TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Tên đề tài: Phát triển ứng dụng web quản lý kho hàng

GVHD: PGS.TS Lê Đức Hậu

Nhóm 11 : Lê Viết Đức Anh – 20226075
Nguyễn Duy Anh – 20226103
Hồ Tuấn Anh – 20226100
Lê Phan Anh – 20226102
Đào Kiên Cường – 20226104
Vũ Duy Dũng – 20226078
Nguyễn Tài Hoan – 20226106
Nguyễn Hữu Khải – 20200310
Đỗ Đình Trung – 20226128

Lớp : (LT) LTHĐT Mã lớp học: 151964

Hà Nội, 01/ 2025

MỤC LỤC

Chương 1 Giới thiệu đề tài

1.1 Đặt vấn đề

Trong bối cảnh cạnh tranh gay gắt và sự chuyển đổi nhanh chóng của nền kinh tế, việc quản lý hiệu quả hàng hóa và nguyên vật liệu trở thành yếu tố cốt lõi để đảm bảo sự thành công và tối ưu hóa hoạt động kinh doanh. Quản lý kho hàng không chỉ đơn thuần là việc kiểm soát số lượng sản phẩm, mà còn bao gồm các khía cạnh quan trọng như lập kế hoạch nhập xuất hàng hóa, phân phối tài nguyên, đảm bảo cung ứng đáp ứng kịp thời nhu cầu thị trường. Tuy nhiên, trong thực tế, nhiều doanh nghiệp đang đối mặt với các thách thức như khó khăn trong việc kiểm soát số liệu tồn kho, thiếu minh bạch thông tin, cũng như khó theo dõi chính xác các luồng hoạt động. Ngoài ra, các doanh nghiệp cũng cần một công cụ hiệu quả để tổ chức lịch trình và theo dõi công việc của đội ngũ liên quan. Xuất phát từ nhu cầu thực tiễn này, chúng em đã lựa chọn đề tài "phát triển hệ thống web quản lý kho hàng".

1.2 Mục tiêu và phạm vi đề tài

Hiện nay, đã có rất nhiều hệ thống quản lý tồn kho được triển khai, chẳng hạn như Zoho Inventory, Odoo Inventory, hay SAP. Tuy nhiên, phần lớn các nền tảng này yêu cầu trả phí để truy cập đầy đủ các tính năng, đồng thời thường không được tùy chỉnh linh hoạt theo nhu cầu của từng doanh nghiệp vừa và nhỏ. Với đề tài “phát triển hệ thống web quản lý kho hàng,” chúng em đặt mục tiêu xây dựng một hệ thống đáp ứng đủ các chức năng cơ bản, đồng thời miễn phí sử dụng và dễ dàng tích hợp, tùy chỉnh theo yêu cầu người dùng.

Hệ thống sẽ bao gồm các tính năng cơ bản như:

- Thêm, sửa, xoá danh mục sản phẩm và thông tin hàng hóa, nhà cung cấp cũng như đối tác.
- Quản lý phiếu nhập kho, xuất kho.
- Báo cáo số liệu, giúp phân tích xu hướng tồn kho theo thời gian.

Chúng em kỳ vọng hệ thống sẽ không chỉ đáp ứng các yêu cầu quản lý tồn kho cơ bản mà còn góp phần giúp các doanh nghiệp tối ưu hóa hoạt động kinh doanh mà không cần tốn kém nhiều chi phí vận hành.

1.3 Định hướng giải pháp

a) Định hướng, phương pháp, công nghệ sử dụng:

Dự án sẽ được xây dựng dựa trên công nghệ web application với kiến trúc client-server. Frontend được phát triển bằng HTML, CSS, JS để tạo giao diện người dùng thân thiện và phản hồi nhanh. Backend sử dụng Spring Boot để xây dựng các API RESTful phục vụ logic nghiệp vụ và giao tiếp với cơ sở dữ liệu. Postman sẽ được sử dụng nhằm mục đích kiểm thử API được viết ở trên. Dữ liệu sẽ được lưu trữ và quản lý bằng PosgresSQL.

b) Mô tả giải pháp:

Giải pháp của chúng em là xây dựng một hệ thống quản lý tồn kho miễn phí, thân thiện và dễ sử dụng, tập trung vào các tính năng cơ bản để hỗ trợ các doanh nghiệp nhỏ và vừa trong việc quản lý hàng hóa và nhà cung cấp. Hệ thống sẽ đảm bảo sự linh hoạt, ổn định và mang lại trải nghiệm sử dụng tốt thông qua giao diện web. Hệ thống cũng tích hợp một số tính năng mở rộng như: Tự động thông báo tình trạng tồn kho khi quá nhiều, phân quyền người dùng theo vai trò (quản lý, nhân viên,...) để bảo vệ tính bảo mật, hỗ trợ thống kê động giúp doanh nghiệp dễ dàng nắm bắt tình trạng tồn kho và luồng hoạt động theo thời gian.

c) Đóng góp chính và kết quả đạt được:

Đóng góp chính: Cung cấp một hệ thống quản lý kho hàng miễn phí, dễ sử dụng, đáp ứng nhu cầu của các doanh nghiệp vừa và nhỏ mà không yêu cầu chi phí để sử dụng đầy đủ tính năng.

Kết quả đạt được: Hệ thống quản lý kho hàng hỗ trợ theo dõi sản phẩm, tối ưu hóa quản lý nhập - xuất kho, và cung cấp báo cáo trực quan, giúp doanh nghiệp cải thiện hiệu quả vận hành và quản lý tài nguyên tốt hơn.

Chương 2 Khảo sát và phân tích yêu cầu

2.1 Khảo sát hiện trạng

Trong giai đoạn khảo sát hiện trạng, mục tiêu chính là thu thập và phân tích các thông tin cần thiết về tình hình hiện tại của các hệ thống quản lý dự án, các yêu cầu của người dùng, cũng như những ứng dụng phần mềm hiện có trên thị trường. Dưới đây là ba nguồn chính mà chúng em sẽ khảo sát để đưa ra đánh giá về hệ thống phần mềm quản lý dự án.

2.1.1 Khảo sát người dùng/khách hàng

Người dùng là yếu tố then chốt trong việc xây dựng phần mềm quản lý tồn kho. Khảo sát người dùng giúp nhận diện rõ các nhu cầu và thách thức mà họ gặp phải trong quản lý. Một số phương pháp khảo sát bao gồm:

- Phỏng vấn trực tiếp doanh nghiệp vừa và nhỏ.
- Bảng câu hỏi về khó khăn trong quản lý tồn kho và các tính năng cần thiết.

Những yêu cầu điển hình mà doanh nghiệp quan tâm:

- Quản lý sản phẩm và nhà cung cấp: Tạo, chỉnh sửa, và quản lý danh sách sản phẩm, nhà cung cấp, và thông tin liên quan.
- Quản lý kho hàng: Theo dõi tình trạng nhập - xuất hàng hóa và lượng hàng tồn kho.
- Thống kê và báo cáo: Báo cáo hàng tồn, phân tích lượng nhập/xuất theo tuần, tháng, hoặc năm.
- Thông báo tự động: Cảnh báo khi hàng tồn đạt ngưỡng tối thiểu.
- Phân quyền: Giới hạn truy cập và hành động dựa trên vai trò.

2.1.2 Các hệ thống đã có

Nghiên cứu các hệ thống quản lý kho hàng nổi bật cung cấp cái nhìn sâu sắc về tính năng phổ biến và điểm mạnh/yếu của từng hệ thống.

Hệ thống	Ưu điểm	Nhược điểm
Zoho Inventory	- Giao diện thân thiện, tích hợp dễ dàng với các nền tảng khác.	- Hạn chế các tính năng phân quyền chi tiết ở bản miễn phí.
Odoo Inventory	- Hệ thống mã nguồn mở, dễ dàng tùy chỉnh.	- Tốn thời gian thiết lập ban đầu, chi phí tùy chỉnh cao.
SAP	- Phù hợp với doanh nghiệp lớn, quản lý chi tiết theo lô.	- Phức tạp, khó tiếp cận đối với doanh nghiệp vừa và nhỏ.

Phân tích cho thấy các hệ thống hiện tại thiên về doanh nghiệp lớn, trong khi các doanh nghiệp nhỏ cần giải pháp tối giản hơn với chi phí hợp lý.

2.1.3 Các ứng dụng tương tự

Ngoài các hệ thống chính, một số ứng dụng tương tự khác cũng được phân tích:

- InFlow Inventory: Dễ sử dụng nhưng hạn chế báo cáo động.
- Sortly: Ứng dụng dành cho di động tốt nhưng thiếu hỗ trợ tích hợp hệ thống lớn.
- QuickBooks Commerce: Mạnh về kết nối thương mại điện tử nhưng có giá thành cao.

Những hệ thống và ứng dụng này có thể cung cấp nguồn cảm hứng trong thiết kế giao diện, tổ chức tính năng, đồng thời gợi ý các khía cạnh cần cải tiến để xây dựng một hệ thống quản lý tồn kho linh hoạt, hiệu quả và phù hợp hơn cho các doanh nghiệp nhỏ.

2.1.4 Mô tả sơ lược các tính năng phần mềm quan trọng cần phát triển

Dựa trên khảo sát người dùng và phân tích các hệ thống tương tự, dưới đây là các tính năng chính mà hệ thống quản lý tồn kho sẽ tập trung phát triển.

Module 1: Quản lý sản phẩm và nhà cung cấp

- Thao tác CRUD: Thêm, sửa, xóa sản phẩm và nhà cung cấp.
- Liên kết dữ liệu: Kết nối sản phẩm với nhà cung cấp (tạo bản ghi trong bảng liên kết khi chưa tồn tại).
- Hiển thị thông tin: Danh sách sản phẩm và nhà cung cấp được trình bày rõ ràng, có hỗ trợ tìm kiếm và lọc theo nhiều tiêu chí.

Module 2: Quản lý nhập - xuất kho

- Tạo phiếu nhập kho: Lưu thông tin sản phẩm nhập, số lượng, và nhà cung cấp.
- Xem chi tiết phiếu xuất kho: Hiển thị cụ thể thông tin hàng hóa xuất khỏi kho.
- Quản lý phiếu nhập - xuất: Liệt kê, tìm kiếm, và phân loại các phiếu nhập - xuất hàng theo ngày tháng hoặc sản phẩm liên quan.

Module 3: Thống kê và báo cáo

- Thống kê tồn kho: Liệt kê sản phẩm tồn nhiều nhất hoặc ít nhất để hỗ trợ quản lý tồn kho.
- Thống kê nhà cung cấp: Danh sách nhà cung cấp cung cấp nhiều sản phẩm nhất hoặc có số lượng phiếu nhập nhiều nhất.
- Xu hướng nhập - xuất: Thống kê lượng nhập - xuất hàng hóa theo các khoảng thời gian như tuần, tháng, năm.

Tổng quan giải pháp:

Hệ thống sẽ tập trung vào việc cung cấp các tính năng cơ bản nhưng hiệu quả, với giao diện thân thiện, trực quan. Phần mềm không chỉ đơn thuần là công cụ quản lý mà còn giúp doanh nghiệp dự đoán, lập kế hoạch nhập - xuất hàng hóa và tối ưu hóa nguồn lực.

2.2 Tổng quan chức năng

2.2.1 Biểu đồ use case tổng quan

Figure 2-1: Use case tổng quan

2.2.2 Biểu đồ use case phân rã XYZ

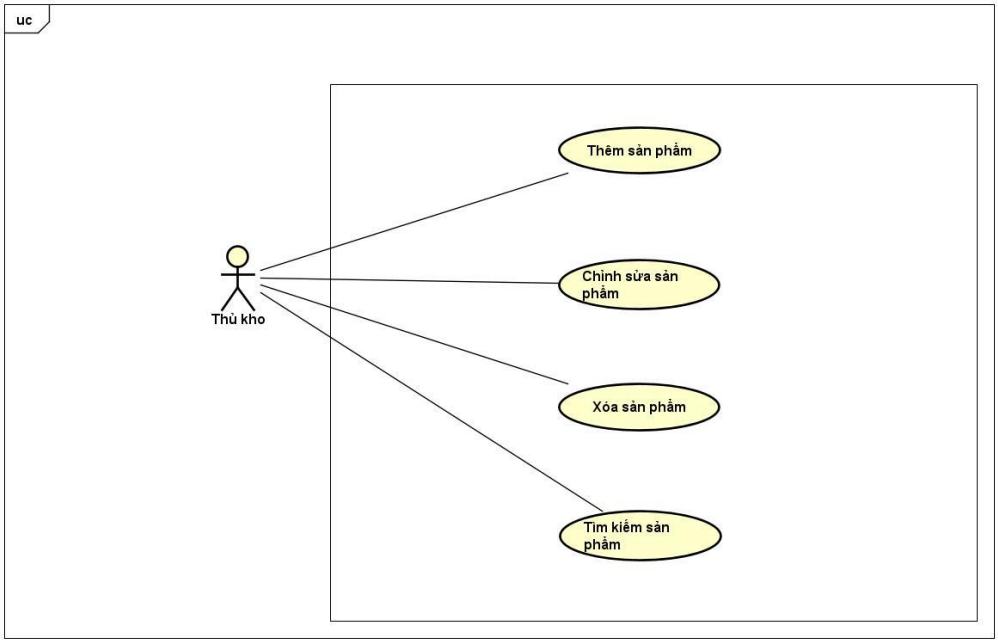


Figure 2-2: Quản lý sản phẩm

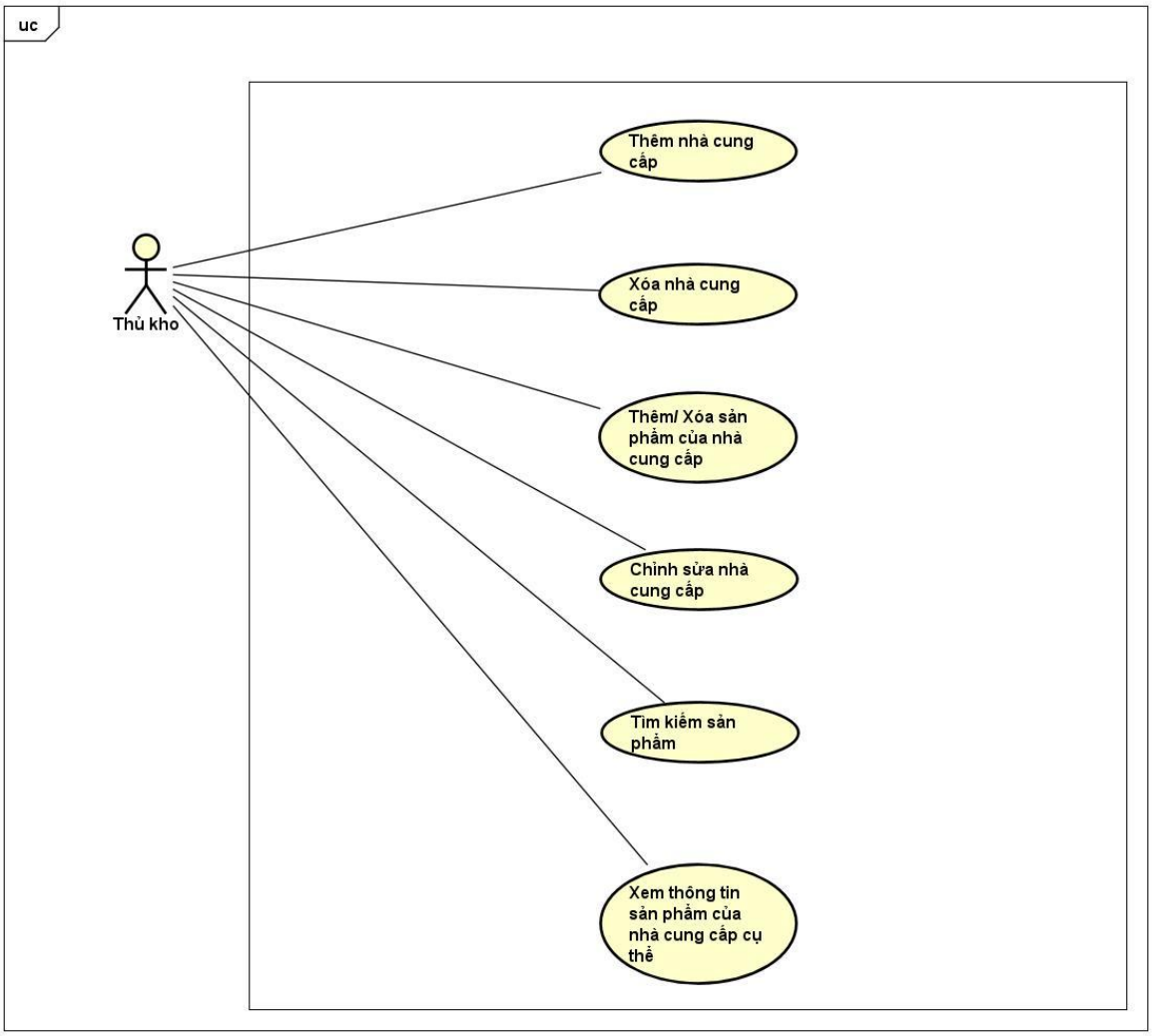


Figure 2-3: Quản lý nhà cung cấp

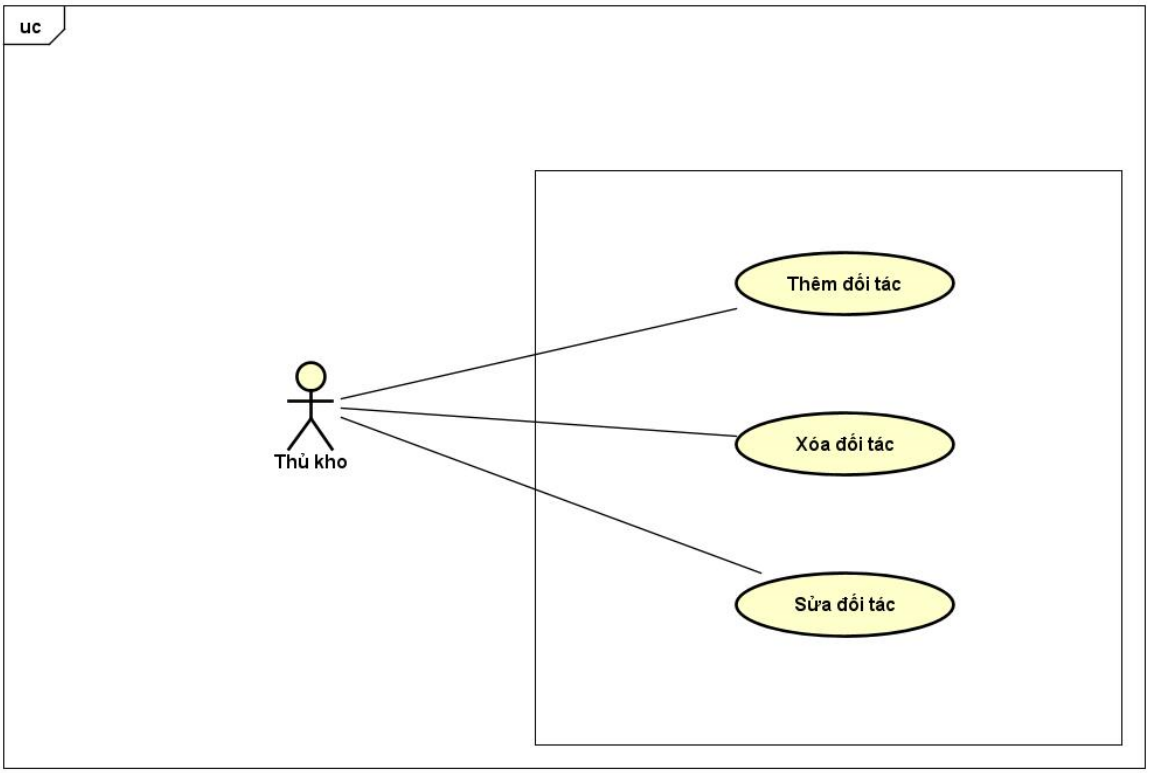


Figure 2-4: Quản lý đối tác

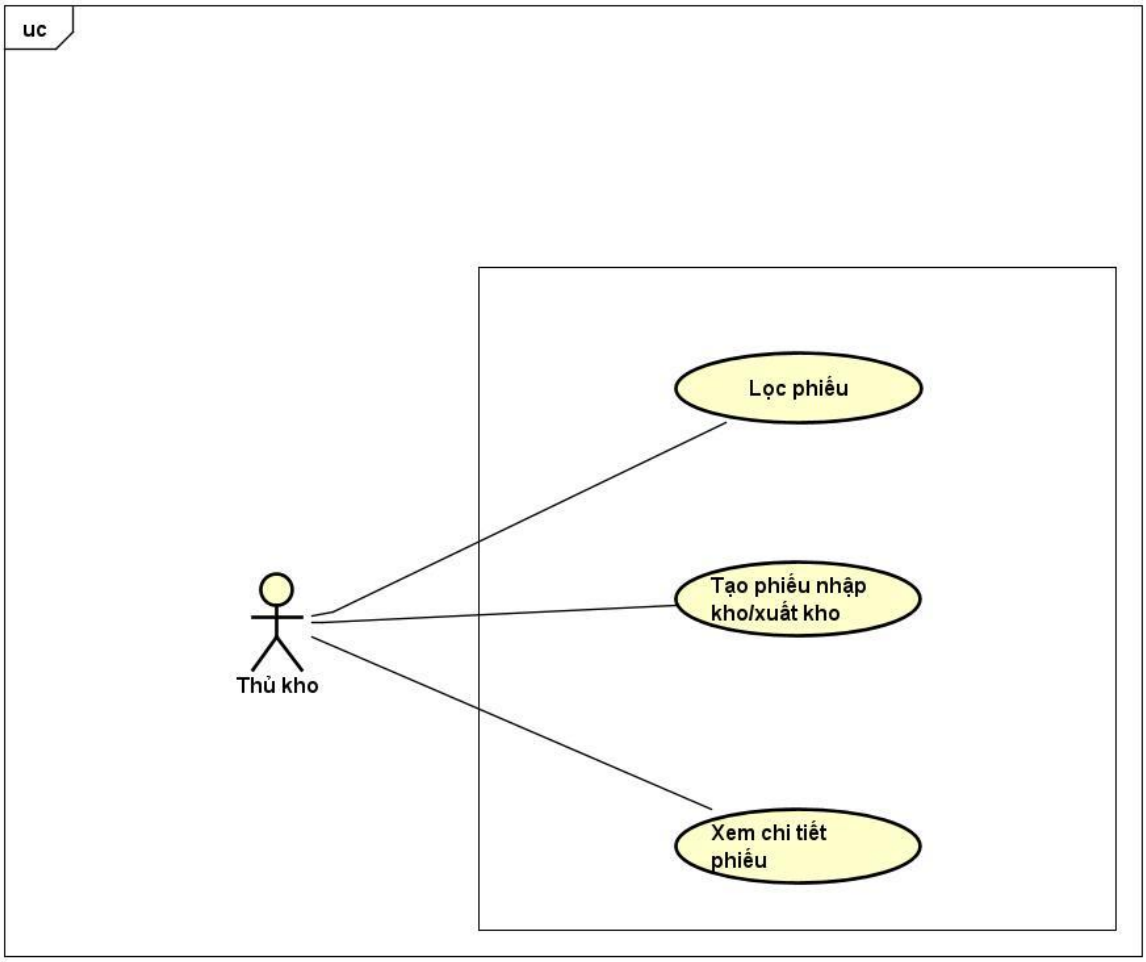


Figure 2-5: Quản lý phiếu nhập/ xuất hàng

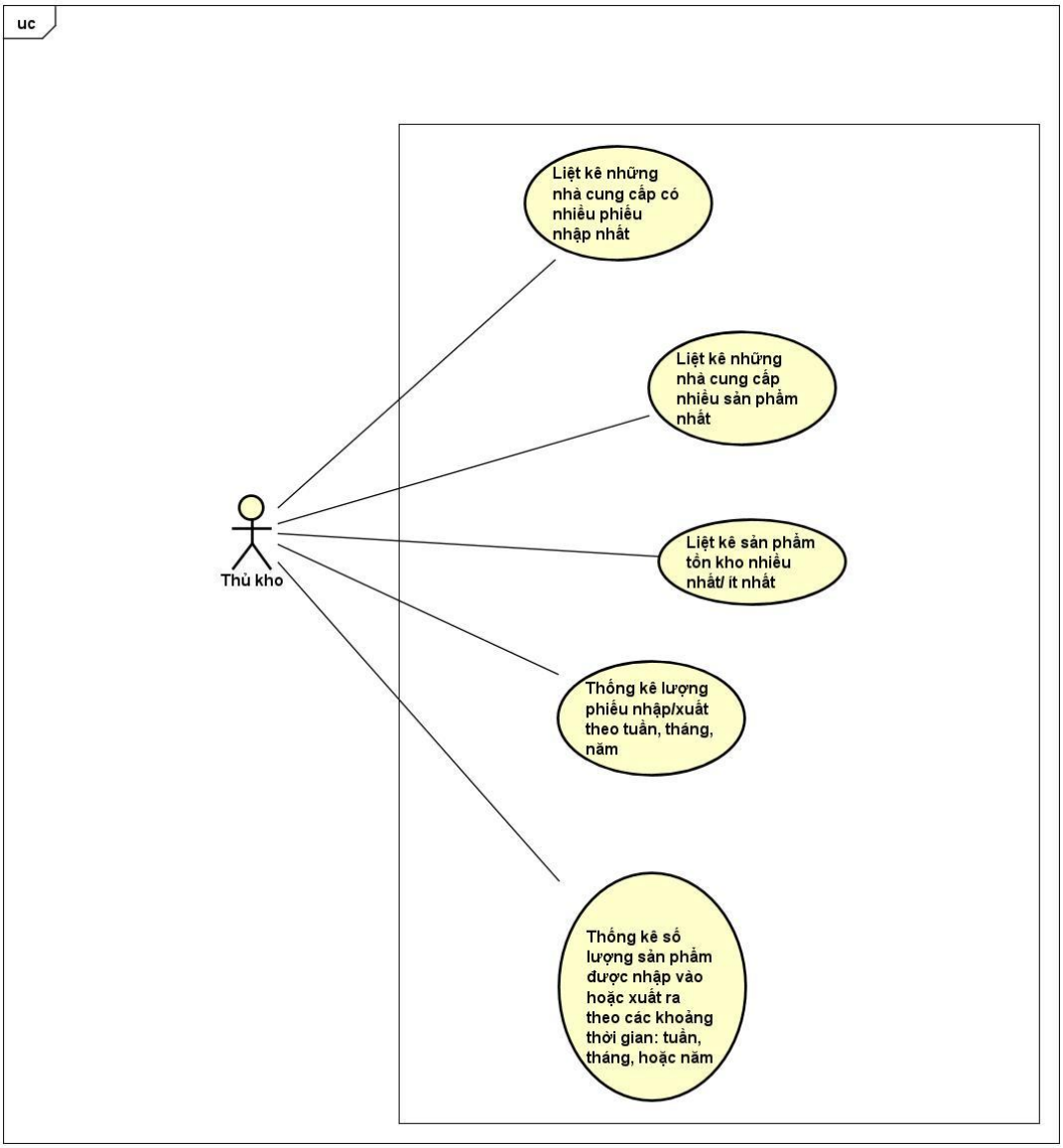


Figure 2-6: Thống kê

Chương 3 Công nghệ sử dụng

Trong chương này, chúng em sẽ trình bày về các công nghệ, công cụ và nền tảng được sử dụng để phát triển hệ thống quản lý dự án. Việc lựa chọn các công nghệ này nhằm đảm bảo hệ thống có tính hiệu quả, dễ bảo trì và khả năng mở rộng trong tương lai.

3.1 Ngôn ngữ lập trình

Java: Được sử dụng để phát triển phần backend, Java cung cấp tính ổn định, hiệu năng cao và khả năng xử lý các yêu cầu nghiệp vụ phức tạp. Java còn hỗ trợ mạnh mẽ cho việc xây dựng các API RESTful thông qua các framework như Spring Boot.

JavaScript: Là ngôn ngữ chính để phát triển phần frontend. JavaScript kết hợp với HTML, CSS giúp xây dựng giao diện người dùng thân thiện và tương tác nhanh.

3.2 Framework và thư viện

Spring: Được sử dụng để phát triển backend, Spring hỗ trợ xây dựng các API RESTful nhanh chóng và tích hợp dễ dàng với cơ sở dữ liệu. Ngoài ra, Spring còn hỗ trợ quản lý các thành phần của ứng dụng một cách hiệu quả nhờ cấu trúc module rõ ràng.

3.3 Cơ sở dữ liệu

PostgresSQL: Hệ quản trị cơ sở dữ liệu quan hệ PostgresSQL được sử dụng để lưu trữ và quản lý dữ liệu của hệ thống. PostgresSQL có khả năng xử lý dữ liệu lớn, hỗ trợ mạnh mẽ các thao tác CRUD (Create, Read, Update, Delete) và dễ dàng tích hợp với Spring Boot.

3.4 Công cụ hỗ trợ phát triển

Postman: Được sử dụng để kiểm thử các API RESTful, đảm bảo rằng backend hoạt động đúng chức năng và dữ liệu truyền tải giữa frontend và backend chính xác.

GitHub: Là công cụ quản lý mã nguồn và hỗ trợ làm việc nhóm. GitHub giúp theo dõi lịch sử thay đổi của dự án và cho phép nhiều thành viên cùng phối hợp phát triển một cách hiệu quả.

IntelliJ Idea Ultimate Edition: Là môi trường phát triển được sử dụng để phát triển backend với Java. IntelliJ Idea cung cấp các tính năng mạnh mẽ để lập trình, kiểm thử, và gỡ lỗi.

Visual Studio Code: Môi trường phát triển có nhiều thư viện và extension đa dạng, là môi trường phát triển tích hợp (IDE) được sử dụng để phát triển phần frontend với HTML và CSS và cung cấp các tính năng mạnh mẽ như gợi ý mã, kiểm tra lỗi, tích hợp dễ dàng với các tiện ích mở rộng, giúp tăng hiệu quả lập trình và đảm bảo mã nguồn dễ bảo trì.

Eclipse: Eclipse được sử dụng như một môi trường phát triển tích hợp (IDE) cho việc lập trình backend bằng Java. Công cụ này cung cấp các tính năng mạnh mẽ như gợi ý mã, quản lý dự án và hỗ trợ debug, giúp tăng hiệu quả phát triển ứng dụng và giảm thiểu lỗi trong quá trình triển khai.

3.5 Công nghệ triển khai

Render: Render là nền tảng triển khai trực tuyến được sử dụng để triển khai hệ thống quản lý dự án. Render hỗ trợ tích hợp liên tục, cung cấp môi trường ổn định và khả năng mở rộng, đồng thời dễ dàng quản lý và tối ưu chi phí cho dự án.

Jenkins: Jenkins được sử dụng để tự động hóa các quy trình triển khai và kiểm thử, đảm bảo hệ thống luôn trong trạng thái sẵn sàng hoạt động. Công cụ này hỗ trợ tích hợp liên tục (CI) và triển khai liên tục (CD), giúp giảm thiểu rủi ro và tăng hiệu quả làm việc nhóm.

3.6 Lý do chọn lựa công nghệ

Tính ổn định: Các công nghệ như Java, Spring Boot và PostgresSQL được sử dụng phổ biến trong các hệ thống doanh nghiệp lớn, đảm bảo tính ổn định và bảo mật.

Dễ sử dụng và bảo trì: JS, HTML, CSS giúp dễ dàng phát triển và mở rộng giao diện.

Hiệu quả và tiết kiệm: Render và Jenkins cho phép triển khai nhanh chóng và tiết kiệm chi phí, phù hợp với các dự án sinh viên.

Chương 4 Phát triển và triển khai ứng dụng

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Kiến trúc phần mềm : kiến trúc ba lớp MVC. Trong đó:

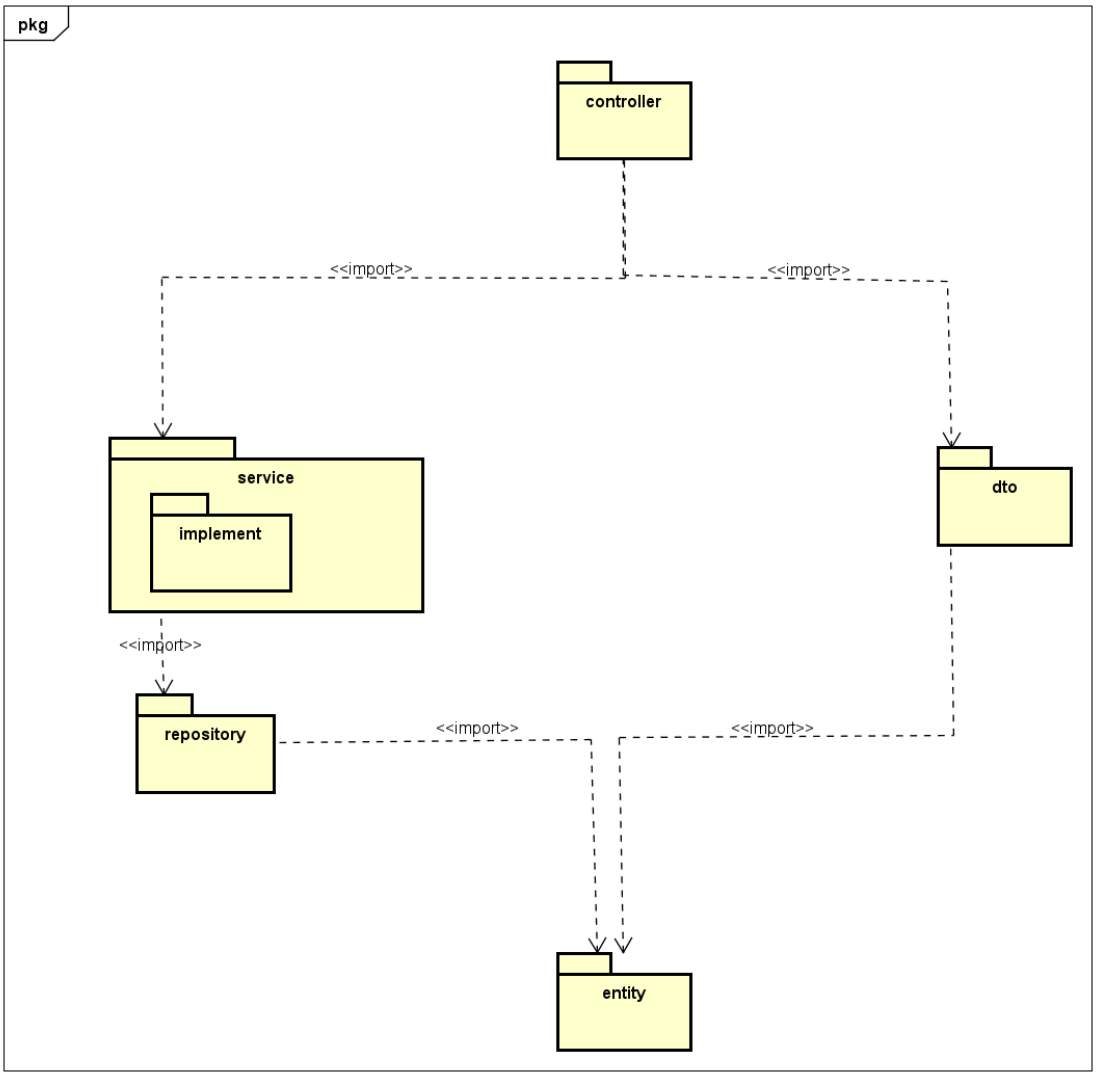
- M (Model): là nơi chứa những nghiệp vụ tương tác với dữ liệu bao gồm:
  - Lưu trữ trạng thái của ứng dụng trong các cấu trúc dữ liệu phù hợp hoặc cơ sở dữ liệu
  - Phản hồi những thay đổi của trạng thái
  - Xác thực và phân quyền người dùng

- V (View):
  - Hiển thị giao diện người dùng.
  - Gửi yêu cầu dữ liệu từ người dùng đến Controller
  - Nhận phản hồi từ Controller và cập nhật dữ liệu trên giao diện

- C (Controller):
  - Vận chuyển dữ liệu được gửi về từ View đến Model thông qua API
  - Kết xuất dữ liệu và thông báo từ Model và gửi phản hồi lên View thông qua API

4.1.2 Thiết kế biểu đồ lớp

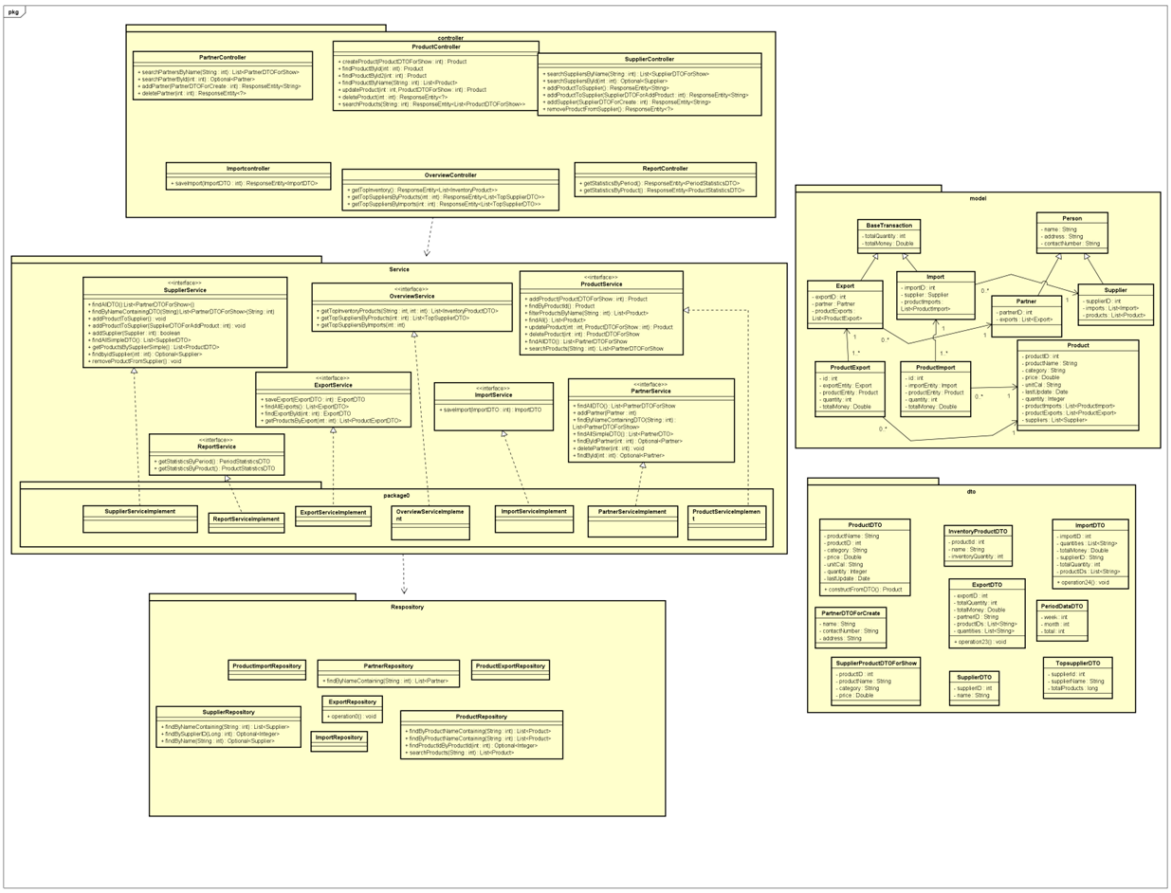
Biểu đồ gói UML:

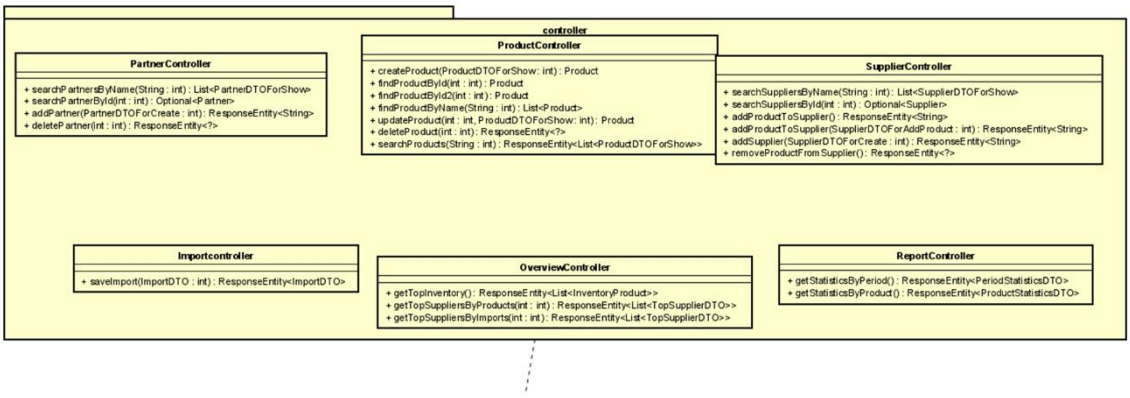


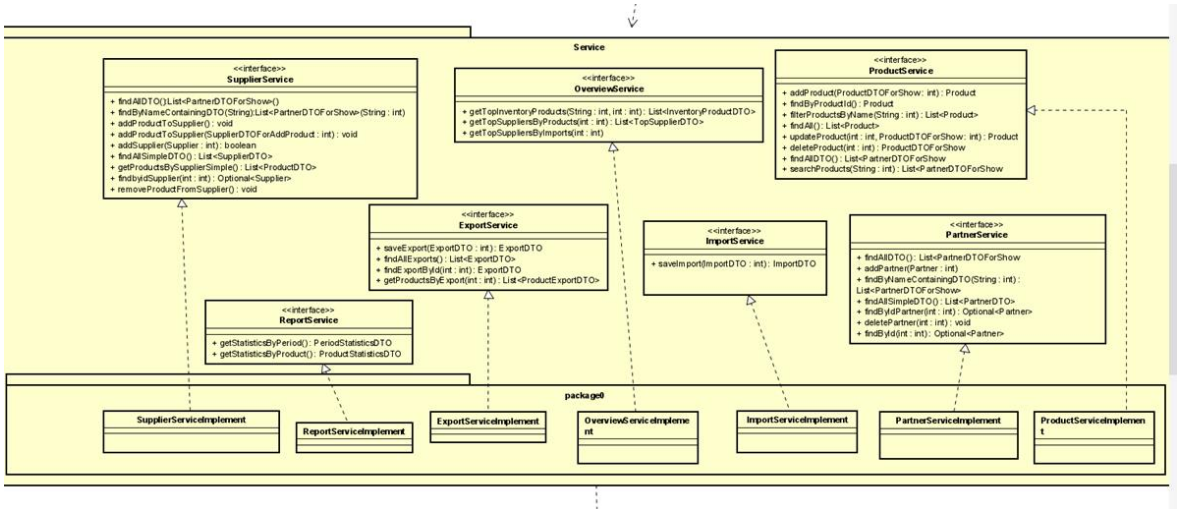
Chi tiết nhiệm vụ của từng package:

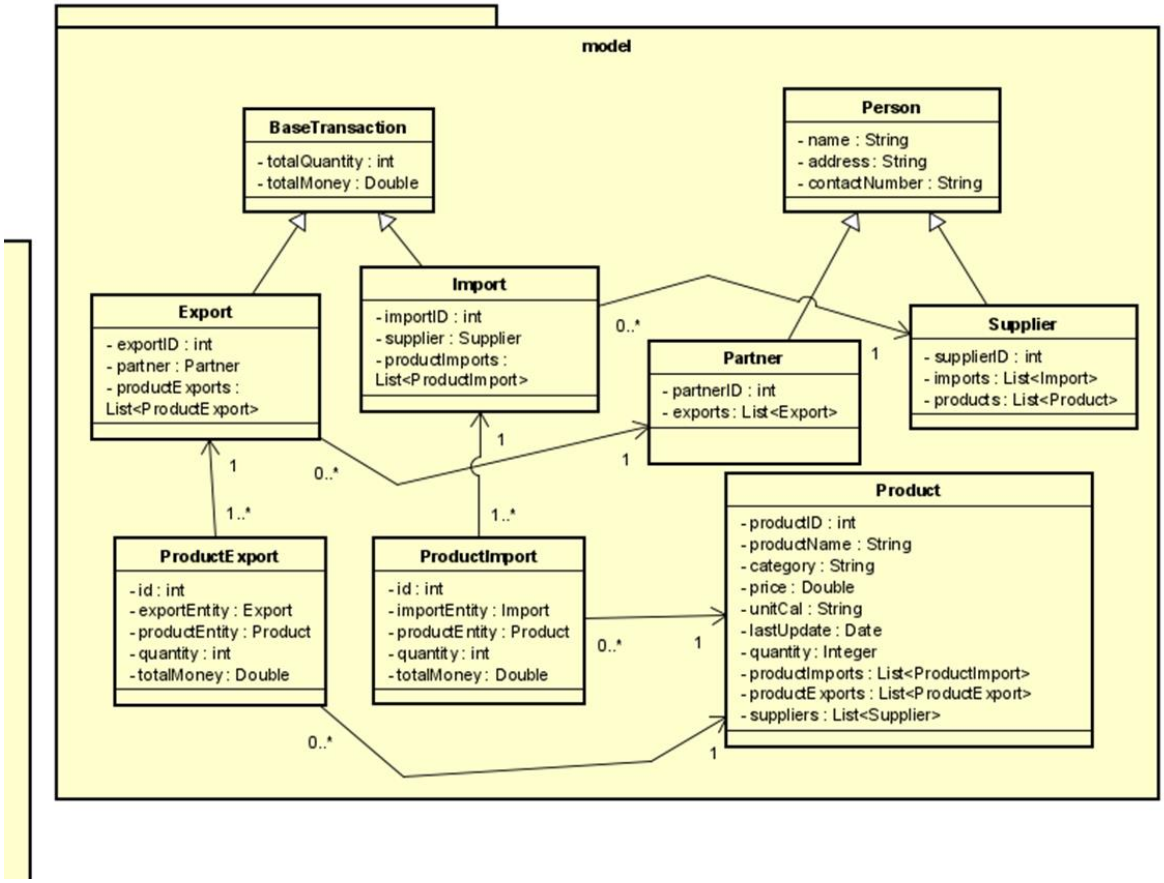
- Package entity: Chứa các lớp khai báo thuộc tính của đối tượng và ánh xạ đến các bảng trong cơ sở dữ liệu.
- Package repository: Chứa các giao diện khai báo các phương thức truy vấn dữ liệu trong database.
- Package dto: Chứa các lớp đóng gói data để chuyển giữa client – server, giúp giảm bớt lượng dữ liệu cần vận chuyển.
- Package service: Chứa các giao diện khai báo các phương thức nghiệp vụ như cập nhật, xử lý, thống kê dữ liệu. Package implement sẽ triển khai cụ thể từng phương thức được khai báo trong interface.
- Package controller: Chứa các lớp khai báo API endpoint cùng các phương thức chuyển đổi dữ liệu được phản hồi từ service sao cho phù hợp với kiểu dữ liệu yêu cầu từ phía client.

Biểu đồ lớp


pkg
controller
PartnerController
ProductController
SupplierController
Importcontroller
OverviewController
ReportController
Service
SupplierService
OverviewService
ProductService
ExportService
ImportService
PartnerService
ReportService
SupplierServiceImplement
ReportServiceImplement
ExportServiceImplement
OverviewServiceImplement
ImportServiceImplement
PartnerServiceImplement
ProductServiceImplement
model
BaseTransaction
Person
Export
Import
Supplier
Partner
Product
ProductExport
ProductImport
dto
ProductDTO
InventoryProductDTO
ImportDTO
ExportDTO
PartnerDTOForCreate
PeriodDataDTO
SupplierProductIDTOForShow
SupplierDTO
TopsupplierDTO
Respository
ProductImportRepository
PartnerRepository
ProductExportRepository
SupplierRepository
ExportRepository
ImportRepository
ProductRepository


controller
PartnerController
+ searchPartnersByName(String : int) : List<PartnerDTOForShow>
+ searchPartnerById(int : int) : Optional<Partner>
+ addPartner(PartnerDTOForCreate : int) : ResponseEntity<String>
+ deletePartner(int : int) : ResponseEntity<?>
ProductController
+ createProduct(ProductDTOForShow: int) : Product
+ findProductById(int : int) : Product
+ findProductById2(int : int) : Product
+ findProductByName(String : int) : List<Product>
+ updateProduct(int : int, ProductDTOForShow: int) : Product
+ deleteProduct(int : int) : ResponseEntity<?>
+ searchProducts(String : int) : ResponseEntity<List<ProductDTOForShow>>
SupplierController
+ searchSuppliersByName(String : int) : List<SupplierDTOForShow>
+ searchSuppliersById(int : int) : Optional<Supplier>
+ addProductToSupplier() : ResponseEntity<String>
+ addProductToSupplier(SupplierDTOForAddProduct : int) : ResponseEntity<String>
+ addSupplier(SupplierDTOForCreate : int) : ResponseEntity<String>
+ removeProductFromSupplier() : ResponseEntity<?>
Importcontroller
+ saveImport(ImportDTO : int) : ResponseEntity<ImportDTO>
OverviewController
+ getTopInventory() : ResponseEntity<List<InventoryProduct>>
+ getTopSuppliersByProducts(int : int) : ResponseEntity<List<TopSupplierDTO>>
+ getTopSuppliersByImports(int : int) : ResponseEntity<List<TopSupplierDTO>>
ReportController
+ getStatisticsByPeriod() : ResponseEntity<PeriodStatisticsDTO>
+ getStatisticsByProduct() : ResponseEntity<ProductStatisticsDTO>


Service
<<interface>>
SupplierService
+ findAllDTO():List<PartnerDTOForShow>()
+ findByNameContainingDTO(String):List<PartnerDTOForShow>(String : int)
+ addProductToSupplier() : void
+ addProductToSupplier(SupplierDTOForAddProduct : int) : void
+ addSupplier(Supplier : int) : boolean
+ findAllSimpleDTO() : List<SupplierDTO>
+ getProductsBySupplierSimple() : List<ProductDTO>
+ findbyIdSupplier(int : int) : Optional<Supplier>
+ removeProductFromSupplier() : void
<<interface>>
OverviewService
+ getTopInventoryProducts(String : int, int : int): List<InventoryProductDTO>
+ getTopSuppliersByProducts(int : int) : List<TopSupplierDTO>
+ getTopSuppliersByImports(int : int)
<<interface>>
ProductService
+ addProduct(ProductDTOForShow: int) : Product
+ findByProductId() : Product
+ filterProductsByName(String : int) : List<Product>
+ findAll(): List<Product>
+ updateProduct(int : int, ProductDTOForShow: int) : Product
+ deleteProduct(int : int) : ProductDTOForShow
+ findAllDTO() : List<PartnerDTOForShow
+ searchProducts(String : int) : List<PartnerDTOForShow
<<interface>>
ExportService
+ saveExport(ExportDTO : int) : ExportDTO
+ findAllExports() : List<ExportDTO>
+ findExportById(int : int) : ExportDTO
+ getProductsByExport(int : int) : List<ProductExportDTO>
<<interface>>
ImportService
+ saveImport(ImportDTO : int) : ImportDTO
<<interface>>
PartnerService
+ findAllDTO() : List<PartnerDTOForShow
+ addPartner(Partner : int)
+ findByNameContainingDTO(String : int) : List<PartnerDTOForShow>
+ findAllSimpleDTO() : List<PartnerDTO>
+ findByIdPartner(int : int) : Optional<Partner>
+ deletePartner(int : int) : void
+ findById(int : int): Optional<Partner>
<<interface>>
ReportService
+ getStatisticsByPeriod() : PeriodStatisticsDTO
+ getStatisticsByProduct() : ProductStatisticsDTO
package0
SupplierServiceImplement
ReportServiceImplement
ExportServiceImplement
OverviewServiceImplement
ImportServiceImplement
PartnerServiceImplement
ProductServiceImplement


model
BaseTransaction
- totalQuantity : int
- totalMoney : Double
Person
- name : String
- address: String
- contactNumber : String
Export
- exportID : int
- partner : Partner
- productExports : List<ProductExport>
Import
- importID : int
- supplier : Supplier
- productImports : List<ProductImport>
Partner
- partnerID : int
- exports: List<Export>
Supplier
- supplierID : int
- imports : List<Import>
- products: List<Product>
Product
- productID : int
- productName : String
- category : String
- price : Double
- unitCal : String
- lastUpdate : Date
- quantity : Integer
- productImports : List<ProductImport>
- productExports : List<ProductExport>
- suppliers : List<Supplier>
ProductExport
- id : int
- exportEntity : Export
- productEntity : Product
- quantity : int
- totalMoney: Double
ProductImport
- id : int
- importEntity : Import
- productEntity : Product
- quantity : int
- totalMoney: Double
0..*
1
1..*

4.1.3 Thiết kế cơ sở dữ liệu

4.1.4 Mục tiêu thiết kế cơ sở dữ liệu

Hệ thống quản lý kho này nhằm theo dõi quá trình nhập/xuất hàng hóa, thông tin sản phẩm, nhà cung cấp, và các đối tác (partners) liên quan, đảm bảo tính toàn vẹn dữ liệu và khả năng mở rộng.

4.1.5 Các thực thể (Entities) và thuộc tính (Attributes)

- **Partner (Đối tác):**
  - `partnerid` (Primary Key): Mã định danh của đối tác.
  - `address`: Địa chỉ của đối tác.
  - `contact_number`: Số điện thoại liên hệ.
  - `name`: Tên của đối tác.
- **Supplier (Nhà cung cấp):**
  - `supplierid` (Primary Key): Mã định danh của nhà cung cấp.
  - `address`: Địa chỉ nhà cung cấp.
  - `contact_number`: Số điện thoại nhà cung cấp.
  - `name`: Tên nhà cung cấp.
- **Product (Sản phẩm):**
  - `productid` (Primary Key): Mã định danh của sản phẩm.
  - `category`: Loại danh mục của sản phẩm.
  - `last_update`: Ngày cập nhật cuối cùng.
  - `price`: Giá sản phẩm.
  - `product_name`: Tên sản phẩm.
  - `quantity`: Số lượng sản phẩm trong kho.
  - `unit_cal`: Đơn vị đo lường (vd: cái, kg, hộp).
- **Import (Nhập hàng):**
  - `importid` (Primary Key): Mã định danh của phiếu nhập.
  - `total_quantity`: Tổng số lượng hàng nhập.
  - `supplierid` (Foreign Key): Liên kết đến nhà cung cấp.
  - `total_money`: Tổng giá trị đơn nhập.
  - `create_date`: Ngày tạo đơn nhập.
- **Export (Xuất hàng):**
  - `exportid` (Primary Key): Mã định danh của phiếu xuất.

    - `total_quantity`: Tổng số lượng hàng xuất.
    - `partnerid` (Foreign Key): Liên kết đến đối tác nhận hàng.
    - `total_money`: Tổng giá trị đơn xuất.
    - `create_date`: Ngày tạo đơn xuất.
- **Product_Import (Chi tiết nhập hàng):**
    - `id` (Primary Key): Mã định danh chi tiết.
    - `quantity`: Số lượng sản phẩm nhập.
    - `importid` (Foreign Key): Liên kết đến bảng nhập hàng.
    - `productid` (Foreign Key): Liên kết đến bảng sản phẩm.
    - `total_money`: Giá trị hàng nhập (tổng).
- **Product_Export (Chi tiết xuất hàng):**
    - `id` (Primary Key): Mã định danh chi tiết.
    - `quantity`: Số lượng sản phẩm xuất.
    - `exportid` (Foreign Key): Liên kết đến bảng xuất hàng.
    - `productid` (Foreign Key): Liên kết đến bảng sản phẩm.
    - `total_money`: Giá trị hàng xuất (tổng).
- **Supplier_Product (Nhà cung cấp - Sản phẩm):**
    - `supplierid` (Foreign Key): Liên kết đến bảng nhà cung cấp.
    - `productid` (Foreign Key): Liên kết đến bảng sản phẩm.

4.1.6 Các mối quan hệ (Relationships)

- **Partner - Export:**
    - Một đối tác (Partner) có thể liên kết với nhiều phiếu xuất hàng (Export).
- **Supplier - Import:**
    - Một nhà cung cấp (Supplier) có thể liên kết với nhiều phiếu nhập hàng (Import).
- **Product - Product_Import:**
    - Một sản phẩm (Product) có thể xuất hiện trong nhiều phiếu nhập hàng (Product_Import).
- **Product - Product_Export:**
    - Một sản phẩm (Product) có thể xuất hiện trong nhiều phiếu xuất hàng (Product_Export).
- **Import - Product_Import:**
    - Một phiếu nhập hàng (Import) có thể bao gồm nhiều chi tiết nhập hàng (Product_Import).

- **Export - Product_Export:**
    - Một phiếu xuất hàng (Export) có thể bao gồm nhiều chi tiết xuất hàng (Product_Export).
- **Supplier - Supplier_Product:**
    - Một nhà cung cấp (Supplier) có thể cung cấp nhiều sản phẩm (Product), và một sản phẩm có thể do nhiều nhà cung cấp cung cấp (mối quan hệ nhiều-nhiều).

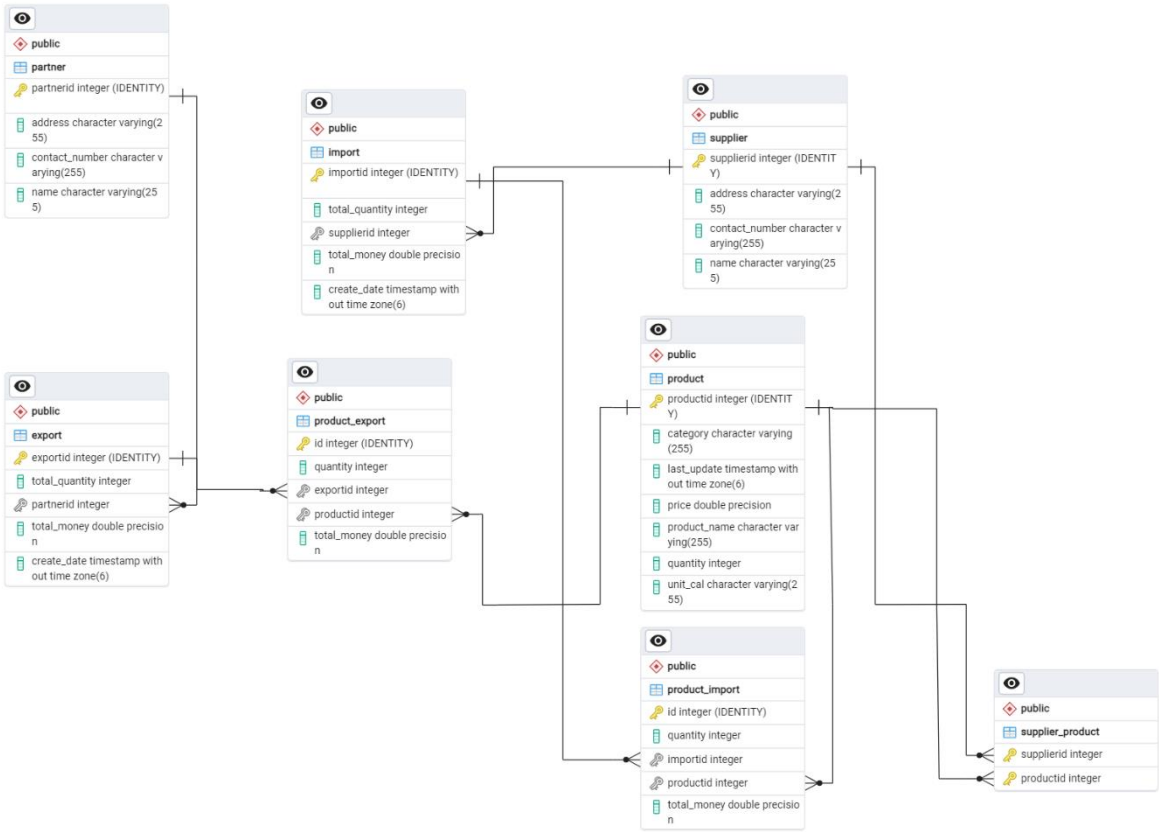


Figure 4-1: Thiết kế cơ sở dữ liệu

- Bảng **Import**:

Tên trường	Kiểu dữ liệu	Ràng buộc
import_id	Integer	PRIMARY KEY
total_quantity	Integer	
supplierid	Integer	FOREIGN KEY
total_money	Double	
create_date	Timestamp	

- Bảng **Partner**:

Tên trường	Kiểu dữ liệu	Ràng buộc
partner_id	Integer	PRIMARY KEY
address	Varchar(255)	
contact_number	Varchar(255)	
name	Varchar(255)	

- Bảng **Supplier**:

Tên trường	Kiểu dữ liệu	Ràng buộc
supplier_id	Integer	PRIMARY KEY
address	Varchar(255)	
contact_number	Varchar(255)	
name	Varchar(255)	

- Bảng **Product**:

Tên trường	Kiểu dữ liệu	Ràng buộc
Product_id	Integer	PRIMARY KEY
category	Varchar(255)	
last_update	Timestamp	
price	double	
Product_name	Varchar(255)	
quantity	Integer	
Unit_cal	Varchar(255)	

- Bảng **Export**:

Tên trường	Kiểu dữ liệu	Ràng buộc
Export_id	Integer	PRIMARY KEY
total_quantity	Integer	
Partner_id	Integer	FOREIGN KEY
Total_money	Double	
Create_date	Timestamp	

- Bảng Product_import:

Tên trường	Kiểu dữ liệu	Ràng buộc

id	Integer	PRIMARY KEY
quantity	Integer	
Export_id	Integer	FOREIGN KEY
Product_id	Integer	FOREIGN KEY
total_money	Double	

- Bảng Product_export:

Tên trường	**Kiểu dữ liệu**	**Ràng buộc**
id	Integer	PRIMARY KEY
quantity	Integer	
Export_id	Integer	FOREIGN KEY
Product_id	Integer	FOREIGN KEY
total_money	Double	

- Bảng Supplier_Product:

Tên trường	**Kiểu dữ liệu**	**Ràng buộc**
supplierid	Integer	FOREIGN KEY
productid	Integer	FOREIGN KEY

Giải thích và lý do thiết kế

- E-R Diagram: Biểu đồ E-R cung cấp cái nhìn tổng quan về cấu trúc dữ liệu và mối quan hệ giữa các thực thể.
- **Tính toàn vẹn dữ liệu:** Sử dụng khóa chính (Primary Key) và khóa ngoại (Foreign Key) để đảm bảo sự liên kết chặt chẽ giữa các thực thể.
- **Tối ưu hiệu suất:** Các trường thường xuyên truy vấn như `productid`, `supplierid`, và `partnerid` có thể được đánh chỉ mục để tăng tốc độ truy vấn.
- **Khả năng mở rộng:** Thiết kế này cho phép dễ dàng mở rộng, ví dụ thêm thuộc tính mới cho sản phẩm hoặc thêm bảng mới như "loại sản phẩm".

4.1.7 Thiết kế giao diện

Mục tiêu:

- Giao diện của ứng dụng hướng tới việc mang lại trải nghiệm người dùng thân thiện, dễ sử dụng và phù hợp với các đối tượng mục tiêu.
- Các yếu tố thiết kế đảm bảo tính nhất quán, dễ nhận biết và hoạt động tốt trên các độ phân giải màn hình máy tính phổ biến.

Thông tin về màn hình

- Độ phân giải hỗ trợ:
    - 1920x1080 (Full HD) là độ phân giải chính.
    - Tương thích với các độ phân giải thấp hơn như 1600x900.
- Kích thước màn hình:
    - Thiết kế tương thích với màn hình máy tính 14inch trở lên.
- Số lượng màu sắc:
    - Ứng dụng hỗ trợ hiển thị trên màn hình 16.7 triệu màu (24-bit color depth).
    - Đảm bảo tương phản và độ rõ ràng cho các màn hình chỉ hỗ trợ 16-bit màu.

Nguyên tắc và chuẩn hoá trong thiết kế giao diện

- Thiết kế nút bấm:
    - Nút có kích thước tối thiểu 24x24 pixel để đảm bảo người dùng quan sát tốt và tương tác tốt khi dùng chuột.
    - Sử dụng hiệu ứng hover hoặc nhấn để phản hồi hành động của người dùng.
- Thiết kế điều khiển:
    - Các mẫu nhập liệu có khung viền rõ ràng, kích thước đủ lớn để nhập liệu đầy đủ và thoải mái.
    - Thanh điều khiển cố định phía bên trái màn hình để chọn các chức năng.

- Vị trí hiển thị các thông điệp phản hồi:
    - Các thông báo được hiển thị dưới dạng các toastify hoặc các alert, sử dụng màu sắc để phân biệt các trạng thái:
        - Màu đỏ cho thông báo lỗi.
        - Màu xanh cho thông báo thành công.
- Phối màu:
    - Tông màu chính: #EBDFD7 (Hồng phấn nhẹ), #000 (Đen).
    - Tông màu phụ: #F2EAE5 (Trắng kem).
    - Tông màu nhấn mạnh: #E65F2B (Cam đất).
- Font chữ:
    - Sử dụng font chữ Opens San, cỡ chữ tối thiểu 12px để dễ đọc.
    - Tất cả tiêu đề đều in đậm.

4.2 Các kỹ thuật lập trình hướng đối tượng đã sử dụng

4.2.1 Dependency injection

Hệ thống này có thể sử dụng **Dependency Injection (DI)** để quản lý các service và repository nhằm tăng tính linh hoạt và dễ kiểm thử.

Ví dụ:

> Các service như `ProductService`, `SupplierService`, `ImportService`, `ExportService` có thể được tiêm vào các controller như `ProductController`, `SupplierController`, `ImportController`, `ExportController` thông qua **constructor injection** hoặc **annotation @Autowired** của Spring.
> `Repository` như `ProductRepository`, `SupplierRepository`, `ExportRepository` có thể được tiêm vào các service tương ứng, giúp tách biệt xử lý logic và truy vấn dữ liệu.

Điều này giúp **giảm sự phụ thuộc cứng** giữa các lớp và tăng khả năng mở rộng của hệ thống.

4.2.2 Singleton pattern

Singleton Pattern giúp duy trì một thể hiện duy nhất của một class trong suốt vòng đời của ứng dụng.

Ví dụ:

Các service như `ProductService`, `SupplierService`, `ReportService`, `ImportService` có thể được khai báo với `@Service`, làm cho chúng trở thành **Singleton Bean** trong Spring.
Các repository như `ProductRepository`, `SupplierRepository`, `ExportRepository` được Spring quản lý như **Singleton**, giúp tiết kiệm bộ nhớ và tránh khởi tạo nhiều lần không cần thiết.

Điều này giúp tối ưu tài nguyên và đảm bảo nhất quán trong truy vấn dữ liệu.

4.2.3 Inheritance (Kế thừa)

Kế thừa giúp tái sử dụng mã nguồn, giảm sự trùng lặp và dễ dàng mở rộng hệ thống.

Ví dụ:

Lớp **`BaseTransaction`** có thể là một lớp cha chứa các thuộc tính chung như `id`, `totalMoney`, và được các lớp `Export` và `Import` kế thừa.
Lớp `Person` có thể là một lớp cha của `Supplier` và `Partner`, giúp nhóm các thuộc tính chung như `name`, `address`, `phoneNumber`.

Kế thừa giúp **giảm trùng lặp code** và tổ chức hệ thống một cách logic hơn.

4.2.4 Polymorphisism (Đa hình)

Đa hình giúp các lớp có thể mở rộng mà không cần sửa đổi code gốc, giúp hệ thống linh hoạt hơn.

Ví dụ:

`BaseTransaction` có thể có một phương thức `calculateTotal()`, nhưng `Export` và `Import` có thể **override** phương thức này để tính toán theo quy tắc riêng của từng loại giao dịch.
`ReportService` có thể có phương thức `generateReport()`, nhưng có thể có nhiều cách triển khai khác nhau như `ProductReportService`, `SupplierReportService`, `ExportReportService`.

Điều này giúp **tăng tính linh hoạt** và cho phép mở rộng hệ thống dễ dàng.

4.2.5 Encapsulation (Đóng gói)

Đóng gói giúp bảo vệ dữ liệu bằng cách giới hạn quyền truy cập và chỉ cung cấp các phương thức cần thiết.

Ví dụ:

> Các lớp `Product`, `Supplier`, `Export`, `Import` có **thuộc tính private** (`id`, `name`, `price`, `totalMoney`) và chỉ có thể được truy cập thông qua **getter/setter**.
> Các lớp `Service` chỉ cung cấp một số phương thức public (`getProductById()`, `getSupplierById()`, `getExportTransactions()`), giúp ẩn đi logic xử lý bên trong.

Điều này giúp **bảo vệ dữ liệu** và đảm bảo rằng các thao tác trên đối tượng luôn tuân theo các quy tắc định sẵn.

4.2.6 Delegation (Ủy thác)

Ủy thác giúp phân chia trách nhiệm giữa các lớp, tránh việc một lớp xử lý quá nhiều logic.

Ví dụ:

> `ProductController` ủy thác việc xử lý nghiệp vụ cho `ProductService`, thay vì thực hiện trực tiếp.
> `ExportService` có thể gọi `ProductService` để kiểm tra sản phẩm trước khi thực hiện giao dịch xuất hàng.
> `ReportService` có thể gọi `ExportRepository` và `ImportRepository` để tổng hợp dữ liệu báo cáo.

Điều này giúp **giảm tải cho controller**, làm cho hệ thống dễ bảo trì và mở rộng hơn.

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Bảng 1 Danh sách thư viện và công cụ sử dụng

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Eclipse Oxygen 64 bit	http://www.eclipse.org/
IDE lập trình	IntelliJ IDEA Ultimate 64 bit	https://www.jetbrains.com/
IDE lập trình	Visual Studio Code 64 bit 1.96.2	https://code.visualstudio.com/
Kiểm thử API	Postman 64 bit	https://www.postman.com/
Quản trị cơ sở dữ liệu quan hệ	PostgreSQL 64 bit 16	https://www.postgresql.org/
Triển khai ứng dụng	Render	https://render.com/
Triển khai ứng dụng	Jenkins 2.491	https://www.jenkins.io/

4.3.2 Kết quả đạt được

Web quản lý kho đã hoàn thành, đảm bảo các tính năng cũng như giao diện. Với phần backend được phát triển bởi Java với Spring Boot, code được viết bằng IntelliJ Ultimate và Eclipse. Frontend được xây dựng bằng Javascrips, HTML, CSS, được code bằng IDE lập trình VSCode. Web được triển khai bằng Jenkins.

Bảng 2 Thống kê về backend

Thông số backend	Giá trị
Tổng số dòng code	2806 dòng

Tổng số gói	9 gói
Tổng số file	63 file
Dung lượng mã nguồn	683 kb

Bảng 3 Thống kê về frontend

Thống số	**Giá trị**
Tổng số dòng code	3227 dòng
Tổng số file	26 file
Dung lượng mã nguồn	6.17 MB

4.4 Triển khai

Mô hình triển khai: Mô hình client-server giao tiếp thông qua giao thức HTTPS

Máy chủ triển khai: Ubuntu Server 22.04 LTS

Cấu hình chi tiết:

- CPU core: 1
- Ram: 4GB
- Dung lượng ổ cứng: 30 GB
- Máy chủ web: Apache Tomcat

Kết quả thử nghiệm:

Performance Test Report - Dec 28, 2024 (#15)

Open in Postman

Postman collection: OOP-PMS

Report exported on: Dec 28, 2024, 11:50:14 (GMT+7)

Test setup

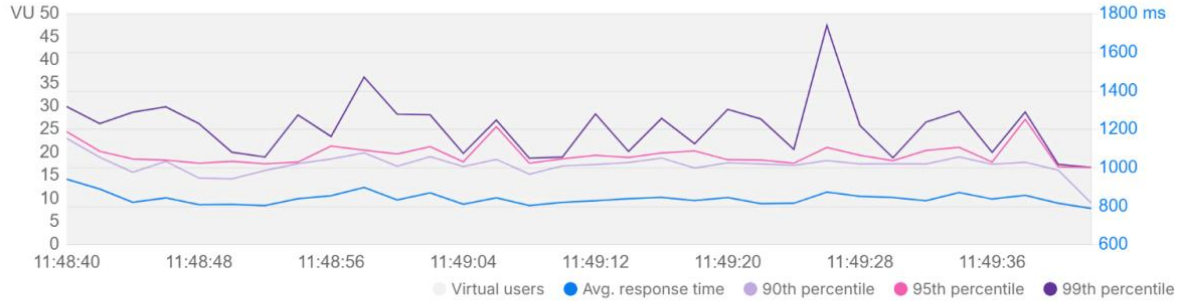
Virtual users 50 VU	Start time Dec 28, 11:48:35 (GMT+7)	Load profile Fixed
Duration 1 minute	End time Dec 28, 11:49:42 (GMT+7)	Environment New Environment

1. Summary

Total requests sent	Throughput	Average response time	Error rate
1,608	23.92 requests/second	842 ms	0.00 %

1.1 Response time

Response time trends during the test duration.



1.2 Throughput

Rate of requests sent per second during the test duration.

Figure 4-20: Kết quả thử nghiệm với 50 người dùng ảo truy cập

Chương 5 Kết luận và hướng phát triển

5.1 Kết luận

Phần lớn các chức năng trong phần mềm gần như đáp ứng yêu cầu đồ án. Trong quá trình làm đồ án còn gặp khá nhiều khó khăn trong lập trình, viết báo cáo cũng như quản lý, sắp xếp thời gian làm việc nhóm. Bên cạnh đó vẫn còn một số lỗi nhóm cần khắc phục như giao diện còn thô sơ, các lỗi tiềm ẩn trong quá trình thực thi, chưa kiểm tra được hiệu suất web, cần cải thiện thêm chức năng để tăng tương tác với người dùng, …

5.2 Hướng phát triển

Trong tương lai, nhóm có ý định phát triển ứng dụng thành ứng dụng đa nên tảng. Nhóm cũng cần cải thiện giao diện thân thiện với người dùng hơn và mở rộng thêm nhiều chức năng khác như cho phép up load file hay tạo báo cáo cho dự án.