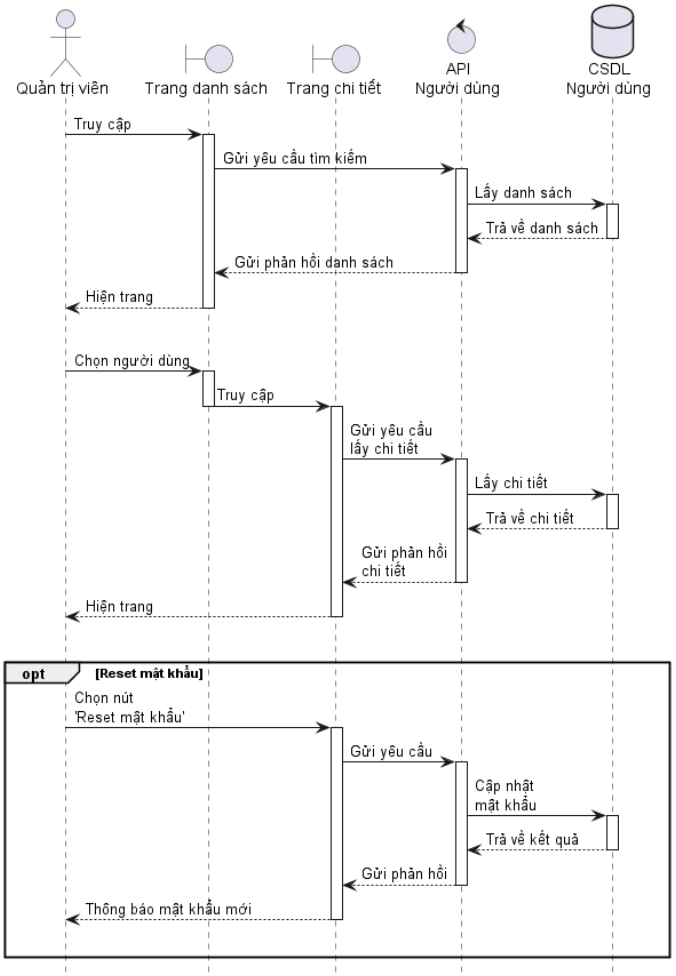

**Hình 2.27:** Biểu đồ tuần tự use case Quản lý người dùng (phần 2)

### 2.3.17 Đặc tả use case Quản lý thông tin cá nhân

Bảng 2.17 mô tả chi tiết use case Quản lý thông tin cá nhân. Hình 2.28 mô tả biểu đồ tuần tự của use case này.

| | | | |
|---|---|---|---|
| **Mã Use case** | UC71 | | |
| **Tên Use case** | Quản lý thông tin cá nhân | | |
| **Tác nhân** | Quản trị viên, Quản lý, Nhân viên | | |
| **Mô tả** | Người dùng muốn xem chi tiết, sửa thông tin người dùng cá nhân | | |
| **Tiền điều kiện** | Đã đăng nhập hệ thống | | |
| **Luồng sự kiện** | **STT** | **Hành động** | **Hệ thống phản hồi** |
| | **1** | Chọn thanh thông tin cá nhân | Hiện màn thông tin |
| | **2** | Chọn nút ’ ⋮ ’ → ’Sửa’ | Hiện form sửa |
| | | Điền thông tin<br>Chọn nút ’Lưu’ | Hiện màn thông tin |
| **Hậu điều kiện** | Xem chi tiết, sửa thông tin thành công | | |

**Bảng 2.17:** Đặc tả use case Quản lý thông tin cá nhân




**Hình 2.28:** Biểu đồ tuần tự use case Quản lý thông tin cá nhân

## 2.4 Yêu cầu phi chức năng

Ngoài các bộ usecase đã được trình bày ở trên, hệ thống còn có các yêu cầu phi chức năng như sau để đạt được mục tiêu của dự án:

- Hệ thống backend phải có thể chạy trên các hệ điều hành Windows, Linux.
- Ứng dụng người dùng phải chạy được trên các thiết bị di động Android, iOS.
- Hệ thống phải có khả năng chịu tải tối thiểu 100 request trên 1 giây.
- Lưu lượng sử dụng mạng nên được tối giản và tối ưu nhất đề phòng sự cố nghẽn mạng.
- Khả năng bảo trì và hồi phục hệ thống trong vòng 1 giờ.
- Dự án có một bộ codebase dễ dàng mở rộng và bảo trì.
- Dễ cài đặt và có hướng dẫn tường minh cho người quản trị.
- Tính bảo mật cao, hệ thống phải có khả năng chống lại các cuộc tấn công từ bên ngoài.

# CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

## 3.1 Ngôn ngữ C# và framework ASP.NET Core

C# là một ngôn ngữ lập trình hướng đối tượng, đa năng, hiện đại và mạnh mẽ, được phát triển bởi Microsoft vào năm 2000 và là một phần của nền tảng .NET. C# có thể được sử dụng để phát triển các ứng dụng web, di động và game trên nhiều nền tảng khác nhau nhờ vào các công cụ như MAUI, Unity và ASP.NET.

Một số tính năng của C# giúp tạo ra các ứng dụng mạnh mẽ và bền bỉ hơn các ngôn ngữ hướng đối tượng Java. Như là, Garbage Collector là một công cụ tự động để quản lý bộ nhớ và thu gom các đối tượng không còn sử dụng trong ứng dụng. Language Integrated Query (LINQ) là một công cụ cho phép người lập trình truy vấn dữ liệu từ các nguồn khác nhau, như SQL Server, Oracle, XML, v.v. C# cũng hỗ trợ các tính năng như generics, delegates, lambda expressions, extension methods, anonymous types, dynamic.

ASP.NET Core là một framework mã nguồn mở cho việc xây dựng các ứng dụng web động, API web và các dịch vụ đám mây. ASP.NET Core được xây dựng trên nền tảng .NET Core, và có thể chạy trên nhiều nền tảng khác nhau. ASP.NET Core cung cấp nhiều tính năng như dependency injection, middleware, routing, authentication, authorization, logging, testing và nhiều hơn nữa.

Thiết kế API server là một trong những lĩnh vực phổ biến của C# và ASP.NET Core. API server là một ứng dụng web chịu trách nhiệm cho việc nhận và xử lý các yêu cầu từ các ứng dụng khác thông qua giao thức HTTP. API server có thể trả về các dữ liệu ở các định dạng khác nhau, như JSON, XML, HTML hoặc bất kỳ định dạng nào khác do người lập trình định nghĩa. API server có thể được sử dụng để cung cấp các chức năng như xác thực người dùng, quản lý phiên bản, phân quyền, bảo mật, caching, logging.

## 3.2 Hệ cơ sở dữ liệu MongoDB

MongoDB là một chương trình cơ sở dữ liệu NoSQL mã nguồn mở được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp và đa dạng và không cố định (hay còn gọi là Big Data). MongoDB sẽ lưu trữ dữ liệu dưới dạng Document JSON. Vì vậy, mỗi một collection sẽ các các kích cỡ và các document khác nhau. Bên cạnh đó, việc các dữ liệu được lưu trữ trong document kiểu JSON dẫn đến chúng được truy vấn rất nhanh. MongoDB cũng hỗ trợ các tính năng như:

- Aggregation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất.
- Replication bao gồm hai hoặc nhiều bản sao của dữ liệu. Trong đó mỗi bản sao có thể đóng vai trò chính và phụ. Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi cơ sở dữ liệu toàn vẹn, không bị mất mát trước những sự cố ngoài ý muốn.
- Indexing có thể được tạo để cải thiện hiệu suất của các tìm kiếm trong MongoDB. Bất kỳ field nào trong MongoDB document đều có thể được index.
- Load balancing bằng cách sử dụng Sharding. Nó chạy trên nhiều máy chủ, cân bằng tải hoặc sao chép dữ liệu để giữ hệ thống luôn hoạt động trong trường hợp có lỗi về phần cứng.

## 3.3 Docker

Docker là một công cụ giúp cho việc tạo ra và triển khai các container để phát triển, chạy ứng dụng được dễ dàng. Các container là môi trường, mà ở đó lập trình viên đưa vào các thành phần cần thiết để ứng dụng của họ chạy được, bằng cách đóng gói ứng dụng cùng với container như vậy, nó đảm bảo ứng dụng chạy được và giống nhau ở các máy khác nhau (Linux, Windows, Desktop, Server ...)

Docker có vẻ rất giống máy ảo (nhiều người từng tạo máy ảo với công cụ ảo hóa như Virtual Box, VMWare), nhưng có điểm khác với VM: thay vì tạo ra toàn bộ hệ thống (dù ảo hóa), Docker lại cho phép ứng dụng sử dụng nhân của hệ điều hành đang chạy Docker để chạy ứng dụng bằng cách bổ sung thêm các thành phần còn thiếu cung cấp bởi container. Cách này làm tăng hiệu xuất và giảm kích thước ứng dụng.

## 3.4 .NET MAUI

.NET Multi-platform App UI (.NET MAUI) là một framework đa nền tảng để tạo các ứng dụng native dành cho thiết bị di động và máy tính với C# và XAML. .NET MAUI là một phần của .NET 6 và là phiên bản tiếp theo của Xamarin.Forms. Nó cung cấp một cách để tạo ứng dụng di động và máy tính đa nền tảng với một mã nguồn duy nhất. .NET MAUI cũng hỗ trợ các tính năng như:

- Có thể chạy trên nhiều nền tảng khác nhau như Android, iOS, macOS, Windows, Linux.
- Hỗ trợ liên kết dữ liệu (data binding) theo kiến trúc thiết kế phần mềm MVVM (model, view, model-view).
- Cung cấp các API hợp nhất từ các nền tảng nằm sau để truy cập các tính năng native như GPS, clipboard, pin, trạng thái mạng, ...
- Một dự án code duy nhất có thể chia sẻ giữa các nền tảng khác nhau và có thể sử dụng thêm các tính năng đặc biệt của từng nền tảng nếu cần.
- Hot reload cho phép người lập trình thay đổi code và xem kết quả ngay lập tức trên thiết bị đang chạy ứng dụng mà không cần phải build lại.

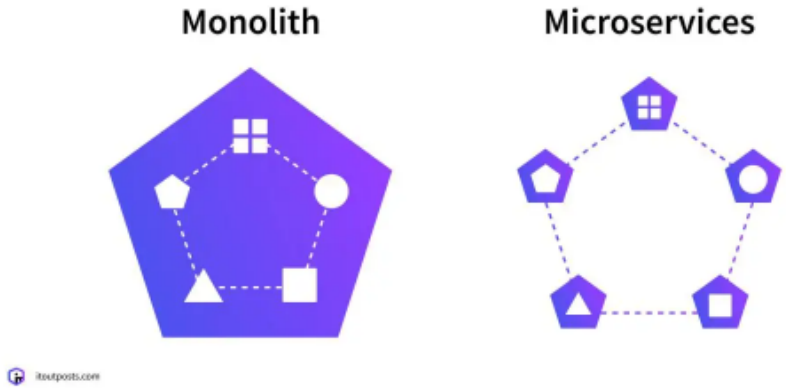
# CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

## 4.1 Thiết kế kiến trúc

### 4.1.1 Lựa chọn kiến trúc phần mềm

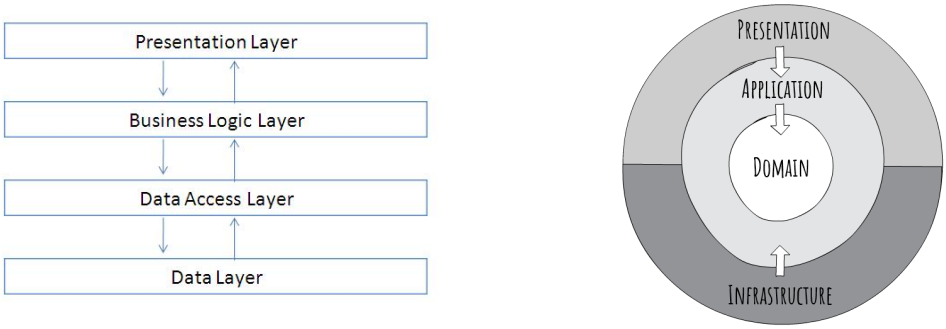
Trong thiết kế hệ thống thông tin, ta thường biết đến rất nhiều cách tiếp cận, thiết kế và cấu trúc dự án khác nhau. Sau khi xem xét kĩ lưỡng, mô hình Client-Server sẽ làm hệ thống có tính mở rộng cao và dễ dàng truy cập. Khi đó dữ liệu của hệ thống sẽ là một server riêng biệt, còn phần giao diện và thao tác dữ liệu sẽ là một hoặc nhiều kiểu client riêng biệt. Với mục tiêu đó, một bộ API sẽ được xây dựng cùng với cơ sở dữ liệu được đóng gói trong một server có thể được host trên các dịch vụ cloud hoặc thậm chí home server. Về phía Client hiện giờ sẽ được phát triển là một ứng dụng di động để người dùng dễ dàng thao tác nhất.

Khi thiết kế **Server**, kiểu hệ thống Monolithic và Microservice là hai cách tiếp cận phổ biến nhất. Tuy nhiên, với đối tượng người dùng là các hộ kinh doanh vừa và nhỏ, việc sử dụng Microservice sẽ quá phức tạp và thêm nhiều phần xử lí thừa. Do đó, kiểu Monolithic sẽ tối giản quá trình phát triển và đẩy nhanh tiến độ rất nhiều. Thiết kế sẽ bao gồm các đầu API cùng với logic xử lí và truy cập lưu trữ cơ sở dữ liệu vào trong một process. Hình 4.1 mô tả sự khác biệt giữa hai kiểu kiến trúc này.




**Hình 4.1:** Kiến trúc Monolithic và Microservice [1]

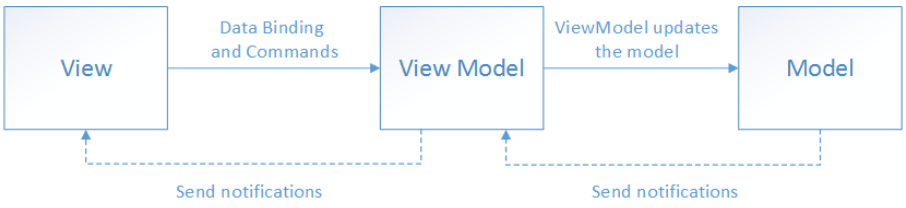
Đi sâu hơn, khi tiếp cận việc thiết kế một bộ mã nguồn tường minh và dễ phát triển, kiến trúc N-tier (hình 4.2a), Clean Architecture (hình 4.2b) được đề xuất rất nhiều. Nhưng trong trường hợp thay đổi hệ cơ sở dữ liệu thì kiến trúc N-tier sẽ khiến ta phải cấu trúc lại rất nhiều vì tầng truy cập dữ liệu (Data Access Layer) nằm rất sâu trong hệ thống. Vì vậy, kiến trúc Clean Architecture sẽ được sử dụng trong hệ thống này vì nó trừu tượng hóa việc truy cập dữ liệu (Application Layer). Bất kể kiểu cơ sở dữ liệu nào, chúng sẽ đều phải kế thừa lớp trừu tượng đó, do đó giải quyết được vấn đề thay đổi công nghệ trong tương lai. Lựa chọn vì sao và thiết kế kiến trúc Clean Architecture sẽ được trình bày chi tiết ở phần 5.1.




**(a)** Kiến trúc N-tier [2] **(b)** Kiến trúc Clean Architecture [3]

**Hình 4.2:** N-tier và Clean Architecture

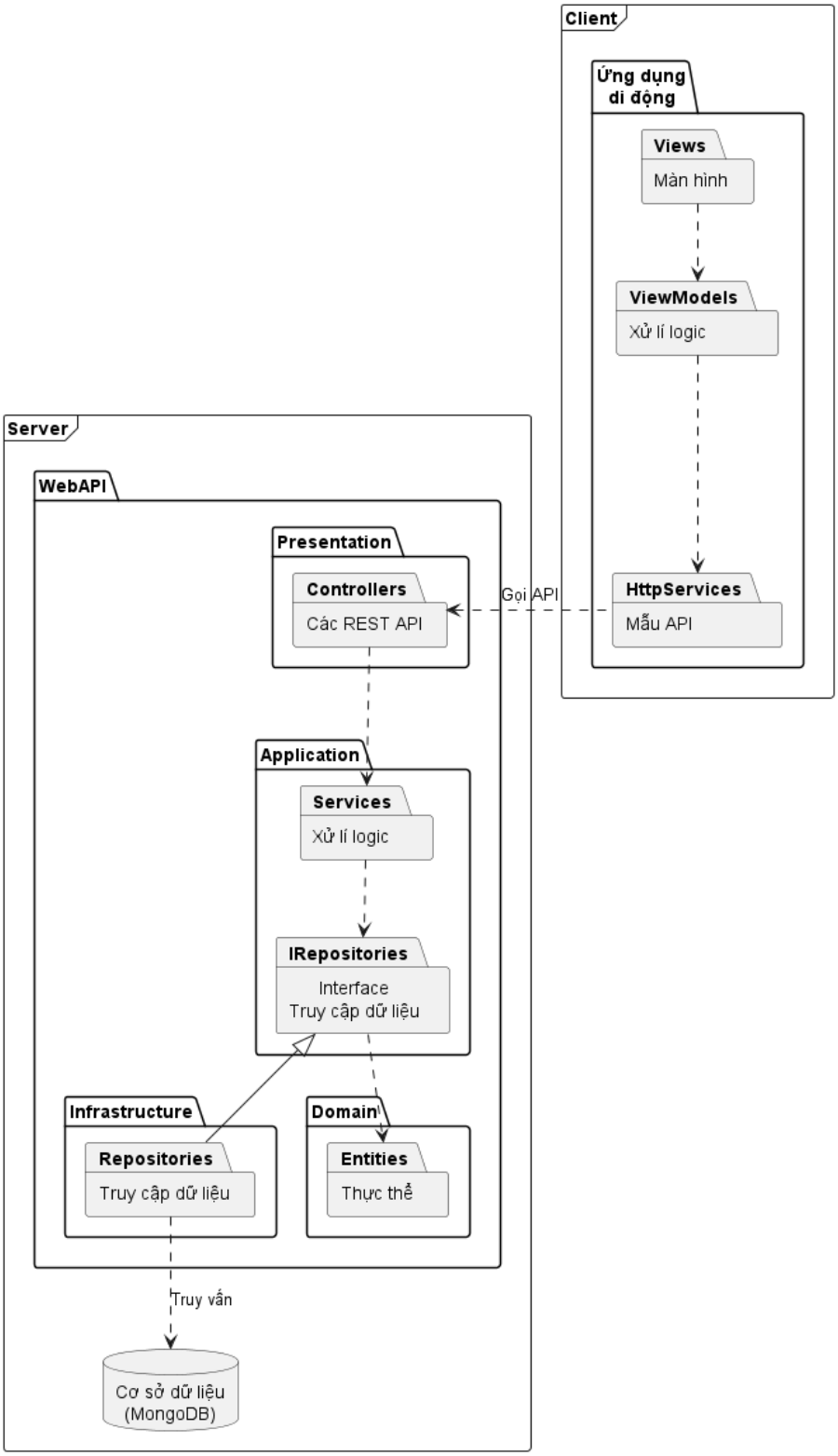
Khi thiết kế **Client**, phổ biến nhất là các kiến trúc MVC, MVP... Nhưng kiến trúc MVVM (Model-View-ViewModel) (4.3) chứng minh được quy trình truyền tải dữ liệu giữa Model và View hiệu quả hơn nhiều. Kiến trúc này giúp phân tách rõ ràng giữa giao diện (View) và dữ liệu (Model) với một lớp trung gian (ViewModel) để xử lí logic và lan truyền hai chiều sự thay đổi trong dữ liệu. Điều này giúp cho việc phát triển ứng dụng di động trở nên dễ dàng hơn để kiểm thử, duy trì và mở rộng. Nó cũng giúp cải thiện việc sử dụng lại code và người thiết kế giao diện có thể làm việc độc lập với nhà phát triển logic. Chi tiết về lựa chọn và thiết kế kiến trúc MVVM sẽ được trình bày ở phần 5.2.




**Hình 4.3:** Kiến trúc MVVM [4]

### 4.1.2 Thiết kế tổng quan

Hình 4.4 là biểu đồ gói tổng quan của hệ thống. Hệ thống được chia thành hai phần chính là **Server** và **Client**. Trong đó, **Server** sẽ chứa các thành phần liên quan đến logic xử lí và truy cập lưu trữ cơ sở dữ liệu. **Client** sẽ chứa các thành phần liên quan đến giao diện người dùng để giao tiếp với hệ thống.




**Hình 4.4:** Biểu đồ gói

Trong phần **Client**, hệ thống áp dụng mô hình MVVM và được chia thành các gói như sau:

- **Views**: Gói này sẽ chứa các giao diện người dùng. Trong hệ thống này, nó sẽ chứa các màn hình như danh sách sản phẩm (ProductListPage), chi tiết sản phẩm (ProductDetailsPage), danh sách hóa đơn (InvoiceListPage),...
- **ViewModels**: Gói này sẽ chứa các thành phần liên quan đến logic xử lí và truyền tải dữ liệu giữa View và Model. Trong hệ thống này, ứng với mỗi trang sẽ có một ViewModel tương ứng. Ví dụ, trang danh sách sản phẩm sẽ có ProductListViewModel, trang chi tiết sản phẩm sẽ có ProductDetailsViewModel,...
- **HttpServices**: Gói này là tượng trưng cho lớp Model của mô hình MVVM. Chúng chứa các đối tượng gọi API để giao tiếp với hệ thống qua môi trường mạng. Trong hệ thống này, nó sẽ chứa các đối tượng hỗ trợ đăng nhập (IdentityService), quản lý sản phẩm (ProductService), quản lý hóa đơn (InvoiceService)...

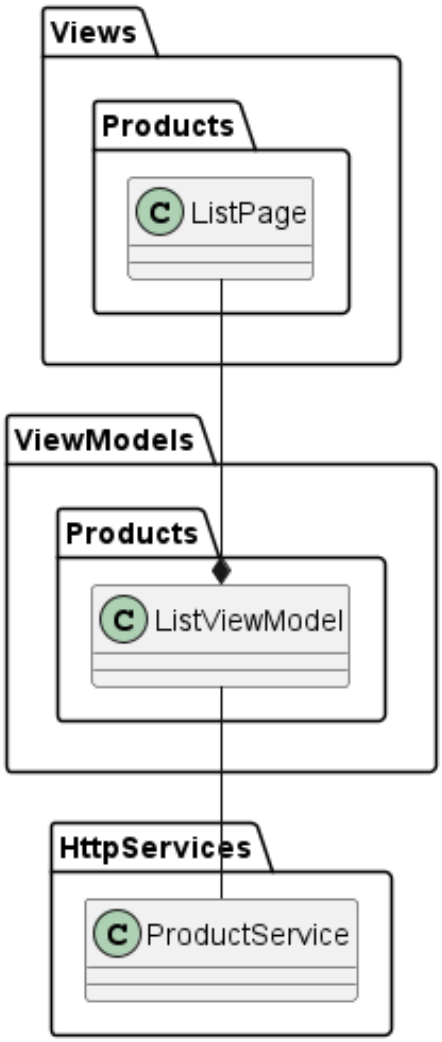
Áp dụng kiến trúc **Clean Architecture** vào hệ thống, **Server** sẽ được chia thành các gói như sau:

- **Presentation**: Gói này gồm các thành phần giao tiếp với hệ thống. Cụ thể, các gói **Controller** trong hệ thống này là các đầu API như đăng nhập (IdentityController), quản lý sản phẩm (ProductController), quản lý hóa đơn (InvoiceController)...
- **Application**: Gói này gồm các gói con như **Services, IRepositories**. Các gói **Services** mang nhiệm vụ lấy và lưu trữ dữ liệu từ gói **IRepositories** rồi xử lí các logic nghiệp vụ. Còn các gói **IRepositories** trừu tượng hóa các câu truy vấn cơ sở dữ liệu thành các hàm riêng. Mục đích này để cho dù dùng bất kì kiểu cơ sở dữ liệu nào (SQL Server, MongoDB) thì quy trình xử lí logic của hệ thống không bị ảnh hưởng.
- **Infrastructure**: Gói **Repositories** này sẽ kế thừa các gói **IRepositories** của gói **Application** và cài đặt các câu truy vấn với một hệ cơ sở dữ liệu nhất định. Trong hệ thống này sẽ dùng MongoDB nên gói này sẽ chứa các đối tượng kết nối và truy vấn cơ sở dữ liệu (MongoDatabase, ProductRepository, InvoiceRepository...).
- **Domain**: Gói **Entities** sẽ chứa các kiểu dữ liệu ứng với các bảng, các tài liệu được lưu trữ trong cơ sở dữ liệu. Chúng được định nghĩa bằng các lớp (User, Product, Invoice,...) và được sử dụng trong các gói **Repositories** và **Services**.

### 4.1.3 Thiết kế chi tiết gói

#### a, Gói Client

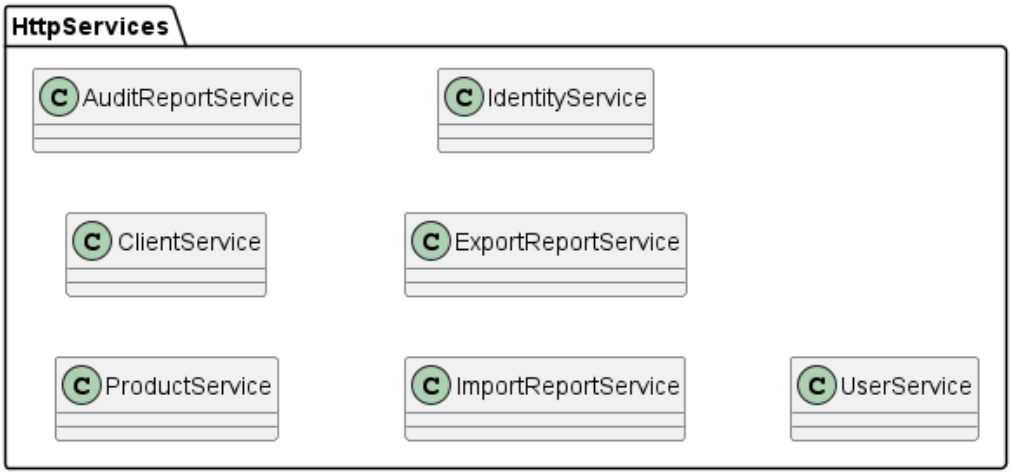
Hầu hết các gói trong **Client** đều tuân thủ theo mô hình MVVM, do đó các gói Views, ViewModels, HttpServices sẽ được thiết kế tương tự nhau. Để tránh lặp lại các biểu đồ, sau đây chỉ View (ProductListPage) sẽ làm mẫu cho các gói khác và được trình bày chi tiết trong hình 4.5.




**Hình 4.5:** Biểu đồ gói ListPage của Product

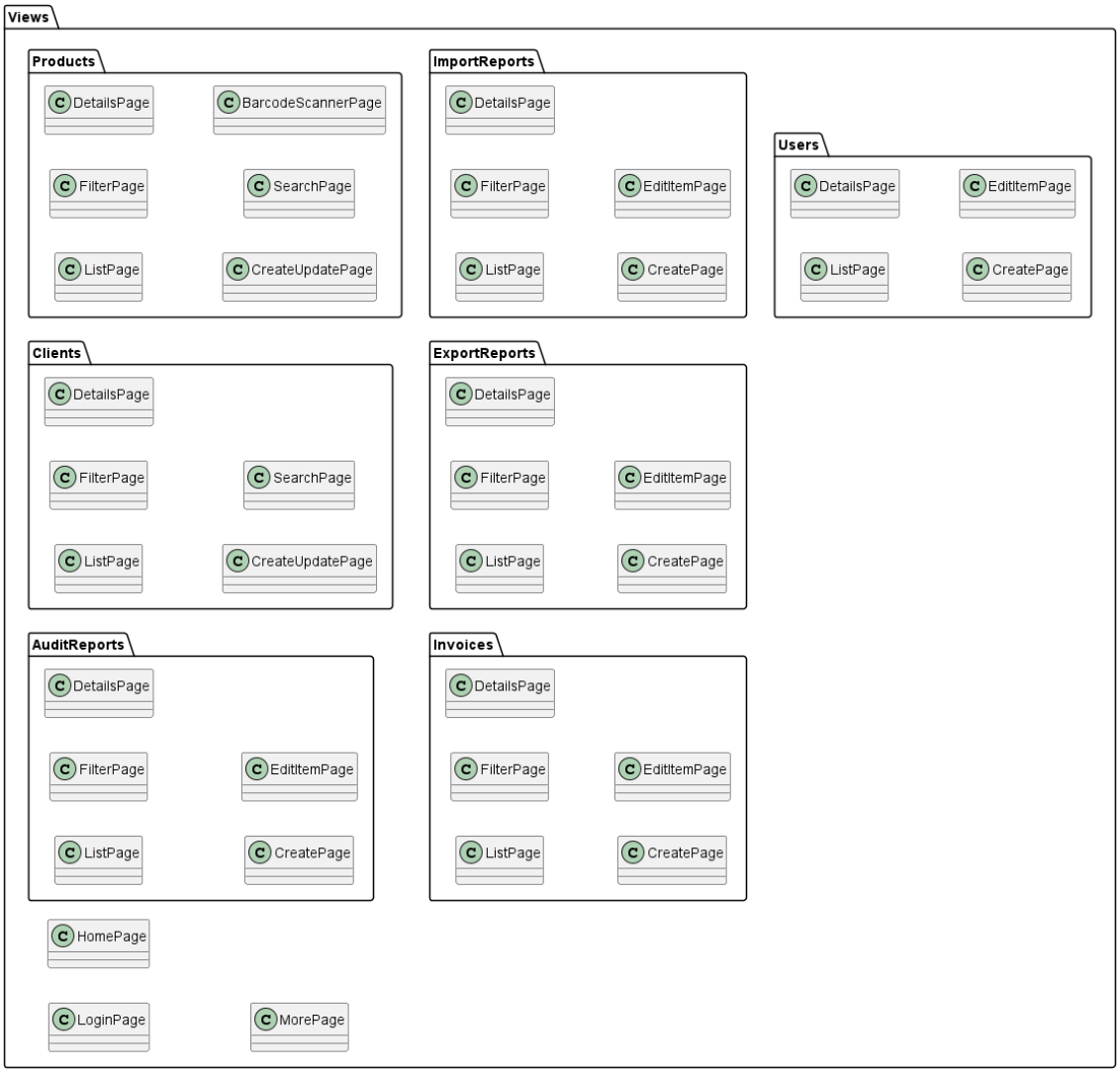
**View** chứa một lớp con **ViewModel** để xử lí logic và truyền tải dữ liệu giữa View và Model. Lớp **ViewModel** sẽ chứa các dữ liệu mà **View** của nó hiển thị, và khi người dùng thực hiện thao tác tới hệ thống, thì **ViewModel** sẽ dùng **HttpService** tương ứng để gọi API và cập nhật dữ liệu. Khi đó, **View** sẽ được cập nhật lại dữ liệu và hiển thị lên giao diện.

Hình 4.6 mô tả các lớp hỗ trợ gọi API trong gói HttpServices.



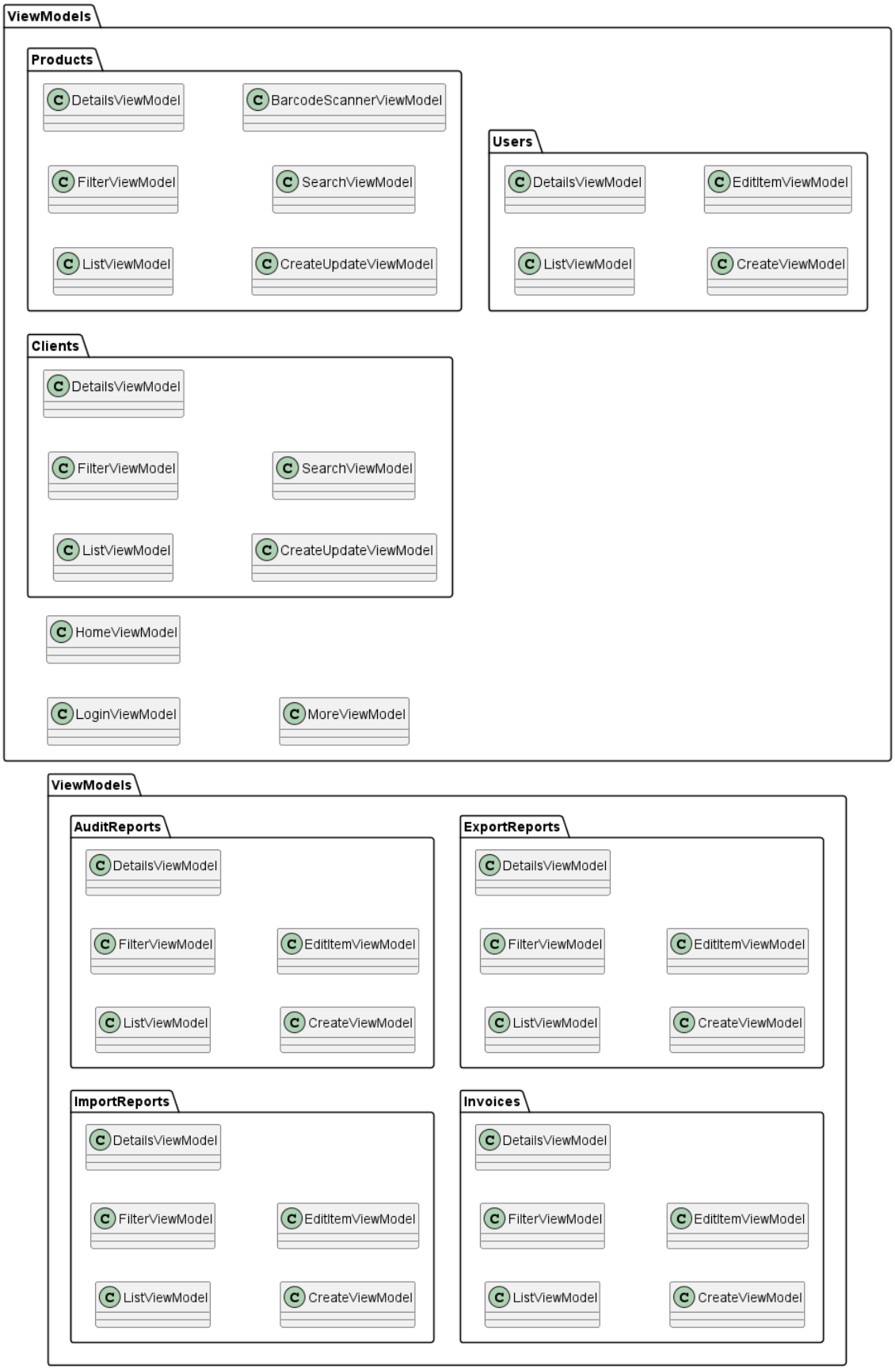

**Hình 4.6:** Các biểu đồ gói HttpServices

Hình 4.7 mô tả các lớp giao diện người dùng trong gói Views.




**Hình 4.7:** Các biểu đồ gói Views

Hình 4.8 mô tả các lớp ViewModels.

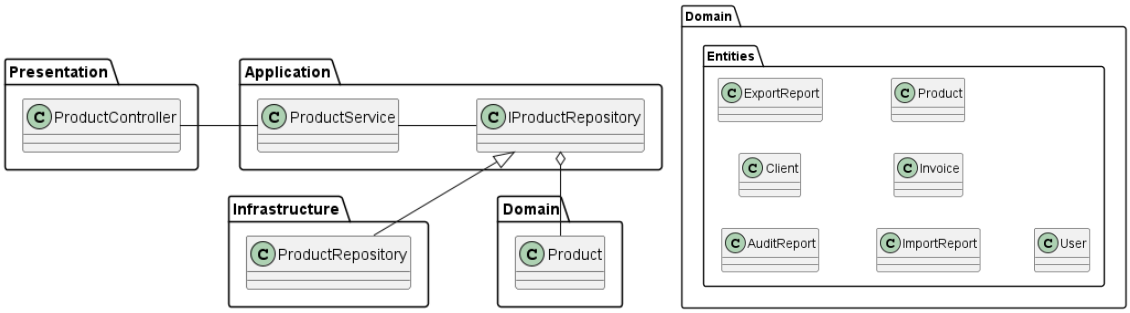



**Hình 4.8:** Các biểu đồ gói ViewModels

### b, Gói Server

Áp dụng kiến trúc Clean Architecture, hầu hết tương ứng với mỗi thực thể thì sẽ luôn có một lớp kế thừa interface **IRepository** được định nghĩa trong tầng **Infrastructure**. Các lớp **Services** sẽ dùng các lớp đó để xử lí logic. Cuối cùng là lớp **Controller** sẽ định nghĩa các đầu API.

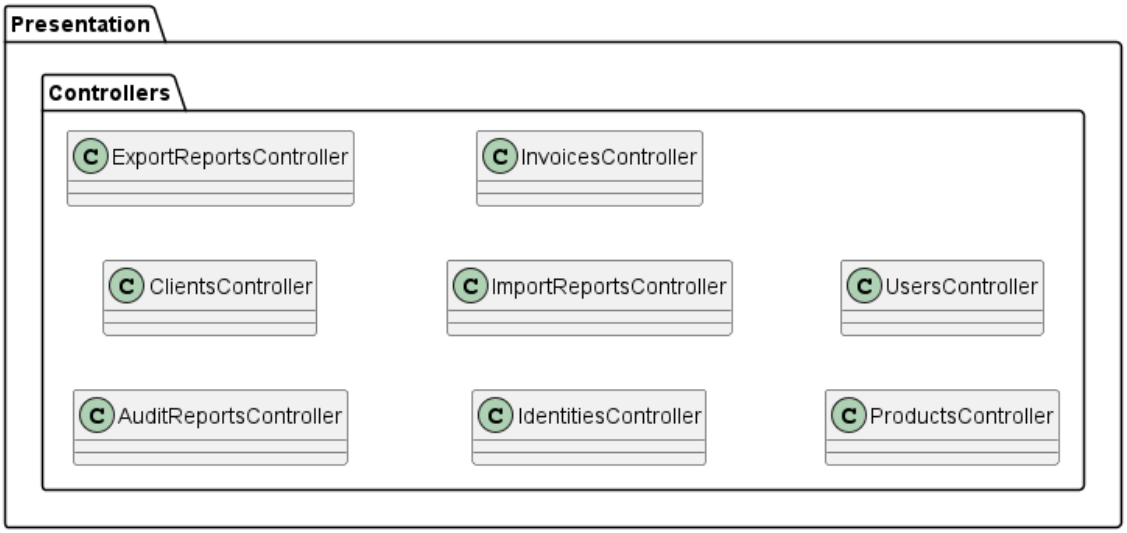
Ví dụ đối với thực thể *Product*, các lớp liên quan tương tác với nhau sẽ được trình bày chi tiết trong hình 4.9a. Hình 4.9b mô tả các thực thể của hệ thống.




**(a)** Biểu đồ các lớp liên quan tới sản phẩm **(b)** Biểu đồ các lớp gói Domain

**Hình 4.9:** Các biểu đồ gói Server

Hình 4.10 mô tả các lớp gói Presentation. Chúng chứa các đầu API để giao tiếp với thiết bị frontend.

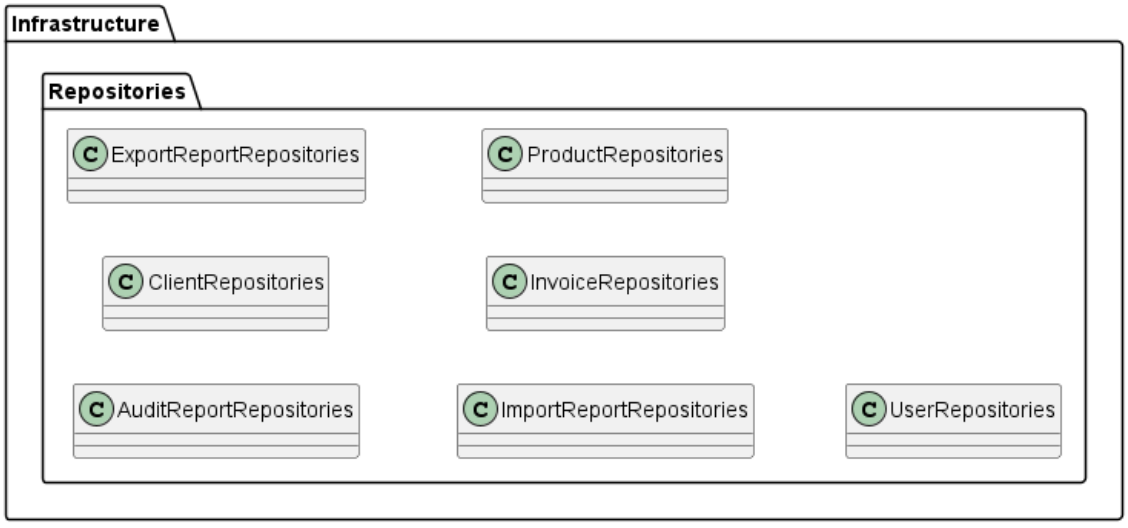



**Hình 4.10:** Biểu đồ gói Presentation

Hình 4.11 mô tả các lớp gói Application.


**Hình 4.11:** Biểu đồ gói Application

Hình 4.12 mô tả các lớp gói Infrastructure.




**Hình 4.12:** Biểu đồ gói Infrastructure

## 4.2 Thiết kế chi tiết

### 4.2.1 Thiết kế giao diện

Bảng 4.1 mô tả quy chuẩn thiết kế giao diện của hệ thống cho các màn hình và các thiết kế thành phần.

| | |
|---|---|
| Số lượng màu sắc hỗ trợ | 16 triệu màu |
| Font chữ sử dụng | Open Sans |
| Màu nền chính | #FFFFFF (trắng) |
| Màu nền phụ | #F2F0F4 (xám nhạt) |
| Màu chữ chính | #000000 (đen) |
| Màu chữ phụ | #696969 (xám) |
| Màu cảnh báo | #FF5e52 (đỏ) |
| Màu thành công | #35A854 (xanh lá) |
| Màu thông tin | #247ECE (xanh biển) |
| Ngôn ngữ hiển thị | Tiếng Việt |
| Hỗ trợ chế độ màu tối | Không |
| Vị trí thông báo lỗi | Giữa màn hình |
| Múi giờ mặc định | UTC+7 |

**Bảng 4.1:** Quy chuẩn thiết kế giao diện

### 4.2.2 Thiết kế lớp

Để minh họa thiết kế lớp mà vẫn ngắn gọn súc tích, use case Quản lý phiếu nhập kho (được mô tả trong bảng use case 2.6 và biểu đồ 2.15) sẽ được chọn làm một ví dụ mẫu trong quá trình xử lí dữ liệu giữa các lớp với nhau. Sau đây là một số bảng thiết kế thuộc tính lớp, biểu đồ trình tự luồng truyền thông điệp giữa các đối tượng tham gia trong use case.

Bảng 4.2 mô tả thiết kế lớp **ImportReportController** trong gói **Presentation**. Lớp này sẽ định nghĩa các đầu API để giao tiếp với thiết bị frontend.

| **ImportReportController** | |
|---|---|
| **Tên thuộc tính** | **Mô tả** |
| ImportReportService service | Đối tượng để gọi các hàm xử lí logic |
| **Tên hàm** | **Mô tả** |
| Get(string id) | Lấy thông tin phiếu nhập kho bằng id |
| Post(ImportReport entity) | Tạo phiếu nhập kho |
| Cancel(string id) | Hủy phiếu nhập kho |
| Delete(string id) | Xóa phiếu nhập kho |

**Bảng 4.2:** Bảng thiết kế lớp ImportReportController

Bảng 4.3 mô tả thiết kế lớp **ImportReportRepository** trong gói **Application**. Lớp này sẽ định nghĩa các hàm xử lí dữ liệu.

| **ImportReportService** | |
|---|---|
| **Tên thuộc tính** | **Mô tả** |
| ImportReportRepository service | Đối tượng để gọi các hàm truy vấn cơ sở dữ liệu |
| ProductRepository service | Đối tượng để gọi các hàm truy vấn cơ sở dữ liệu |
| **Tên hàm** | **Mô tả** |
| Get(string id) | Lấy thông tin phiếu nhập kho bằng id |
| Create(ImportReport entity) | Tạo phiếu nhập kho |
| Cancel(string id) | Hủy phiếu nhập kho |
| Delete(string id) | Xóa phiếu nhập kho |

**Bảng 4.3:** Bảng thiết kế lớp ImportReportService

Bảng 4.4 mô tả thiết kế lớp **ImportReportRepository** trong gói **Infrastructure**. Lớp này sẽ định nghĩa các hàm truy vấn cơ sở dữ liệu.

| **ImportReportRepository** | |
|---|---|
| **Tên thuộc tính** | **Mô tả** |
| MongoDatabase database | Đối tượng kết nối cơ sở dữ liệu |
| **Tên hàm** | **Mô tả** |
| Get(string id) | Lấy thông tin phiếu nhập kho bằng id |
| Create(ImportReport entity) | Tạo phiếu nhập kho |
| Update(string id, ...) | Cập nhật phiếu nhập kho |
| Delete(string id) | Xóa phiếu nhập kho |

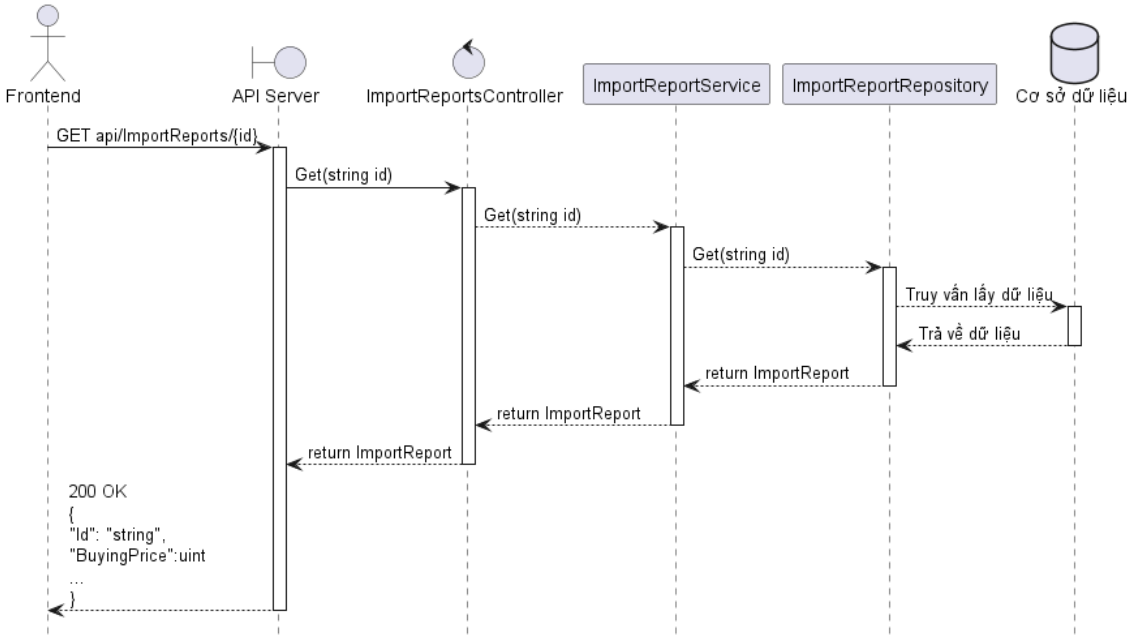
**Bảng 4.4:** Bảng thiết kế lớp ImportReportRepository

Bảng 4.5 mô tả thiết kế lớp **ProductRepository** trong gói **Infrastructure**. Lớp này sẽ định nghĩa các hàm truy vấn cơ sở dữ liệu.

| **ProductRepository** | |
|---|---|
| **Tên thuộc tính** | **Mô tả** |
| MongoDatabase database | Đối tượng kết nối cơ sở dữ liệu |
| **Tên hàm** | **Mô tả** |
| Get(string id) | Lấy thông tin sản phẩm bằng id |
| GetAll(string[] ids) | Lấy thông tin nhiều sản phẩm bằng các id |
| Create(Product entity) | Tạo sản phẩm |
| Update(string id, ...) | Cập nhật sản phẩm |
| Delete(string id) | Xóa sản phẩm |

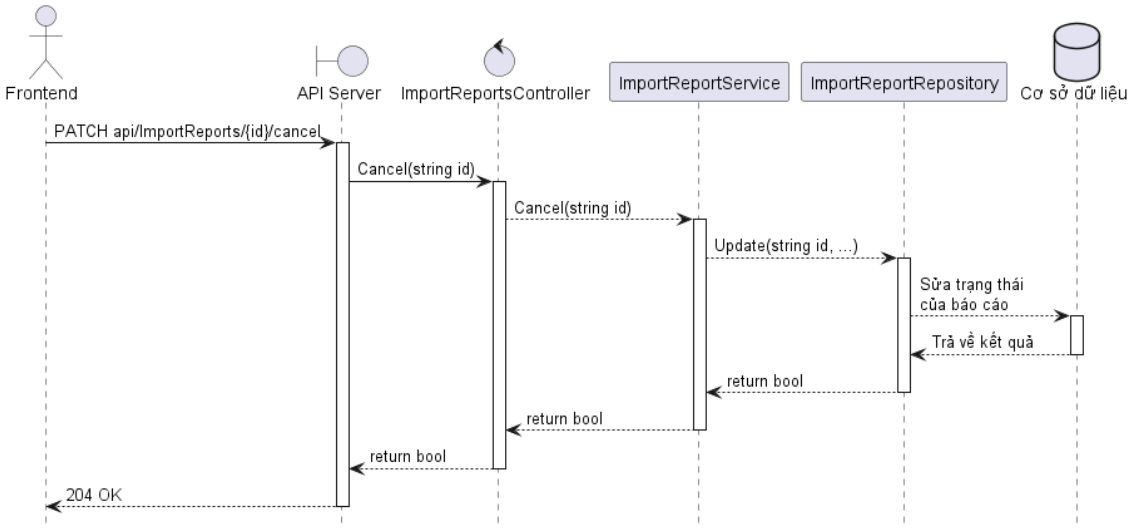
**Bảng 4.5:** Bảng thiết kế lớp ProductRepository

Biểu đồ 4.13 mô tả quá trình các thiết bị frontend gọi API để lấy thông tin phiếu nhập kho. Cụ thể, khi người dùng thực hiện thao tác lấy thông tin phiếu nhập kho, thì đối tượng **ImportReportController** sẽ nhận yêu cầu và gọi đến đối tượng **ImportReportService** để xử lí logic. Sau đó, **ImportReportService** sẽ gọi đến đối tượng **ImportReportRepository** để lấy thông tin từ cơ sở dữ liệu. Cuối cùng, **ImportReportRepository** sẽ trả về kết quả cho **ImportReportService**, **ImportReportService** sẽ trả về kết quả cho **ImportReportController**, **ImportReportController** sẽ trả về kết quả cho thiết bị frontend.



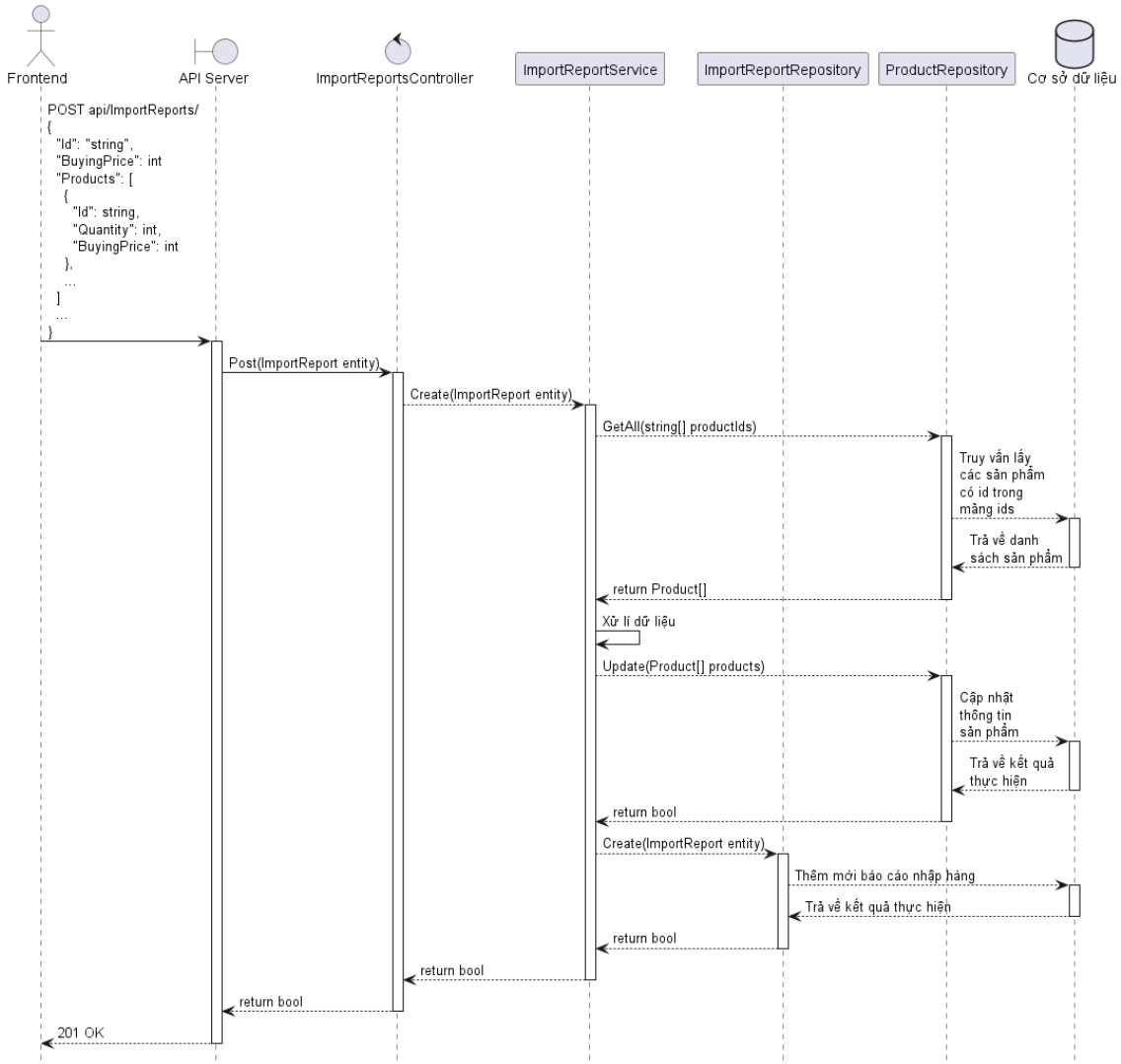

**Hình 4.13:** Biểu đồ trình tự Lấy thông tin phiếu nhập kho

Biểu đồ 4.14 mô tả quá trình thiết bị frontend gọi API để hủy phiếu nhập kho.



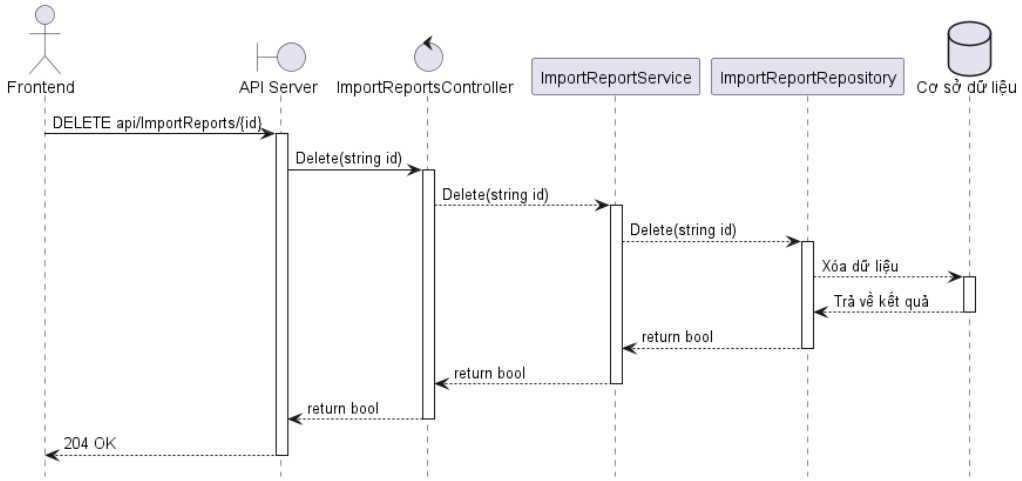

**Hình 4.14:** Biểu đồ trình tự Hủy phiếu nhập kho

Biểu đồ 4.15 mô tả quá trình thiết bị frontend gọi API để tạo phiếu nhập kho.




**Hình 4.15:** Biểu đồ trình tự Tạo phiếu nhập kho

Biểu đồ 4.16 mô tả quá trình thiết bị frontend gọi API để xóa phiếu nhập kho.

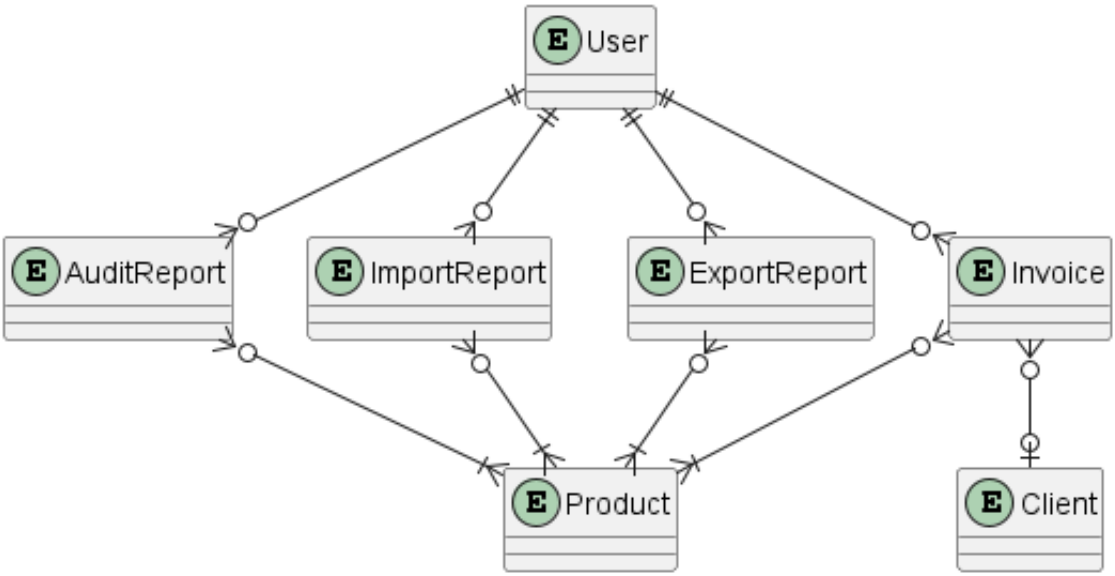



**Hình 4.16:** Biểu đồ trình tự Xóa phiếu nhập kho

### 4.2.3 Thiết kế cơ sở dữ liệu

#### a, Biểu đồ thực thể liên kết

Biểu đồ thực thể liên kết (E-R diagram) của hệ thống được trình bày trong hình 4.17. Trong đó, các thực thể chính bao gồm: **User** (Người dùng), **Client** (Khách hàng), **Product** (Sản phẩm), **AuditReport** (Phiếu kiểm kho), **ImportReport** (Phiếu nhập kho), **ExportReport** (Phiếu xuất kho), **Invoice** (Hóa đơn).




**Hình 4.17:** Biểu đồ thực thể liên kết

Các thực thể **AuditReport**, **ImportReport**, **ExportReport**, **Invoice** đều chứa một hoặc nhiều thực thể **Product** (ghi chép thay đổi hàng hóa), và chứa chính xác một thực thể **User** (người làm phiếu). Ngoài ra thực thể Invoice còn có thể chứa một thực thể **Client** (khách hàng mua hàng).

#### b, Chi tiết thực thể User

Bảng 4.6 là bảng cơ sở dữ liệu **User**, lưu trữ thông tin cá nhân người dùng. Cụ thể người dùng có thể là Quản lí, Nhân viên hay Quản trị viên.

| Tên trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| Id | string | Mã định danh |
| Name | string | Tên người dùng |
| Phonenumber | string | Số điện thoại |
| DateCreated | DateTime | Ngày tạo |

**Bảng 4.6:** Bảng cơ sở dữ liệu User

Bảng 4.7 là bảng cơ sở dữ liệu **UserInfo**, lưu trữ thông tin vắn tắt của người dùng. Thực thể này được đính kèm vào các thực thể **AuditReport**, **ImportReport**, **ExportReport**, **Invoice**. Điều này giúp cho việc cho dù người dùng bị xóa khỏi hệ thống thì các phiếu vẫn có thể lưu trữ được thông tin người tạo phiếu đơn.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Name | string | Tên người dùng |

**Bảng 4.7:** Bảng cơ sở dữ liệu UserInfo

### c, Chi tiết thực thể Client

Bảng 4.8 là bảng cơ sở dữ liệu **Client**, lưu trữ thông tin cá nhân về khách hàng.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Name | string | Tên khách hàng |
| Phonenumber | string | Số điện thoại |
| DateCreated | DateTime | Ngày tạo |
| Email | string? | Địa chỉ email |
| Address | string? | Địa chỉ |
| Description | string? | Mô tả |

**Bảng 4.8:** Bảng cơ sở dữ liệu Client

Bảng 4.9 là bảng cơ sở dữ liệu **ClientInfo**, lưu trữ thông tin vắn tắt của khách hàng. Thực thể này được đính kèm vào thực thể **Invoice**. Điều này giúp cho việc cho dù khách hàng bị xóa khỏi hệ thống thì hóa đơn vẫn còn thông tin khách mua hàng.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Name | string | Tên khách hàng |
| Phonenumber | string | Số điện thoại |

**Bảng 4.9:** Bảng cơ sở dữ liệu ClientInfo

### d, Chi tiết thực thể Product

Bảng 4.10 là bảng cơ sở dữ liệu **Product**, lưu trữ thông tin về sản phẩm.

| Tên trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| Id | string | Mã định danh |
| Name | string | Tên sản phẩm |
| Barcode | string | Mã vạch |
| Group | string? | Nhóm sản phẩm |
| Brand | string? | Thương hiệu |
| StoragePosition | string? | Vị trí lưu trữ |
| Description | string? | Mô tả |
| BuyingPrice | uint | Giá nhập |
| SellingPrice | uint | Giá bán |
| InStock | uint | Số lượng tồn kho |
| DateCreated | DateTime | Ngày tạo |

**Bảng 4.10:** Bảng cơ sở dữ liệu Product

### e, Chi tiết thực thể AuditReport

Bảng 4.11 là bảng cơ sở dữ liệu **AuditReport**, lưu trữ thông tin phiếu kiểm kho.

| Tên trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| Id | string | Mã định danh |
| Author | UserInfo (bảng 4.7) | Thông tin người tạo phiếu |
| DateCreated | DateTime | Ngày tạo |
| DateDeleted | DateTime? | Ngày xóa |
| ProductItems | List<ARPI> (bảng 4.12) | Danh sách sản phẩm |

**Bảng 4.11:** Bảng cơ sở dữ liệu AuditReport

Bảng 4.12 là bảng cơ sở dữ liệu **AuditReportProductItem**, lưu trữ thông tin chi tiết về các sản phẩm trong phiếu kiểm kho.

| Tên trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| Id | string | Mã sản phẩm |
| Name | string | Tên sản phẩm |
| Barcode | string | Mã vạch |
| OriginalQuantity | uint | Số lượng ban đầu |
| AdjustedQuantity | uint | Số lượng đã điều chỉnh |

**Bảng 4.12:** Bảng cơ sở dữ liệu AuditReportProductItem

### f, Chi tiết thực thể ExportReport

Bảng 4.13 mô tả bảng cơ sở dữ liệu **ExportReport** của phiếu xuất kho.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Author | UserInfo (bảng 4.7) | Thông tin người tạo phiếu |
| InvoiceId | string? | Mã hóa đơn |
| DateCreated | DateTime | Ngày tạo |
| DateCancelled | DateTime? | Ngày hủy |
| DateDeleted | DateTime? | Ngày xóa |
| ProductItems | List<ERPI> (bảng 4.14) | Danh sách sản phẩm |

**Bảng 4.13:** Bảng cơ sở dữ liệu ExportReport

Bảng 4.14 mô tả bảng cơ sở dữ liệu **ExportReportProductItem**, lưu trữ thông tin chi tiết về các sản phẩm trong phiếu xuất kho. Cho dù sản phẩm bị xóa khỏi hệ thống thì phiếu xuất kho vẫn còn thông tin.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã sản phẩm |
| Name | string | Tên sản phẩm |
| Barcode | string | Mã vạch |
| Quantity | uint | Số lượng |

**Bảng 4.14:** Bảng cơ sở dữ liệu ExportReportProductItem

### g, Chi tiết thực thể ImportReport

Bảng 4.15 là bảng cơ sở dữ liệu **ImportReport**, lưu thông tin phiếu nhập kho.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Author | UserInfo (bảng 4.7) | Thông tin người tạo phiếu |
| DateCreated | DateTime | Ngày tạo |
| DateCancelled | DateTime? | Ngày hủy |
| DateDeleted | DateTime? | Ngày xóa |
| ProductItems | List<IRPI> (bảng 4.16) | Danh sách sản phẩm |

**Bảng 4.15:** Bảng cơ sở dữ liệu ImportReport

Bảng 4.16 là bảng cơ sở dữ liệu **ImportReportProductItem**, lưu trữ thông tin chi tiết về các sản phẩm trong phiếu nhập kho.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã sản phẩm |
| Name | string | Tên sản phẩm |
| Barcode | string | Mã vạch |
| Quantity | uint | Số lượng |
| Price | uint | Giá nhập |

**Bảng 4.16:** Bảng cơ sở dữ liệu ImportReportProductItem

### h, Chi tiết thực thể Invoice

Bảng 4.17 là bảng cơ sở dữ liệu **Invoice**, lưu trữ thông tin hóa đơn.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã định danh |
| Author | UserInfo (bảng 4.7) | Thông tin người tạo phiếu |
| Client | ClientInfo (bảng 4.9) | Thông tin khách hàng |
| ExportReportId | string | Mã phiếu xuất kho |
| GrandTotal | uint | Thành tiền |
| DateCreated | DateTime | Ngày tạo |
| DatePaid | DateTime? | Ngày thanh toán |
| DateCancelled | DateTime? | Ngày hủy |
| DateDeleted | DateTime? | Ngày xóa |
| ProductItems | List<IPI> (bảng 4.18) | Danh sách sản phẩm |

**Bảng 4.17:** Bảng cơ sở dữ liệu Invoice

Bảng 4.18 là bảng cơ sở dữ liệu **InvoiceProductItem**, lưu trữ thông tin chi tiết về các sản phẩm trong hóa đơn.

| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| Id | string | Mã sản phẩm |
| Name | string | Tên sản phẩm |
| Barcode | string | Mã vạch |
| Quantity | uint | Số lượng |
| Price | uint | Giá bán |

**Bảng 4.18:** Bảng cơ sở dữ liệu InvoiceProductItem

## 4.3 Xây dựng ứng dụng

### 4.3.1 Thư viện và công cụ sử dụng

| Công cụ | Mục đích | Địa chỉ URL |
|---|---|---|
| Visual Studio Code | IDE lập trình | Link |
| Visual Studio 2022 | IDE lập trình | Link |
| C# | Ngôn ngữ lập trình | Link |
| .NET 7.0 | Framework | Link |
| .ASP.NET Core 7.0 | API Framework | Link |
| .NET MAUI | UI Framework | Link |
| MongoDB | Cơ sở dữ liệu | Link |
| BarcodeScanning.MAUI | Thư viện quét mã vạch | Link |
| PlantUML | Vẽ biểu đồ | Link |
| Docker | Công cụ Deploy | Link |

**Bảng 4.19:** Danh sách thư viện và công cụ sử dụng

### 4.3.2 Kết quả đạt được

Dự án đã được đóng gói thành hai sản phẩm: ứng dụng di động và ứng dụng server. Ứng dụng di động được đóng gói thành file APK, có thể cài đặt trên các thiết bị chạy hệ điều hành Android. Ứng dụng server được đóng gói thành Docker Compose file, có thể triển khai trên các máy chủ đã cài đặt Docker. Các sản phẩm đã hoàn thiện đầy đủ các chức năng được đề xuất.

Ứng dụng Server có các thành phần chính như sau:

- **API**: API server, cung cấp các API để ứng dụng di động gọi.
- **Database**: Cơ sở dữ liệu MongoDB.

Ứng dụng di động có các thành phần chính như sau:

- **MAUI App**: Ứng dụng di động, cung cấp giao diện người dùng.

Bảng 4.20 là bảng thống kê về ứng dụng Server và ứng dụng di động.

| Tiêu chí | Thông số |
|---|---|
| Số dòng code | 16,030 dòng |
| Số lớp | 106 lớp |
| Số gói | 12 gói |
| Dung lượng mã nguồn | 12.9 MB |
| Dung lượng ứng dụng Android | 35 MB |

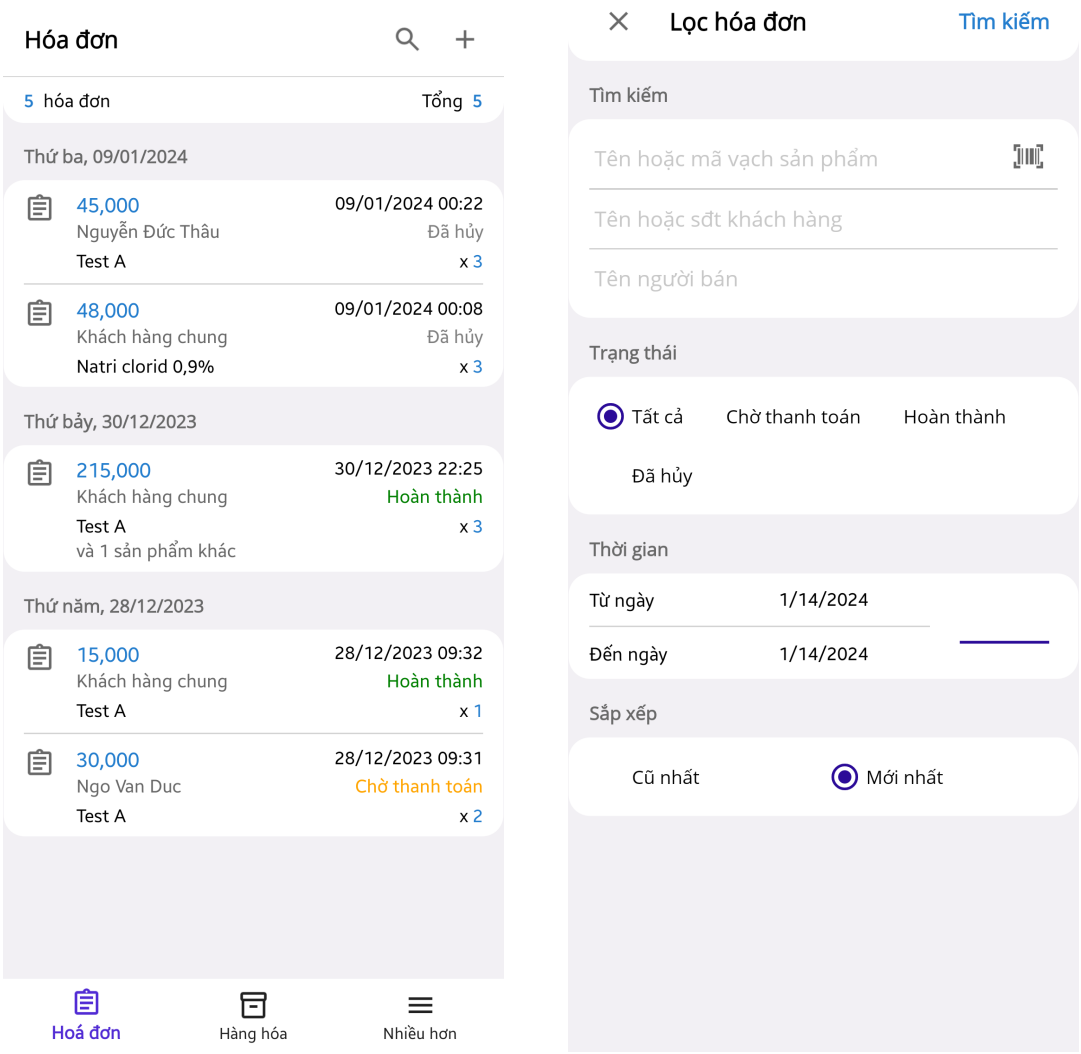
**Bảng 4.20:** Thông tin thống kê

### 4.3.3 Minh họa các chức năng chính

Để minh họa cho sản phẩm cuối của hệ thống và giữ cho đề mục ngắn gọn, sau đây là các màn hình cho một số chức năng chính tiêu biểu, được chụp từ ứng dụng di động trên hệ điều hành Android.

#### a, Chức năng hóa đơn

Hình 4.18a là màn hình danh sách hóa đơn. Tại đây người dùng có thể xem danh sách hóa đơn đã tạo, và có thể tạo hóa đơn mới bằng cách nhấn vào nút " + " ở góc phải màn hình. Người dùng có thể lọc danh sách hóa đơn bằng cách nhấn vào nút hình kính lúp để hiện màn hình bộ lọc. Hình 4.18b là màn hình bộ lọc hóa đơn. Tại đây người dùng có thể lọc danh sách hóa đơn theo các trường tên sản phẩm, tên khách hàng, tên người bán, ngày tạo, và theo trạng thái hóa đơn.

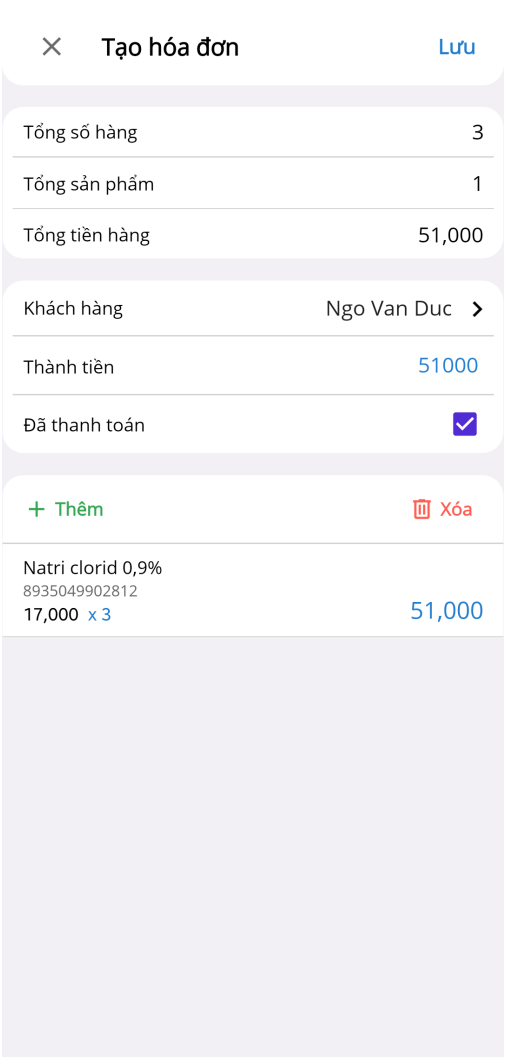



**(a)** Màn hình danh sách hóa đơn **(b)** Màn hình lọc hóa đơn

**Hình 4.18:** Các màn hình hóa đơn

Hình 4.19a là màn hình tạo mới hóa đơn. Tại đây người dùng có thể thêm sản phẩm vào hóa đơn bằng cách nhấn vào nút " + Thêm ". Thêm nữa, người dùng có thể chọn khách hàng đã lưu trong hệ thống từ trước vào hóa đơn bằng cách nhấn vào thanh " Khách hàng ". Sau đó, mục thành tiền sẽ là giá tiền hóa đơn có thể thay đổi tùy vào người bán hàng. Mặc định hóa đơn sẽ có trạng thái "Chưa thanh toán", để thay đổi, người dùng có thể tích vào ô " Đã thanh toán ". Ngoài ra, để xóa sản phẩm khỏi hóa đơn, người dùng có thể vuốt sang trái sản phẩm hay vuốt sang phải để xem chi tiết hơn sản phẩm đó.

Hình 4.19b là màn hình chi tiết hóa đơn. Ở đây chứa mọi thông tin chi tiết về hóa đơn, hơn nữa người dùng có thể hủy hay xóa hóa đơn, và có thể thanh toán hóa đơn. Mọi chức năng đó có thể thực hiện bằng cách nhấn vào nút " ⋮ " ở góc phải màn hình.




**(a)** Màn hình tạo mới hóa đơn

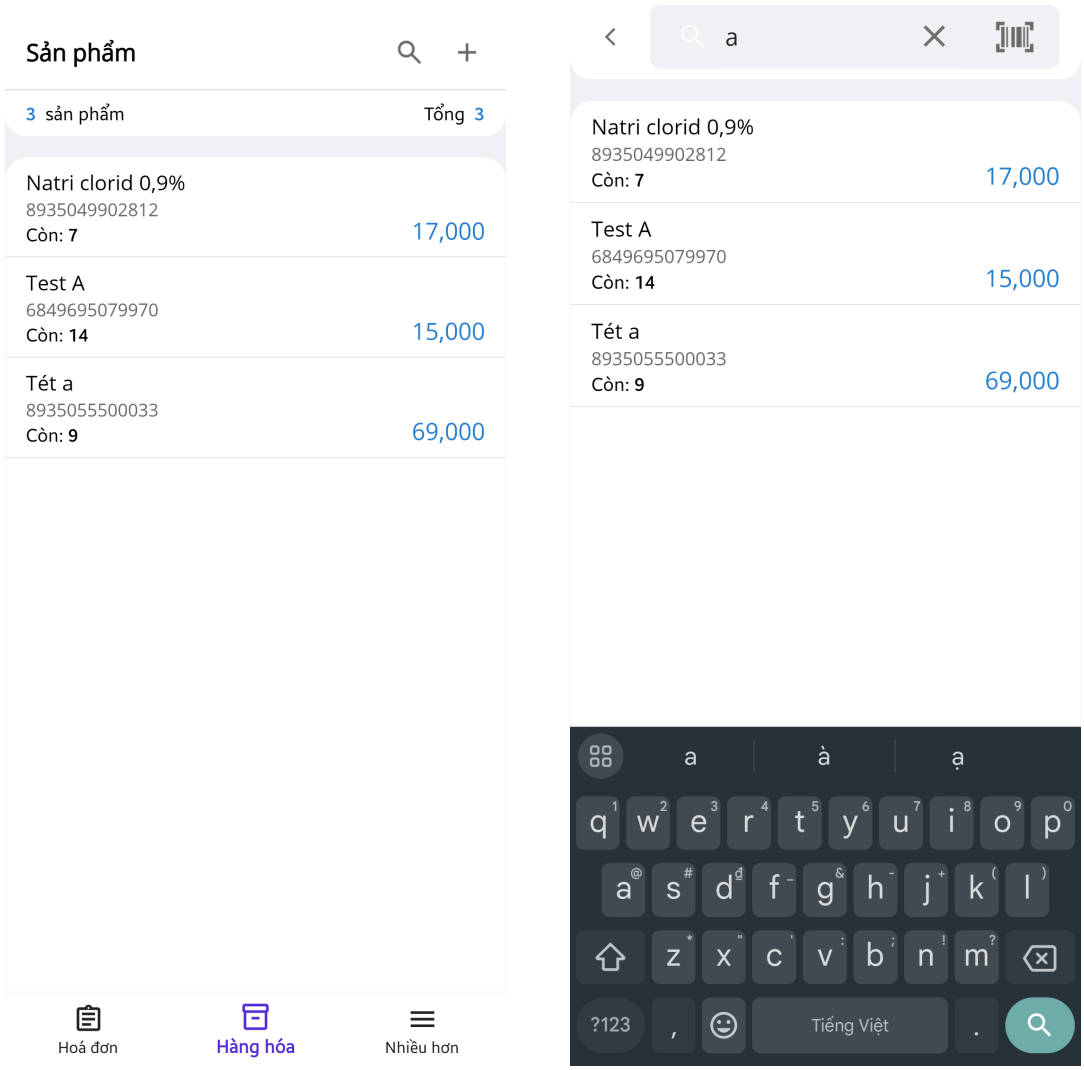

**(b)** Màn hình chi tiết hóa đơn

**Hình 4.19:** Các màn hình hóa đơn

### b, Chức năng sản phẩm

Hình 4.20a là màn hình danh sách sản phẩm. Tại đây người dùng có thể xem danh sách sản phẩm, và có thể tạo sản phẩm mới bằng cách nhấn vào nút " + " ở góc phải màn hình.

Hình 4.20b là màn hình tìm kiếm sản phẩm. Màn hình này được tạo ra hỗ trợ các chức năng khác của hệ thống, như đang trong quá trình người dùng tạo hóa đơn, tạo phiếu nhập kho, tạo phiếu xuất kho, v.v. Người dùng có thể tìm kiếm bằng tên hay quét mã vạch sản phẩm.

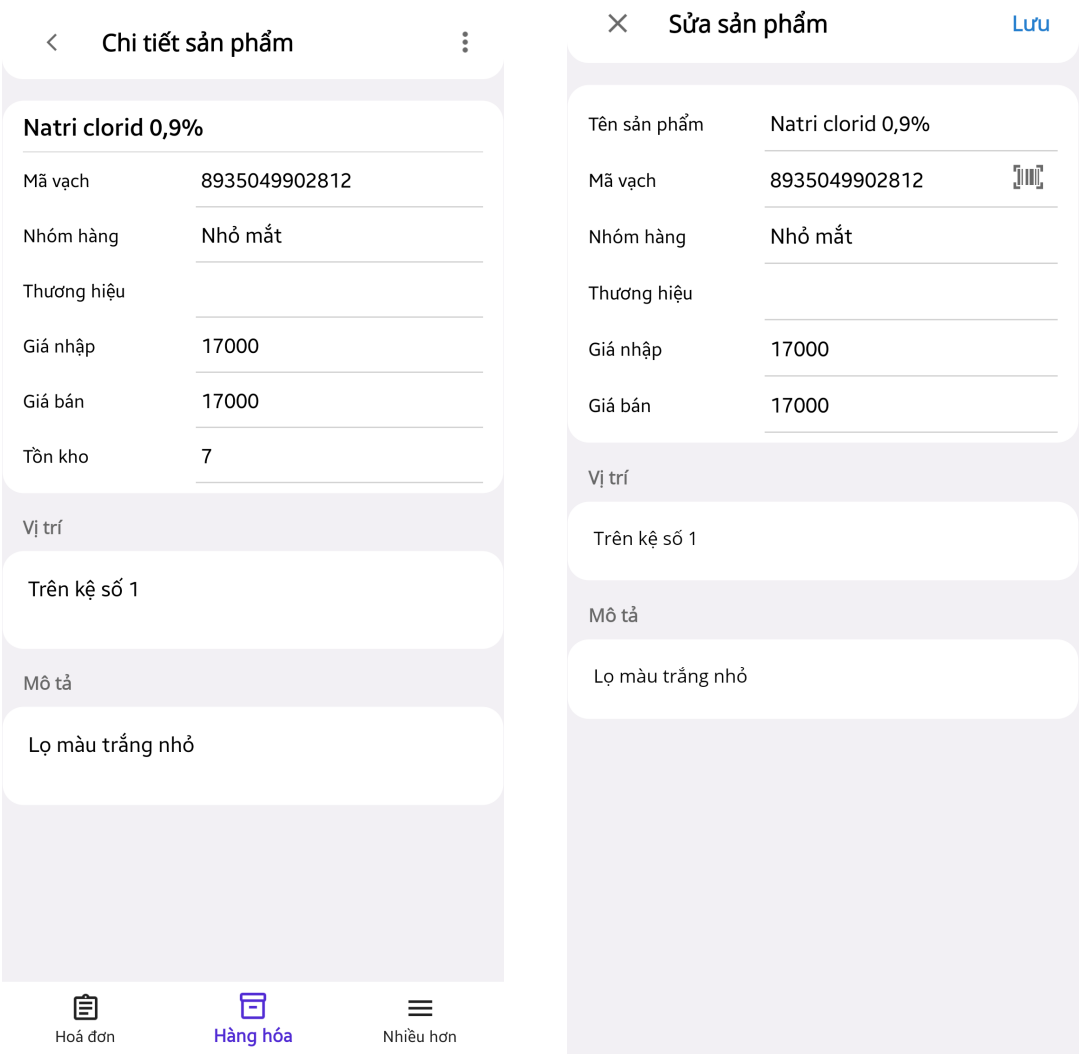



**(a)** Màn hình danh sách sản phẩm **(b)** Màn hình tìm kiếm sản phẩm

**Hình 4.20:** Các màn hình sản phẩm

Hình 4.21a là màn hình chi tiết sản phẩm. Tại đây người dùng có thể xem chi tiết sản phẩm, và có thể cập nhật hay xóa sản phẩm bằng cách nhấn vào nút " ⋮ " ở góc phải màn hình.

Hình 4.21b là màn hình cập nhật sản phẩm. Màn hình này có cấu trúc tương tự như màn hình tạo mới sản phẩm, nhưng có thêm một số trường thông tin đã được điền sẵn khi người dùng chọn sửa sản phẩm từ màn hình chi tiết.




**(a)** Màn hình chi tiết sản phẩm **(b)** Màn hình cập nhật sản phẩm

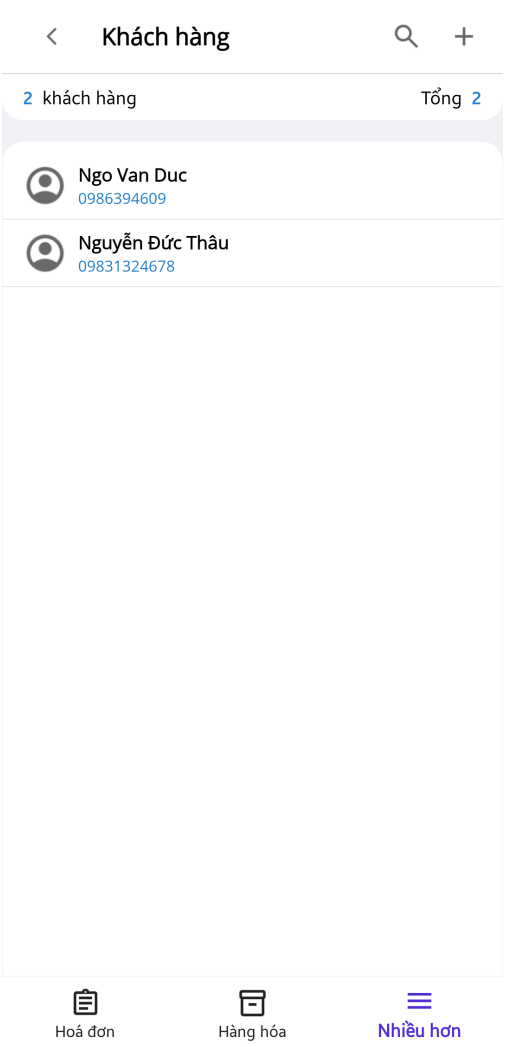
**Hình 4.21:** Các màn hình sản phẩm

### c, Các chức năng khác

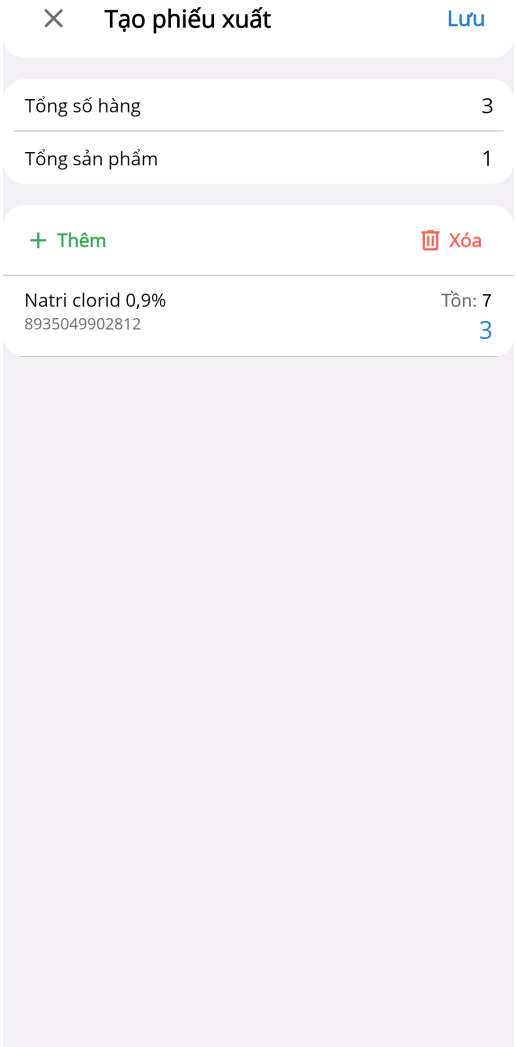
Do hệ thống có cấu trúc giao diện gần tương tự nhau rất nhiều, sau đây là các màn hình của một phần các chức năng khác để minh họa nhanh.

Hình 4.22a là màn hình danh sách khách hàng. Người dùng có thể tạo khách hàng mới bằng cách nhấn vào nút " + " ở góc phải màn hình.

Hình 4.22b là màn hình tạo phiếu xuất kho. Tại đây người dùng có thể thêm sản phẩm vào phiếu xuất kho bằng cách nhấn vào nút " + Thêm ". Ngoài ra, để xóa sản phẩm khỏi phiếu xuất kho, người dùng có thể vuốt sang trái sản phẩm hay vuốt sang phải để xem chi tiết hơn sản phẩm đó.




**(a)** Màn hình danh sách khách hàng

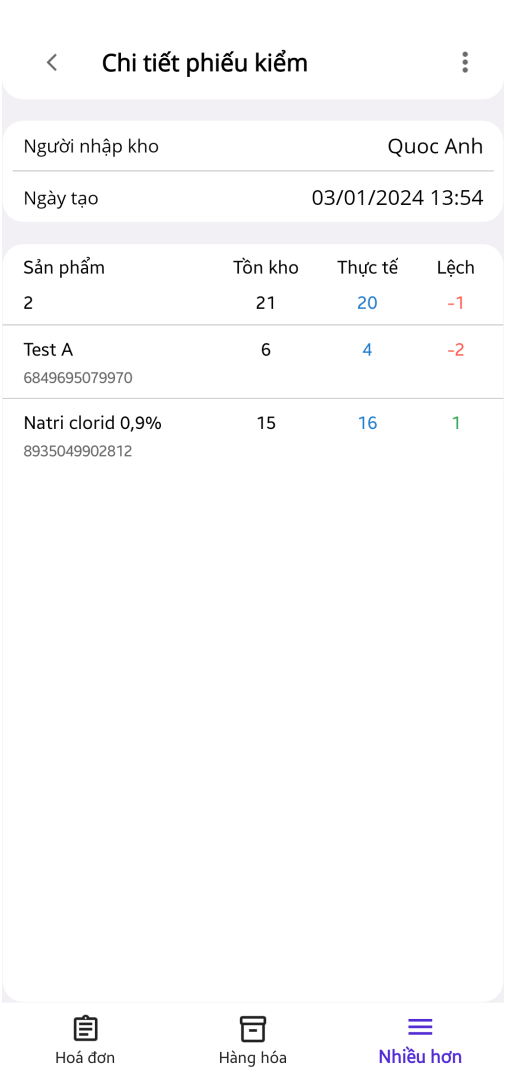



**(b)** Màn hình tạo phiếu xuất kho

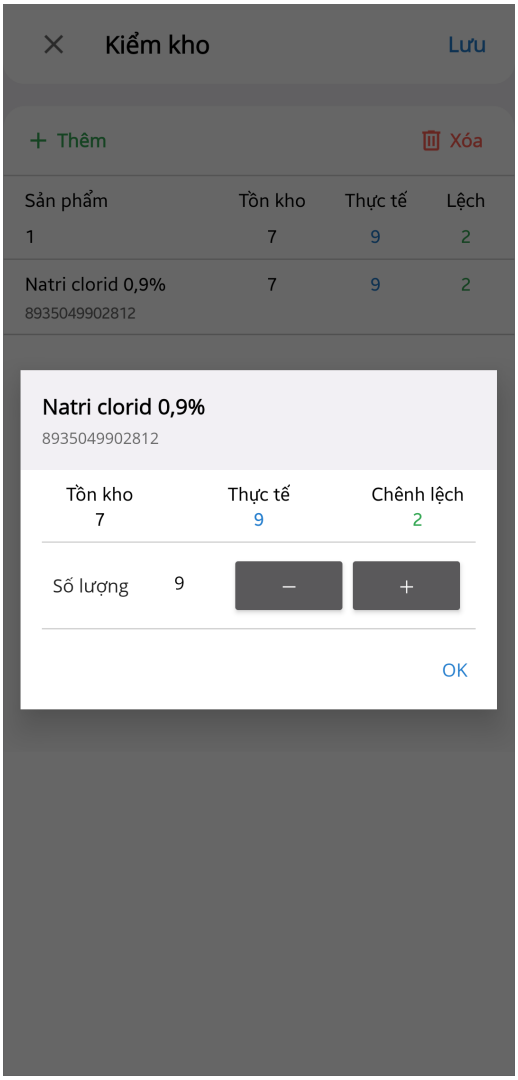
**Hình 4.22:** Các màn hình khác

Hình 4.23a là màn hình chi tiết phiếu kiểm kho. Xóa phiếu kiểm kho bằng cách nhấn vào nút " ⋮ " ở góc phải màn hình. Người dùng có thể xem chi tiết sản phẩm kiểm kho bằng cách nhấn vào sản phẩm đó mà không phải thủ công tìm sản phẩm đó.

Hình 4.23b là màn hình tạo phiếu kiểm kho. Tại đây người dùng có thể thêm sản phẩm vào phiếu kiểm kho bằng cách nhấn vào nút " + Thêm ". Ngoài ra, để xóa sản phẩm khỏi phiếu kiểm kho, người dùng có thể vuốt sang trái sản phẩm hay vuốt sang phải để xem chi tiết hơn sản phẩm đó. Bấm vào sản phẩm nào đó để hiện hộp thoại chỉnh các số liệu của sản phẩm đó.




**(a)** Màn hình chi tiết phiếu kiểm kho




**(b)** Màn hình tạo phiếu kiểm kho

**Hình 4.23:** Các màn hình khác

## 4.4 Kiểm thử

Các chức năng kiểm thử chính được lựa chọn ở đây là để một phần phản ánh lại khả năng của hệ thống, các kịch bản kiểm thử khác quá dài và không thể trình bày hết trong báo cáo này. Các kịch bản được mô tả trong bảng 4.21.

**Bảng 4.21:** Các kịch bản kiểm thử

| Mã | Mô tả | Quy trình | Kết quả mong đợi | Đạt? |
|---|---|---|---|---|
| TC01 | Đăng nhập | 1. Nhập tên đăng nhập, mật khẩu đúng | Tự động chuyển sang màn hình chính | Đạt |
| TC02 | Đăng nhập | 1. Nhập tên đăng nhập, mật khẩu sai | Hiện thông báo lỗi | Đạt |
| TC03 | Tìm hóa đơn bằng tên sản phẩm | 1. Nhập từ khóa "Natri"<br>2. Chọn nút "Tìm kiếm" | Chuyển sang màn hình danh sách và hiện hóa đơn có sản phẩm tên chứa "Natri" | Đạt |
| TC04 | Tìm hóa đơn bằng mã vạch sản phẩm | 1. Bấm nút hình mã vạch<br>2. Quay camera về mã vạch sản phẩm "8642314687523" | Chuyển sang màn hình danh sách và hiện hóa đơn có sản phẩm mã vạch "8642314687523" | Đạt |
| TC05 | Bỏ trống tất cả các trường tìm kiếm hóa đơn | 1. Bỏ trống tất cả các trường<br>2. Chọn nút "Tìm kiếm" | Hiện tất cả hóa đơn | Đạt |
| TC06 | Thêm sản phẩm hết hàng vào phiếu tạo hóa đơn | 1. Chọn nút "+ Thêm"<br>2. Nhập tên sản phẩm "Natri" và bấm tìm<br>3. Chọn sản phẩm có "Còn: 0" | 1. Hiện màn hình tìm kiếm sản phẩm<br>2. Hiện danh sách sản phẩm chứa tên "Natri"<br>3. Quay về màn hình tạo hóa đơn và hiện thông báo lỗi "Hết hàng" | Đạt |

| Tiếp của bảng 4.21 | | | | |
|---|---|---|---|---|
| **Mã** | **Mô tả** | **Quy trình** | **Kết quả mong đợi** | **Đạt?** |
| TC07 | Thêm sản phẩm vào phiếu tạo hóa đơn | 1. Chọn nút "+ Thêm"<br>2. Nhập tên sản phẩm "Trà" và bấm tìm<br>3. Chọn "Trà Actiso" | 1. Hiện màn hình tìm kiếm sản phẩm<br>2. Hiện danh sách sản phẩm chứa tên "Trà"<br>3. Quay về màn hình tạo hóa đơn và hiện sản phẩm đã chọn trong danh sách sản phẩm của hóa đơn | Đạt |
| TC08 | Chỉnh số lượng sản phẩm bán trong phiếu tạo hóa đơn | 1. Chọn sản phẩm "Trà Actiso"<br>2. Bấm dấu cộng 3 lần<br>3. Chọn "OK" | 1. Hiện hộp thoại chỉnh số lượng<br>2. Số lượng tăng lên "4"<br>3. Quay về màn hình tạo hóa đơn, sản phẩm hiện "20,000 x 4" "80,000" và thanh "Thành tiền" tăng lên 80000 | Đạt |
| TC09 | Hoàn thành tạo hóa đơn đã thanh toán | 1. Tích ô "Đã thanh toán"<br>2. Chọn nút "Lưu" | Quay về màn hình danh sách hóa đơn và hiển thị lại trang có hóa đơn đã thanh toán mới nhất | Đạt |
| TC10 | Tạo thêm sản phẩm mới | 1. Chọn nút "+"<br>2. Nhập các trường và chọn nút "Lưu" | 1. Hiện màn hình tạo mới sản phẩm<br>2. Quay về màn hình danh sách sản phẩm và hiện sản phẩm đã thêm trong danh sách sản phẩm | Đạt |
| TC11 | Xóa sản phẩm | 1. Chọn sản phẩm trong màn hình danh sách sản phẩm<br>2. Chọn nút " ⋮ " $\rightarrow$ "Xóa" | 1. Hiện màn hình chi tiết sản phẩm<br>2. Quay về màn hình danh sách sản phẩm và sản phẩm không còn trong danh sách sản phẩm | Đạt |

| Tiếp của bảng 4.21 | | | | |
|---|---|---|---|---|
| **Mã** | **Mô tả** | **Quy trình** | **Kết quả mong đợi** | **Đạt?** |
| TC12 | Cập nhật sản phẩm | 1. Chọn sản phẩm trong màn hình danh sách sản phẩm<br>2. Chọn nút " ⋮ " → "Sửa" | 1. Hiện màn hình chi tiết sản phẩm<br>2. Hiện màn hình sửa sản phẩm với các trường đã được điền sẵn<br>3. Quay về màn hình chi tiết sản phẩm với thông tin mới | Đạt |
| TC13 | Làm mới màn hình danh sách sản phẩm | 1. Kéo từ giữa màn hình xuống dưới một đoạn cho hiện vòng tròn | Màn hình danh sách làm mới dữ liệu và vòng tròn biến mất | Đạt |
| TC14 | Xem chi tiết sản phẩm trong phiếu nhập kho | 1. Chọn phiếu nhập trong màn hình danh sách nhập kho<br>2. Chọn sản phẩm | 1. Hiện màn hình chi tiết phiếu nhập kho<br>2. Hiện màn hình chi tiết sản phẩm | Đạt |
| TC15 | Hủy phiếu nhập kho | 1. Chọn phiếu nhập trong màn hình danh sách nhập kho<br>2. Chọn nút " ⋮ " → "Hủy" | 1. Hiện màn hình chi tiết phiếu nhập kho<br>2. Màn hình chi tiết được làm mới và hiện thêm trường "Ngày hủy phiếu" với thời gian hủy | Đạt |
| TC16 | Xem chi tiết sản phẩm trước khi chọn | 1. Hiện trang tìm kiếm sản phẩm và tìm kiếm<br>2. Quẹt phải một sản phẩm để hiện chữ "Info" | 1. Hiện màn hình tìm kiếm sản phẩm<br>2. Hiện danh sách sản phẩm<br>3. Hiện màn hình chi tiết sản phẩm | Đạt |
| TC17 | Đăng xuất | 1. Chọn nút hình mũi nên chỉ cửa màu đỏ | Quay về màn hình đăng nhập | Đạt |

## 4.5 Triển khai

### 4.5.1 Hướng dẫn triển khai

Hệ thống được thiết kế với mô hình Server-Client. Để triển khai phía Server, cần tối thiểu một máy chủ đã cài đặt Docker. Sau đây là các bước cài đặt và triển khai:

1. Làm theo hướng dẫn cài đặt Docker.
2. Lấy source code từ Github.
   git clone https://github.com/Corn207/Inventoice.git
3. Tạo volume để lưu trữ cơ sở dữ liệu.
   docker volume create inventoice
4. Chạy docker compose.
   docker compose -f docker-compose.yml -f docker-compose.prod.yml up -d

Về phía Client, hệ thống có thể được triển khai trên các thiết bị chạy hệ điều hành Android hoặc iOS. Với các thiết bị Android, có thể download ứng dụng từ đây và cài đặt trực tiếp.

### 4.5.2 Triển khai thử nghiệm

Hệ thống đang được triển khai thử nghiệm trên máy chủ có cấu hình như sau:

- CPU: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
- RAM: 12 GB
- Ổ cứng: SSD 128 GB
- Hệ điều hành: Ubuntu 23.04 lunar
- Docker version 24.0.7, build afdd53b

Hệ thống server hiện tại đang được triển khai tại địa chỉ `https://ivt.corn207.top/api/`. Hệ thống có thể được truy cập bằng cách gửi các yêu cầu HTTP đến địa chỉ này. Ngoài ra, hệ thống có trang liệt kê các đầu API hiện có tại địa chỉ `https://ivt.corn207.top/swagger/index.html`. Để thêm tính bảo mật của server và chống DDoS, domain của server đang được quản lý bởi Cloudflare và giới hạn truy cập chỉ trong Việt Nam.

Bảng 4.22 là bảng thống kê thông tin về hệ thống server đang được triển khai thử nghiệm. Các thông số được tính đến hết ngày 14/01/2024.

| **Tiêu chí** | **Thông số** |
|---|---|
| Số lượng người dùng | 3 người dùng |
| Số sản phẩm | 11 sản phẩm |
| Số hóa đơn | 12 hóa đơn |
| Số các phiếu xuất nhập tồn | 21 phiếu |
| Số khách hàng | 6 khách hàng |
| Thời gian trung hình phản hồi | dưới 1 giây |

**Bảng 4.22:** Thông tin thống kê hệ thống thử nghiệm

### 4.5.3 Phản hồi người dùng

Hệ thống đã được triển khai thử nghiệm và sử dụng bởi một số người dùng. Dựa trên form phản hồi của người dùng về hệ thống và ứng dụng, tôi đã tổng hợp lại số phản hồi. Bảng 4.23 thống kê chung phản hồi tính đến ngày 13/01/2024.

**Bảng 4.23:** Thông tin phản hồi người dùng

| **Chức năng hài lòng nhất** | |
|---|---|
| **Chức năng** | **Tỷ lệ chọn** |
| Quản lý sản phẩm | 18.2% |
| Quản lý khách hàng | 27.3% |
| Quản lý hóa đơn | 27.3% |
| Quản lý kiểm kho | 9.1% |
| Quản lý nhập kho | 9.1% |
| Quản lý xuất kho | 18.2% |
| **Cải thiện chức năng** | |
| **Chức năng** | **Nội dung** |
| Quản lý sản phẩm | Thêm chức năng chụp ảnh sản phẩm |
| Quản lý khách hàng | Thêm chức năng nghe, gọi |
| Quản lý hóa đơn | Cải thiện trang thanh toán |
| Quản lý kiểm kho | Không |
| Quản lý nhập kho | Thêm thống kê nhập kho |
| Quản lý xuất kho | Không |
| **Tỷ lệ đáp ứng nhu cầu** | 100% |
| **Đánh giá độ hài lòng chức năng** | |
| 1 sao | 0% |
| 2 sao | 0% |

| | |
|---|---|
| 3 sao | 18.2% |
| 4 sao | 72.7% |
| 5 sao | 9.1% |
| **Đề xuất chức năng mới** | |
| Thêm quản lý hạn sử dụng | |
| Theo dõi thông số bảo quản | |
| Nhắn tin nội bộ hệ thống | |
| Lịch sử xuất nhập | |
| Thống kê | |
| Chụp ảnh sản phẩm | |
| **Đánh giá độ hài lòng giao diện** | |
| 1 sao | 0% |
| 2 sao | 0% |
| 3 sao | 0% |
| 4 sao | 81.8% |
| 5 sao | 18.2% |
| **Góp ý giao diện** | |
| Nhiều màu hơn | |
| Tùy chỉnh cỡ chữ | |
| Thiết kế bắt mắt hơn | |

# CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

## 5.1 Kiến trúc Clean Architecture

### 5.1.1 Dẫn dắt vấn đề

Trong quá trình thiết kế kiến trúc hệ thống, việc chia nhỏ mã nguồn và phân tách thành các tầng lớp riêng mang lại nhiều lợi ích. Mỗi tầng lớp đó sẽ mang một vài chức năng cụ thể, giúp cho việc phát triển, bảo trì trở nên dễ dàng hơn. Việc này sẽ giúp các đội lập trình viên có thể tập trung vào nhiệm vụ của họ mà không phải bận tâm đến những thay đổi của các thành viên khác. Miễn sao kiến trúc hệ thống có một quy chuẩn chung, các thành viên sẽ có thể độc lập làm việc với nhau. Ví dụ như trong kiến trúc N-tier, chúng được chia thành các tầng lớp mang nhiệm vụ cụ thể:

- **Presentation Layer**: Tầng lớp này sẽ chịu trách nhiệm hiển thị giao diện người dùng, nhận các sự kiện từ người dùng và gửi đến tầng **Business Logic Layer** để xử lí các yêu cầu đó.
- **Business Logic Layer**: Tầng lớp này sẽ xử lý các nghiệp vụ của hệ thống. Nó sẽ nhận các yêu cầu từ tầng trên nó (**Presentation Layer**), sử dụng các tham số đó để truy cập dữ liệu ở tầng dưới (**Data Access Layer**) và cuối cùng thực hiện xử lí dữ liệu.
- **Data Access Layer**: Tầng lớp này chịu trách nhiệm truy cập dữ liệu từ cơ sở dữ liệu. Nó sẽ nhận các yêu cầu từ tầng (**Business Logic Layer**), truy cập vào cơ sở dữ liệu và gửi lại kết quả. Tầng này cũng định nghĩa các hàm sẵn với các câu truy vấn cơ sở dữ liệu đã được chuyên môn hóa và tối ưu sẵn, giúp cho việc sử dụng lại code được tận dụng tối đa.
- **Data Layer**: Tầng lớp này sẽ chịu trách nhiệm lưu trữ dữ liệu của hệ thống. Thường chúng sẽ là các hệ cơ sở dữ liệu, hay đơn thuần chỉ là các cách thức lưu trữ dữ liệu khác như file, cache, v.v.

Như đã thấy ở trên, kiến trúc hệ thống đã có sự chuyên môn hóa rõ ràng. Nhưng trong thực tế, việc thay đổi trong công nghệ sử dụng và sự thích nghi theo thị trường sẽ luôn tạo ra vấn đề. Ví dụ như khi một công nghệ mới ra đời, nó có thể giúp cho việc phát triển hệ thống trở nên dễ dàng hơn, tuy nhiên nó lại không tương thích với kiến trúc hệ thống hiện tại. Điều này dẫn đến việc phải thay đổi kiến trúc hệ thống để phù hợp với công nghệ mới.

Một ví dụ thực tiễn, khi một hệ thống khi lần đầu thiết kế, họ dùng một hệ cơ sở dữ liệu phù hợp với số lượng người dùng hiện giờ, việc thiết kế trước cho tương lai với số lượng người dùng không định hình trước được là rất tốn kém. Tuy nhiên, khi hệ thống phát triển, số lượng người dùng tăng lên, hệ cơ sở dữ liệu hiện tại không còn phù hợp nữa. Và việc thay đổi hệ cơ sở dữ liệu sẽ ảnh hưởng đến toàn bộ hệ thống. Điều này dẫn đến việc phải thay đổi toàn bộ hệ thống để phù hợp với hệ cơ sở dữ liệu mới. Điều này sẽ tốn kém thời gian và công sức của các lập trình viên. Và đây cũng là một trong những vấn đề mà kiến trúc Clean Architecture hướng đến.

### 5.1.2 Giải pháp

Để giải quyết vấn đề trên, một kiến trúc mới đã ra đời, đó là Clean Architecture. Kiến trúc này sẽ tách biệt hoàn toàn các thành phần của hệ thống với nhau, giúp cho việc thay đổi một thành phần không ảnh hưởng đến các thành phần khác. Điều này sẽ giúp cho việc phát triển và bảo trì hệ thống trở nên dễ dàng hơn. Kiến trúc này sẽ được chia thành các tầng lớp hầu như rất tương tự kiến trúc N-tier, tuy nhiên nó có một số điểm khác biệt để giải quyết vấn đề các tầng phụ thuộc nhau trong kiến trúc N-tier. Các tầng lớp của kiến trúc Clean Architecture sẽ được chia thành các tầng lớp sau:

- **Presentation Layer**: Tương tự như kiến trúc N-tier.
- **Infrastructure Layer**: Tầng lớp này tương ứng với tầng **Data Access Layer** của kiến trúc N-tier nhưng khác ở chỗ, giờ nó phụ thuộc vào tầng **Application Layer**.
- **Application Layer**: Tầng này gần như tương tự với tầng **Business Logic Layer** của kiến trúc N-tier, tuy nhiên nó không phụ thuộc vào tầng **Infrastructure Layer**.
- **Domain Layer**: Tầng này sẽ chứa các thực thể của hệ thống, các thực thể này sẽ được định nghĩa bởi người thiết kế hệ thống. Tầng này sẽ không phụ thuộc vào bất kì tầng nào khác.

Lí do kiến trúc này thay thứ tự phụ thuộc của nhau đều có mục đích đặc biệt để giải quyết vấn đề, đặc biệt trong trường hợp này là vấn đề thay đổi hệ cơ sở dữ liệu. Với kiến trúc Clean Architecture, việc thay đổi hệ cơ sở dữ liệu sẽ không ảnh hưởng đến các tầng khác. Vì các tầng khác không phụ thuộc vào tầng **Infrastructure Layer** nữa. Đây là một trong các lợi ích của kiến trúc này.

Cụ thể, đặt ra một ví dụ giả thiết, ta đang thiết kế hệ thống với kiến trúc N-tier với hệ cơ sở dữ liệu SQL Server. Bởi vì tính chất của hệ thống giả dụ là thay đổi cấu trúc dữ liệu thường xuyên như là mạng xã hội,... khiến các thực thể biểu diễn

cho các bảng trong SQL Server thường xuyên phải cấu trúc lại schema rất nhiều. Điều này khiến việc duy trì và phát triển gặp nhiều bất tiện và tốn kém thời gian. Vì vậy, ta sẽ quyết định dùng hệ cơ sở dữ liệu MongoDB để thay thế vì đặc tính của nó là document database, có thể thay đổi cấu trúc dữ liệu một cách dễ dàng. Tuy nhiên, với kiến trúc N-tier, việc thay đổi hệ cơ sở dữ liệu sẽ ảnh hưởng đến tầng **Data Access Layer** và suy ra, các tầng bên trên như **Business Logic Layer**, v.v. cũng phải thay đổi theo. Nếu ta biết trước được điều này, cách tốt nhất là phải có một kiến trúc có khả năng chống chịu được thay đổi này.

Chi tiết lựa chọn làm tầng **Infrastructure Layer** phụ thuộc ngược lại tầng **Application Layer** sẽ được trình bày ở như sau. Sự thay đổi trong mã nguồn trong tầng nằm trên sẽ không ảnh hưởng đến tầng nằm dưới. Vì vậy, tầng **Application Layer** sẽ phụ thuộc vào tầng **Infrastructure Layer**. Nhưng làm sao để tầng **Application Layer** có thể gọi các hàm truy vấn cơ sở dữ liệu mà không phụ thuộc vào tầng **Infrastructure Layer** được. Ở đây chúng ta sẽ áp dụng một kĩ thuật trong lập trình hướng đối tượng (Object Oriented Programming) để giải quyết vấn đề này.

Thay vì chúng ta định nghĩa đầy đủ logic gọi truy vấn cơ sở dữ liệu trong một tầng. Chúng ta sẽ chỉ định nghĩa đầu vào và đầu ra của các hàm truy vấn cơ sở dữ liệu. OOP cung cấp khái niệm Interface cho việc định nghĩa hàm này. Ví dụ, việc truy vấn sản phẩm cần đầu vào là một biến kiểu chữ Id và đầu ra là một biến kiểu Product. Ta chỉ việc viết một interface có tên là IProductRepository với một hàm có tên là GetProductById() trong chính tầng **Application Layer** hoặc các tầng khác ngang hàng hoặc thấp hơn. Còn khi ta định nghĩa cách gọi truy vấn cơ sở dữ liệu, ta sẽ viết một class có tên là ProductRepository kế thừa từ interface IProductRepository và định nghĩa hàm GetProductById() trong tầng **Infrastructure Layer**. Và tầng **Application Layer** sẽ gọi hàm GetProductById() thông qua interface IProductRepository. Điều này sẽ giúp cho việc thay đổi hệ cơ sở dữ liệu không ảnh hưởng đến tầng **Application Layer**.

Sau khi đã thiết kế xong được 2 tầng **Application Layer** và **Infrastructure Layer**. Ở mã nguồn sản phẩm cuối, người lập trình chỉ cần nối các tầng với nhau và có thể chọn tầng **Infrastructure Layer** phù hợp với hệ cơ sở dữ liệu mà họ muốn. Một kĩ thuật trong lập trình hướng đối tượng khác để thực hiện sự lựa chọn này là Dependency Injection. Chúng mang vai trò như một trung tâm khởi tạo và duy trì vòng đời của các lớp trong hệ thống, ngoài ra chúng còn cho phép người lập trình có thể lựa chọn các lớp nào sẽ là lớp được sử dụng cho interface nào. Dẫn tới kĩ thuật này sẽ đưa vào các lớp của tầng **Application Layer** một đối tượng của tầng **Infrastructure Layer** thông qua interface. Việc chọn hệ cơ sở dữ liệu dễ dàng chỉ bằng thay đổi một dòng code đơn giản.

### 5.1.3 Kết quả đạt được

Áp dụng các kĩ thuật được mô tả trong phần 5.1.2, hệ thống của chúng ta có thể đáp ứng được sự biến đổi và tiến hóa của công nghệ. Không những thế, nó còn đem lại khả năng mở rộng, khả năng thích ứng với mọi môi trường. Một hệ thống nay có thể cho phép người quản lý tha hồ lựa chọn hệ cơ sở dữ liệu mà họ muốn mà không ảnh hưởng đến các thành phần khác của hệ thống.

Ví dụ vì hệ thống theo kiến trúc *Clean Architecture* nên ta có thể tạo ra hai biến thể của tầng **Infrastructure Layer** là **SQLServerInfrastructureLayer** và **MongoDBInfrastructureLayer**. Và hệ thống triển khai đã có sẵn SQL Server, họ chỉ cần nối tầng **SQLServerInfrastructureLayer** vào tầng **Application Layer** thông qua **Dependency Injection**. Và khi họ muốn chuyển sang MongoDB, họ chỉ cần thay đổi một dòng code trong **Dependency Injection** để nối tầng **MongoDBInfrastructureLayer** vào tầng **Application Layer**.

Những ví dụ trên đã được thấy trong thực tế rất nhiều khi Discord chuyển từ PostgreSQL sang Cassandra, khi Facebook chuyển từ MySQL sang Cassandra, v.v. Điều này cho thấy kiến trúc *Clean Architecture* là một kiến trúc phù hợp với thực tế và có thể giải quyết được nhiều vấn đề và thay đổi trong hệ cơ sở dữ liệu là một ví dụ điển hình.

Khi sau một thời gian thực hành và nghiên cứu, và được biết đến những vấn đề trên đã thúc đẩy tôi tìm hiểu kĩ hơn về các kiến trúc hệ thống. Chúng khiến tôi thấy thú vị và thấy được một trong các ứng dụng của lập trình hướng đối tượng tạo nên sự linh hoạt và dễ dàng trong việc phát triển hệ thống. Và đồ án tốt nghiệp này là một cơ hội tốt để tôi có thể áp dụng những kiến thức đã học vào thực tế.

## 5.2 Kiến trúc Model-View-ViewModel

### 5.2.1 Dẫn dắt vấn đề

Các kĩ sư phần mềm thường sử dụng nhiều mô hình, công cụ khác nhau để hỗ trợ cho việc phát triển và kiểm thử ứng dụng và website. Việc phân tách ứng dụng thành các thành phần riêng biệt giúp cho đội ngũ phát triển có thể độc lập làm việc với nhau. Một trong những mô hình được sử dụng phổ biến nhất là mô hình Model-View-Controller (MVC). Mô hình này phân tách ứng dụng thành 3 thành phần chính:

- **Model**: Thành phần này chịu trách nhiệm cho việc lưu trữ dữ liệu và xử lý dữ liệu.
- **View**: Thành phần này chứa các giao diện người dùng và hiển thị dữ liệu.
- **Controller**: Thành phần này chịu trách nhiệm điều hướng các yêu cầu từ người dùng từ **View** tới **Model**.

Nhưng với sự phát triển của công nghệ, các ứng dụng ngày càng phức tạp hơn. Điểm bất cập của mô hình này là logic xử lí UI và xử lí dữ liệu từ **Model** thường đi cùng với nhau khiến cho việc phát triển và bảo trì trở nên khó khăn hơn. Hơn nữa, với xu hướng giao diện ứng dụng ngày càng hiện đại, sẽ rất khó để áp dụng các kĩ thuật xác thực đầu vào người dùng. Hơn nữa, mô hình này gây cho việc sử dụng lại code trở nên khó khăn và quy trình kiểm thử cũng rất rắc rối. Mô hình này được sử dụng rộng rãi trong các ứng dụng web. Tuy nhiên, nó lại không phát huy được điểm mạnh của các ứng dụng di động.

### 5.2.2 Giải pháp

Vì vậy, một mô hình khác được tạo ra để phù hợp với các ứng dụng di động, đó là mô hình Model-View-ViewModel (MVVM). Mô hình này được sử dụng rộng rãi trong các ứng dụng di động và ứng dụng web đơn trang (Single Page Application - SPA). Mô hình này cũng phân tách ứng dụng thành 3 thành phần chính:

- **Model**: Thành phần này chịu trách nhiệm cho việc lưu trữ dữ liệu và xử lý dữ liệu.
- **View**: Thành phần này chứa các giao diện người dùng và hiển thị dữ liệu.
- **ViewModel**: Thành phần này đóng vai trò trung gian giữa **View** và **Model**. Nó sẽ xử lí các hoạt động truyền tải dữ liệu giữa **View** và **Model**.

Điều khác biệt của mô hình này so với mô hình MVC là ở **ViewModel**. Nó không hề biết đến, thậm chí cần đến sự tồn tại của **View**, điều này có thể tách được logic xử lí dữ liệu của giao diện người dùng với thiết kế giao diện người dùng. Sự phân hóa này có thể giúp cho người thiết kế và người lập trình được tự do hơn. Cách **ViewModel** truyền tải dữ liệu không còn đơn thuần là lắng nghe thay đổi từ **View** và bảo **Model** phải thay đổi theo nữa. Mà nó sẽ thông qua các **Binding**. **Binding** được hiểu như là một đối tượng đặc biệt trong việc chuyển hóa từ kiểu dữ liệu này thành kiểu dữ liệu khác. Lợi dụng điều này, mô hình tạo lên một cơ chế tự động thay đổi dữ liệu gọi là **Two-way Data Binding**. Cơ chế này sẽ tự động cập nhật dữ liệu từ **View** tới **ViewModel** và ngược lại. Khi đó **ViewModel** như là một biến thể dữ liệu của **Model** dành riêng để trình diễn.

Ban đầu mô hình này được thiết kế cho các ứng dụng WPF (Windows Presentation Foundation) của Microsoft cho máy tính Windows. Tuy nhiên bởi vì lợi ích của nó, rất nhiều nền tảng, framework khác đã ứng dụng vào trong các ứng dụng của họ. Dần dần, nó đã trở thành một mô hình phổ biến trong các ứng dụng di động và ứng dụng web đơn trang.

### 5.2.3 Kết quả đạt được

Khi lựa chọn công nghệ để thiết kế ứng dụng điện thoại, đơn thuần ban đầu tôi chỉ muốn tiếp tục sử dụng hệ sinh thái .NET để tăng khả năng sử dụng lại code. Và tôi đã tìm hiểu về .NET MAUI, một framework cho phép viết ứng dụng di động, máy tính bằng ngôn ngữ C#. Tuy nhiên, khi tôi tìm hiểu sâu hơn về framework này, rất nhiều tài liệu đều hướng đến sử dụng mô hình MVVM. Từ đây, tôi được biết thêm một mô hình hiện đại mà tôi chưa từng biết đến ngoài mô hình quá đỗi nổi tiếng MVC. Tôi đã tìm hiểu sâu hơn về mô hình này và thấy được sự linh hoạt của nó. Đây là một sự tình cờ tuyệt vời và cũng là một trong những điều mới tôi học được trong quá trình làm đồ án tốt nghiệp này.

# CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1 Kết luận

Sau khi hoàn thiện hệ thống đề xuất, tôi nhận thấy rằng hệ thống đề xuất đã đạt được những kết quả như mong đợi. So sánh với các hệ thống hiện giờ trên thị trường, hệ thống đề xuất của tôi mang những điểm lợi như sau:

- Giao diện người dùng đơn giản và tường minh, chỉ cần một ứng dụng mà có thể làm được toàn bộ chức năng quản lí. Người dùng không phải cao siêu hay là tín đồ công nghệ nhiều mà vẫn hiểu được cách thức hoạt động của hệ thống.
- Hệ thống có tính mở rộng cực cao, có thể được xây dựng với ứng dụng web, ứng dụng máy tính mà không phải thay đổi hệ backend nhiều. Hệ cơ sở dữ liệu không còn là vấn đề khi người dùng muốn thay đổi nền tảng.
- Sự bảo toàn dữ liệu, không hề có bên thứ ba nào tham gia trong quá trình cài đặt và sử dụng hệ thống nếu người dùng mong muốn.
- Chi phí duy trì hệ thống hoàn toàn dựa vào quy mô triển khai của người dùng.
- Bởi vì đối tượng sử dụng của hệ thống là các hộ kinh doanh vừa và nhỏ, hệ thống có sức chịu tải lớn và độ phản hồi rất cao.

Tuy nhiên, hệ thống vẫn còn một số hạn chế do thời gian thực hiện có hạn, cũng như khả năng của tôi còn hạn chế. Các hạn chế của hệ thống như sau:

- Hệ thống chưa có những tính năng để phục hồi hay sao lưu dữ liệu.
- Hệ thống thiếu các tính năng thông báo về những thay đổi trong quy trình hoạt động.
- Sự chuyên môn hóa của hệ thống chưa cao, chưa có những tính năng đặc thù cho từng loại hình kinh doanh.
- Các chức năng hệ thống hiện giờ là đủ vận hành nhưng chưa có các tính năng nâng cao để tăng trải nghiệm người dùng hay tăng hiệu quả công việc.

Về mặt chức năng của hệ thống, tôi có thể nói là đã hoàn thành được những gì mà mình đã đề ra ban đầu. Tôi khá là hài lòng khi được hoàn thiện một ứng dụng có thể giúp ích cho nhiều người, cũng như là một sản phẩm thực thụ có thể sử dụng được trong thực tế. Đồng thời cũng là một trải nghiệm cá nhân có ích để tôi theo đuổi ngành công nghệ phần mềm thương mại. Tuy nhiên, vẫn còn một số điểm chưa được hoàn thiện, tôi sẽ cố gắng hoàn thiện trong tương lai. Đây cũng là cơ hội tốt để tôi ghi nhận và tích lũy sau này.

Về mặt kĩ thuật, tôi thấy mình đã học được một số kĩ năng mới sau khi hoàn thành đồ án này. Tôi đã có thể tự tạo ra một hệ thống hoàn chỉnh, từ frontend đến backend, từ cơ sở dữ liệu đến giao diện người dùng. Hơn nữa, tôi đã biết ứng dụng các nguyên lí nâng cao trong thiết kế một hệ thống thông tin vào lí thuyết mà tôi được dạy và quan sát trong thực tế. Những trải nghiệm và khó khăn khi áp dụng kiến trúc hệ thống như *Clean Architecture*, *Model-View-ViewModel* vào một ứng dụng thực tế cũng là một điều mà tôi rất trân trọng. Nó là một thước đo tốt cho sự ham học hỏi và tìm tòi cái mới. Cùng với đó, tôi cũng đã có thể tự tìm hiểu và áp dụng những công nghệ mới vào sản phẩm của mình. Ví dụ như lần đầu được học và sử dụng *.NET MAUI* trong thiết kế ứng dụng điện thoại. Trước đây tôi chỉ biết thiết kế website, nhưng giờ đây khả năng đưa trải nghiệm người dùng tới gần hơn với ứng dụng điện thoại cũng là một điều tôi rất vui mừng.

## 6.2 Hướng phát triển

Về định hướng tương lai của hệ thống, tôi nhận thấy rằng bài toán giúp các hộ kinh doanh vừa và nhỏ trong thời đại công nghệ 4.0 này là một bài toán thú vị và có nhiều tiềm năng. Dựa trên các phản hồi của người dùng trong giai đoạn hệ thống triển khai thử nghiệm, hệ thống sẽ cần thêm nhiều chức năng chuyên môn hóa hơn để phù hợp với nhu cầu người dùng. Thêm vào đó, hệ thống cũng cần có những tính năng mới để đạt hiệu quả hơn. Các tính năng đó có thể kể đến như:

- Tính năng quản lí chi phí, giúp người dùng hiệu quả hơn việc thu chi trong quá trình kinh doanh.
- Tính năng quản lí sản phẩm cần thêm những chức năng cảnh báo hạn sử dụng, tái đặt hàng, thanh lí hàng tồn kho.
- Tính năng quản lí đơn hàng, cho phép người dùng tạo sẵn đơn hàng để có thể sử dụng lại trong tương lai.
- Tính năng báo cáo, tổng hợp các thông tin quan trọng, tổng quan chỉ số hỗ trợ người dùng.
- Cải thiện độ tương tác của ứng dụng, giúp người dùng tiết kiệm trong việc nhập liệu.
- Chức năng in ấn, chia sẻ hóa đơn cũng là một tính năng được chú trọng.
- Khả năng sao lưu và phục hồi dữ liệu, giúp chống chịu với những rủi ro có thể xảy ra.
- Thông báo thời gian thực, cung cấp người dùng thông tin hoạt động kinh doanh đang diễn ra như thế nào.

Đây là những tính năng mà tôi nghĩ nếu có thêm vào hệ thống sẽ giúp ích rất nhiều cho người dùng. Tuy nhiên, để có thể hoàn thiện những tính năng này, tôi cần phải có thêm nhiều kiến thức và kinh nghiệm hơn nữa. Đây cũng là một trong những động lực để tôi tiếp tục học hỏi và nghiên cứu trong tương lai.

# TÀI LIỆU THAM KHẢO


[1] I. Outposts, *Monolith vs microservices: Which is better to choose?* [Online]. Available: https://itoutposts.com/blog/monolith-vs-microservices-which-is-better-to-choose/ (visited on 01/15/2024).

[2] R. R. Singh, *Creating asp.net applications with n-tier architecture*. [Online]. Available: https://www.codeproject.com/Articles/439688/Creating-ASP-NET-application-with-n-tier-architect/ (visited on 01/15/2024).

[3] J. Taylor, *Clean architecture with .net core: Getting started*. [Online]. Available: https://jasontaylor.dev/clean-architecture-getting-started/ (visited on 01/15/2024).

[4] Microsoft, *Model-view-viewmodel (mvvm)*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm/ (visited on 01/15/2024).

# PHỤ LỤC