**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Ảnh có chứa văn bản, ký hiệu

Mô tả được tạo tự động**

**BÁO CÁO DỰ ÁN CÔNG NGHỆ**

**Ngành: Hệ thống thông tin**

**MIGRATE HỆ THỐNG QUẢN LÝ ĐỀ TÀI, DỰ ÁN THUỘC CÁC CHƯƠNG TRÌNH TRỌNG ĐIỂM CẤP NHÀ NƯỚC**

|  |  |
| --- | --- |
| Người thực hiện  Mã sinh viên  Lớp  Giảng viên hướng dẫn | : Lưu Đạt Tuấn Minh  : 20021392  : INT3131 20  : Nguyễn Ngọc Hóa |

**HÀ NỘI - 2023**

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 3](#_Toc153137148)

[Chương 1 Giới thiệu 4](#_Toc153137149)

[1.1. Mô tả tổng quan 4](#_Toc153137150)

[1.1.1. Giới thiệu Hệ thống quản lý đề tài, dự án thuộc các chương trình trọng điểm cấp nhà nước 4](#_Toc153137151)

[1.1.2. Tình hình hiện tại của trang web 4](#_Toc153137152)

[1.1.3. Lí do lựa chọn ASP.NET CORE 2](#_Toc153137153)

[1.1.4. Kế hoạch Migrate 2](#_Toc153137154)

[1.1.5. Phạm vi và mục tiêu 2](#_Toc153137155)

[1.2. Phân tích hệ thống 3](#_Toc153137156)

[1.2.1. Cấu trúc hệ thống 3](#_Toc153137157)

[1.2.2. Chức năng quản lý 5](#_Toc153137158)

[1.2.3. Đánh giá sơ bộ 6](#_Toc153137159)

[Chương 2 Thiết kế hệ thống mới 8](#_Toc153137160)

[2.1. Phương pháp thiết kế 8](#_Toc153137161)

[2.2. Kiến trúc hệ thống 12](#_Toc153137162)

[2.2.1. Xây dựng kiến trúc 12](#_Toc153137163)

[2.2.2. Thiết kế cơ sở dữ liệu 13](#_Toc153137164)

[2.2.3. Thiết kế logic xử lý 17](#_Toc153137165)

[2.2.4. Thiết kế bảo mật 18](#_Toc153137166)

[Chương 3 Triển khai và kiểm thử 20](#_Toc153137167)

[3.1. Triển khai 20](#_Toc153137168)

[3.1.1. Phạm vi triển khai 20](#_Toc153137169)

[3.1.2. Lập kế hoạch 20](#_Toc153137170)

[3.1.3. Seed data 20](#_Toc153137171)

[3.2. Kiểm thử 20](#_Toc153137172)

[Chương 4 Kết quả 21](#_Toc153137173)

[Tài liệu tham khảo 22](#_Toc153137174)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1 Giao diện của trang web 4](#_Toc153137175)

[Hình 1.2 Sơ đồ quy trình nghiệp vụ chung 4](#_Toc153137176)

[Hình 1.3 Form nhập nhiệm vụ cho chương trình 2](#_Toc153137177)

[Hình 1.4 Giao diện quản lý chương trình 4](#_Toc153137178)

[Hình 1.5 Giao diện quản lý nhiệm vụ 5](#_Toc153137179)

[Hình 1.6 Decompiled code của hệ thống 7](#_Toc153137180)

[Hình 1.7 Hệ thống khi bị treo 8](#_Toc153137181)

[Hình 2.1 Mô hình thác nước 11](#_Toc153137182)

[Hình 2.2 Kiến trúc ASP.NET CORE WEB API 12](#_Toc153137183)

[Hình 2.3 IUnitOfWork pattern 13](#_Toc153137184)

[Hình 2.4 DepartmentModel 13](#_Toc153137185)

[Hình 2.5 FieldModel 14](#_Toc153137186)

[Hình 2.6 ExpertModel 14](#_Toc153137187)

[Hình 2.7 ProgramModel 15](#_Toc153137188)

[Hình 2.8.1 TaskModel(first half) 15](#_Toc153137189)

[Hình 2.9 DocType&FileTypeModel 16](#_Toc153137190)

[Hình 2.10 GiaiDoanModel 16](#_Toc153137191)

[Hình 2.11 ProductModel 17](#_Toc153137192)

[Hình 2.12 Tính năng phân quyền 19](#_Toc153137193)

[Hình 4.1 Sản phẩm cuối cùng 21](#_Toc153137194)

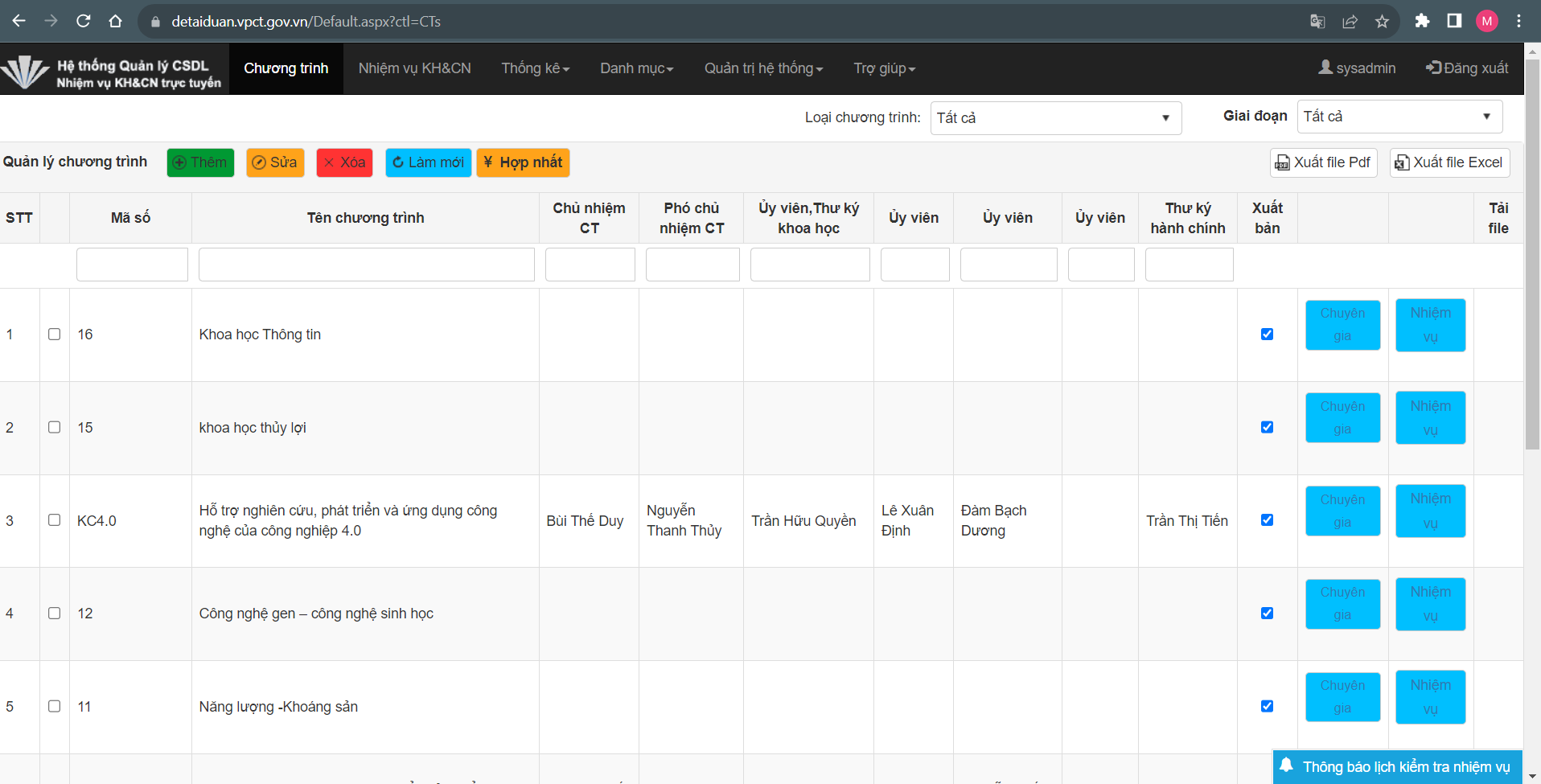
[Hình 4.2 Download file Request trong Swagger 21](#_Toc153137195)

# **Giới thiệu**

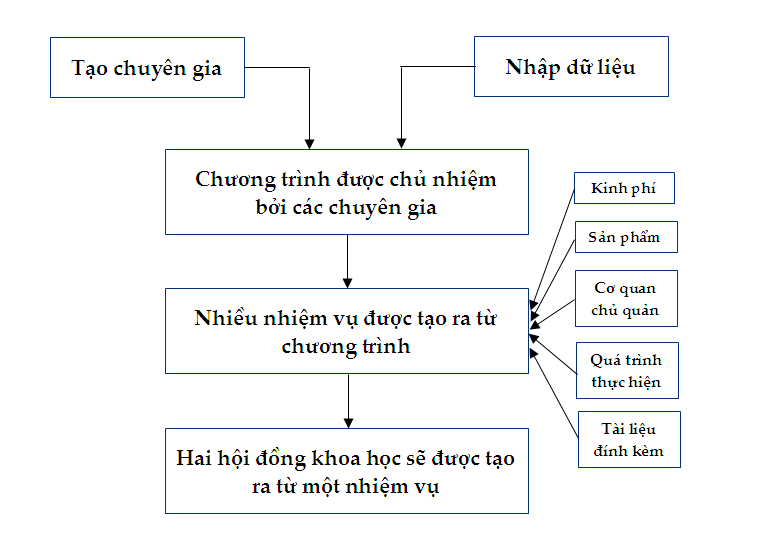
## Mô tả tổng quan

### Giới thiệu Hệ thống quản lý đề tài, dự án thuộc các chương trình trọng điểm cấp nhà nước

Hệ thống quản lý đề tài đóng vai trò quan trọng trong việc theo dõi và quản lý các dự án thuộc các chương trình trọng điểm cấp nhà nước. Hiện tại, hệ thống được xây dựng trên nền tảng Webforms, tuy nhiên, để nâng cao hiệu suất, tính bảo mật và khả năng mở rộng, quyết định chuyển đổi sang Asp.net Core đã được đưa ra.



Hình 1.1 Giao diện của trang web

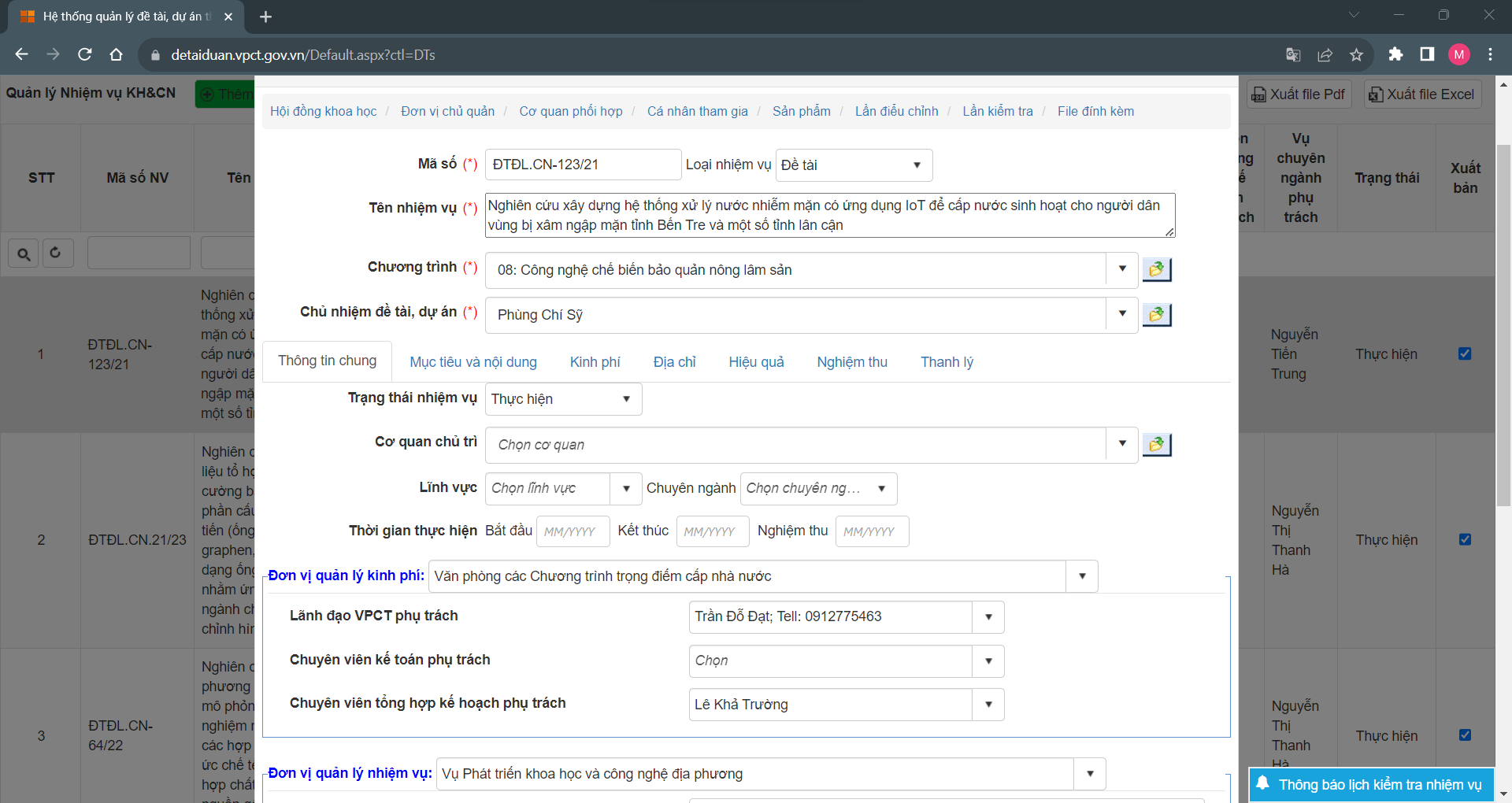


Hình 1.2 Sơ đồ quy trình nghiệp vụ chung

### Tình hình hiện tại của trang web

Hệ thống quản lý đề tài hiện tại dựa trên công nghệ Webforms đã phục vụ công việc quản lý dự án một phần hiệu quả. Tuy nhiên, một số yêu cầu nghiệp vụ của trang web không đáp ứng chính xác yêu cầu nghiệp vụ, và một số yêu cầu kỹ thuật không đạt tiêu chuẩn, gây ra các lỗi treo phần mềm.

Ngoài các lỗi về backend, frontend của Website không thân thiện với người dùng, do sử dụng các chức năng kéo thả căn bản của Webforms. Do đó, hệ thống cần được nâng cấp để đảm bảo đạt đầy đủ yêu cầu nghiệp vụ, các chức năng được nâng cấp để phù hợp với việc bảo trì lâu dài và dễ dàng nâng cấp lên các phiên bản mới nhất trong tương lai và một UI thân thiện hơn với người dùng.



Hình 1.3 Form nhập nhiệm vụ cho chương trình

### Lí do lựa chọn ASP.NET CORE

Do hệ thống cũng sử dụng nền tảng .NET và cũng để đảm bảo các quy trình nghiệp vụ được thực thi đầy đủ thì việc lựa chọn một nền tảng .NET vốn được dùng trong việc xây dựng phần mềm web sẽ giúp mang lại nhiều thuận tiện về mặt kỹ thuật. ASP.NET CORE cũng cung cấp hiệu suất cao, hỗ trợ tốt và lâu dài cho các tính năng hiện đại như API và microservices, đồng thời tối ưu hóa quá trình phát triển và triển khai.

### Kế hoạch Migrate

Dự án chuyển đổi sẽ bao gồm các bước cụ thể như phân tích hệ thống hiện tại, thiết kế kiến trúc mới, việc chuyển đổi mã nguồn và kiểm thử kỹ thuật. Việc xây dựng mã nguồn sẽ được thực hiện thông qua Backend là ASP.NET CORE API và Frontend là React. Điều này sẽ giúp việc chuyển đổi Frontend trong tương lai nếu có nhu cầu thuận tiện hơn.

### Phạm vi và mục tiêu

Phạm vi dự án:

* Phân tích hệ thống hiện tại: Xác định cấu trúc, tính năng và hạn chế của hệ thống quản lý đề tài và dự án hiện tại được xây dựng trên nền tảng Webforms.
* Thiết kế kiến trúc mới: Xác định và thiết kế kiến trúc mới dựa trên ASP.NET CORE API và React để đảm bảo tính mở rộng, hiệu suất và khả năng bảo trì.
* Phát triển ứng dụng: Xây dựng ứng dụng quản lý đề tài và dự án mới sử dụng ASP.NET CORE API và React theo yêu cầu và thiết kế đã xác định.
* Kiểm thử và đảm bảo chất lượng: Thực hiện kiểm thử chức năng, hiệu suất và bảo mật để đảm bảo tính ổn định và an toàn của hệ thống mới.
* Triển khai và huấn luyện: Triển khai hệ thống mới và cung cấp huấn luyện cho người dùng cuối và quản trị viên.

Mục tiêu dự án:

* Nâng cấp hệ thống hiện tại: Chuyển đổi hệ thống quản lý đề tài và dự án từ Webforms sang một kiến trúc hiện đại và linh hoạt hơn.
* Tối ưu hóa hiệu suất: Tăng cường hiệu suất và khả năng mở rộng của hệ thống để đáp ứng nhu cầu người dùng và tương lai.
* Cải thiện trải nghiệm người dùng: Xây dựng giao diện người dùng thân thiện, dễ sử dụng và phản hồi nhanh chóng bằng React.
* Đảm bảo tính ổn định và an toàn: Kiểm thử kỹ lưỡng và đảm bảo tính ổn định, bảo mật của hệ thống mới trước khi triển khai.

## Phân tích hệ thống

### Cấu trúc hệ thống

1. Giao diện đăng nhập



Hình 1.4 Giao diện đăng nhập hiện tại

1. Giao diện hệ thống

A screenshot of a computer

Description automatically generated

Hình 1.4 Giao diện quản lý chương trình

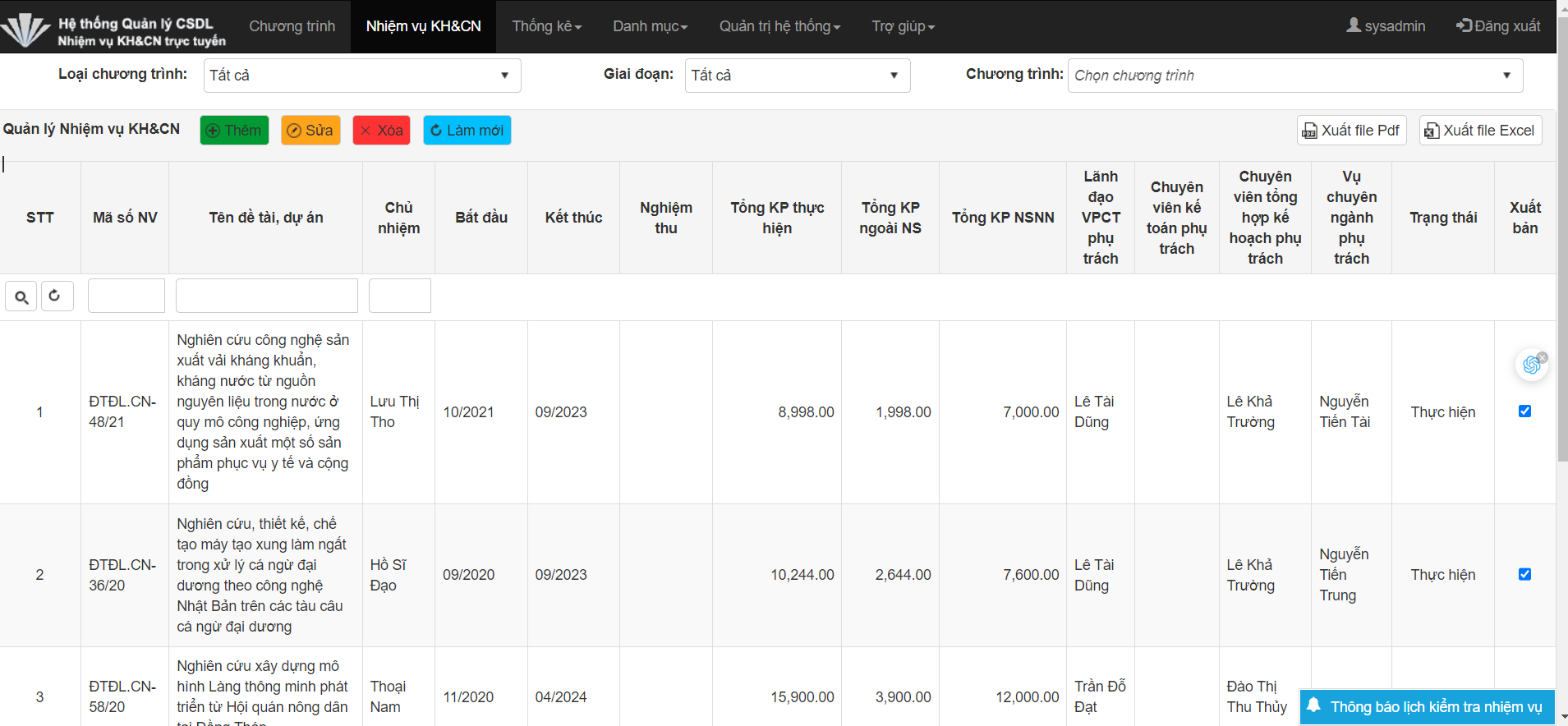
Tổng quan: Có 2 loại chương trình:

* Chương trình trọng điểm cấp nhà nước
* Chương trình độc lập quỹ gen

Mỗi chương trình có nhiều nhiệm vụ (đề tài) và có nhiều chuyên gia (từng chức vụ) tham gia.

**Lưu ý: Hệ thống có rất nhiều dữ liệu (model) ngoài chương trình và nhiệm vụ. Phần Kiến trúc hệ thống mới sẽ đi sâu hơn.**

* Chương trình:
* Hiển thị chi tiết chương trình
* Xuất bản chương trình
* Xuất file PDF, Excel
* Cung cấp bộ lọc tìm kiếm: Loại chương trình, Giai đoạn
* Thêm, sửa, xóa, làm mới, hợp nhất
* Nhiệm vụ
* Hiển thị chi tiết nhiệm vụ (1 chương trình có nhiều nhiệm vụ)
* Cung cấp bộ lọc tìm kiếm: Loại chương trình, Giai đoạn, chương trình
* Xuất file PDF, Excel
* Thêm, sửa, xóa, làm mới
* Xuất bản nhiệm vụ



Hình 1.5 Giao diện quản lý nhiệm vụ

### Chức năng quản lý

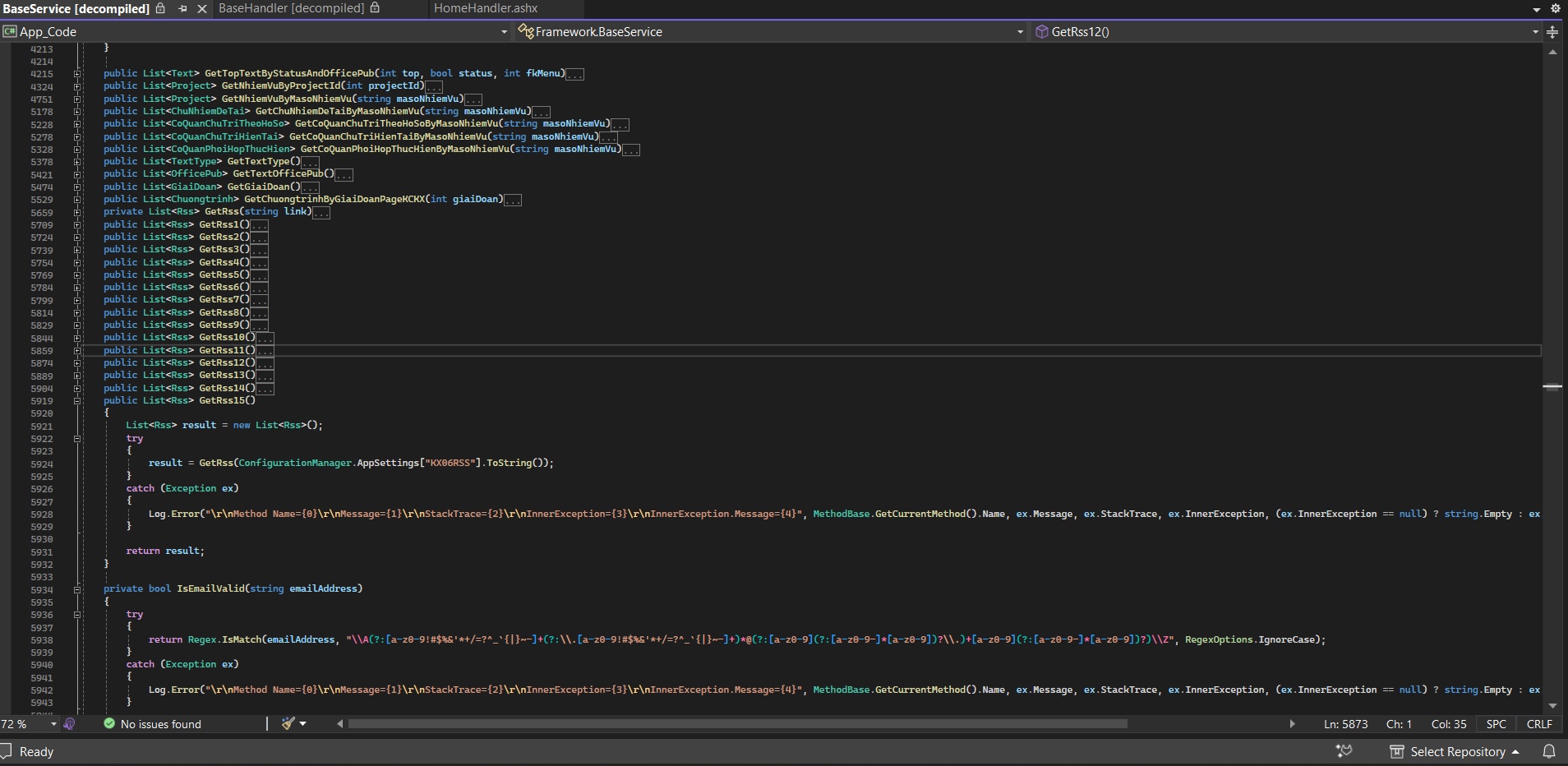
* 1. **Quản lý danh mục:**
* Chuyên gia
* Giai đoạn
* Lĩnh vực khoa học
* Chuyên ngành
* Đơn vị chủ quản
* Cơ quan chủ trì
* Học hàm
* Học vị
* Chức danh
* Chức vụ
* Dạng sản phẩm
* Loại sản phẩm
* Loại chương trình
* Cơ quan quản lý
* Loại công văn
* Loại file đính kèm nhiệm vụ
  1. **Thống kê:**
* Báo cáo kết quả thực hiện chương trình KH&CN
* Xem, in, quản lí các mẫu báo cáo của chương trình
  1. **Phân quyền người dùng, nhóm người dùng:**
* Quản lý quyền truy cập và vai trò của người dùng và nhóm người dùng trong hệ thống.
  1. **Quản lí nhật ký đăng nhập, nhật ký lỗi:**
* Ghi nhận và quản lý nhật ký đăng nhập và nhật ký lỗi để theo dõi hoạt động và xử lý sự cố.
  1. **Cấu hình hệ thống:**
* Cho phép quản trị viên cấu hình hệ thống theo yêu cầu cụ thể của tổ chức.

Mô tả chi tiết về các yêu cầu chức năng quản lý sẽ giúp định hình rõ ràng vai trò và khả năng của hệ thống trong việc quản lý và tổ chức thông tin, đồng thời đảm bảo tính bảo mật và hiệu suất của hệ thống khi triển khai.

### Đánh giá sơ bộ

1. Đánh giá về cấu trúc và kiến trúc của hệ thống:

* Phân tích cấu trúc code, sự phụ thuộc và các vấn đề kỹ thuật khác của hệ thống hiện tại.
* Đánh giá cấu trúc code dựa trên thông tin API chia sẻ dữ liệu của trang web và decompiled code. Phát hiện rằng cấu trúc code không linh hoạt và khó mở rộng do tất cả các phần của hệ thống được đặt trong cùng một file.



Hình 1.6 Decompiled code của hệ thống

1. Đánh giá về hiệu suất và tải trọng:

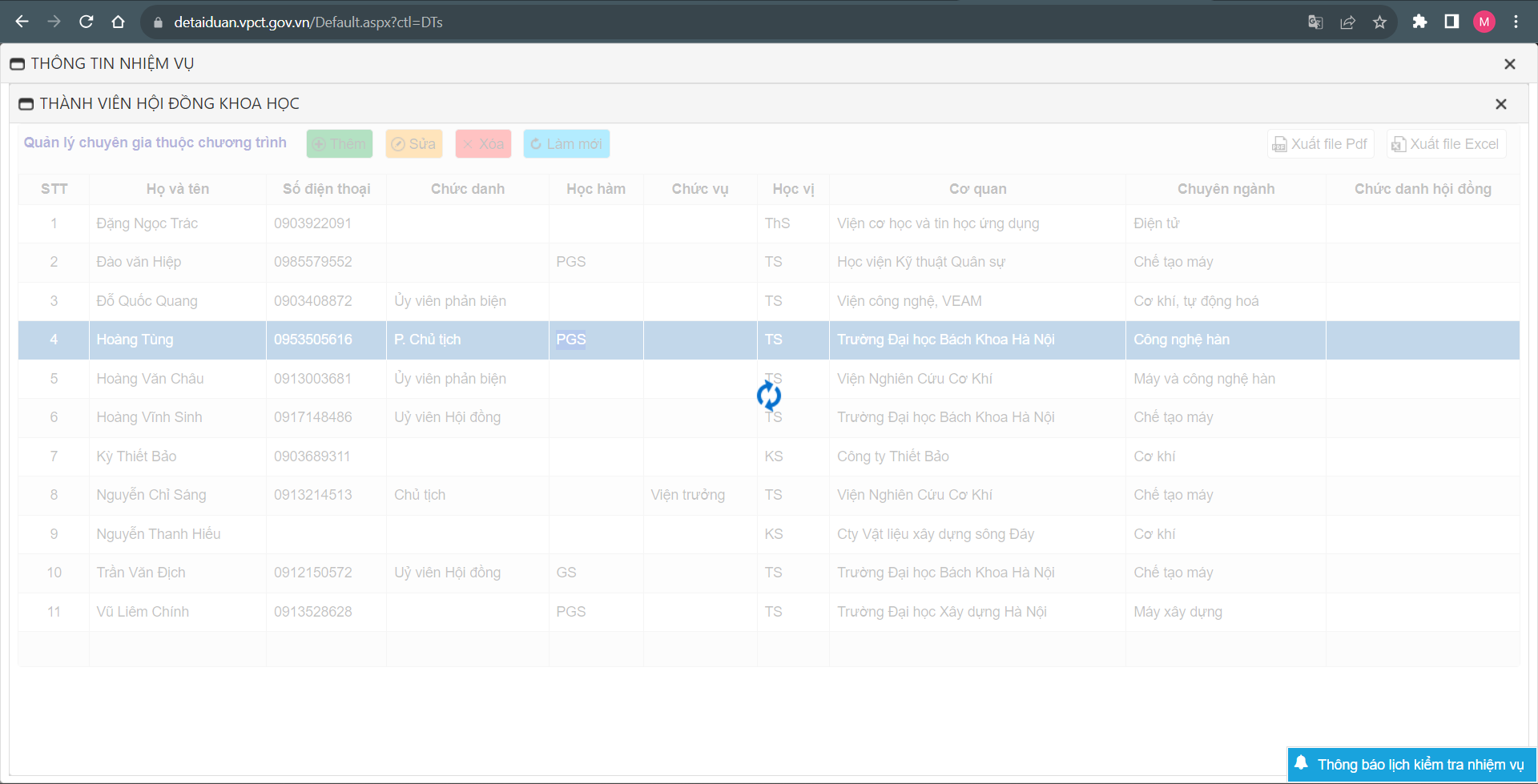
* Xác định các vấn đề liên quan đến hiệu suất và tải trọng của hệ thống hiện tại, bao gồm tốc độ phản hồi, thời gian phản hồi và khả năng mở rộng.
* Phân tích mã đọc dữ liệu từ cơ sở dữ liệu và ánh xạ vào đối tượng, nhận thấy rủi ro về hiệu suất khi xử lý quá nhiều dữ liệu.

1. Đánh giá về bảo mật:

* Xác định các vấn đề bảo mật có thể tồn tại trong hệ thống hiện tại và đề xuất cách giải quyết.
* Phát hiện rằng việc trực tiếp kết nối cơ sở dữ liệu bằng ADO.NET có thể dẫn đến các vấn đề về bảo mật và quản lý kết nối, và đề xuất sử dụng Entity Framework hoặc Dapper để quản lý kết nối cơ sở dữ liệu hiệu quả hơn.

1. Đánh giá về khả năng mở rộng và bảo trì:

* Xem xét khả năng mở rộng và bảo trì của hệ thống hiện tại, bao gồm khả năng thêm tính năng mới và sửa lỗi.
* Phân tích rủi ro khi tất cả các phần của hệ thống được đặt trong cùng một file, ảnh hưởng đến khả năng mở rộng và tái sử dụng mã nguồn.



Hình 1.7 Hệ thống khi bị treo

# **Thiết kế hệ thống mới**

## Phương pháp thiết kế

Phần thiết kế hệ thống mới sẽ tuân theo phương pháp phát triển theo phương pháp "waterfall". Phương pháp này bao gồm việc chia quá trình phát triển thành các giai đoạn rõ ràng, bao gồm thiết kế, lập trình, kiểm thử, sửa lỗi và tài liệu. Mỗi giai đoạn sẽ được hoàn thành trước khi chuyển sang giai đoạn tiếp theo, đảm bảo một phương pháp phát triển có hệ thống và có tổ chức.

Quá trình phát triển sẽ được chia thành 5 giai đoạn:

Thiết kế → Lập trình → Kiểm thử → Sửa lỗi → Tài liệu

**Thiết kế**

Giai đoạn thiết kế bao gồm việc tạo ra tài liệu thiết kế, bao gồm các yêu cầu thiết kế phần mềm. Những yêu cầu này sẽ được xem xét bởi người liên hệ chính cho sản phẩm cuối cùng, và sau khi được phê duyệt, sẽ tiếp tục sang giai đoạn "lập trình".

**Lập trình**

Giai đoạn lập trình bao gồm việc phát triển ứng dụng dựa trên các đặc tả thiết kế chứa trong tài liệu này. Giai đoạn này được chia thành ba phần riêng biệt: Giao diện API, Giao diện chính và Thiết kế cơ sở dữ liệu.

**Kiểm thử**

Giai đoạn kiểm thử sẽ tài liệu hóa tất cả các quy trình người dùng và xác minh tính năng của chúng. Người kiểm thử sẽ tham gia trong giai đoạn này để đảm bảo hệ thống hoạt động như dự kiến.

**Sửa lỗi**

Trong giai đoạn kiểm thử, giai đoạn sửa lỗi sẽ diễn ra đồng thời, khi phát triển sửa chữa bất kỳ lỗi nào, và kiểm thử xác nhận sự sửa lỗi.

**Tài liệu**

Cuối cùng, trong giai đoạn tài liệu, tài liệu cuối cùng sẽ được chuẩn bị và trình bày cho người dùng. Tài liệu sẽ bao gồm tài liệu nguồn và người dùng, cũng như các vấn đề và hạn chế đã biết.

**Giả định**

Kiến trúc cơ sở dữ liệu sẽ dựa trên SQL. Người dùng hiện tại sử dụng các công nghệ của Microsoft và yêu cầu duy trì tình trạng hiện tại. Do đó, Microsoft SQL Server 2019 Enterprise là hệ thống cơ sở dữ liệu được ưa chuộng.

**Công nghệ Web**

ASP.NET Core API sẽ được sử dụng để xây dựng phần back-end, trong khi React sẽ được sử dụng để phát triển giao diện người dùng. Điều này sẽ đảm bảo tính tương thích và hiệu suất cao cho ứng dụng của bạn.

Để duy trì tính tương thích về sau, React sẽ được sử dụng cho việc viết mã phía máy khách. Điều này sẽ giúp tạo ra một giao diện người dùng linh hoạt và hiệu quả hơn.

**Hỗ trợ Trình duyệt**

Giao diện người dùng web sẽ hỗ trợ Chrome và sau này sẽ trở nên linh hoạt hơn. Các trình duyệt phổ biến nhất phải được kiểm thử kỹ lưỡng và triển khai.

**Ràng buộc**

- Vai trò và quyền hạn người dùng sẽ được triển khai bằng cách kiểm soát quyền truy cập dựa trên vai trò (RBAC).

- Backend API sẽ được phát triển bằng ASP.NET Core và C#, giới hạn việc sử dụng các ngôn ngữ lập trình khác.

- Frontend sẽ được viết bằng React.

- Hệ thống quản lý cơ sở dữ liệu sẽ là SQL Server Management Studio 19.

- Trang web sẽ hỗ trợ các phiên bản mới nhất của các trình duyệt web phổ biến, như Google Chrome, Mozilla Firefox và Microsoft Edge.

**Môi trường Hệ thống**

- Framework Phát triển: ASP.NET Core, React

- Ngôn ngữ Lập trình: C#

- Hệ thống Quản lý Cơ sở dữ liệu: SQL Server Management Studio 19

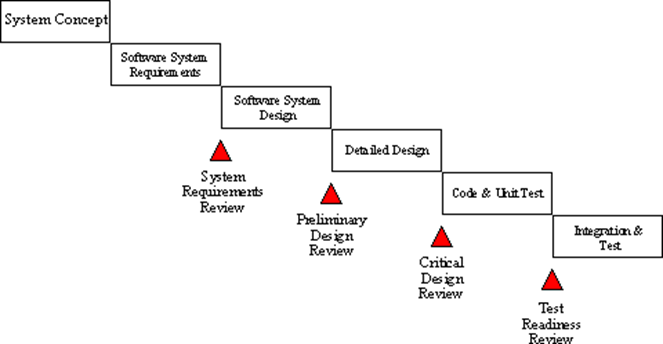
- Máy chủ Web: IIS (Dịch vụ Thông tin Internet)

- Công cụ Phát triển: Visual Studio, SQL Server Management Studio

**Phương pháp Thiết kế**

Phương pháp thiết kế theo phương pháp "waterfall" được chọn cho dự án này, vì nó dường như phù hợp nhất với quá trình thiết kế. Một mô tả ngắn về phương pháp "waterfall":

"Tất cả các dự án có thể được quản lý tốt hơn khi chia thành một cấu trúc phân cấp của các phần như các giai đoạn, các bước, các hoạt động, các nhiệm vụ và các bước. Trong các dự án phát triển hệ thống, bản diễn giản nhất của phương pháp này được gọi là phương pháp "waterfall", như được thể hiện trong hình vẽ dưới đây [1]:



Hình 2.1 Mô hình thác nước

Nhìn vào biểu đồ này, mà được thiết kế cho các hệ thống quốc phòng lớn, hãy lưu ý rằng điều này giả định rằng yêu cầu hệ thống đã được xác định và kiểm tra kỹ lưỡng, đây có lẽ là bước quan trọng nhất đối với sự thành công của dự án.

Tuy nhiên, biểu đồ này minh họa một số nguyên tắc quan trọng của một phương pháp phát triển tốt:

- Công việc được thực hiện theo từng giai đoạn,

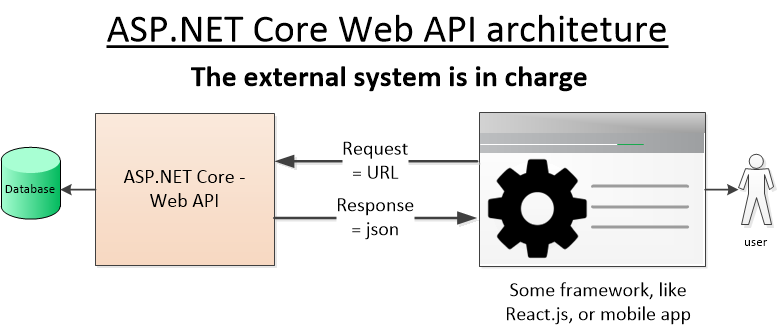
- Kiểm tra nội dung được tiến hành giữa các giai đoạn, và

- Kiểm tra đại diện cho cổng chất lượng và điểm quyết định cho việc tiếp tục.

Phương pháp "waterfall" cung cấp một chuỗi phát triển có trình tự và giúp đảm bảo tính đầy đủ của tài liệu và kiểm tra thiết kế để đảm bảo chất lượng, đáng tin cậy và khả năng bảo trì của phần mềm phát triển. Mặc dù hiện nay hầu hết mọi người đều phê phán "phương pháp waterfall" là chậm chạp và cồng kềnh, nhưng nó thực sự minh họa một số nguyên tắc cơ bản của quá trình phát triển vòng đời." Phương pháp thiết kế này sẽ được sử dụng cho một phương pháp tiếp cận theo giai đoạn. Không có giai đoạn nào có thể bắt đầu, cho đến khi giai đoạn chính của nó đã hoàn thành, ngoại trừ Kiểm thử và Sửa lỗi, có thể chạy đồng thời.

## Kiến trúc hệ thống

### Xây dựng kiến trúc



Hình 2.2 Kiến trúc ASP.NET CORE WEB API

Kiến trúc của dự án đi theo thiết kế sử dụng ASP.NET CORE Web API là Backend, còn React sẽ là Frontend.

Operating System: Windows 11 Pro

Database: SQL Server Management Studio 19

Web Server: IIS (Internet Information Services)

Code: React & C#

Backend được thiết kế với mô hình 3 tầng với 3 project chính, theo mô hình IunitOfWork pattern.

* Core: Chứa các lớp model, định nghĩa cấu trúc cơ sở dữ liệu và các DTO (Data Transfer Objects) để truyền dữ liệu giữa các tầng.
* Repository: Chứa interface IUnitOfWork và các query theo Repository của từng model. Điều này giúp tách biệt logic truy cập dữ liệu và cung cấp một cách cấu trúc để thao tác với cơ sở dữ liệu.
* API: Chứa các controller và logic xử lý, sử dụng IUnitOfWork để tương tác với cơ sở dữ liệu. Điều này giúp tách biệt logic kinh doanh và truy cập dữ liệu.

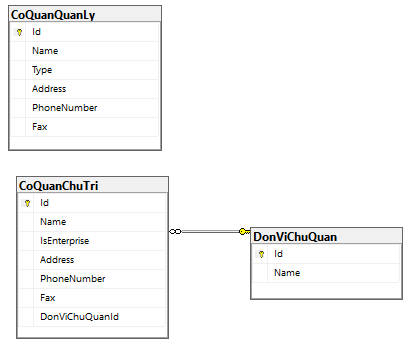


Hình 2.3 IUnitOfWork pattern

### Thiết kế cơ sở dữ liệu

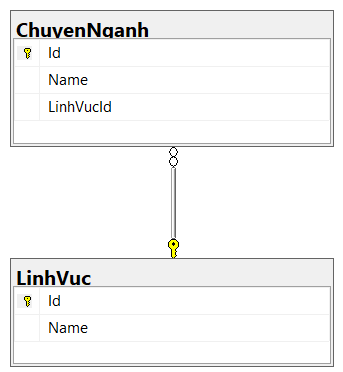
Để dễ quản lý các model, chúng sẽ được phân chia dựa trên nhóm và chức năng của chúng dựa trên yêu cầu nghiệp vụ.

Nhóm Department:



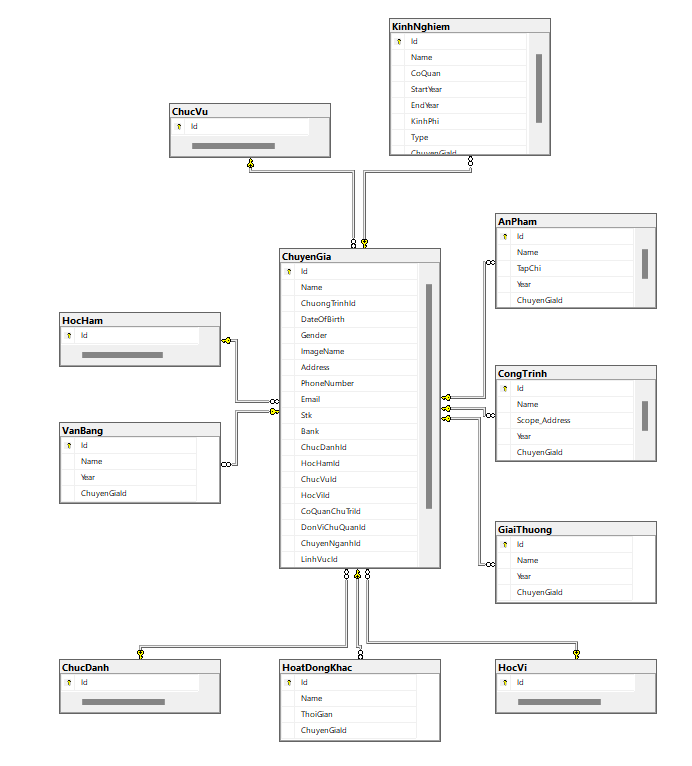
Hình 2.4 DepartmentModel

Nhóm FieldModel:



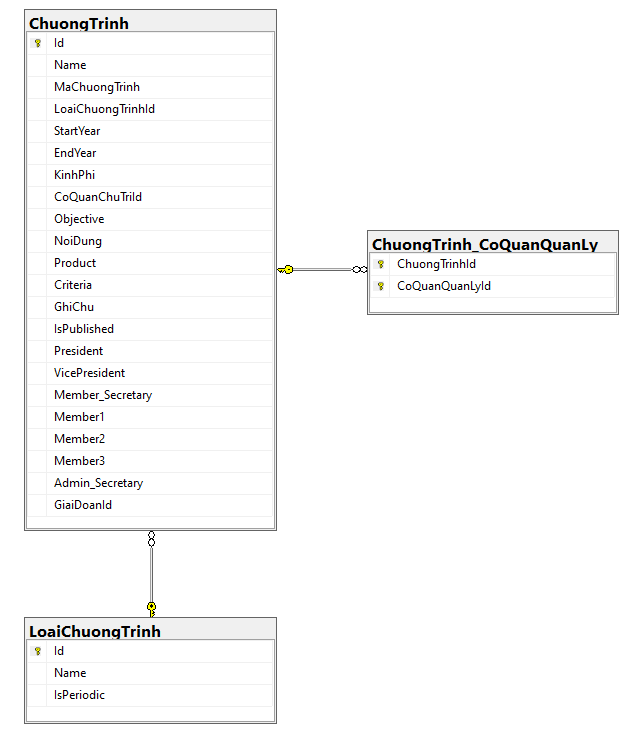
Hình 2.5 FieldModel

Nhóm ExpertModel:



Hình 2.6 ExpertModel

Nhóm Program:

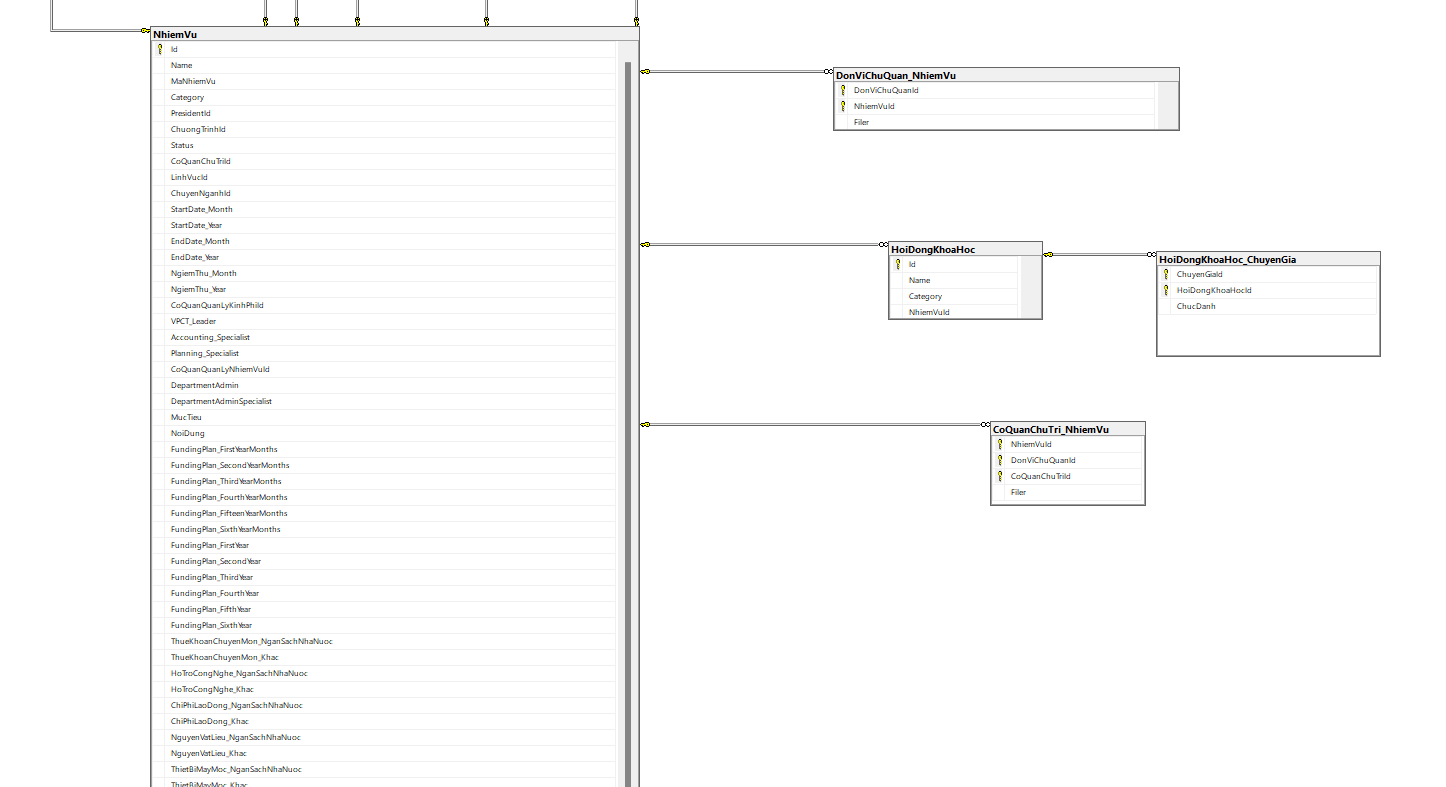


Hình 2.7 ProgramModel

Nhóm Task

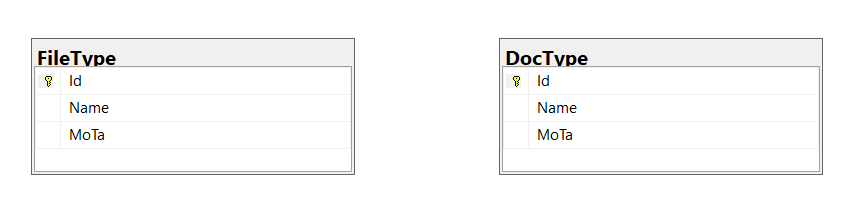


Hình 2.8.1 TaskModel(first half)



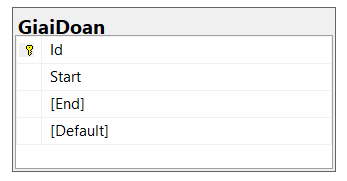
Hình 2.8.2 TaskModel(second half)

Nhóm DocType&FileType



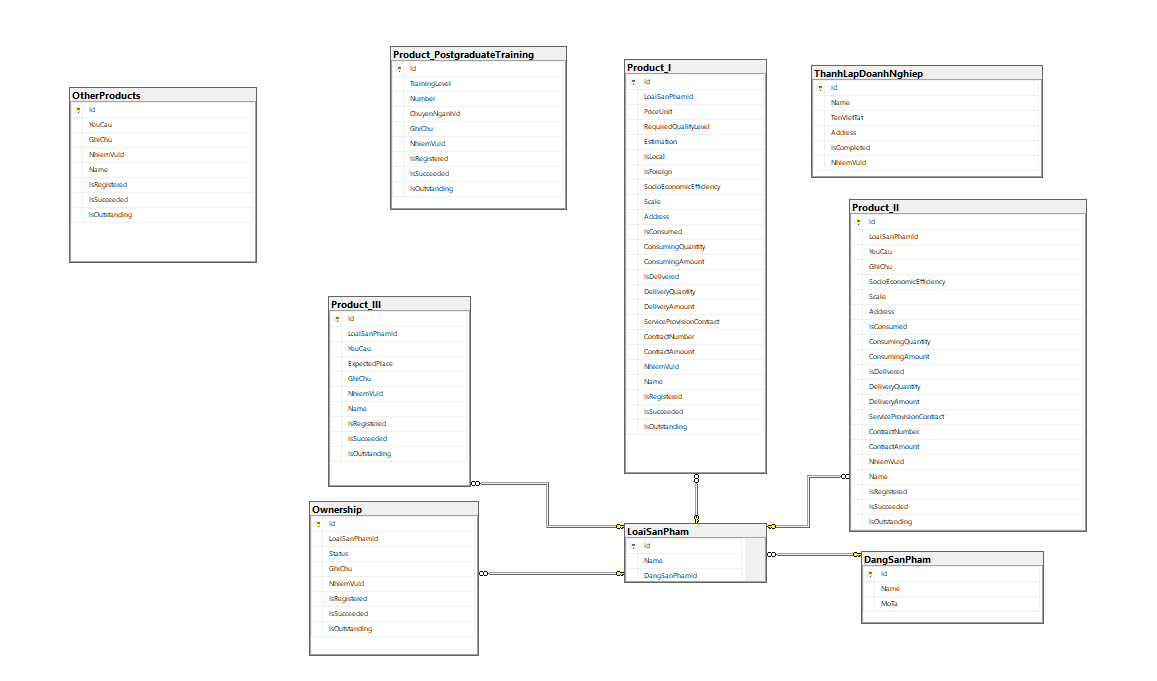
Hình 2.9 DocType&FileTypeModel

Nhóm GiaiDoan



Hình 2.10 GiaiDoanModel

Nhóm Product



Hình 2.11 ProductModel

### Thiết kế logic xử lý

API sử dụng container dependency injection (DI) tích hợp sẵn được cung cấp bởi ASP.NET Core. DI được sử dụng để quản lý các phụ thuộc giữa các thành phần khác nhau của ứng dụng. Nó thúc đẩy sự liên kết lỏng lẻo và cho phép dễ dàng kiểm thử và bảo trì hơn.

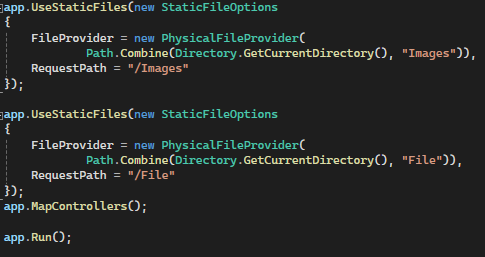
Ngoài ra, có thể thêm các thành phần:

• Middleware: Các thành phần Middleware được sử dụng để xử lý các vấn đề chung, như ghi nhật ký, xác thực, và xử lý ngoại lệ.

• Dịch vụ Bên ngoài: API có thể tích hợp với các dịch vụ bên ngoài, như máy chủ cơ sở dữ liệu, dịch vụ email, hoặc các API của bên thứ ba. Những dịch vụ này thường được truy cập thông qua các client dịch vụ hoặc repository.

Các endpoints trong API sẽ cung cấp các action để thực hiện nhiệm vụ:

1. CRUD Operations: Các hoạt động CRUD (Create, Read, Update, Delete) trên các đối tượng trong hệ thống.
2. Upload ảnh: Hệ thống sẽ cung cấp một endpoint để cho phép người dùng tải lên các hình ảnh và lưu trữ chúng trong cơ sở dữ liệu hoặc hệ thống tệp.
3. Upload và Download file: API hỗ trợ các action để tải lên và tải xuống các tệp từ hệ thống, cung cấp khả năng lưu trữ và truy cập các tệp liên quan đến dự án.



1. Hiển thị thông tin tìm kiếm của model theo DTO: Các endpoint sẽ hỗ trợ việc truy xuất thông tin từ cơ sở dữ liệu và hiển thị theo định dạng DTO (Data Transfer Object) để đảm bảo rằng dữ liệu được trả về đáp ứng đúng yêu cầu của người dùng.

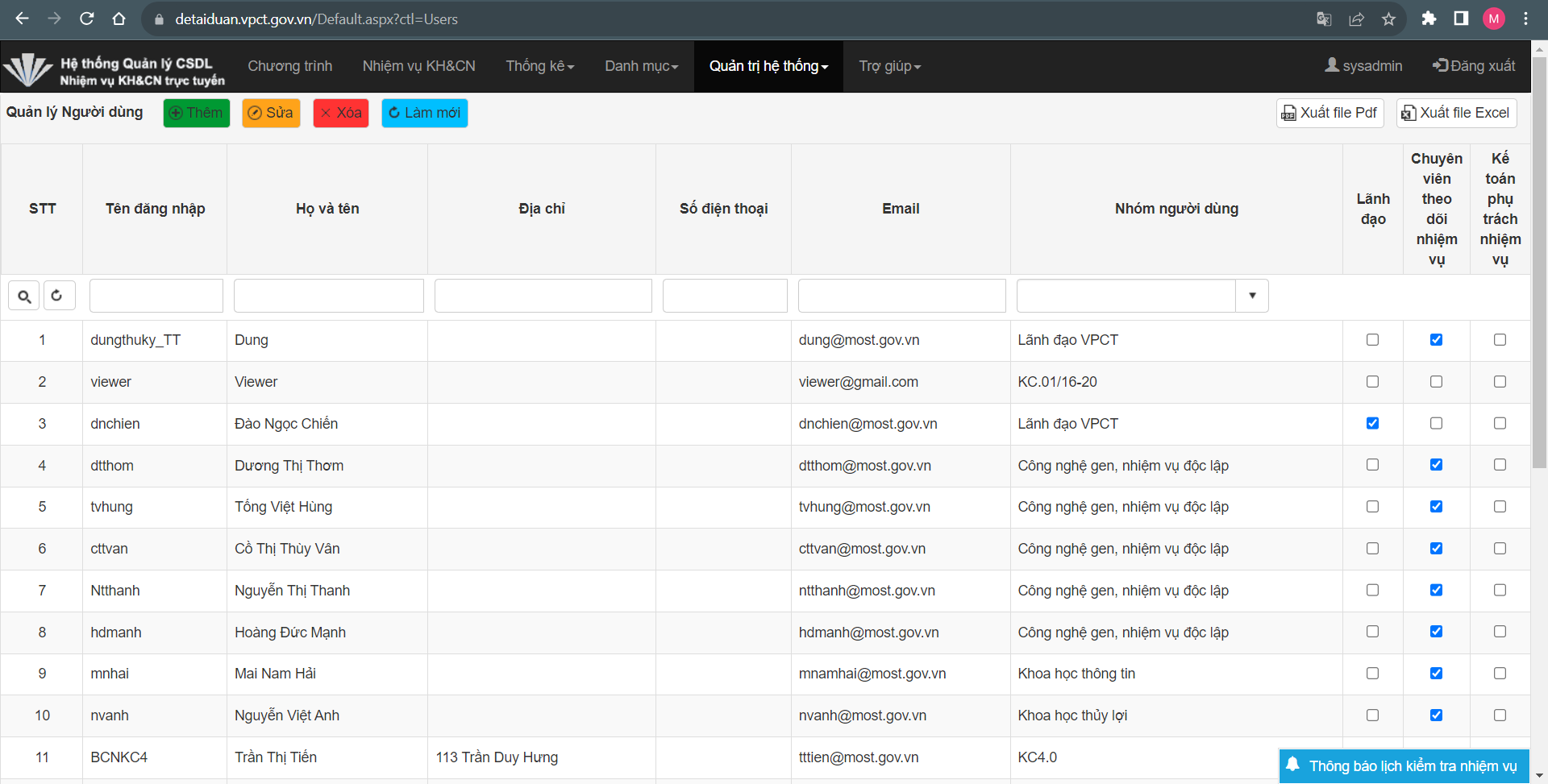
### Thiết kế bảo mật

Phần thiết kế bảo mật cho dự án bao gồm các yếu tố sau:

Xác thực người dùng: Người dùng sẽ phải có tài khoản mới có thể truy cập vào trang web. Điều này đảm bảo rằng chỉ người dùng được xác thực mới có thể truy cập vào các tính năng của hệ thống.

Sử dụng JWT và ASP.NET Core Identity: Hệ thống sẽ sử dụng JSON Web Tokens (JWT) để xác thực và ủy quyền người dùng. ASP.NET Core Identity sẽ được tích hợp để quản lý thông tin người dùng, bao gồm xác thực và quản lý tài khoản.

Phân quyền theo vai trò (role-based authorization): Do thời gian thực hiện của dự án có hạn, việc phân quyền chỉ có thể đáp ứng theo vai trò. Điều này có nghĩa là các tính năng và tài nguyên sẽ được bảo vệ dựa trên vai trò của người dùng, bao gồm Lãnh đạo, Chuyên viên và Kế toán.



Hình 2.12 Tính năng phân quyền

Mặc dù phân quyền theo nhóm và phân quyền theo permission không được triển khai trong giai đoạn đầu của dự án, nhưng kiến trúc bảo mật có thể được mở rộng sau này để hỗ trợ các yêu cầu phức tạp hơn về phân quyền.

# **Triển khai và kiểm thử**

## Triển khai

### Phạm vi triển khai

* Quản lý thông tin dự án, tiến độ, nguồn lực, chuyên gia, ngân sách, và các chức năng quản lý khác.
* Đảm bảo các báo cáo và thống kê vẫn có sẵn và hoạt động một cách hiệu quả trong hệ thống mới.
* Đảm bảo tính bảo mật và quản lý người dùng hiệu quả.
* Sử dụng Axios để gửi các yêu cầu HTTP từ ứng dụng React tới API và xử lý các phản hồi trả về.

### Lập kế hoạch

Trong phần này, chúng ta sẽ xác định kế hoạch cụ thể cho quá trình triển khai. Đầu tiên, chúng ta sẽ xác định các bước chuẩn bị, bao gồm việc cài đặt môi trường triển khai, cấu hình máy chủ, và chuẩn bị cơ sở dữ liệu mới. Sau đó, chúng ta sẽ xác định các bước triển khai cụ thể, bao gồm việc triển khai mã nguồn, cài đặt và cấu hình hệ thống mới, và kiểm tra tính năng.

### Seed data

Chuyển đổi dữ liệu từ định dạng cũ sang mới, và xác nhận tính toàn vẹn và chính xác của dữ liệu sau khi di chuyển.

## Kiểm thử

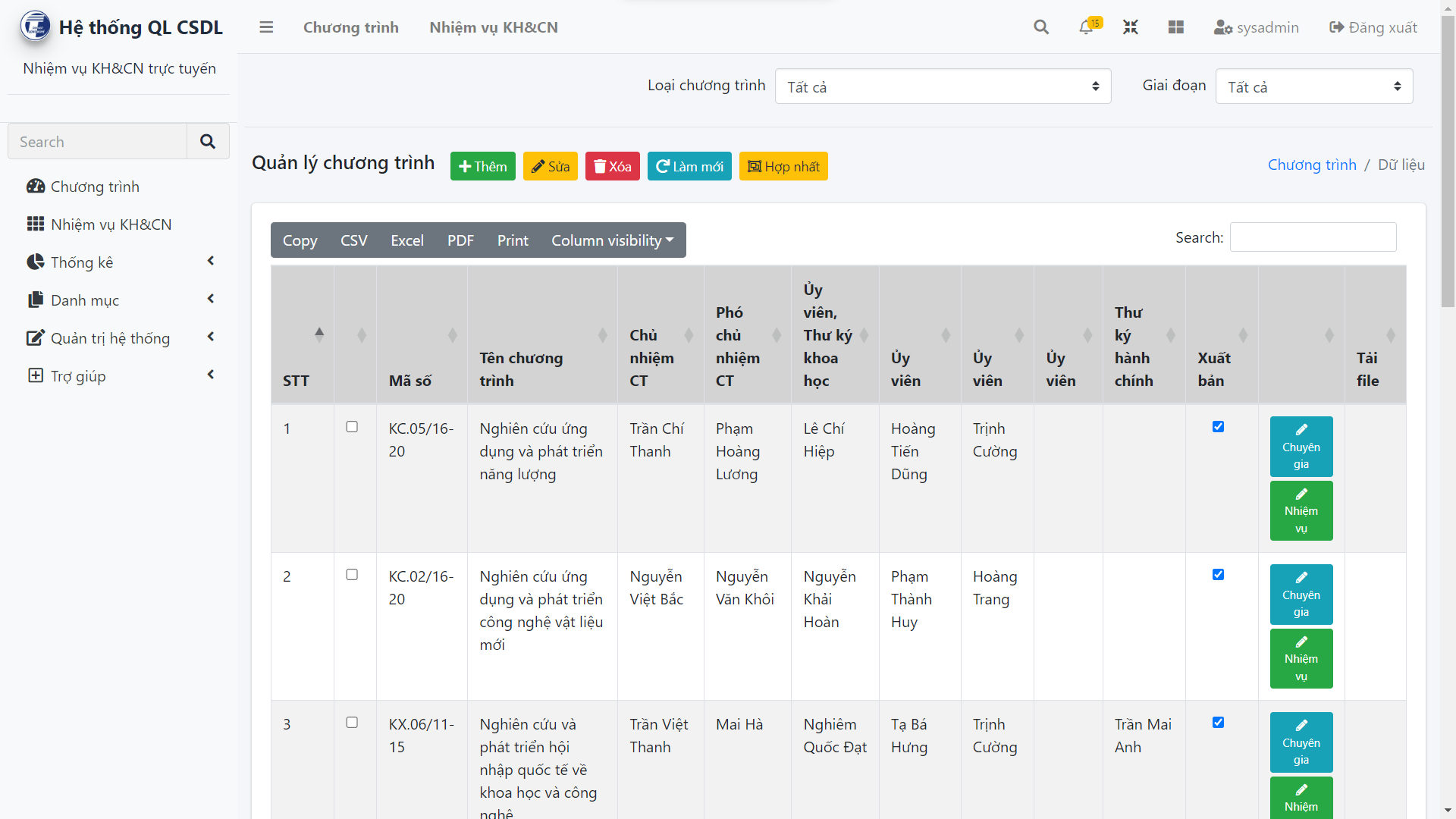
Trước khi triển khai hệ thống mới, chúng ta sẽ tiến hành kiểm thử kỹ lưỡng để đảm bảo tính ổn định và chất lượng của hệ thống. Chúng ta sẽ tận dụng các tính năng của Swagger để thực hiện kiểm thử API một cách hiệu quả.

Sử dụng Swagger để Kiểm thử API: Swagger cung cấp giao diện người dùng tương tác (UI) để thử nghiệm các endpoint API một cách trực quan. Chúng ta sẽ sử dụng giao diện này để gửi các yêu cầu tới API, kiểm tra các phản hồi trả về, và đảm bảo tính đúng đắn của các chức năng API.

Kiểm thử Tính năng và Hiệu suất: Ngoài việc sử dụng Swagger để kiểm thử cú pháp và logic của API, chúng ta cũng sẽ thực hiện kiểm thử tính năng và hiệu suất của hệ thống. Điều này bao gồm việc xác nhận rằng các chức năng hoạt động đúng như mong đợi và hệ thống có thể xử lý tải cao một cách hiệu quả.

Kiểm thử Bảo mật và Tính toàn vẹn của Dữ liệu: Swagger cũng có thể được sử dụng để kiểm thử bảo mật của API, bao gồm xác thực, phân quyền và bảo mật dữ liệu. Chúng ta sẽ đảm bảo rằng API được bảo vệ an toàn và dữ liệu được truy cập và xử lý một cách an toàn.

# **Kết quả**

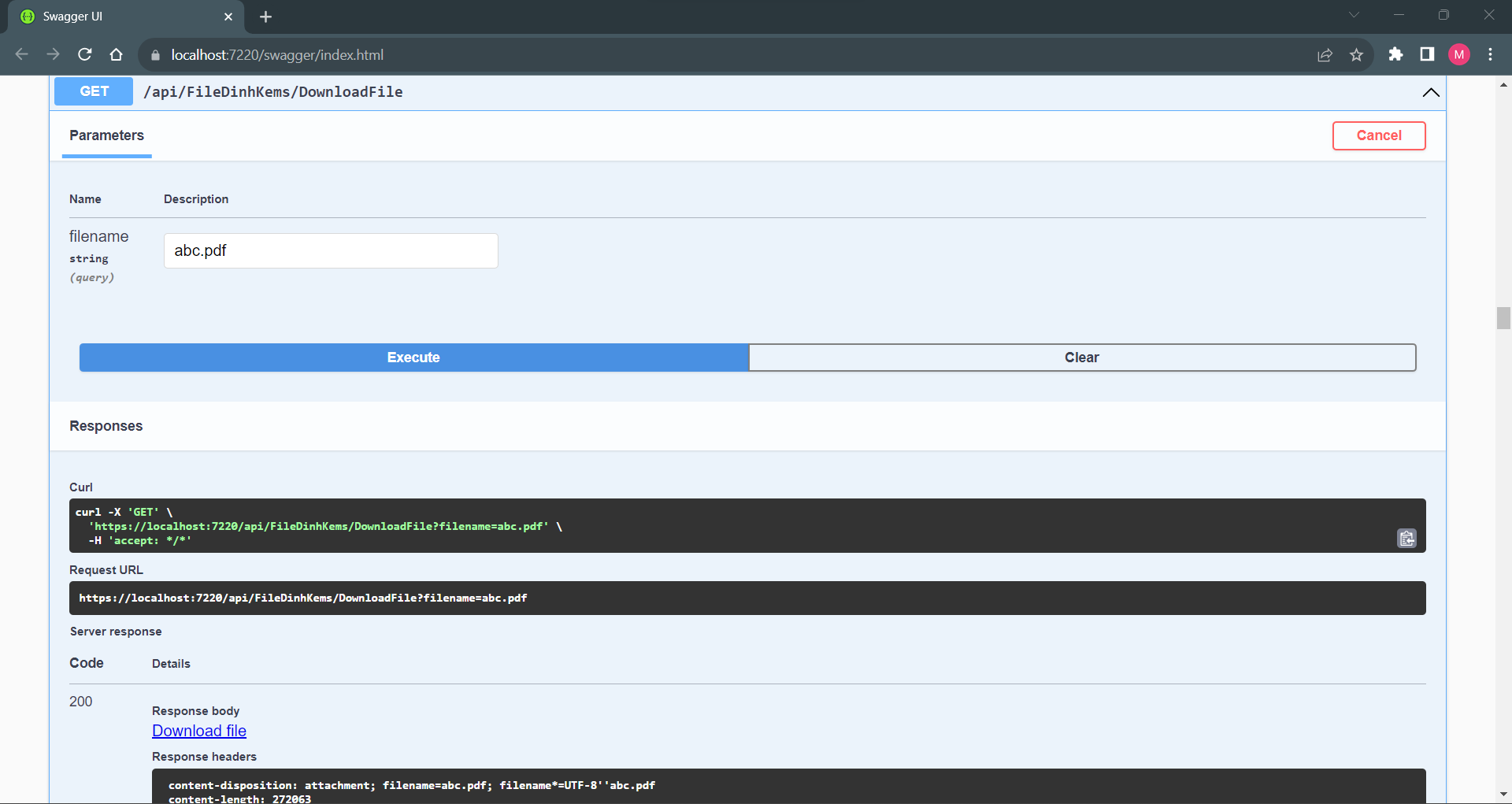


Hình 4.1 Sản phẩm cuối cùng

Có thể thấy một số cải tiến về Frontend. Trong đó:

* Thanh Navbar được chuyển thành cột, giúp tạo không gian thoáng hơn cho quy trình nghiệp vụ người dùng
* Một số nhược điểm css như con trỏ chuột được khắc phục.
* Thanh Navbar chủ yếu dùng cho hoạt động chính như Indexing các mục dữ liệu quan trọng, thông báo và quản lý tài khoản.

Hoạt động Backend có thể dễ dàng được bảo trì và nâng cấp, nhờ việc xây dựng theo một mô hình cụ thể.



Hình 4.2 Download file Request trong Swagger

**Tài liệu tham khảo**

|  |  |
| --- | --- |
| [1] | B. C. Green, "eCommerce Site and Business Management Portal," Brian C. Green, Philadelphia, USA, 2007. |