

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC YERSIN ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THỰC TẬP**

**Phát triển SuperAdminBackend bằng EntityFramework**

**GVHD: Th.S Nguyễn Đức Tấn**

**SVTH:**

**MSSV:**

**Lớp: Công nghệ thông tin K19**

**Khóa học : 2022 – 2025**

Đà Lạt, 10 - 2025

**Lời Cảm Ơn**

Trong quá trình học tập môn Lập trình Web, em đã có cơ hội áp dụng kiến thức lý thuyết vào thực hành thông qua việc phát triển dự án này. Em xin gửi lời cảm ơn chân thành đến ThS. Nguyễn Đức Tấn – giảng viên hướng dẫn – vì đã tận tình hỗ trợ, cung cấp kiến thức nền tảng và góp ý chuyên môn kịp thời, giúp em hoàn thành báo cáo đúng tiến độ và đáp ứng yêu cầu học thuật.

Em cũng xin cảm ơn Trường Đại học Yersin Đà Lạt, đặc biệt là Khoa Công nghệ Thông tin, vì đã xây dựng chương trình đào tạo gắn liền với thực tiễn, giúp sinh viên như em tiếp cận các công nghệ web hiện đại và rèn luyện kỹ năng lập trình.

Qua dự án này, em đã học được cách kết nối lý thuyết với thực hành, xây dựng nền tảng vững chắc cho việc phát triển ứng dụng web trong tương lai.

Em xin chân thành cảm ơn!

**Mục Lục**

[I. TỔNG QUAN KIẾN THỨC HỌC PHẦN 5](#_Toc211808792)

[1.1 Giới thiệu về môn Lập trình Web 5](#_Toc211808793)

[1.2 Kiến thức cốt lõi đã học 5](#_Toc211808794)

[1.2.1. Kiến trúc Client-Server và Mô hình MVC 5](#_Toc211808795)

[1.2.2. Công nghệ phát triển Frontend 5](#_Toc211808796)

[1.2.3. Công nghệ phát triển Backend 6](#_Toc211808797)

[1.2.4. Bảo mật và xác thực 6](#_Toc211808798)

[1.2.5. Quản lý cơ sở dữ liệu 6](#_Toc211808799)

[1.3 Mục tiêu áp dụng trong dự án 7](#_Toc211808800)

[1.3.1. Phát triển ứng dụng web hoàn chỉnh 7](#_Toc211808801)

[1.3.2. Ứng dụng công nghệ hiện đại 7](#_Toc211808802)

[1.3.3. Rèn luyện kỹ năng chuyên môn 7](#_Toc211808803)

[1.3.4. Giải quyết bài toán thực tế 8](#_Toc211808804)

[1.3.5. Mô phỏng quy trình làm việc chuyên nghiệp 8](#_Toc211808805)

[1.3.6. Chuẩn bị cho môi trường làm việc thực tế 8](#_Toc211808806)

[II. GIỚI THIỆU ĐỀ TÀI 9](#_Toc211808807)

[2.1. Lý do chọn đề tài 9](#_Toc211808808)

[2.2. Mục tiêu đề tài 9](#_Toc211808809)

[2.3. Đối tượng sử dụng 9](#_Toc211808810)

[2.4. Phạm vi đề tài 9](#_Toc211808811)

[III. PHÂN TÍCH HỆ THỐNG 9](#_Toc211808812)

[3.1 Khảo sát bài toán thực tế 9](#_Toc211808813)

[3.1.1. Quy trình quản lý kho truyền thống 9](#_Toc211808814)

[3.1.2. Vấn đề tồn tại 10](#_Toc211808815)

[3.2. Yêu cầu chức năng 10](#_Toc211808816)

[3.2.1. Chức năng cơ bản 10](#_Toc211808817)

[3.2.2. Chức năng nâng cao 11](#_Toc211808818)

[3.3. Yêu cầu phi chức năng 11](#_Toc211808819)

[IV. THIẾT KẾ HỆ THỐNG 12](#_Toc211808820)

[4.1. Kiến trúc tổng thể 12](#_Toc211808821)

[4.1.1. Mô hình 3 lớp 12](#_Toc211808822)

[4.1.2. Sơ đồ tổng quát 12](#_Toc211808823)

[4.2. Sơ đồ Use Case 13](#_Toc211808824)

[4.2.1. Actor: Admin 13](#_Toc211808825)

[4.2.2. Actor: User 13](#_Toc211808826)

[4.2.3. Sơ đồ 14](#_Toc211808827)

[Sơ đồ 1: Đăng nhập và Phân quyền 14](#_Toc211808828)

[Sơ đồ 2: Quản lý sản phẩm 14](#_Toc211808829)

[Sơ đồ 3: Quản lý danh mục 14](#_Toc211808830)

[Sơ đồ 4: Quản lý nhân viên 15](#_Toc211808831)

[Sơ đồ 5: Báo cáo và Xuất/Nhập dữ liệu 15](#_Toc211808832)

[4.3. Thiết kế cơ sở dữ liệu 15](#_Toc211808833)

[4.3.1. Các bảng chính 15](#_Toc211808834)

[4.3.2. Quan hệ giữa các bảng 16](#_Toc211808835)

[4.3.3. Sơ đồ ERD (Entity-Relationship Diagram) 17](#_Toc211808836)

[5.1. Công nghệ sử dụng 18](#_Toc211808837)

[5.1.1. Backend 18](#_Toc211808838)

[5.1.2. Frontend 18](#_Toc211808839)

[5.1.3. Tools & Environment 18](#_Toc211808840)

[5.2. Cấu trúc project 19](#_Toc211808841)

[5.2.1. Backend (quanlikho) 19](#_Toc211808842)

[5.2.2. Frontend (quanlikhoFE) 19](#_Toc211808843)

[5.3. Các tính năng đã triển khai 20](#_Toc211808844)

[5.3.1. Authentication & Authorization 20](#_Toc211808845)

[5.3.2. Quản lý sản phẩm 20](#_Toc211808846)

[5.3.3. Quản lý danh mục 21](#_Toc211808847)

[5.3.4. Quản lý nhân viên 21](#_Toc211808848)

[5.3.5. Báo cáo và thống kê 22](#_Toc211808849)

[5.3.6. Giao diện người dùng 22](#_Toc211808850)

[VI. KẾT QUẢ VÀ ĐÁNH GIÁ 23](#_Toc211808851)

[6.1. Kết quả đạt được 23](#_Toc211808852)

[6.1.1. Chức năng hoàn thành 23](#_Toc211808853)

[6.1.2. Chất lượng kỹ thuật 26](#_Toc211808854)

[6.2. Hạn chế và khó khăn gặp phải 27](#_Toc211808855)

[6.2.1. Hạn chế của hệ thống 27](#_Toc211808856)

[6.2.2. Khó khăn trong quá trình thực hiện 27](#_Toc211808857)

[6.3. Bài học kinh nghiệm 27](#_Toc211808858)

[6.3.1. Kiến thức kỹ thuật 27](#_Toc211808859)

[6.3.2. Kỹ năng mềm 28](#_Toc211808860)

[6.4. Hướng phát triển 28](#_Toc211808861)

[6.4.1. Tính năng bổ sung 28](#_Toc211808862)

[6.4.2. Cải tiến kỹ thuật 28](#_Toc211808863)

[6.4.3. Tối ưu hóa 29](#_Toc211808864)

[VII. KẾT LUẬN 29](#_Toc211808865)

[7.1. Tổng kết đề tài 29](#_Toc211808866)

[7.2. Kiến thức đã áp dụng 29](#_Toc211808867)

[7.2.1. Kiến thức lý thuyết 29](#_Toc211808868)

[7.2.2. Kỹ năng thực hành 30](#_Toc211808869)

[7.3. Ý nghĩa và giá trị 30](#_Toc211808870)

[7.3.1. Giá trị học tập 30](#_Toc211808871)

[7.3.2. Giá trị thực tiễn 30](#_Toc211808872)

[7.4. Lời cảm ơn cuối 31](#_Toc211808873)

**Danh Mục Hình Ảnh**

|  |  |  |
| --- | --- | --- |
| **Hình** | **Mô tả** | **Trang** |
| Hình 6.1.1. | Tổng thể giao diện trang web | **24** |
| Hình 6.1.1.2 | Giao diện chức năng quản lý sản phẩm (danh sách, tìm kiếm, lọc) | **24** |
| Hình 6.1.1.3 | Giao diện chức năng thêm sản phẩm (form nhập liệu) | **25** |
| Hình 6.1.1.4 | Giao diện chức năng quản lý danh mục | **25** |
| Hình 6.1.1.5 | Giao diện chức năng thêm danh mục | **26** |
| Hình 6.1.1.6 | Giao diện chức năng quản lý nhân viên (danh sách, phân quyền) | **26** |

**Danh Mục Các Từ Viết Tắt**

|  |  |
| --- | --- |
| **Từ viết tắt** | **Ý nghĩa đầy đủ** |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| DTO | Data Transfer Object |
| ERD | Entity-Relationship Diagram |
| HTTP | HyperText Transfer Protocol |
| HTTPS | HyperText Transfer Protocol Secure |
| JWT | JSONWeb Token |
| LINQ | Languae Integrated Query |
| MVC | Model-View-Controller |
| ORM | Object-Relational Mapping |
| REST | Representational State Transfer |

# CHƯƠNG 1: TÌM HIỂU VỀ LẬP TRÌNH WEB API (CƠ SỞ LÝ THUYẾT)

## 1.1. Khái niệm

**Giao diện lập trình ứng dụng web** (*Web Application Programming Interface* – *Web API*) là một **giao diện trung gian** cho phép các hệ thống phần mềm khác nhau **tương tác và trao đổi dữ liệu** thông qua mạng **World Wide Web**.  
Về bản chất, Web API định nghĩa một **tập hợp các quy tắc, giao thức (chủ yếu là HTTP/HTTPS)** và **định dạng thông điệp** mà các ứng dụng có thể sử dụng để **yêu cầu và thực thi các chức năng từ máy chủ từ xa**.

  Khái niệm “**giao diện**” trong Web API không đề cập đến giao diện người dùng đồ họa (GUI), mà là **một giao diện lập trình**, hay còn gọi là **một hợp đồng kỹ thuật** (*technical contract*) được xác định rõ ràng. Hợp đồng này quy định chi tiết cách thức **một hệ thống bên ngoài (client)** có thể **tương tác** với **hệ thống cung cấp dịch vụ (server)**.

**Các thành phần chính của hợp đồng API:**

* **Endpoints (Điểm cuối):**  
  Là các định danh tài nguyên thống nhất (*Uniform Resource Identifiers* – URIs) xác định các tài nguyên hoặc dịch vụ cụ thể.  
  Ví dụ: endpoint /api/products có thể đại diện cho tài nguyên là **tập hợp các sản phẩm**.
* **HTTP Methods (Các phương thức HTTP):**  
  Là các **động từ chỉ định hành động** cần thực hiện trên tài nguyên.  
  Các phương thức phổ biến gồm:
  + **GET:** Truy xuất tài nguyên.
  + **POST:** Tạo mới tài nguyên.
  + **PUT / PATCH:** Cập nhật tài nguyên hiện có.
  + **DELETE:** Xóa tài nguyên.
* **Data Formats (Định dạng dữ liệu):**  
  Là **quy cách biểu diễn dữ liệu** được trao đổi.  
  Định dạng phổ biến nhất là **JSON (JavaScript Object Notation)** do tính **tinh gọn, dễ phân tích cú pháp và độc lập ngôn ngữ lập trình**.

  Web API đóng vai trò **trụ cột trong các kiến trúc phần mềm hiện đại** như **Service-Oriented Architecture (SOA)** và **Microservices**.  
Bằng cách **trừu tượng hóa và đóng gói logic nghiệp vụ của backend**, Web API cho phép **tách rời lớp trình bày (presentation layer – frontend)** khỏi **lớp xử lý nghiệp vụ (business logic layer – backend)**.

**Lợi ích của Web API:**

* **Tính độc lập trong phát triển:**  
  Cho phép các nhóm **frontend** và **backend** phát triển **song song**, chỉ cần tuân thủ **đặc tả API** đã thống nhất.
* **Khả năng tương tác và tái sử dụng:**  
  Một **backend API** có thể phục vụ **nhiều loại client** khác nhau như ứng dụng web, di động, hoặc hệ thống bên thứ ba.  
  Điều này giúp **tối đa hóa khả năng tái sử dụng logic nghiệp vụ**.
* **Tính linh hoạt và khả năng mở rộng:**  
  Các thành phần frontend và backend có thể **triển khai, bảo trì và mở rộng quy mô độc lập**, giúp tăng hiệu suất và độ linh hoạt của hệ thống.

**Kiểu kiến trúc phổ biến: RESTful API**

  Một trong những kiểu kiến trúc phổ biến nhất của Web API là **REST (Representational State Transfer)**, tập trung vào việc **biểu diễn trạng thái của tài nguyên** thông qua các phương thức HTTP tiêu chuẩn.  
Các API tuân thủ nguyên tắc REST được gọi là **RESTful API**, và đây cũng là **phương pháp được áp dụng trong dự án này**.

## 1.2. Đặc điểm

* **Giao thức không trạng thái (Stateless):**  
  Mỗi yêu cầu (request) từ client đến server **chứa đầy đủ thông tin cần thiết** để server hiểu và xử lý, **không lưu trữ trạng thái của client** giữa các yêu cầu.
* **Dựa trên HTTP/HTTPS:**  
  Sử dụng các phương thức HTTP như **GET, POST, PUT, DELETE** để thực hiện các thao tác **CRUD** (Create – Read – Update – Delete).
* **Định dạng dữ liệu linh hoạt:**  
  Hỗ trợ nhiều định dạng dữ liệu, phổ biến nhất là **JSON** và **XML**.  
  Trong đó, **JSON** được ưu tiên do **gọn nhẹ và dễ xử lý**.
* **Khả năng mở rộng:**  
  Dễ dàng **mở rộng và tích hợp** với nhiều loại client khác nhau như **web, mobile, hoặc desktop**.

## 1.3. Nguyên lý hoạt động

Quy trình hoạt động cơ bản của Web API gồm ba bước chính:

1. **Client gửi yêu cầu (Request):**  
   Ứng dụng client (ví dụ: trình duyệt web hoặc ứng dụng di động) **tạo một yêu cầu HTTP** đến **endpoint** cụ thể của Web API.  
   Yêu cầu bao gồm **phương thức HTTP**, **URL**, **header**, và có thể có **body** (chứa dữ liệu gửi đi – thường ở định dạng JSON).
2. **Server xử lý yêu cầu:**  
   Server **nhận yêu cầu**, **xác thực** (nếu cần), **phân tích**, và **thực thi logic nghiệp vụ** tương ứng (ví dụ: truy vấn cơ sở dữ liệu, xử lý dữ liệu, tính toán,…).
3. **Server gửi phản hồi (Response):**  
   Sau khi xử lý xong, server tạo **phản hồi HTTP** gửi lại cho client.  
   Phản hồi bao gồm **mã trạng thái (status code)** như *200 OK*, *404 Not Found*, *500 Internal Server Error* và phần **body** (thường chứa dữ liệu ở định dạng JSON).

## 1.4. Công nghệ triển khai ASP.NET Core API

  Dự án sử dụng **ASP.NET Core Web API** trên nền tảng **.NET 8**, một framework mạnh mẽ, **mã nguồn mở**, **đa nền tảng** của **Microsoft** để xây dựng các ứng dụng web hiện đại, bao gồm Web API.

**Công nghệ sử dụng:**

* **Ngôn ngữ:** C#
* **Framework:** ASP.NET Core 8
* **Cơ sở dữ liệu:** SQL Server (tương tác thông qua **Entity Framework Core**)
* **Xác thực:** Sử dụng **JWT (JSON Web Tokens)** để **bảo vệ các endpoint API**

## 1.5. Công nghệ triển khai để khai thác API (Consuming API)

  Để khai thác (consume) Web API đã xây dựng, dự án sử dụng một ứng dụng **ASP.NET Core MVC** có tên **quanlikhoFE**.

**Cách thức hoạt động:**

* **Tầng giao tiếp:**  
  File **ApiService.cs** trong dự án *quanlikhoFE* đóng vai trò **trung gian**, chịu trách nhiệm **gửi các yêu cầu HTTP (GET, POST, PUT, DELETE)** đến các endpoint của Web API và **nhận dữ liệu trả về**.
* **Tầng hiển thị:**  
  Dữ liệu sau khi được nhận sẽ được **hiển thị cho người dùng** thông qua các **View (.cshtml)**, được xây dựng bằng **Razor syntax** trong ASP.NET Core MVC.

# CHƯƠNG 2: TRIỂN KHAI ỨNG DỤNG WEB API

## 2.1. Phát biểu bài toán ứng dụng

  Ứng dụng được xây dựng là **“Hệ thống Quản lý Kho” (Warehouse Management System)**.  
Mục tiêu của hệ thống là hỗ trợ doanh nghiệp **quản lý các hoạt động nhập, xuất và tồn kho** một cách **hiệu quả, chính xác và tự động hóa**, giúp tiết kiệm thời gian và giảm thiểu sai sót trong quản lý.

**Ý nghĩa của hệ thống:**

* Tối ưu hóa quy trình quản lý kho.
* Cung cấp dữ liệu **thời gian thực** về lượng hàng tồn kho.
* Hỗ trợ **ra quyết định kinh doanh** dựa trên các báo cáo thống kê.
* Giảm thiểu sai sót so với phương pháp quản lý thủ công.

## 2.2. Phân tích yêu cầu của ứng dụng

### 2.2.1. Yêu cầu chức năng (Functional Requirements)

#### 2.2.1.1. Quản lý xác thực và phân quyền

* Cho phép **đăng nhập bằng username và password**.
* **Xác thực người dùng** bằng **JWT token**.
* **Phân quyền** theo vai trò (Admin, User).
* **Đăng xuất** và **tự động hết hạn phiên làm việc (session)**.
* **Lưu trữ thông tin đăng nhập** tạm thời trong session để tái sử dụng.

#### 2.2.1.2. Quản lý danh mục sản phẩm

* Thêm **danh mục sản phẩm mới** (tên, mô tả).
* Xem **danh sách tất cả danh mục**.
* Cập nhật thông tin danh mục (tên, mô tả).
* Xóa danh mục.
* Hiển thị **số lượng sản phẩm** trong từng danh mục.

#### 2.2.1.3. Quản lý sản phẩm

* Thêm sản phẩm mới (tên, danh mục, số lượng, đơn giá).
* Xem danh sách sản phẩm với đầy đủ thông tin chi tiết.
* **Tìm kiếm sản phẩm** theo tên.
* **Lọc sản phẩm** theo danh mục.
* Cập nhật thông tin sản phẩm (tên, danh mục, số lượng, đơn giá).
* Xóa sản phẩm.
* Hiển thị **tổng giá trị tồn kho** = *số lượng × đơn giá*.

#### 2.2.1.4. Quản lý nhà cung cấp

* Thêm **nhà cung cấp mới** với thông tin đầy đủ (tên, địa chỉ, số điện thoại, email).
* Xem danh sách nhà cung cấp.
* Cập nhật thông tin nhà cung cấp.
* Xóa nhà cung cấp (kiểm tra ràng buộc với phiếu nhập).
* **Tìm kiếm nhà cung cấp** theo tên hoặc địa chỉ.

#### 2.2.1.5. Quản lý kho hàng

* Thêm **kho hàng mới** (tên, địa chỉ, diện tích, mô tả).
* Xem danh sách kho hàng.
* Cập nhật thông tin kho.
* Xóa kho (kiểm tra tồn kho trước khi xóa).
* Quản lý **vị trí lưu trữ** và **mô tả chi tiết** trong kho.

#### 2.2.1.6. Quản lý nhập kho

* Tạo **phiếu nhập kho mới** từ nhà cung cấp.
* Chọn **nhà cung cấp** và **kho nhập**.
* Thêm/xóa **chi tiết sản phẩm nhập** (sản phẩm, số lượng, đơn giá).
* Tính **tổng giá trị phiếu nhập** tự động.
* Xác nhận phiếu nhập → **tự động cập nhật tồn kho**.
* Xem **danh sách** và **chi tiết phiếu nhập kho**.
* In phiếu nhập kho (HTML, PDF).
* Hủy phiếu nhập (nếu chưa xác nhận).

#### 2.2.1.7. Quản lý xuất kho

* Tạo **phiếu xuất kho mới**.
* Chọn **kho xuất** và thêm chi tiết sản phẩm.
* **Kiểm tra số lượng tồn kho** trước khi xuất.
* Xác nhận phiếu xuất → **cập nhật tồn kho**.
* Xem danh sách, chi tiết, và lịch sử phiếu xuất.
* In phiếu xuất kho (HTML, PDF, Excel).
* Hủy phiếu xuất (nếu chưa xác nhận).

#### 2.2.1.8. Quản lý tồn kho

* Xem **tồn kho theo sản phẩm** và **theo kho hàng**.
* **Cảnh báo hết hàng** khi dưới mức tối thiểu.
* **Chuyển hàng giữa các kho** khác nhau.
* **Kiểm kê định kỳ** để đối chiếu số liệu.
* **Điều chỉnh tồn kho** (tăng/giảm) khi cần thiết.

#### 2.2.1.9. Báo cáo và thống kê

* **Dashboard tổng quan** hệ thống (sản phẩm, kho, người dùng, giá trị hàng tồn).
* Báo cáo **tồn kho tổng hợp** theo thời gian.
* Báo cáo **nhập – xuất** theo ngày, tháng, năm.
* **Thống kê sản phẩm bán chạy / tồn đọng**.
* Báo cáo theo **nhà cung cấp** hoặc **danh mục sản phẩm**.
* **Xuất báo cáo ra Excel hoặc PDF**.
* **Import / Export dữ liệu** (sản phẩm, danh mục) qua file CSV.

#### 2.2.1.10. Quản lý người dùng

* Xem danh sách người dùng.
* Thêm người dùng mới.
* Phân quyền (Admin / User).
* Cập nhật hoặc xóa người dùng.
* Quản lý thông tin đăng nhập và trạng thái hoạt động.

**2.2.2 Yêu cầu phi chức năng (Non-Functional Requirements)**

Các yêu cầu phi chức năng là những tiêu chí không trực tiếp liên quan đến tính năng của hệ thống, nhưng ảnh hưởng đến **hiệu quả, độ tin cậy, bảo mật và trải nghiệm người dùng**. Dưới đây là các yêu cầu phi chức năng chính của hệ thống “Quản lý Kho”:

**2.2.2.1 Bảo mật (Security)**

* Hệ thống sử dụng cơ chế **xác thực bằng JWT (JSON Web Token)** để đảm bảo chỉ người dùng hợp lệ mới có quyền truy cập vào các tài nguyên được bảo vệ.
* Thực hiện **phân quyền người dùng** (Admin, User) giúp kiểm soát quyền hạn đối với các hành động như thêm, sửa, xóa dữ liệu.
* Mã hóa thông tin nhạy cảm (mật khẩu, token) trước khi lưu trữ trong cơ sở dữ liệu.
* Áp dụng giao thức **HTTPS** để mã hóa dữ liệu trong quá trình truyền giữa client và server.
* Ngăn chặn các cuộc tấn công phổ biến như **SQL Injection**, **Cross-Site Scripting (XSS)** và **Cross-Site Request Forgery (CSRF)** thông qua việc kiểm tra và xác thực đầu vào.

**2.2.2.2 Hiệu năng (Performance)**

* API được tối ưu hóa để xử lý nhanh chóng các yêu cầu từ client, hạn chế độ trễ (latency) trong phản hồi.
* Sử dụng **caching** cho các yêu cầu lặp lại, đặc biệt là dữ liệu ít thay đổi như danh mục sản phẩm hoặc danh sách kho.
* Hệ thống có khả năng xử lý đồng thời nhiều yêu cầu từ người dùng (concurrent requests) mà không làm giảm hiệu năng.
* Tối ưu truy vấn cơ sở dữ liệu bằng cách sử dụng **chỉ mục (index)** và giới hạn dữ liệu trả về thông qua **phân trang (pagination)**.

**2.2.2.3 Khả năng sử dụng (Usability)**

* API được thiết kế tuân theo chuẩn **RESTful**, dễ hiểu, dễ tích hợp và tương thích với nhiều loại client (web, mobile, ứng dụng bên thứ ba).
* Cấu trúc endpoint rõ ràng, nhất quán, có tài liệu mô tả (API Documentation) hỗ trợ cho việc phát triển frontend hoặc tích hợp hệ thống.
* Thông báo lỗi và mã trạng thái HTTP (HTTP Status Codes) được trả về chính xác, giúp lập trình viên dễ dàng xử lý và gỡ lỗi.

**2.2.2.4 Độ tin cậy (Reliability)**

* Hệ thống đảm bảo hoạt động ổn định và sẵn sàng cao (high availability).
* Thực hiện cơ chế **sao lưu dữ liệu định kỳ (backup)** và **phục hồi sau sự cố (recovery)**.
* Khi có lỗi xảy ra, hệ thống ghi log chi tiết để hỗ trợ giám sát và xử lý sự cố.
* Cơ sở dữ liệu được thiết kế đảm bảo tính toàn vẹn dữ liệu (data integrity) thông qua các ràng buộc khóa chính, khóa ngoại, và kiểm tra logic nghiệp vụ.

## 2.3 Thiết kế cơ sở dữ liệu

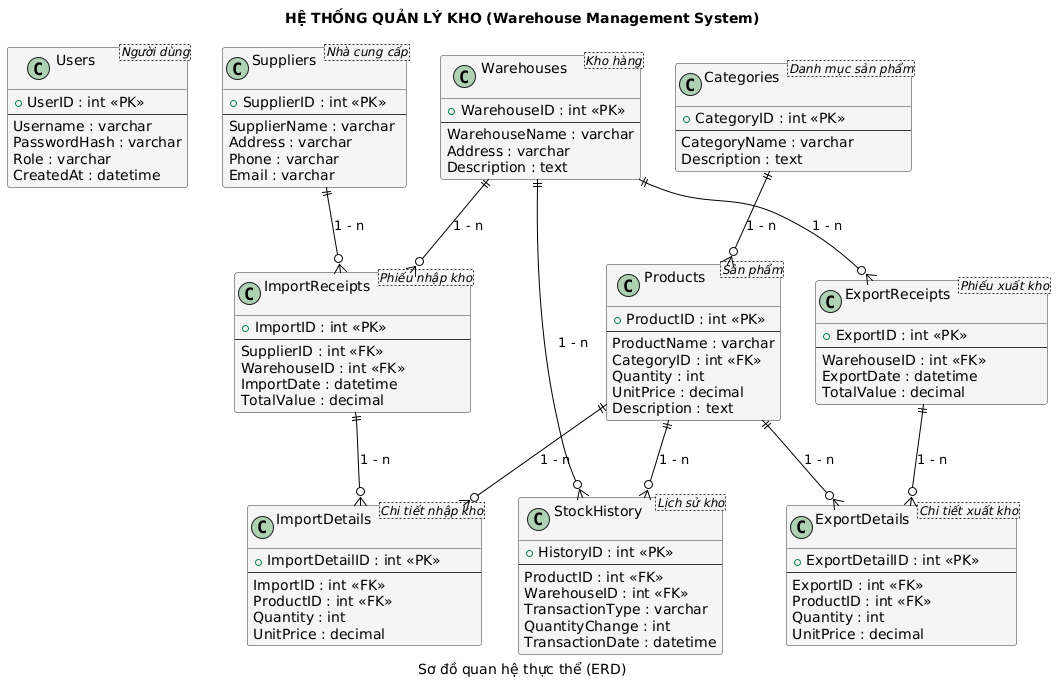
Cơ sở dữ liệu được thiết kế theo mô hình **quan hệ (Relational Database)**, đảm bảo tính chặt chẽ trong việc lưu trữ và truy xuất dữ liệu cho các chức năng quản lý kho, sản phẩm, người dùng và giao dịch.

### 2.3.1 Mô tả các bảng chính

|  |  |  |
| --- | --- | --- |
| **Tên bảng** | **Mô tả** | **Các trường chính** |
| **Users** | Lưu thông tin tài khoản người dùng | UserID, Username, PasswordHash, Role, CreatedAt |
| **Categories** | Lưu danh mục sản phẩm | CategoryID, CategoryName, Description |
| **Products** | Lưu thông tin sản phẩm | ProductID, ProductName, CategoryID, Quantity, UnitPrice, Description |
| **Suppliers** | Lưu thông tin nhà cung cấp | SupplierID, SupplierName, Address, Phone, Email |
| **Warehouses** | Lưu thông tin kho hàng | WarehouseID, WarehouseName, Address, Description |
| **ImportReceipts** | Quản lý phiếu nhập kho | ImportID, SupplierID, WarehouseID, ImportDate, TotalValue |
| **ImportDetails** | Chi tiết các sản phẩm nhập kho | ImportDetailID, ImportID, ProductID, Quantity, UnitPrice |
| **ExportReceipts** | Quản lý phiếu xuất kho | ExportID, WarehouseID, ExportDate, TotalValue |
| **ExportDetails** | Chi tiết các sản phẩm xuất kho | ExportDetailID, ExportID, ProductID, Quantity, UnitPrice |
| **StockHistory** | Lưu lịch sử nhập/xuất kho | HistoryID, ProductID, WarehouseID, TransactionType, QuantityChange, TransactionDate |

### 2.3.2 Mối quan hệ giữa các bảng

* **Users** có quan hệ *1-nhiều* với các bảng log hoạt động (nếu có).
* **Categories – Products**: Quan hệ *1-nhiều* (một danh mục có nhiều sản phẩm).
* **Suppliers – ImportReceipts**: Quan hệ *1-nhiều* (một nhà cung cấp có nhiều phiếu nhập).
* **Warehouses – ImportReceipts / ExportReceipts**: Quan hệ *1-nhiều* (một kho có thể có nhiều phiếu nhập/xuất).
* **ImportReceipts – ImportDetails**: Quan hệ *1-nhiều*.
* **ExportReceipts – ExportDetails**: Quan hệ *1-nhiều*.
* **Products – ImportDetails / ExportDetails / StockHistory**: Quan hệ *1-nhiều* (một sản phẩm có thể xuất hiện trong nhiều giao dịch).



## 2.4 Thiết kế giao diện người dùng

Phần này mô tả thiết kế giao diện (UI) cho ứng dụng **Hệ thống Quản lý Kho** được triển khai bằng **ASP.NET Core MVC**. Mục tiêu là tạo giao diện trực quan, dễ sử dụng, tối ưu cho nhiều thiết bị (responsive) và hỗ trợ các thao tác nghiệp vụ (CRUD, in ấn, xuất báo cáo).

### 2.4.1 Layout chính

Giao diện được tổ chức theo mô hình chuẩn gồm **Header**, **Sidebar**, **Main Content Area** và **Footer**:

* **Header / Navigation**
  + Logo + tên ứng dụng ở góc trái.
  + Menu chính (Dashboard, Quản lý, Báo cáo) — có thể là menu ngang hoặc collapsed trên thiết bị nhỏ.
  + Thông tin người dùng (avatar, tên) kèm dropdown menu (Profile, Settings, Logout).
  + Nút đăng xuất rõ ràng.
* **Sidebar Menu**
  + Dạng vertical navigation, hỗ trợ mở rộng/thu gọn (collapsible).
  + Các mục chính:
    - Dashboard
    - Quản lý Danh mục
    - Quản lý Sản phẩm
    - Quản lý Nhà cung cấp
    - Quản lý Kho hàng
    - Nhập kho
    - Xuất kho
    - Báo cáo & Thống kê
* **Main Content Area**
  + **Breadcrumb** (điều hướng phân cấp) để người dùng biết vị trí.
  + Tiêu đề trang (page title) + action buttons (Add New, Export, Filter, v.v.).
  + Khu vực nội dung hiển thị bảng (tables), form, hoặc biểu đồ tùy theo trang.
  + Các điều khiển phân trang (pagination), sắp xếp (sort) và lọc (filter).
* **Footer**
  + Thông tin bản quyền.
  + Các links hữu ích (Help, Docs).
  + Thông tin phiên bản ứng dụng.

Gợi ý UI: sử dụng layout grid (container → row → col) của Bootstrap để dễ quản lý responsive và alignment.

### 2.4.2 Các trang chính

Mô tả từng trang/case chính trong hệ thống cùng các thành phần UI quan trọng:

* **Trang Đăng nhập (/Auth/Login)**
  + Form login: username, password.
  + Remember me checkbox.
  + Hiện validation messages rõ ràng (invalid credentials, required fields).
  + Responsive (hiển thị tốt trên mobile).
  + Hỗ trợ hiển thị/ẩn mật khẩu (eye toggle).
* **Dashboard (/Home/Index)**
  + Các **cards** tóm tắt: tổng số sản phẩm, tổng giá trị tồn kho, số kho, số nhà cung cấp.
  + Biểu đồ (chart) tồn kho theo kho (bar/stacked bar).
  + Biểu đồ nhập/xuất theo tháng (line/area chart).
  + Danh sách hoạt động gần đây (activity feed).
  + Quick actions (Create Import, Create Export, Add Product).
* **Trang Quản lý Sản phẩm (/Products)**
  + DataTable nâng cao: search, sort, filter, pagination.
  + CRUD: Create (form), Read (table/detail), Update (modal hoặc page), Delete (confirm dialog).
  + Modal dialogs cho quick edit / quick view.
  + Bulk actions (delete, export, change category).
  + Export (CSV/Excel/PDF).
* **Trang Quản lý Danh mục (/Categories)**
  + Danh sách đơn giản với inline editing (edit tên, mô tả).
  + Hỗ trợ drag & drop để reorder (nếu cần).
  + Thêm/xóa/cập nhật nhanh.
* **Trang Quản lý Kho hàng (/Warehouses)**
  + List view với thông tin kho (tên, địa chỉ, diện tích, trạng thái).
  + Stock summary per warehouse (tổng Sản phẩm, tổng giá trị).
  + Quản lý vị trí/zone trong kho (nếu hệ thống hỗ trợ vị trí chi tiết).
* **Trang Nhập kho (/StockInbounds)**
  + Giao diện master-detail: header (thông tin phiếu), detail (dòng sản phẩm).
  + Dynamic product lines (thêm/xóa dòng).
  + Tính toán realtime (tổng số lượng, tổng giá trị).
  + Nút In / Xuất PDF.
* **Trang Xuất kho (/StockOutbounds)**
  + Tương tự trang nhập nhưng có thêm validation tồn kho trước khi xác nhận.
  + Hỗ trợ barcode scanning (nếu tích hợp hardware).
  + Hỗ trợ batch processing cho nhiều đơn xuất cùng lúc.
* **Trang Báo cáo (/Reports)**
  + Giao diện chọn báo cáo (report templates).
  + Date range picker (day/week/month/custom).
  + Tùy chọn export (PDF, Excel).
  + Chart visualizations tương tác (zoom/filter).

### 2.4.3 Tính năng giao diện

Các tính năng UI nhằm nâng cao trải nghiệm người dùng, độ phản hồi và khả năng tương tác:

* **Responsive Design**
  + Sử dụng **Bootstrap 5** (hoặc framework tương đương).
  + Áp dụng **mobile-first**: tối ưu cho điện thoại, sau đó cho tablet/desktop.
  + Touch-friendly controls (buttons lớn, spacing hợp lý).
* **Interactive Elements**
  + **Modal dialogs** cho thao tác nhanh (create/edit).
  + **Toast notifications** hiển thị kết quả thao tác (success, error, info).
  + **Loading spinners / skeletons** khi chờ tải dữ liệu.
  + **Progress bars** cho các thao tác dài (import/export).
  + **Tooltips** và help text cho các control phức tạp.
* **Accessibility & UX**
  + Đảm bảo contrast, kích thước font hợp lý, hỗ trợ bàn phím (tab navigation).
  + Messages lỗi rõ ràng, hướng dẫn sửa lỗi.
  + Thời gian phản hồi tối ưu, giảm thông báo spam.

## 2.5 Thiết kế các thành phần API và Consuming API

Hệ thống được chia thành hai dự án chính:

* **quanlikho:** Dự án **ASP.NET Core Web API** phục vụ xử lý nghiệp vụ và cung cấp dữ liệu cho các client.
* **quanlikhoFE:** Dự án **ASP.NET Core MVC** đóng vai trò giao diện người dùng, sử dụng các API được cung cấp từ hệ thống backend.

### 2.5.1 Phần Web API (Dự án quanlikho)

Kiến trúc của dự án Web API được thiết kế theo mô hình nhiều lớp (multi-layer architecture), bao gồm:

**a. Models Layer**

Lưu trữ các lớp thực thể (Entity classes) tương ứng với các bảng trong cơ sở dữ liệu.  
Các thực thể được định nghĩa kèm theo các ràng buộc dữ liệu (Data Annotations) và thuộc tính điều hướng (Navigation Properties) để thể hiện quan hệ giữa các bảng.  
Ngoài ra, các trường kiểm soát thay đổi như CreatedAt, UpdatedAt được thêm vào để hỗ trợ truy vết dữ liệu.

**b. DTOs Layer**

Lớp trung gian dùng để trao đổi dữ liệu giữa client và server.  
Bao gồm các lớp:

* **Request DTOs:** Xác định cấu trúc dữ liệu đầu vào.
* **Response DTOs:** Xác định cấu trúc dữ liệu trả về cho client.  
  Việc ánh xạ giữa các lớp Entity và DTO được thực hiện tự động thông qua công cụ **AutoMapper**.

**c. Data Layer**

Chứa lớp **AppDbContext** để quản lý truy cập cơ sở dữ liệu thông qua **Entity Framework Core**.  
Lớp này khai báo các DbSet tương ứng với các bảng, cấu hình quan hệ, và hỗ trợ thao tác với dữ liệu (migration, seeding,…).

**d. Repository Layer**

Là tầng trung gian giữa Data Layer và Service Layer.  
Sử dụng mô hình **Generic Repository** để cung cấp các thao tác chung (CRUD), đồng thời có thể mở rộng thành các repository đặc thù cho từng nghiệp vụ.  
Tầng này giúp tách biệt logic truy cập dữ liệu ra khỏi nghiệp vụ chính của ứng dụng.

**e. Service Layer**

Thực hiện các **logic nghiệp vụ (business logic)** của hệ thống, bao gồm xử lý dữ liệu, xác thực nghiệp vụ, và điều phối các thao tác liên quan đến nhiều bảng.  
Tầng này đồng thời quản lý giao dịch (transaction) và đảm bảo tính toàn vẹn dữ liệu.

**f. Controller Layer**

Cung cấp các **API endpoint** theo mô hình **RESTful**.  
Các controller chịu trách nhiệm nhận yêu cầu từ client, gọi đến các service tương ứng để xử lý và trả về phản hồi với mã trạng thái HTTP thích hợp.

### 2.5.2 Phần Consuming API (Dự án quanlikhoFE)

Phần frontend được phát triển theo mô hình **ASP.NET Core MVC**, đóng vai trò giao tiếp với API backend để hiển thị và thao tác dữ liệu.

**a. ApiService**

Đóng vai trò lớp giao tiếp với hệ thống Web API.  
Sử dụng **HttpClient** để gửi và nhận dữ liệu qua các phương thức HTTP.  
ApiService cũng đảm nhiệm việc quản lý token JWT, xử lý lỗi và cấu hình URL gốc cho các yêu cầu.

**b. Controllers (MVC Layer)**

Các controller trong dự án MVC chịu trách nhiệm gọi **ApiService**, xử lý dữ liệu nhận được, và truyền tới các View thông qua **ViewModel**.  
Chúng cũng thực hiện quản lý session, điều hướng người dùng và xử lý tương tác từ giao diện.

**c. ViewModels**

Là lớp trung gian giữa Controller và View, giúp định nghĩa cấu trúc dữ liệu hiển thị.  
ViewModel chứa các thuộc tính có gắn các ràng buộc kiểm tra (validation) và định dạng hiển thị.

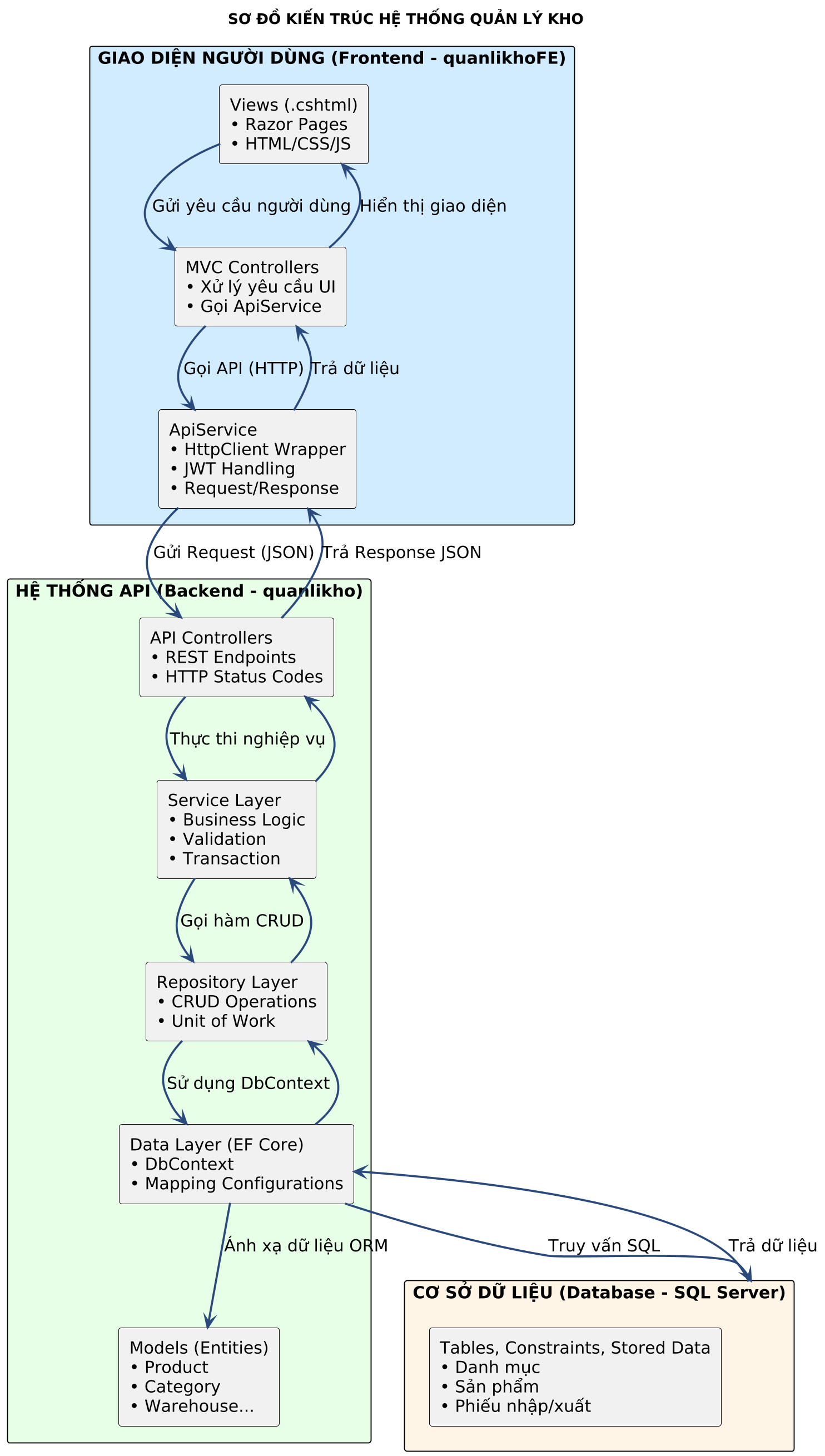
**d. Views**

Gồm các trang giao diện được xây dựng bằng **Razor Templates**.  
Views được thiết kế theo hướng **tái sử dụng (Partial Views)** và kế thừa **Layout chung** để đảm bảo tính thống nhất.  
Các trang hỗ trợ **kiểm tra dữ liệu phía client** và có thể tích hợp với các thư viện giao diện như Bootstrap hoặc jQuery.

**e. Middleware**

Các thành phần trung gian (middleware) đảm nhận các chức năng:

* Xác thực người dùng (Authentication Middleware).
* Ghi log và xử lý lỗi (Logging & Error Handling).
* Cho phép truy cập từ các domain khác (CORS Handling).  
  Các middleware này giúp tăng tính bảo mật và khả năng mở rộng cho ứng dụng.



### 2.5.3. Kiến trúc API Endpoints

Authentication Endpoints:

• POST /api/auth/login - Đăng nhập

• POST /api/auth/logout - Đăng xuất

• GET /api/auth/profile - Thông tin user

• PUT /api/auth/profile - Cập nhật profile

Product Management:

• GET /api/products - Danh sách sản phẩm

• GET /api/products/{id} - Chi tiết sản phẩm

• POST /api/products - Tạo sản phẩm mới

• PUT /api/products/{id} - Cập nhật sản phẩm

• DELETE /api/products/{id} - Xóa sản phẩm

• GET /api/products/search?q={query} - Tìm kiếm

Category Management:

• GET /api/categories - Danh sách danh mục

• POST /api/categories - Tạo danh mục

• PUT /api/categories/{id} - Cập nhật danh mục

• DELETE /api/categories/{id} - Xóa danh mục

Supplier Management:

• GET /api/suppliers - Danh sách nhà cung cấp

• POST /api/suppliers - Tạo nhà cung cấp

• PUT /api/suppliers/{id} - Cập nhật nhà cung cấp

• DELETE /api/suppliers/{id} - Xóa nhà cung cấp

Warehouse Management:

• GET /api/warehouses - Danh sách kho

• POST /api/warehouses - Tạo kho

• PUT /api/warehouses/{id} - Cập nhật kho

• DELETE /api/warehouses/{id} - Xóa kho

Stock Inbound:

• GET /api/stockinbounds - Danh sách phiếu nhập

• GET /api/stockinbounds/{id} - Chi tiết phiếu nhập

• POST /api/stockinbounds - Tạo phiếu nhập

• PUT /api/stockinbounds/{id}/confirm - Xác nhận phiếu nhập

Stock Outbound:

• GET /api/stockoutbounds - Danh sách phiếu xuất

• GET /api/stockoutbounds/{id} - Chi tiết phiếu xuất

• POST /api/stockoutbounds - Tạo phiếu xuất

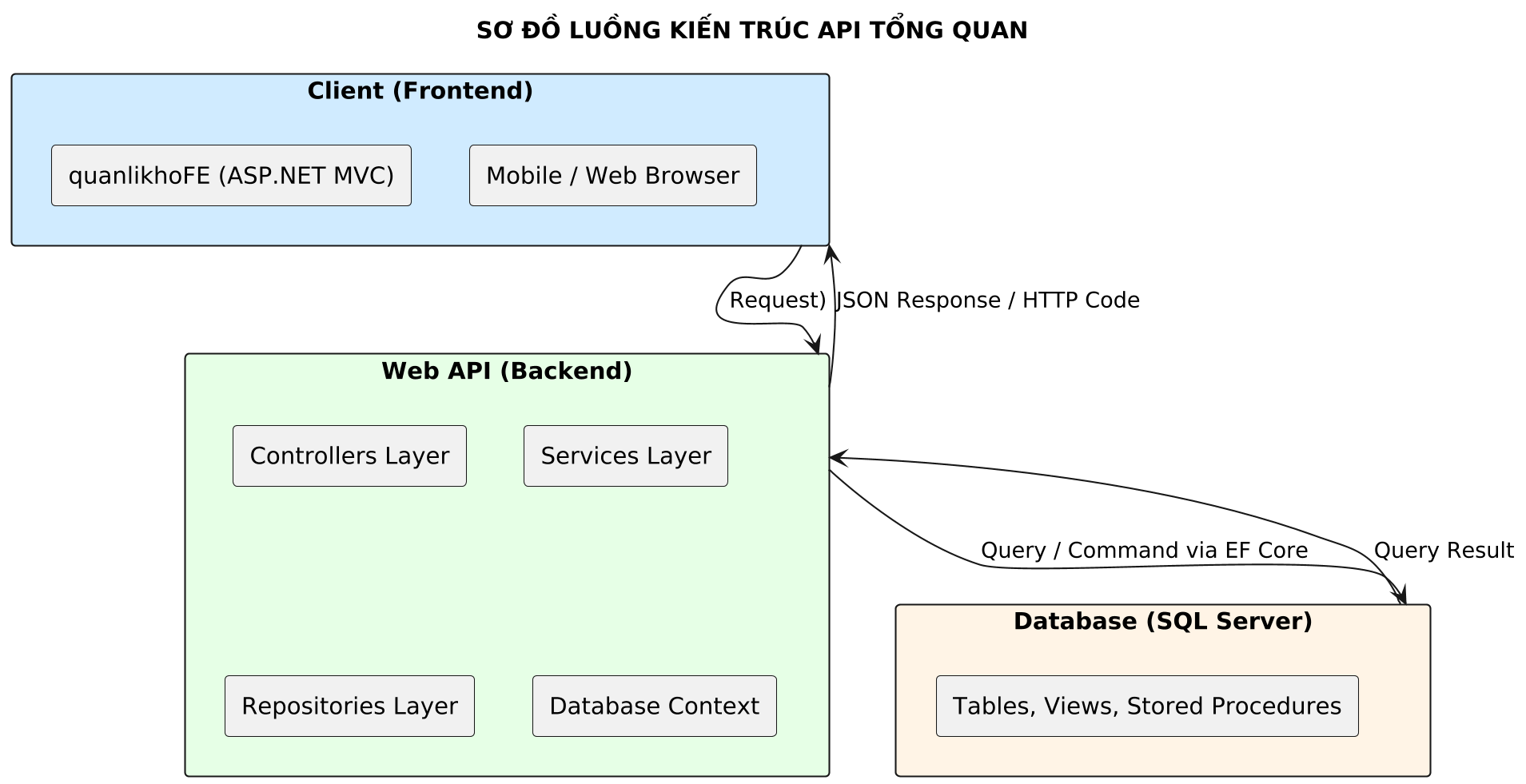
• PUT /api/stockoutbounds/{id}/confirm - Xác nhận phiếu xuất

Reports:

• GET /api/reports/stock-summary - Báo cáo tồn kho

• GET /api/reports/inbound-outbound - Báo cáo nhập/xuất

• GET /api/reports/dashboard-stats - Thống kê dashboard



### 2.5.4 Bảo mật API

|  |  |  |
| --- | --- | --- |
| **Nhóm bảo mật** | **Cơ chế** | **Mục đích** |
| **Authentication** | JWT Bearer Token | Xác thực người dùng khi truy cập API |
|  | Token Expiration & Refresh | Tự động hết hạn, yêu cầu đăng nhập lại sau thời gian định sẵn |
|  | Login Attempt Limiting | Giới hạn số lần đăng nhập sai để chống brute-force |
| **Authorization** | Role-Based Access Control (RBAC) | Phân quyền theo vai trò (Admin/User) |
|  | Resource-Based Permission | Giới hạn truy cập vào từng tài nguyên cụ thể |
| **Data Protection** | Input Validation, SQL Injection Prevention | Bảo vệ dữ liệu và cơ sở dữ liệu |
|  | XSS & CSRF Protection | Ngăn chặn tấn công script và giả mạo yêu cầu |
| **Communication Security** | HTTPS, CORS Configuration | Bảo mật đường truyền và giới hạn nguồn truy cập |
|  | Request Rate Limiting | Ngăn tấn công DDoS hoặc spam request |
|  | API Versioning | Duy trì tính ổn định khi nâng cấp API |

## 2.6 Triển khai và cài đặt

### 2.6.1 Môi trường phần mềm

**Development Environment**

* Operating System: Windows 10/11
* IDE: Visual Studio 2022 Community/Professional
* SDK: .NET 8.0
* Database: SQL Server 2019/2022 hoặc LocalDB
* Web Server: IIS Express (Development), Kestrel (Production)

**Runtime Dependencies**

* ASP.NET Core 8.0 Runtime
* Entity Framework Core 8.0
* SQL Server Client Libraries
* JWT Authentication Libraries

**Development Tools**

* Package Manager: NuGet
* Version Control: Git
* API Testing: Swagger UI, Postman
* Database Management: SQL Server Management Studio

### 2.6.2 Môi trường phần cứng

**Minimum Requirements**

* CPU: Intel Core i3 hoặc tương đương
* RAM: 8 GB
* Storage: 10 GB trống
* Network: Kết nối Internet ổn định

**Recommended Requirements**

* CPU: Intel Core i5 hoặc AMD Ryzen 5
* RAM: 16 GB
* Storage: SSD ≥ 20 GB
* Network: High-speed Internet

**Production Server**

* CPU: Multi-core server processor
* RAM: 32 GB trở lên
* Storage: SSD có backup
* Network: Redundant network connections

### 2.6.3 Các bước cài đặt

**Bước 1 – Chuẩn bị môi trường**

1. Cài đặt Visual Studio 2022 (ASP.NET and web development workload)
2. Cài .NET 8.0 SDK
3. Cài SQL Server hoặc SQL Server Express
4. Cài Git để quản lý mã nguồn

**Bước 2 – Clone và setup project**

1. Clone repository từ source control
2. Mở file quanlikho.sln
3. Restore NuGet Packages
4. Build solution

**Bước 3 – Cấu hình database**

1. Cập nhật ConnectionString trong appsettings.json
2. Mở Package Manager Console → chọn project quanlikho
3. Chạy lệnh: Update-Database
4. Kiểm tra database được tạo thành công

**Bước 4 – Cấu hình JWT**

1. Tạo JWT Secret Key (tối thiểu 32 ký tự)
2. Cập nhật JWT settings trong appsettings.json
3. Cấu hình Issuer và Audience

**Bước 5 – Chạy ứng dụng**

1. Mở Solution Properties → chọn **Multiple Startup Projects**
2. Chọn cả quanlikho và quanlikhoFE → set Start
3. Nhấn **F5** để chạy thử API và Frontend

# CHƯƠNG 3: KẾT QUẢ CHƯƠNG TRÌNH

**3.1. SẢN PHẨM HOÀN THIỆN**

### 3.1.1. Backend API (quanlikho)

**Kiến trúc và Components**

* RESTful API với 9 controllers chính: *Auth, Products, Categories, Suppliers, Warehouses, StockInbounds, StockOutbounds, Reports, StockHistory*
* 13 entity models định nghĩa cấu trúc dữ liệu
* 10 DTOs cho việc trao đổi dữ liệu an toàn
* Repository pattern với 7 repositories
* Service layer với 8 business services
* JWT authentication và role-based authorization

**Database và Migration**

* Entity Framework Core với Code-First approach
* 8 database migrations cho schema evolution
* Seed data cho development và testing
* Proper indexing cho performance optimization

**API Documentation**

* Swagger UI integration
* Comprehensive API documentation
* Request/Response examples
* Authentication flow documentation

### 3.1.2. Frontend MVC (quanlikhoFE)

**Architecture Components**

* 10 MVC controllers xử lý user interface
* ApiService class để consume Web API
* 15+ Views với Razor templates
* ViewModels cho data binding
* Authentication middleware

**User Interface**

* Responsive design với Bootstrap 5
* Modern và intuitive user experience
* Real-time form validation
* Interactive dashboards với charts
* Mobile-friendly interface

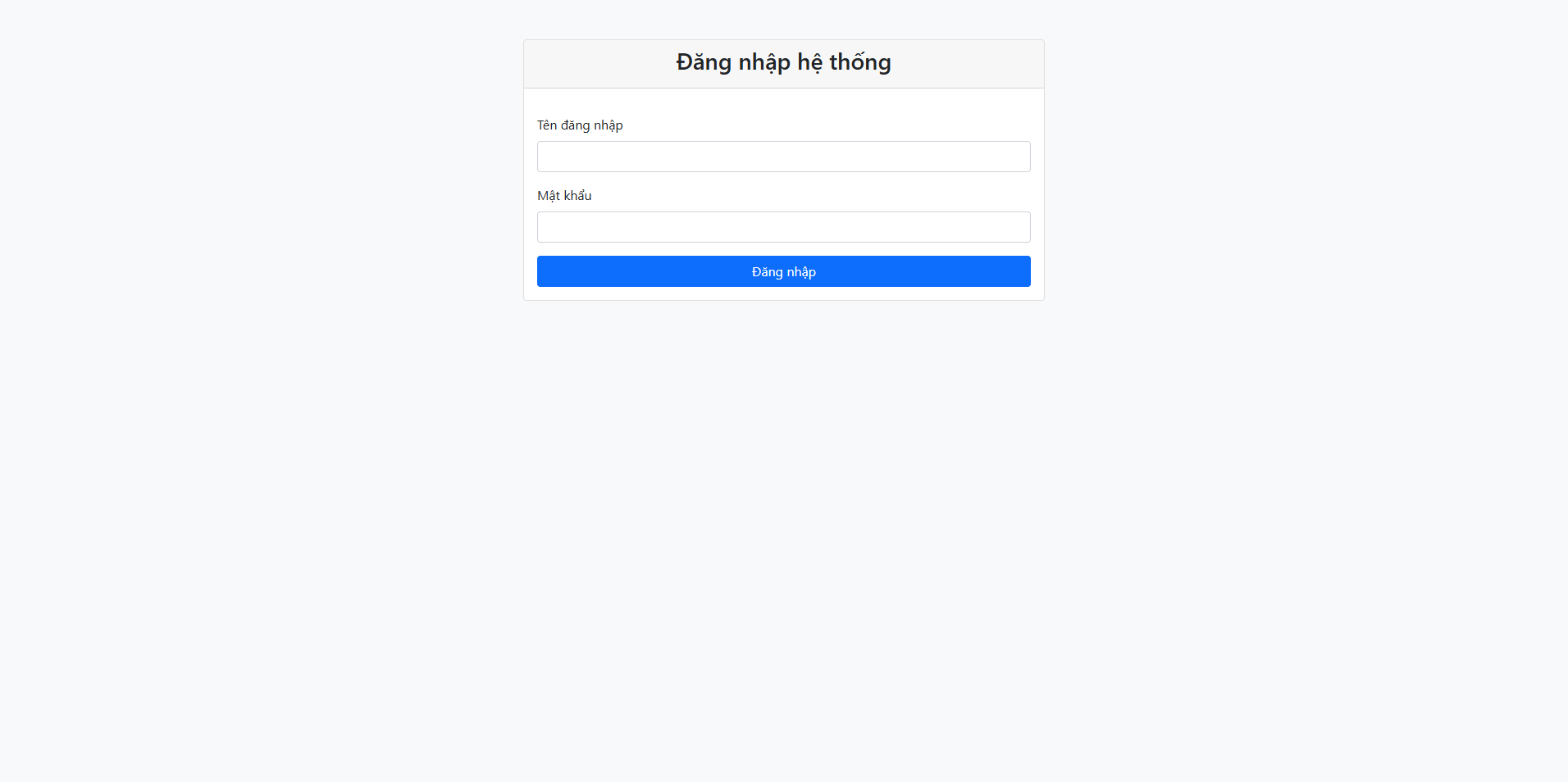
**Integration Features**

* Seamless API integration
* Session management
* Error handling và user feedback
* Loading states và progress indicators

## 3.2. Chức năng đã triển khai

### 3.2.1. Module xác thực và phân quyền

✅ Đăng nhập, đăng xuất, JWT token, role-based access, middleware, timeout handling



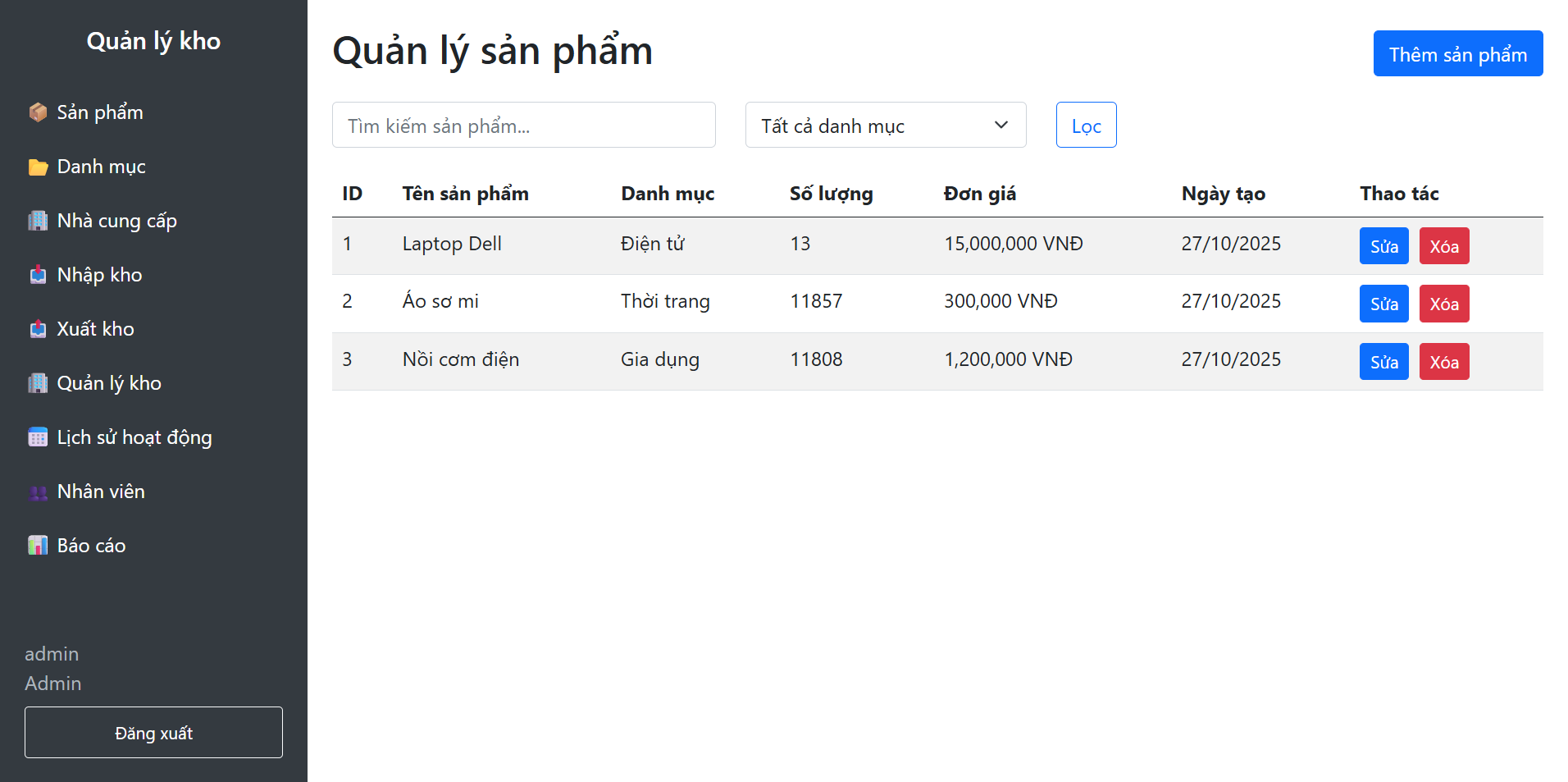
### 3.2.2. Module quản lý danh mục

✅ CRUD danh mục



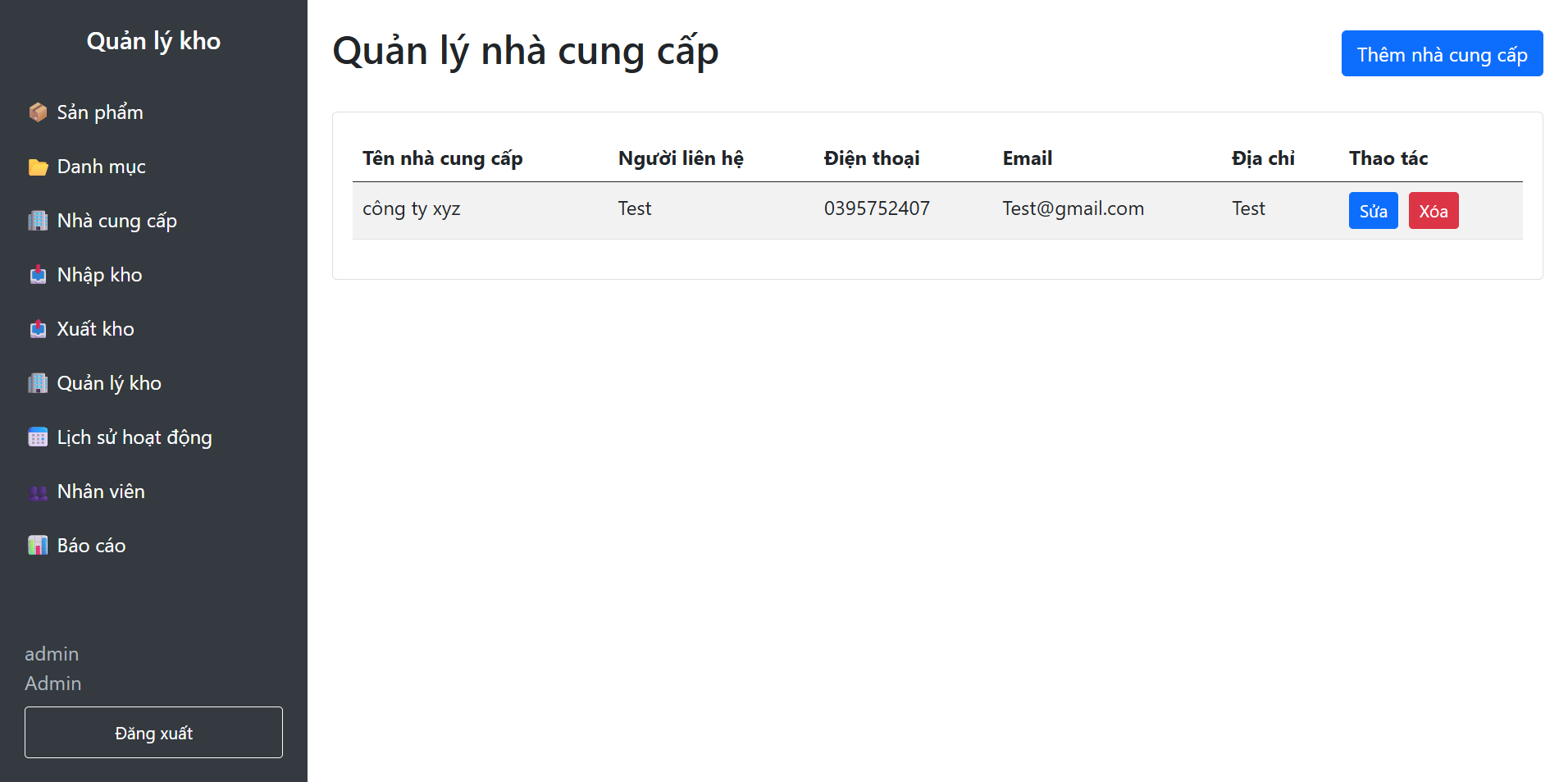
### 3.2.3. Module quản lý sản phẩm

✅ CRUD sản phẩm, advanced filtering, category association.



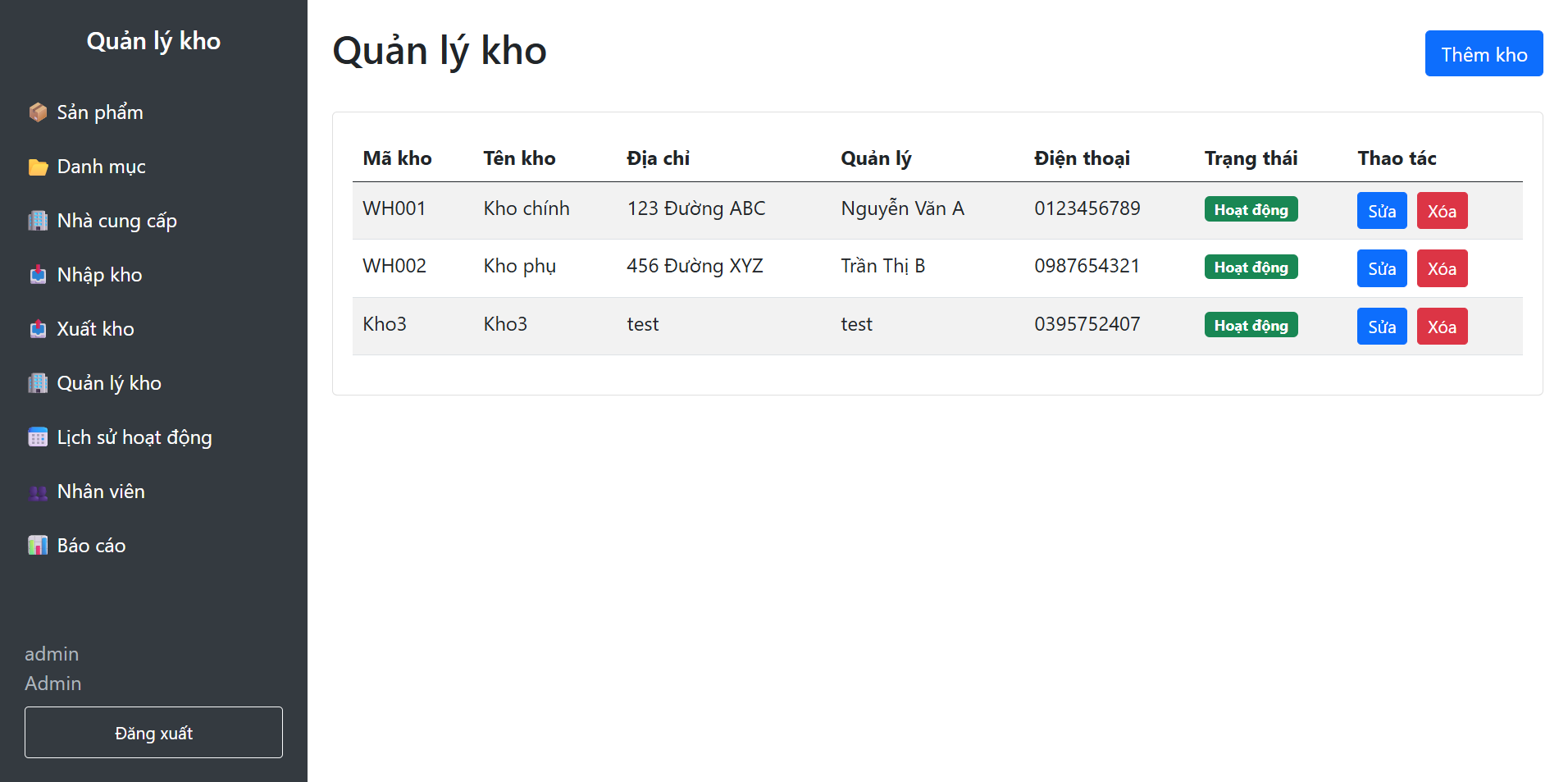
### 3.2.4. Module quản lý nhà cung cấp

✅ CRUD, validation, performance tracking, tích hợp quy trình nhập kho.



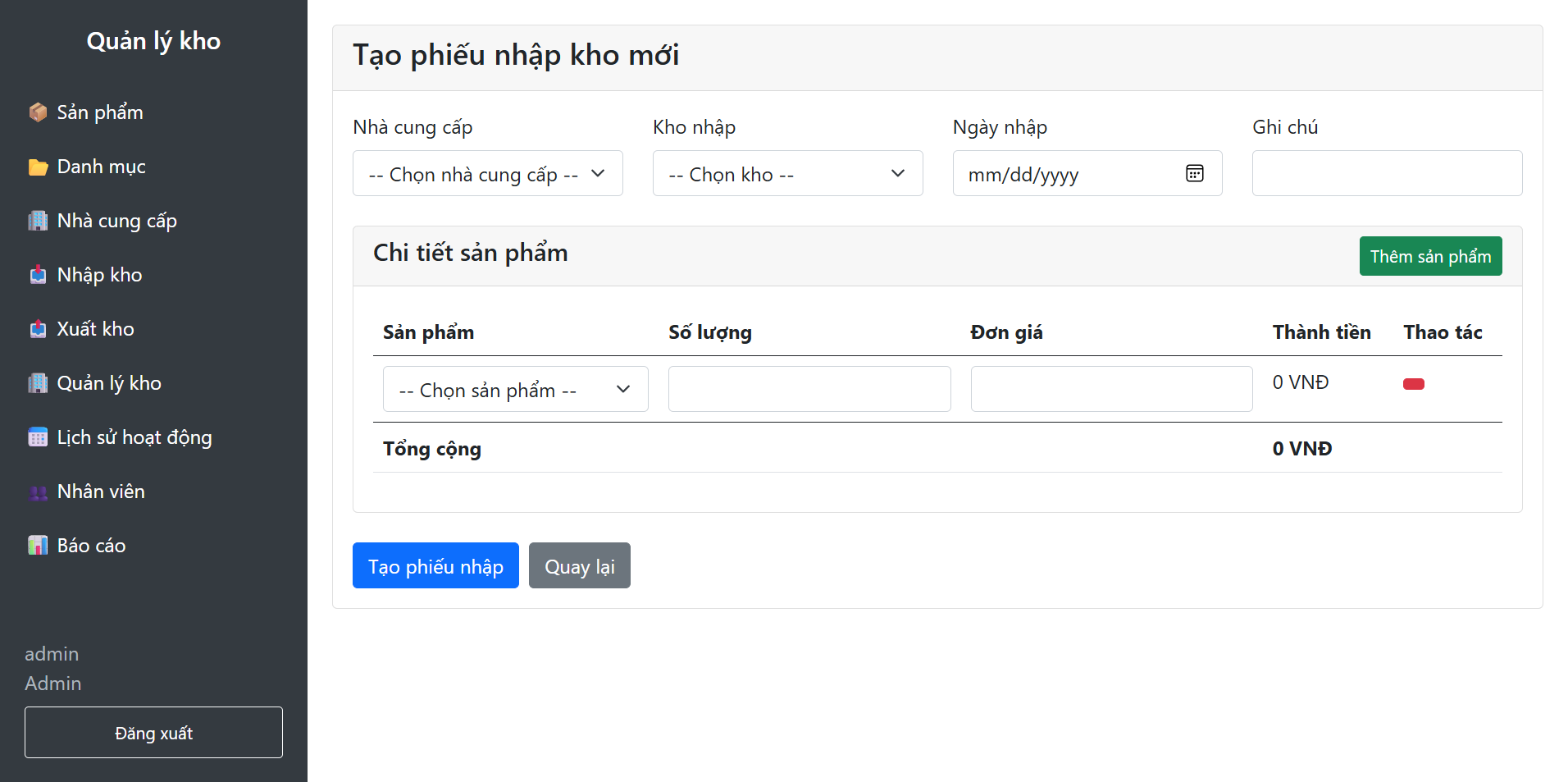
### 3.2.5. Module quản lý kho hàng

✅ CRUD kho, theo dõi dung lượng, báo cáo theo kho, quản lý địa chỉ.



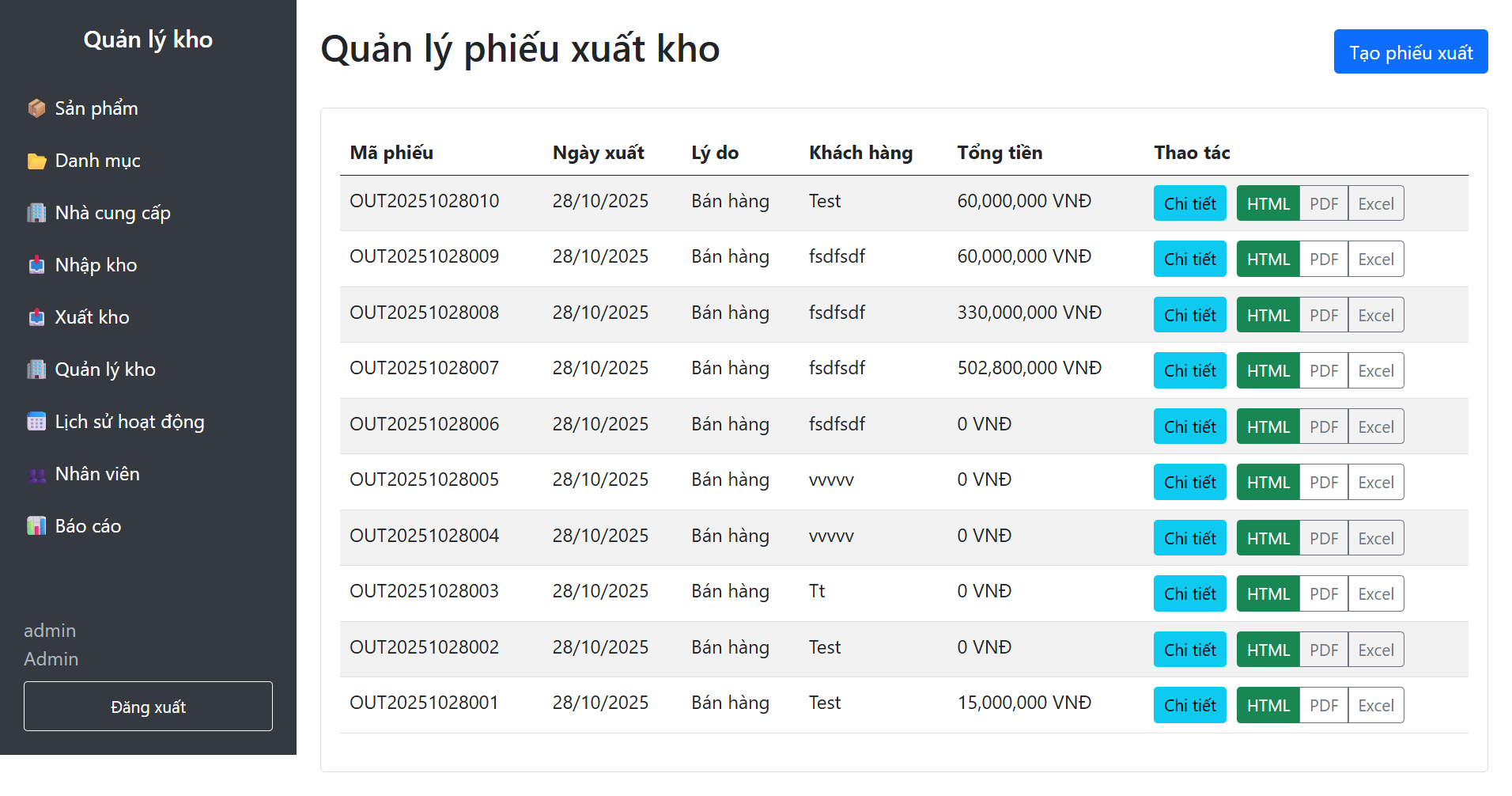
### 3.2.6. Module nhập kho

✅ Phiếu nhập, dynamic lines, tính giá tự động, in phiếu, cập nhật tồn kho.



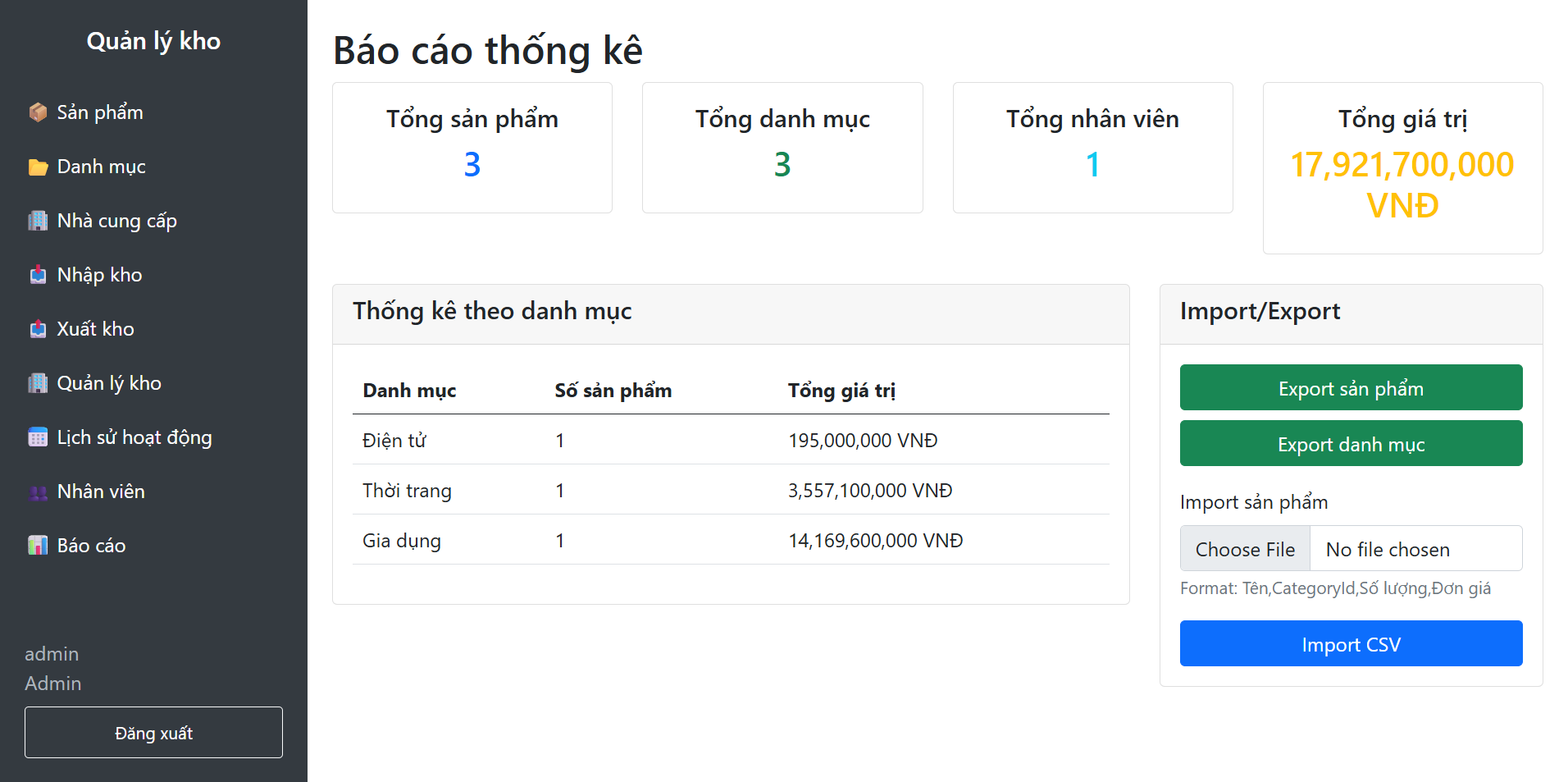
### 3.2.7. Module xuất kho

✅ Phiếu xuất, kiểm tra tồn, xuất hàng tự động, batch processing.



### 3.2.8. Module báo cáo và thống kê

✅ Báo cáo tồn kho, nhập/xuất, dashboard charts, export JSON.



# KẾT LUẬN

Dự án **“Hệ thống Quản lý Kho”** đã được triển khai thành công với **kiến trúc tách biệt giữa backend Web API và frontend MVC**, thể hiện sự hiểu biết sâu sắc về **các công nghệ web hiện đại** và **best practices** trong phát triển phần mềm.

**1. Thành tựu kỹ thuật**

* Áp dụng thành công kiến trúc **RESTful API** với **ASP.NET Core 8**
* Triển khai **Repository Pattern** và **Service Layer** đảm bảo khả năng bảo trì (maintainability)
* Sử dụng **Entity Framework Core** với **Code-First approach**
* Implement **JWT authentication** và **role-based authorization**
* Tích hợp **Swagger UI** cho tài liệu và kiểm thử API
* Thiết kế **giao diện web responsive** với **Bootstrap 5**

**2. Giá trị nghiệp vụ**

* Đáp ứng đầy đủ các yêu cầu **quản lý kho cơ bản**
* **Giao diện người dùng thân thiện và trực quan**
* **Hệ thống báo cáo và thống kê toàn diện (comprehensive)**
* **Theo dõi và quản lý hàng tồn kho theo thời gian thực (real-time)**
* Hỗ trợ **multi-warehouse management**

**3. Đóng góp học thuật**

* Minh họa rõ ràng **quy trình xây dựng và tiêu thụ Web API**
* Thể hiện hiểu biết về **kiến trúc phần mềm hiện đại**
* Áp dụng **industry best practices** trong phát triển ứng dụng
* Cung cấp **tài liệu và kiểm thử (documentation & testing)** đầy đủ

**4. Hạn chế và hướng phát triển**

* Cần bổ sung **unit testing và integration testing** để nâng cao độ tin cậy
* Có thể triển khai **caching** để tối ưu hiệu năng
* **Logging system** cần được mở rộng và tích hợp hệ thống giám sát
* Tích hợp **notification system** (email, SMS, hoặc SignalR notifications)
* Phát triển thêm **mobile application** để mở rộng khả năng truy cập

Dự án này là một ví dụ điển hình cho việc **xây dựng ứng dụng web enterprise với kiến trúc API-first**, phù hợp cho **việc học tập, nghiên cứu và ứng dụng thực tế**.  
Hệ thống không chỉ đáp ứng các yêu cầu kỹ thuật mà còn mang lại **giá trị thực tiễn** trong **quản lý kho hàng của doanh nghiệp**.