

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**



VÕ THỊ KIM YẾN – LÊ HỮU PHƯỚC

**PHÁT TRIỂN HỆ THỐNG QUẢN LÝ KHO HÀNG
WMSPY BẰNG DJANGO VÀ REACTJS**

**KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN**

TP. HỒ CHÍ MINH, 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH



VÕ THỊ KIM YẾN – LÊ HỮU PHƯỚC

PHÁT TRIỂN HỆ THỐNG QUẢN LÝ KHO HÀNG
WMSPY BẰNG DJANGO VÀ REACTJS

Mã số sinh viên:

1851050194

1851050120

KHÓA LUẬN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn: ThS. Lê Ngọc Hiếu

TP. HỒ CHÍ MINH, 2021

LỜI CẢM ƠN

Lời đầu tiên chúng em xin gửi lời cảm ơn đến các thầy cô Khoa Công nghệ Thông tin và các thầy cô trường Đại Học Mở Thành Phố Hồ Chí Minh một lời cảm ơn chân thành và sâu sắc nhất. Các thầy, các cô đã luôn nhiệt tình giúp đỡ, động viên, khích lệ chúng em học tập cũng như truyền đạt tận tình, tận tụy các kiến thức chuyên ngành một cách rõ ràng, dễ hiểu nhất, giúp cho chúng em có một nền tảng kiến thức vững vàng và đã tạo điều kiện để chúng em có được ngày hôm nay.

Chúng em xin gửi đến lời cảm ơn chân thành và sâu sắc đến thầy Ths. Lê Ngọc Hiếu, là người đã luôn bên cạnh đồng hành, hướng dẫn và luôn dành thời gian quý báu của mình cho chúng em trong suốt quá trình làm đồ án.

Trong suốt quá trình học tập và phát triển, bản thân chúng em nhận thấy rằng trình độ chuyên môn của mình còn nhiều khuyết điểm, chúng em mong rằng, những đóng góp ý kiến của các thầy cô sẽ là động lực thúc đẩy giúp chúng em khắc phục các khuyết điểm để dần biến những khuyết điểm đó trở thành những ưu điểm nổi bật và kinh nghiệm quý báu nhất để phát triển toàn diện hơn về trình độ chuyên môn cũng như các kỹ năng cần có của một lập trình viên.

Và cuối cùng, chúng em xin cảm ơn chân thành đến gia đình, người thân và bạn bè đã luôn động viên, tạo điều kiện giúp đỡ chúng em trong suốt quá trình học tập, hoàn thành đồ án tốt nghiệp.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TÓM TẮT ĐỒ ÁN

Hệ thống quản lý kho hàng WMSPY là một hệ thống quản lý các công việc thường ngày của một kho hàng, là giải pháp phần mềm hiển thị hàng trong kho của các doanh nghiệp cũng như quản lý việc xuất nhập hàng từ kho đến các hệ thống cửa hàng sỉ và lẻ của doanh nghiệp. Hệ thống cũng cung cấp giải pháp tối ưu hóa không gian cũng như các thao tác lao động trong kho. Qua đó, giúp tiết kiệm được thời gian, chi phí cũng như sức lao động của chủ kho hàng.

Với sự phát triển của nền tảng công nghệ web hiện nay, hệ thống quản lý kho hàng WMSPY được phát triển dưới dạng web, giúp cho người dùng có thể thao tác dễ dàng hơn, dễ truy cập, theo dõi đơn hàng của mình hơn, theo đó tiết kiệm chi phí phần cứng cũng như thời gian cài đặt cho doanh nghiệp.

Mục lục

Chương 1. Tổng quan đề tài.....	13
1.1. Giới thiệu đề tài	13
1.1.1. Lý do chọn đề tài.....	13
1.1.2. Mục tiêu	13
1.1.3. Phạm vi nghiên cứu.....	14
Chương 2. Cơ sở lý thuyết ReactJS.....	15
2.1. Lý thuyết về React.....	15
2.1.1. Giới thiệu React	15
2.1.2. Ưu điểm, khuyết điểm React.	15
2.2. Lý thuyết về ReactJS	15
2.2.1. ReactJS là gì.....	15
2.2.2. JSX	17
2.2.3. Virtual DOM	19
2.2.4. Component	20
2.2.5. Vòng đời của component	21
2.2.6. Khái niệm về Props và State, Global State	23
2.2.7. React Event	26
2.2.8. React Forms	28
2.2.9. React Router.....	30
2.2.10. React Hook.....	32
2.3. React Axios	33
2.4. React Cookies	Error! Bookmark not defined.
2.5. Lý thuyết về Redux	35
2.5.1. Khái niệm về Redux.....	35

2.5.2. Hoạt động	35
2.5.3. Redux Toolkit	37
Chương 3. Lý thuyết Python Django	41
3.1. Giới thiệu Python Django.....	41
3.2. Tìm hiểu về model, meta options và migrations	43
3.3. Url Dispatcher	46
3.4. View	46
3.5. Authentication và Authorization	47
3.6. Query	49
3.7. Giới thiệu Django Rest Framework	49
3.8. View	50
3.9. ViewSet	50
3.10. ViewSet	51
3.11. Routers.....	52
3.12. Request và Response	53
3.12.1. Request.....	53
3.12.2. Response	53
3.13. Serializer.....	54
3.14. Authentication	55
3.15. Giới thiệu OAuth2	56
3.15.1. OAuth2.....	56
3.15.2. Django OAuth Toolkit	56
3.16. Cors.....	60
3.17. Tích hợp Swagger.....	60
3.18. Cloundinary storage	62
Chương 4. Phân tích hệ thống WMS.PY	64

4.1.	Xác định yêu cầu	64
4.2.	Tổng quan nghiệp vụ hệ thống	65
4.3.	Phân tích yêu cầu.....	68
4.4.	Thiết kế cơ sở dữ liệu	70
4.4.1.	Bảng tài khoản người dùng – User	70
4.4.2.	Bảng nhà cung cấp – Supplier	71
4.4.3.	Bảng sản phẩm – Item.....	71
4.4.4.	Bảng dãy kệ hàng – Row Location	72
4.4.5.	Bảng cột kệ hàng – Shelf Column	72
4.4.6.	Bảng tầng kệ hàng – Floor Column	72
4.4.7.	Bảng vị trí – Location	73
4.4.8.	Bảng sản phẩm tại vị trí – Item Location.....	73
4.4.9.	Bảng đơn nhập – PO	74
4.4.10.	Bảng chi tiết đơn nhập – PO Detail	74
4.4.11.	Bảng đơn xuất – SO	75
4.4.12.	Bảng chi tiết đơn xuất – SO Detail	75
4.4.13.	Bảng biên lai nhập – Receipt	76
4.4.14.	Bảng chi tiết biên lai nhập – Receipt Detail	76
4.4.15.	Bảng biên lai xuất – Order	77
4.4.16.	Bảng chi tiết biên lai xuất – Order Detail	77
4.4.17.	Bảng lưu kho tại vị trí – ImportView.....	78
4.4.18.	Bảng xuất kho tại vị trí – ExportView	78
4.5.	Lược đồ cơ sở dữ liệu.....	79
4.6.	Lược đồ usecase	79
4.6.1.	Đặc tả use case đăng nhập	80
4.6.2.	Đặc tả use case thống kê	82

4.6.3.	Đặc tả use case quản lý sản phẩm	83
4.6.4.	Đặc tả use case tạo đơn hàng.	83
4.6.5.	Đặc tả use case xóa đơn hàng	84
4.6.6.	Đặc tả use case duyệt đơn hàng.	85
4.6.7.	Đặc tả use case nhập hàng.	85
4.6.8.	Đặc tả use case xuất hàng.	86
4.6.9.	Đặc tả use case quản lý biên lai	87
4.7.	Xây dựng giao diện hệ thống	88
4.7.1.	Thiết kế giao diện.....	88
Chương 5.	Kết luận và hướng phát triển.....	97
5.1.	Kết luận	97
5.1.1.	Ưu điểm:	97
5.1.2.	Hạn chế:	97
5.2.	Hướng phát triển.....	98

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Cụm từ đầy đủ
PO	Purchase Order
SO	Shipment Order
Admin	Administrator

DANH MỤC HÌNH VẼ

Hình 2.1. Dự án ReactJS được tạo thành công.....	16
Hình 2.2. DOM.....	17
Hình 2.3 Virtual DOM trong ReactJS	20
Hình 2.4. Vòng đời của component.....	21
Hình 2.5 Hoạt động của Redux	36
Hình 3.1 Cấu trúc project Django.	42
Hình 3.2 Giao diện sau khi chạy thành công.....	43
Hình 3.3 Mô hình hoạt động của Django	43
Hình 3.4 Group và Permission trang Admin.....	48
Hình 3.5 Các APIs đã tạo	51
Hình 3.6 Giao diện tạo application chứng thực.....	58
Hình 3.7 Giao diện tạo thành công.....	58
Hình 3.8 Lấy token thành công	59
Hình 3.9 Sử dụng token vừa tạo để thực hiện API.....	59
Hình 3.10 Giao diện Swagger	62
Hình 3.11 Giao diện MEDIA LIBRARY trên Cloundinary	63
Hình 3.12 Cây thư mục lưu avatar user vừa tạo.....	64
Hình 4.1 Các loại Location trong kho hàng	66
Hình 4.2 Quy trình nhập hàng	67
Hình 4.3 Quy trình xuất hàng	67
Hình 4.4 Bảng User	70
Hình 4.5 Bảng Supplier	71
Hình 4.6 Bảng Item	71
Hình 4.7 Bảng RowLocation	72
Hình 4.8 Bảng ShelfColumn	72
Hình 4.9 Bảng ShelfFloor	72
Hình 4.10 Bảng Location	73
Hình 4.11 Bảng ItemLocation	73
Hình 4.12 Bảng PO	74
Hình 4.13 Bảng PODetail.....	74
Hình 4.14 Bảng SO	75

Hình 4.15 Bảng SODetail.....	75
Hình 4.16 Bảng Receipt	76
Hình 4.17 Bảng ReceiptDetail.....	76
Hình 4.18 Bảng Order	77
Hình 4.19 Bảng OrderDetail	77
Hình 4.20 Bảng ImportView	78
Hình 4.21 Bảng ExportView	78
Hình 4.22 Lược đồ cơ sở dữ liệu của hệ thống	79
Hình 4.23 Lược đồ use case của toàn hệ thống.....	79
Hình 4.24 Giao diện đăng nhập.....	88
Hình 4.25 Giao diện trang chủ	89
Hình 4.26 Giao diện menu	90
Hình 4.27 Giao diện tạo yêu cầu nhập kho.	91
Hình 4.28 Giao diện xem danh sách đơn hàng nhập kho.....	91
Hình 4.29 Giao diện xem chi tiết đơn hàng nhập kho.....	92
Hình 4.30 Giao diện thay đổi trạng thái đơn hàng	92
Hình 4.31 Giao diện xác nhận xóa đơn hàng	93
Hình 4.32 Giao diện thêm biên lai của đơn nhập.....	93
Hình 4.33 Giao diện biểu mẫu tạo biên lai.....	94
Hình 4.34 Giao diện chỉnh sửa một biên lai.....	94
Hình 4.35 Giao diện nhập kho.....	95
Hình 4.36 Giao diện cập nhật sản phẩm đang nhập kho	95
Hình 4.37 Giao diện hiển thị danh sách đã nhập kho.....	96
Hình 4.38 Giao diện của một đơn hàng sau khi nhập kho	96
Hình 4.39 Giao diện cập nhật sản phẩm xuất kho.....	96

DANH MỤC BẢNG

Bảng 4.1 Mô tả chi tiết use case của hệ thống	80
Bảng 4.2 Đặc tả use case đăng nhập	81
Bảng 4.3 Đặc tả use case quản lý tài khoản	82
Bảng 4.4 Đặc tả use case thống kê.	82
Bảng 4.5 Đặc tả use case quản lý sản phẩm.....	83
Bảng 4.6 Đặc tả use case tạo đơn hàng.	84
Bảng 4.7 Đặc tả use case xóa đơn hàng	85
Bảng 4.8 Đặc tả use case nhập hàng	86
Bảng 4.9 Đặc tả use case xuất hàng.	87
Bảng 4.10 Đặc tả usecase quản lý biên lai	88

MỞ ĐẦU

Trong thời kỳ của nền công nghiệp 4.0, thế giới đang đổ dồn vào các cuộc cách mạng phát triển nền công nghiệp mạnh mẽ, khiến cho cuộc sống con người được nâng cao và cải thiện lên rất nhiều. Với sự thừa hưởng của nền công nghiệp mạnh mẽ này, con người đã trở nên gần gũi với nhau hơn rất nhiều nhờ thông qua Internet, là một kiệt tác sáng tạo của nhân loại. Internet mang con người gần lại với nhau hơn, kết nối, chia sẻ một mạng lưới thông tin lớn một cách nhanh chóng và an toàn, đáng tin cậy. Bên cạnh đó nhờ vậy mà các trang web dần được hình thành qua mạng lưới thông tin khổng lồ này.

Với sự phát triển mạnh mẽ và rộng lớn, các trang web dần được hình thành và đóng một vai trò hết sức quan trọng trong việc sẻ chia dữ liệu của mọi người với nhau. Thao tác trên giao diện, hiệu quả, năng suất giao dịch, chia sẻ thông tin nhanh chóng, các trang web đơn giản đáp ứng gần như đủ nhu cầu cho một ứng dụng cơ bản hiện nay. Không chỉ thế, việc sử dụng trang web được đánh giá cao là vì có thể truy cập ở bất cứ đâu nhanh chóng và không phải bị giới hạn về dung lượng phần cứng hay thời gian lắp đặt phần mềm đối với người dùng.

Với những tiện dụng và lợi ích của trang web, đối với một doanh nghiệp, việc sở hữu một trang web hay một ứng dụng web để quảng bá sản phẩm hay phục vụ cho nhu công việc là một việc hết sức cần thiết. Việc truy cập và quản lý nhân viên, đơn hàng, hay chỉ đơn giản là xem các báo cáo công việc thường ngày thì một trang ứng dụng web phục vụ cho các công việc đó đang ngày càng nhiều và phổ biến trong các công ty kể cả trong và ngoài nước.

Với việc đáp ứng nhu cầu riêng của từng công ty, đặc biệt là các công ty logistic, trước đây, đa số các công ty doanh nghiệp dùng những ứng dụng phần mềm offline tại kho hàng, thì nay với việc đáp ứng nhu cầu mở rộng kho bãi, cũng như đáp ứng việc tương tác với khách hàng trong và ngoài nước trong tương lai, áp dụng triển khai xây dựng trang web phục vụ là việc làm cần thiết cho hiện tại mà các doanh nghiệp đang hướng tới. Và cùng với tiện ích kèm theo là đáp ứng được nghiệp vụ của logistic và nghiệp vụ riêng phù hợp website giúp doanh nghiệp có những bước tiến phát triển hơn, đột phá hơn về sứ vụ thống lĩnh thị trường logistic trong nước và quốc tế.

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Trong thời đại công nghiệp hóa hiện đại hóa ngày nay việc trao đổi mua bán hàng hóa đã quá phổ biến, bên cạnh đó còn vấn đề tồn kho khi hàng hóa chưa bán kịp nhất là trong thời buổi dịch bệnh hiện nay. Việc quản lý kho hàng giúp đáp ứng nguồn hàng cho tương lai và hiện tại, giải quyết các vấn đề lưu trữ hàng hóa. Hiện nay các trang website đã quá phổ biến mang lại nhiều tiện lợi và hiệu quả cao trong công việc. Vì vậy chúng em chọn làm một website để giúp giải quyết những vấn đề về quản lý kho hàng. Chúng em đã xem và nghiên cứu nhiều công nghệ và chúng em quyết định chọn nghiên cứu, tìm hiểu về thư viện thiết kế giao diện phổ biến hiện nay là ReactJS và thư viện hỗ trợ viết API là Django Rest, kết hợp cả hai thư viện này để xây dựng website quản lý kho hàng mang tên WMSPY.

1.1.1. Lý do chọn đề tài.

Về phần thực tế đời sống: để giúp các doanh nghiệp giải quyết các vấn đề khó khăn về logistic, mang lại sự tiện lợi, nhanh chóng, linh động về sản phẩm, thời gian... Website cho phép người dùng thực hiện các thao tác nhanh, dễ sử dụng cho cả nhà cung cấp hàng hóa và chủ nhà kho. Người dùng chỉ cần đăng nhập và thực hiện các thao tác đơn giản để làm việc.

Về phần lý thuyết: ReactJS là một thư viện phổ biến hiện nay giúp việc thiết kế giao diện dễ dàng hơn, mang lại nhiều hiệu quả cao. Django để phát triển xây dựng hệ thống phía back-end vì Django để phát triển xây dựng hệ thống phía back-end vì Django là một trong những web framework mạnh mẽ và phổ biến nhất hiện nay. Kết hợp với Django Rest Framework để tạo ra bộ API cho mình với năng suất và hiệu quả cao.

1.1.2. Mục tiêu

Mục tiêu của chúng em tìm hiểu xây dựng một hệ thống quản lý kho hàng đầy đủ các chức năng cần thiết, một website hoàn chỉnh về phần back-end và front-end.

Về phía back-end nghiên cứu sử dụng Django và Django Rest Framework phát triển back-end phía Server và các Rest API cho Client sử dụng.

Về phía front-end chúng em nghiên cứu và sử dụng React, ReactJS, Redux... Xác định cấu trúc và hoạt động của thành phần của ReactJS, gọi API từ server. Kết hợp xây dựng một trang website tiện dụng cho doanh nghiệp.

1.1.3. Phạm vi nghiên cứu

Tìm hiểu khái quát các khái niệm của ReactJS hiểu định nghĩa, các thuộc tính quan trọng như props, state... hiểu cách hoạt động vòng đời của ReactJS. Ngoài ra chúng em còn tìm hiểu về các Hook của ReactJS, tìm hiểu về Redux, sử dụng thêm thư viện Material-UI để hỗ trợ phân giao diện thêm linh động và tiện dụng hơn.

Nghiên cứu sử dụng Django và Django Rest Framework để phát triển Backend phía Server và các Rest API cho Client sử dụng. Tìm hiểu cấu trúc và xây dựng một project Django và kết hợp với DRF để tìm hiểu cách tạo và xây dựng một bộ API riêng mình nhằm thực hiện hóa những mục tiêu được đề ra trong quá trình làm đồ án của chúng em.

Chương 2. CƠ SỞ LÝ THUYẾT REACTJS

2.1. Lý thuyết về React

2.1.1. Giới thiệu React

React là thư viện JavaScript phổ biến nhất để xây dựng giao diện người dùng (UI), được phát triển bởi ông lớn Facebook được ra mắt như một công cụ mã nguồn mở năm 2013 [1]. React gồm ReactJS và React Native. ReactJs dùng để phát triển các ứng dụng single page còn React Native để phát triển các ứng dụng di động.

React sử dụng phổ biến bởi nhiều công ty lớn như: Netflix, Airbnb, American Express, Facebook, WhatsApp, eBay và Instagram [1].

2.1.2. Ưu điểm, khuyết điểm React.

React có nhiều ưu điểm như dễ sử dụng có nhiều tài liệu và cộng đồng người dùng React lớn. React có khả năng tái sử dụng lại cao, việc tái sử dụng các component mang lại lợi thế cho các lập trình viên. React cho phép viết các component tùy chỉnh đi được viết bằng JSX (là một phần mở rộng cú pháp ngôn ngữ Javascript và HTML). Bằng cách dùng DOM ảo React mang lại hiệu suất mượt mà và nhanh hơn. React còn có ưu điểm là thân thiện SEO...

Bên cạnh đó React còn có vài nhược điểm như: khó tiếp cận cho người mới học, React nặng hơn so với các framework khác như Angular, React phát triển phía tầng view mà không có hỗ trợ như model, controller. React liên tục phát triển và cập nhật đòi hỏi người dùng phải chịu khó tìm hiểu.

2.2. Lý thuyết về ReactJS

2.2.1. ReactJS là gì

ReactJS rất phổ biến ngày nay đã có hơn 100.000 ngàn website sử dụng nó và kèm theo đó là số lượng lập trình viên sử dụng rất lớn [2]. Vậy ReactJs là gì? Và tại sao nó lại phổ biến như vậy?

ReactJS là một thư viện JavaScript mã nguồn mở để xây dựng giao diện người dùng, phát triển ở dạng single page. ReactJS cho phép viết các đoạn mã nhỏ gộp lại gọi là các component nhờ vậy nó có khả năng tái sử dụng cao [2]. ReactJS sử dụng cho cả hai phía client và server, tập trung giải quyết phần View MVC (Model-View-

Controller). ReactJS là một Single Page Application nó có thể giúp chỉnh sửa dữ liệu của trang mà không cần nạp lại trang.

Hoạt động của ReactJS hoạt động dựa trên HTML và Javascript. ReactJS dùng các component để chạy vào trình duyệt, sử dụng Javascript để xử lý các chức năng liên quan. React sử dụng DOM ảo hiển thị các component mang lại hiệu năng cao.

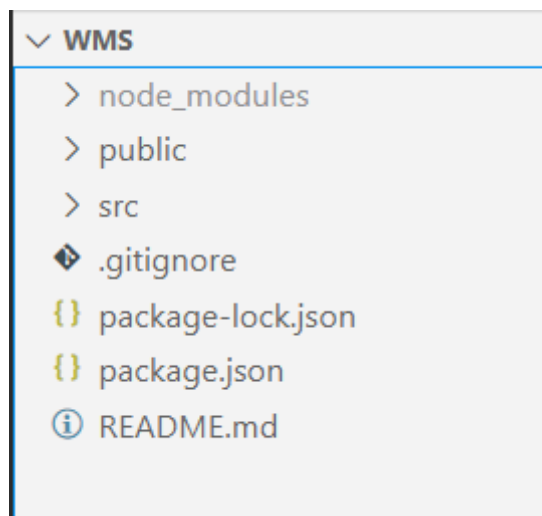
- **Cài đặt**

Cài đặt môi trường:

1. Cài đặt NodeJS tại <https://nodejs.org/en/download/>, cần cài đặt node phiên bản từ 14 trở đi, npm từ phiên bản 5.6 trở lên.
2. Tạo ứng dụng bằng lệnh

```
npx create-react-app wms
```

- Sau khi tạo thư mục và file sẽ được tạo sau:



Hình 2.1. Dự án ReactJS được tạo thành công.

Trong đó thư mục *public* chứa tập tin *index.js* là trang HTML duy nhất. Trong thư mục *src* có tập tin *index.js* và chương trình bắt đầu chạy tại đây, cũng tại thư mục này tạo các component cho riêng mình. Tập tin *package.json* tập tin cấu hình của npm, chứa thông tin của ứng dụng, các dependency cần cài cho project.

Chạy ứng dụng bằng lệnh

```
npm start
```

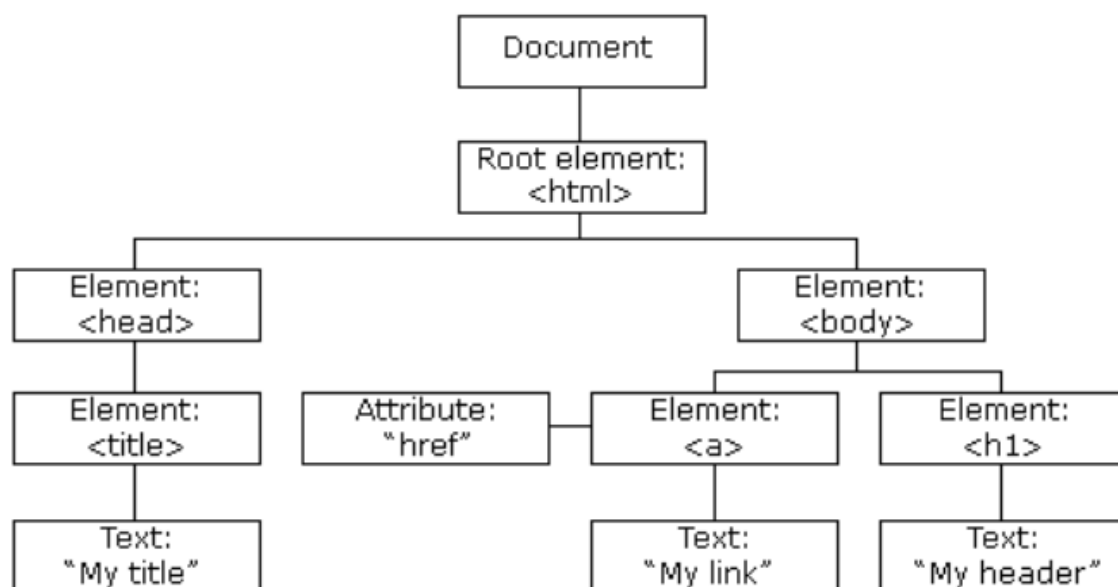
Truy cập vào đường dẫn: <http://localhost:3000/> để xem kết quả.



Hình 2.2 Giao diện sau khi chạy thành công ReactJS

2.2.2. JSX

Trong một trang website bất kỳ phần chính là những HTML documents. Trình duyệt đọc các documents hiển thị nội dung. Trong quá trình đọc đó trình duyệt sẽ tạo Document Object Model (DOM) là một tree đại diện cho một cấu trúc website hiển thị như nào, có thể thay đổi cây DOM bằng Javascript.



Hình 2.3. DOM (nguồn ảnh: <https://www.w3schools.com/>)

JSX (JavaScript Syntax Extension – JS XML) là một cú pháp mở rộng của Javascript cho phép viết HTML bên trong Javascript, các thẻ HTML được thành các component của ReactJS.

Các thuộc tính trong JSX được đặt theo quy tắc quy tắc lowerCamelCase, ví dụ className. Khi trả về nhiều thành phần cần bọc lại trong một thành phần container.

Có thể nhúng bất kỳ biểu thức nào trong Javascript bằng cách đặt chúng trong cặp dấu ngoặc nhọn {}. Ví dụ như user.lastName, 3+4...

```
<div>

  <h1>Xin chào, {user.name}</h1>

</div>
```

Ngoài ra có thể dùng trong vòng lặp, các câu lệnh điều kiện... Ví dụ dưới đây khai báo một danh sách các học sinh được nhúng vào dấu ngoặc nhọn nếu mảng đó lớn hơn 0 nghĩa là có học sinh thì render các học sinh ngược lại hiển thị báo danh sách rỗng.

```
import React from "react";

const Students = () => {

  const students = [

    { id: 1, name: "Kim Yến" },

    { id: 2, name: "Hữu Phước" },

  ];

  return (

    <div>

      {students.length > 0

        ? students.map((item) => {

            return <h1>{item.name}</h1>;

          })

        : "Danh sách rỗng"}

    </div>

  );

};
```

Lợi ích dùng JSX:

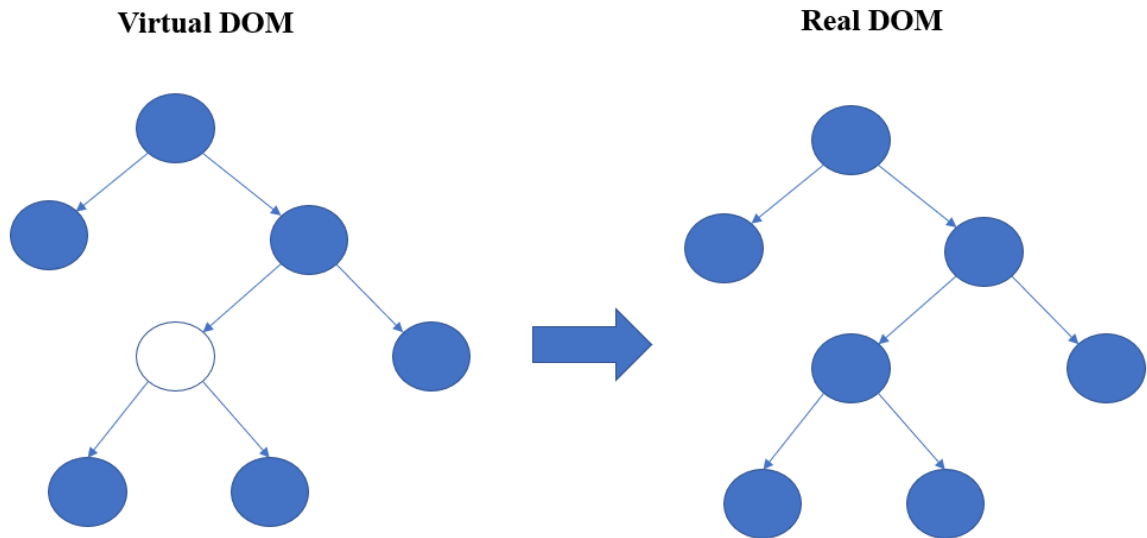
- Trong Javascript quản lý các component phức tạp cần gọi nhiều hàm hoặc các object thì với JSX giúp quản lý các component dễ dàng, người dùng dễ dàng định nghĩa hơn.
- JSX không làm thay đổi ngữ nghĩa của Javascript.
- Cú pháp đơn giản gần giống HTML.
- Có thể phát hiện các lỗi khi chương trình biên dịch mang đến sự an toàn.
- Dễ tái sử dụng, mở rộng và bảo trì khi tập trung các component lại.

React không bắt buộc sử dụng JSX, nhưng thật sự JSX mang lại nhiều hiệu quả cao.

2.2.3. Virtual DOM

Khi một website bình thường sẽ sử dụng HTML để cập nhập lại cây DOM, cách làm này sẽ ổn với các website đơn giản, mô hình nhỏ. Đối với các website lớn cấu trúc phức tạp, cần xử lý tương tác của người dùng nhiều thì việc này ảnh hưởng tới performance website nghiêm trọng vì mỗi khi cây DOM phải tải lại khi người dùng nhấn vào tính năng. Các nhà phát triển của ReactJS nhận ra điều này và họ dùng sử dụng DOM ảo (virtual DOM) nhằm cải thiện hiệu năng vì dùng DOM thật như truyền thống.

Tổng quát DOM ảo có các tính chất như DOM thật nhưng nó trực tiếp tương tác lên màn như DOM thật. Trong DOM ảo khi một node trong DOM thay đổi thì nó tìm được node đó bằng cách so sánh giữa DOM và DOM ảo, khi tìm thấy nó sẽ thay đổi node đó mà không làm ảnh hưởng tới các node khác. Hoạt động như vậy mang lại hiệu suất cao và website mượt mà hơn.



Hình 2.4 Virtual DOM trong ReactJS.

2.2.4. Component

Component trong ReactJS là các View được chia nhỏ ra. Component giúp cho việc tổ chức giao diện người dùng dễ dàng hơn và khả năng tái sử dụng cao. Component giúp quản lý và bảo trì được diễn ra đơn giản hơn. Component giống như hàm trả về các thành phần HTML. Component có hai loại là Function Component và Class Component.

2.2.4.1. Function Component

Function Component là một dạng hàm Javascript hoặc ES6 trả về một phần tử React nhận đối số props nếu cần [3].

Ví dụ:

```
const Example = () => {
    return <h1>Xin chào, tôi là function component</h1>
}
export default Example;
```

2.2.4.2. Class Component

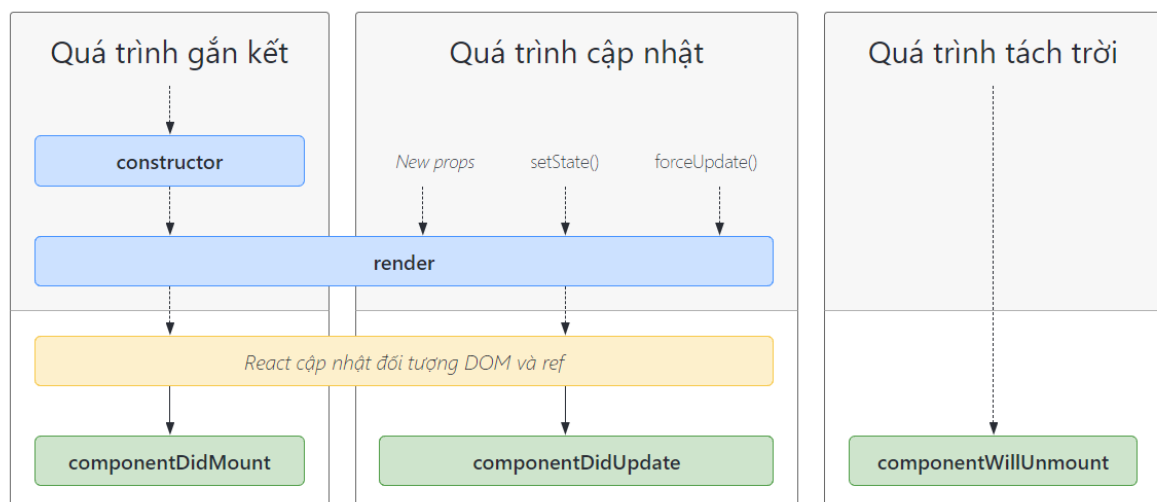
Class Component là một class trong ES6 chúng phức tạp hơn function component về nhiều thứ như: các phương thức khởi tạo, vòng đời, hàm render, quản lý các state.

Một component trong class phải được kế thừa từ thành phần `React.Component` và bắt buộc phải có hàm `render()` trả về các thành phần HTML. Ví dụ:

```
class Welcome extends React.Component {  
  render() {  
    return <h2>Xin Chào, tôi là class Component</h2>;  
  }  
}  
  
export default Welcome;
```

2.2.5. Vòng đời của component

Giống như con người có vòng đời sinh ra, lớn và mất đi. Component trong ReactJS cũng vậy có vòng đời của mình nó được tạo ra mang tên là Mounting, được thay đổi, cập nhật là quá trình Updating và cuối cùng là quá trình bị hủy bỏ Unmounting. Nói cụ thể hơn quá trình Mounting mà component được thêm vào DOM, Unmounting là component đó bị bỏ ra khỏi DOM. Vậy vòng đời của component trong ReactJS có 3 giai đoạn: Mounting, Updating và Unmounting.



Hình 2.5. Vòng đời của component (nguồn ảnh: <https://sentory.vn/>).

2.2.5.1. Quá trình Mounting

Quá trình này chỉ xảy ra đúng một lần lúc nó được gọi. Nó thực hiện gọi các phương thức sau:

- `constructor()` được gọi khi component được tạo ra.
- `render()` phương thức này bắt buộc và trả ra HTML cho DOM.
- Static `getDerivedStateFromProps()` được gọi ngay trước khi render thành phần trong DOM.
- `componentDidMount()` sau khi các component render ra xong thì phương thức này được gọi.

2.2.5.2. Quá trình Updating

Quá trình này xảy ra nhiều lần mỗi khi có props, state hoặc `forceUpdate` thay đổi. Khi thực hiện sẽ gọi các phương thức sau:

- static `getDerivedStateFromProps()` khi component được cập nhật phương thức này được gọi.
- `shouldComponentUpdate()` sẽ trả về giá trị boolean cho biết React có tiếp tục render hay không.
- `render()` được gọi khi component cập nhật và render lại HTML cho DOM.
- `getSnapshotBeforeUpdate()` được gọi trước khi props và state được cập nhật.
- `componentDidUpdate()` sau khi cập nhật phương thức này sẽ được gọi.

2.2.5.3. Quá trình Unmounting

Quá trình diễn ra đúng một lần giống Mounting nhưng nó được gọi khi component xóa khỏi DOM. Khi component được unmounted phương thức dưới đây sẽ được gọi:

- `componentWillUnmount()` được gọi khi một thành phần ngắt kết nối và hủy, `ClearTimeout` hoặc `interval` nếu có dùng. Reset dữ liệu trên Redux nếu cần thiết.

2.2.6. Khái niệm về Props và State, Global State

Trong ReactJS, props (tên đầy đủ properties) là đối số không thể thay đổi được truyền từ component cha xuống con, props không tự thay đổi giá từ component hiện tại mà phải thay đổi từ thằng cha truyền xuống. Props tạo sự đa dạng cho các component.

Trong ví dụ dưới đây thể hiện sự đa dạng, ta có thể tạo nhiều Box với các màu khác nhau, khi props truyền vào khác nhau.

```
function Box(props) {  
  return <div style={{ backgroundColor: props.color }}></div>;  
}
```

truyền props và hàm box

```
function App() {  
  return (  
    <>  
      <Box color="black"></Box>  
      <Box color="red"></Box>  
    </>  
  );  
}
```

Về Props truyền trong class component. Nếu component có dùng phương thức constructor thì props phải luôn truyền vào constructor và truyền cho React. Component bằng phương thức super.

```
class Demo extends React.Component {  
  constructor() {  
    super();  
    this.state = { name: "" };  
  }  
  handlechangeName = (e) => {  
    this.setState({ name: e.target.value });  
  }  
}
```

```

    };

    render() {
      return (
        <div>
          <input
            type="text"
            value={this.state.name}
            onChange={this.handleChangeName}
          />
          <h1>Xin Chào, {this.state.name}</h1>
        </div>
      );
    }
  }

  export default Demo;

```

Ví dụ dưới đây nói về cách truyền Props trong Class Component.

```

class DeMo extends React.Component {
  render() {
    return <h1>Xin chào, {this.props.name}</h1>;
  }
}

export default DeMo;

```

State là đối tượng lưu giá trị của các thuộc tính trong component giống Props nhưng State khác Props ở chỗ State có thể thay đổi lại được các thuộc tính. State được bởi sử dụng bởi một component hiện tại. Khi đối tượng state thay đổi thì component sẽ re-render.

Trong Class Component, giá trị State thay đổi khi được gọi phương thức `this.setState()`. Truy cập thuộc tính của đối tượng state bằng `this.state.propertyName`.

Ví dụ dưới đây minh họa về việc sử dụng State thay đổi tên mỗi khi người dùng nhập tên mới.

```
class Demo extends React.Component {  
  constructor() {  
    super();  
    this.state = { name: "" };  
  }  
  handlechangeName = (e) => {  
    this.setState({ name: e.target.value });  
  };  
  render() {  
    return (  
      <div>  
        <input  
          type="text"  
          value={this.state.name}  
          onChange={this.handlechangeName}  
        />  
        <h1>Xin Chào, {this.state.name}</h1>  
      </div>  
    );  
  }  
}
```

Global State giống với State thay đổi được giá trị của các thuộc tính trong component nhưng nó được dùng ở nhiều component khác nhau thay vì chỉ dùng được dùng ở một component hiện tại như State. Ví dụ sử dụng Global State khi lưu thông tin của user đang đăng nhập, thông tin của giỏ hàng...

2.2.7. React Event

Trong một website việc tương tác giữa người dùng và trang website là một điều không thể thiếu như kiện click hay submit form... Trong React việc xử lý các sự kiện này là một điều dễ dàng. Việc xử lý các sự kiện trong React giống với việc xử lý các sự kiện trên DOM. Có một số khác về cú pháp như tên các theo quy tắc lowerCamelCase thay vì chữ thường, ở ES6 cần truyền một hàm xử lý sự kiện thay vì truyền chuỗi.

Ví dụ gọi sự kiện ở HTML:

```
<button onclick=" handleOnClick() ">
  click
</button>
```

Ví dụ gọi sự kiện ở ReactJS:

```
<button onclick="handleOnClick() ">
  click
</button>
```

Phương thức trong Class của Javascript mặc định không bị ràng buộc, nhưng trong ReactJS nếu không ràng buộc this trước khi truyền vào sự kiện thì this sẽ bị undefined khi phương thức này được thực thi. Vì vậy, nên kết buộc this vào component bằng phương thức bind. Các phương thức định nghĩa dạng arrow function thì không cần thao tác này vẫn đảm bảo ràng buộc this.

Ví dụ dùng bind

```
class Button extends React.Component {
  constructor(props) {
    super(props);
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick(event) {
    console.log(event);
  }
}
```

```
render() {  
  return (  
    <button type="button" onClick={this.handleClick}>  
      Click Here  
    </button>  
  );  
}
```

Ví dụ dùng arrow function:

```
import React from "react";  
  
class Button extends React.Component {  
  handleClick() {  
    console.log("this");  
  }  
  
  render() {  
    return (  
      <button onClick={() => this.handleClick()}>  
        Click  
      </button>  
    );  
  }  
}
```

2.2.8. React Forms

Trong HTML Forms, các thành phần hoạt động hơi khác so với các thành phần DOM khác trong React. Trong ReactJS Forms sẽ linh động và được tương tác nhiều hơn. Có hai cách xử lý dữ liệu Form trong ReactJS.

- Uncontrolled components.
- Controlled components.

Controlled components những thành phần dữ liệu trong form được xử lý dưới dạng state hoặc store. Còn **uncontrolled components** nhận dữ liệu trực tiếp từ DOM. Ví dụ **Uncontrolled components** khi muốn nhận dữ liệu ta phải nhấn nút submit dữ liệu mới được nhận về còn **controlled components** nhận dữ liệu thực tiếp không cần thao tác nhấn submit.

Để viết Uncontrolled components khá đơn giản chỉ cần dùng một tham chiếu trực tiếp tới DOM không cần bắt các sự kiện event thay đổi trong Form. Với ưu điểm cài đặt đơn giản và nhanh từ đó tương tác người dùng trở nên tốt hơn nó phù hợp với những hệ thống có chức năng đơn giản. Ví dụ dưới đây đặt một biến input ref tham chiếu tới DOM.

```
class FormName extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
  
  handleOnSubmit = (e) => {  
    alert("name:  " + this.input.value);  
    e.preventDefault();  
  };  
  
  render() {  
    return (  
      <form onSubmit={this.handleOnSubmit}>  
        <label>
```

```

        Name:

        <input type="text" ref={(input) => (this.input = input)}
/>

    </label>

    <input type="submit" value="Submit" />

</form>

);

}

}

```

Controlled components dữ liệu của form quản lý dưới dạng state dữ liệu được lấy liên tục, có thể truy cập giá trị các trường bằng lệnh `event.target.value`.

Ví dụ này thể hiện input được lấy giá khi liên tục khi người dùng nhập.

```

import React from "react";

class FormName extends React.Component {

  constructor(props) {

    super(props);

    this.state = { name: "" };

  }

  changeInput = (e) => {

    this.setState({ name: e.target.value });

  };

  render() {

    return (

      <>

        <form>

          <input

            placeholder="Name"

            value={this.state.name}

            onChange={this.changeInput}

```

```
        />

    </form>

    <p>{this.state.name}</p>

</>

);

}

}

export default FormName;
```

2.2.9. React Router

React Router là một thư viện định tuyến (routing) tiêu chuẩn trong React. Nó cho phép điều hướng giữa các View của các component khác nhau trong React, cho phép thay đổi URL của trình duyệt và giữ giao diện người dùng đồng bộ với URL ^[4].

React Router có ba gói chính:

- React-router: cung cấp các thành phần routing chính cho ứng dụng.
- React-router-native: sử dụng cho các ứng dụng di động.
- React-router-dom: sử dụng cho các ứng dụng Web.

Cài đặt môi trường

```
npm install react-router-dom --save
```

Import React Router

```
import { BrowserRouter as Router, Link, NavLink, Route } from
"react-router-dom";
```

Các thành phần trong React Router

BrowserRouter: xử lý các URL động được sử dụng phổ biến hơn, theo dõi lịch sử bộ định tuyến nhờ dùng History API trong HTML5.

HashRouter: xử lý các URL tĩnh nó dùng sử dụng hàm băm (hash) của window.location.hash để lưu lịch sử.

Route: là một thành phần quan trọng nhất trong React Router. Nó ánh xạ một component tương ứng lên một URL tương ứng, một component được chỉ định sẽ render lên giao diện. Trong Router có các tham số như:

- **Path:** là đường dẫn URL.
- **Exact:** tham số này giúp cho Router này hoạt động tuyệt đối trên giá trị phù hợp với đường dẫn URL.
- **Component:** tham số chỉ định component được hiển thị ứng với URL.

Ví dụ trong ví dụ này đường dẫn URL là “/” và component được hiển thị là HomePage.

```
<Route
  path="/"
  exact
  component={HomePage}
></Route>
```

Link: trong HTML dùng thẻ <a> để chuyển tiếp các trang thì trong React Router dùng link để chuyển qua các trang. Link dùng tham số to để chuyển trang giống href trong thẻ a. Ví dụ:

```
<Link to="/user"> User</Link>
```

NavLink: giống như Link nhưng nó có thể tạo style cho nó.

```
<NavLink
  exact
  activeStyle={{
    backgroundColor: "black",
    color: "red",
  }}
  className="nav-link"
  to="/"
```

Redirect: dùng để chuyển hướng đến route khác, nhưng vẫn duy trì URL cũ. Để sử dụng Redirect chỉ cần import dòng sau:

```
import { Redirect } from "react-router-dom";
```

Switch: chứa các Route để render các component có path khớp.

```
<Switch>
  <Route exact path="/">
    <HomePage />
  </Route>
  <Route path="/login">
    <Login />
  </Route>
</Switch>
```

2.2.10. React Hook

React Hook là đặc trưng mới được thêm vào từ phiên bản React 16.8 có thể dùng state, life cycle và các đặc trưng khác của React mà không cần dùng tới class [5]. React Hook chỉ sử dụng được cho Functional Component, không dùng cho Class Component.

Ưu điểm sử dụng Hook là loại bỏ rào cản hướng đối tượng (OOP), Hook ra đời không phá những thứ code đã chạy tốt, Hook không xóa bỏ những kiến thức đã biết về ReactJS, không xóa bỏ Class Component.

Một số Hook phổ biến như: useState, useEffect...

2.2.10.1. useState

useState là một Hook cơ bản, dùng để sử dụng state trên Functional Component. Nó là một nên function nên có input và output. Input là initialState (giá trị hoặc function), Output là một mảng có hai phần tử tương ứng là state (giá trị hiện tại state) và setState

(hàm để cập nhật lại state). Khi sử dụng `useState` chỉ cần sử dụng cú pháp Array destructuring.

```
const [color, setColor] = useState("red");
```

2.2.10.2. `useEffect`

`useEffect` là một Hook cơ bản, nó thực thi ít nhất một sau mỗi lần render những lần sau sẽ được thực thi nếu nó dependencies thay đổi. **`useEffect`** giúp thực hiện các hành động trên Function component mà không dùng các phương thức trong vòng đời của React component như `componentDidMount()`, `componentDidUpdate()`, `componentWillUnmount()`.

Dùng `useEffect` với điều kiện:

Không khai báo dependencies mang ý nghĩa luôn được thực thi.

```
useEffect(() => {  
    //something  
});
```

Để mảng rỗng mang ý nghĩa thực thi một lần duy nhất sau khi render.

```
useEffect(() => {  
    //something  
}, []);
```

Có tham số ý nghĩa thực thi một lần sau lần render đầu, thực thi tiếp nếu dependencies thay đổi.

```
useEffect(() => {  
    //something  
}, [filter]);
```

2.3. React Axios

Axios là một thư viện HTTP Client cho phép hỗ trợ cho ứng dụng cung cấp API dễ dàng được sử dụng cả trên Node.js và trình duyệt nhờ promise.

Cài đặt:

```
npm install axios  
npm install react-axios
```

Lợi ích sử dụng React Axios:

- Dữ liệu trả về dưới dạng JSON
- Nó có đầy đủ các phương thức ứng với bất kỳ HTTP ví dụ để truy cập phương thức GET chỉ cần dùng hàm get(),
- Cú pháp trong React Axios ít và tiện hơn.
- Khả năng xử lý lỗi tốt trả về các lỗi như 400 hay 500.

Ví dụ sử dụng React Axios lấy dữ liệu từ API, ví dụ này dùng phương thức get() lấy danh sách học sinh, nếu thành công trả về danh sách học sinh, nếu xảy ra lỗi thì hiển thị lỗi ra console.log ở catch.

```
import React from "react";  
import axios from "axios";  
  
export default class StudentList extends React.Component {  
  state = {  
    students: [],  
  };  
  
  componentDidMount() {  
    axios  
      .get(`http://127.0.0.1:8000/users`)  
      .then((res) => {  
        const students = res.data;  
        this.setState({ students });  
      })  
      .catch((error) => console.log(error));  
  }  
  
  render() {
```

```
return (  
  <ul>  
    {this.state.students.map((student) => (  
      <li>{student.name}</li>  
    ))}  
  </ul>  
);  
}
```

2.4. Lý thuyết về Redux

2.4.1. Khái niệm về Redux

Redux là một thư viện Javascript giúp quản lý State, mà State này đặc biệt ở chỗ nó có thể dự đoán được sử dụng kiến trúc uni-directional data flow. Nó giúp cho ứng dụng hoạt động nhất quán ở các môi trường khác nhau để kiểm tra. Redux ra đời lấy cảm hứng từ tư tưởng của ngôn ngữ Elm và kiến trúc Flux của Facebook [7]. Nhà phát triển Redux nói rằng: “Nếu ứng dụng của bạn đã chạy ổn, không nhất thiết phải dùng Redux”. Redux không chỉ dùng kèm với ReactJs mà còn nhiều thư viện khác nữa.

2.4.2. Hoạt động

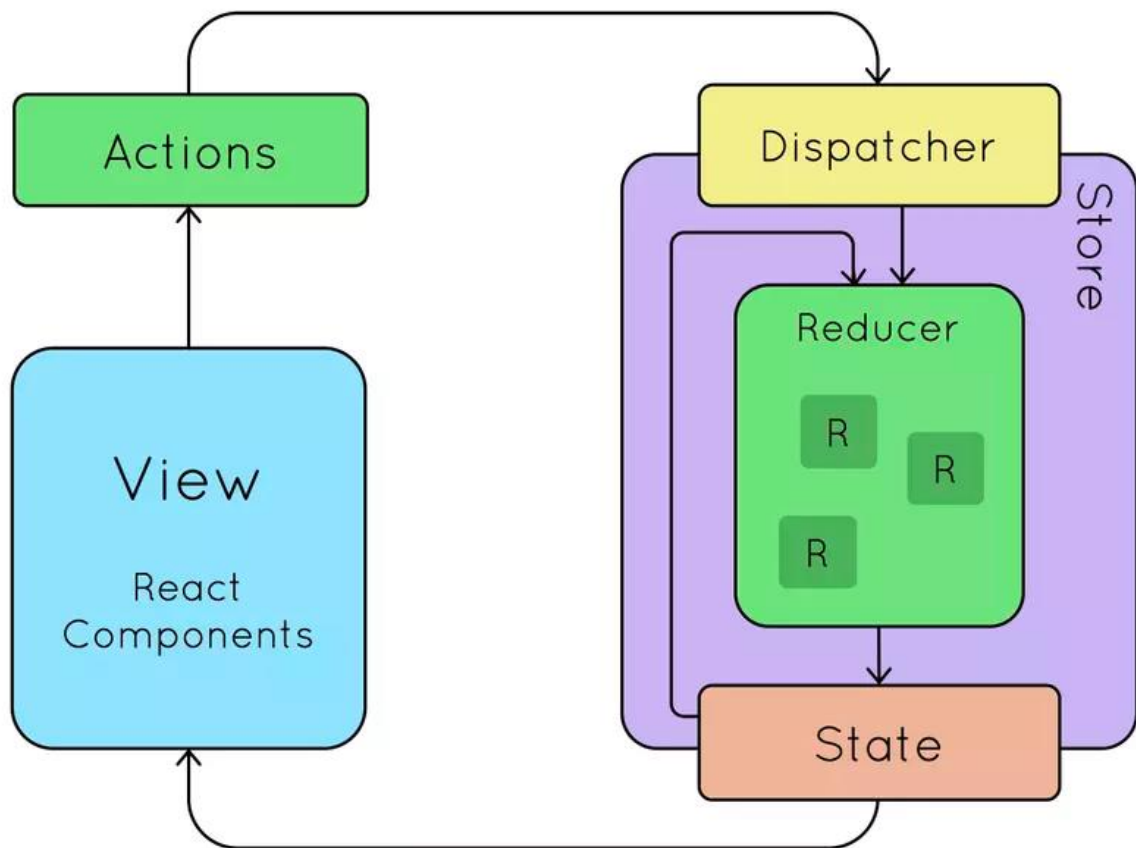
Redux sử dụng kiến trúc uni-directional data flow nghĩa là data flow đi đúng một chiều. Redux gồm ba thành phần chính Actions, Store, View.

Action là các sự kiện mà qua đó dữ liệu được gửi từ ứng dụng đến store Redux. Dữ liệu có thể là tương tác của người dùng, gọi API hoặc gửi Form. Action được truyền vào Store

Store là cập nhật trạng thái mới, các thành phần quan sẽ lắng nghe những thay đổi đó và thực hiện những thay đổi cần thiết. Trong Store có **Reducer** là một thành phần quan trọng nó dựa trên các Action được gửi vào chúng sẽ đưa trạng thái hiện tại, cập nhật nếu cần và trả về trạng thái mới. Các trạng thái mới này được lưu trữ dưới dạng các đối tượng.

View là một thành phần sẽ hiển thị lên giao diện của người dùng.

Từ store khởi tạo một State truyền lên giao diện khi người dùng thực hiện một Action thì sẽ truyền lên Store, ở Store cập nhập thay đổi trả lại State nếu có State có thay đổi.



Hình 2.6 Hoạt động của Redux (nguồn ảnh: <https://topdev.vn/>).

Cài đặt:

```
npm install --save redux
```

Ví dụ mẫu dùng Redux đếm số tăng khi người dùng chọn increment, giảm khi người dùng chọn decrement.

```
const redux = require("redux");

const countReducer = (state = { counter: 0 }, action) => {

  if (action.type === "increment") {

    return {
```

```

        counter: state.counter + 1,
    };
}
if (action.type === "decrement") {
    return {
        counter: state.counter - 1,
    };
}
return state;
};

const store = redux.createStore(countReducer);

const countSubscribe = () => {
    const lastetate = store.getState();
    console.log(lastetate);
};

store.subscribe(countSubscribe);

store.dispatch({ type: "increment" });
store.dispatch({ type: "decrement" });

```

2.4.3. Redux Toolkit

Redux Toolkit là thư viện giúp viết Redux tốt hơn, cơ bản vẫn là Redux nhưng giúp cài đặt sẵn một số hàm mang lại hiệu năng nhanh hơn, đơn giản và dễ dàng , và nó giải quyết ba vấn đề chính.

- Cấu hình Store của Redux khá phức tạp.
- Phải có cài nhiều gói để Redux trở nên hữu ích.

- Redux yêu cầu nhiều mã soạn sẵn.

Redux Toolkit bao gồm:

configureStore(): tạo một createStore để cung cấp một cấu hình mặc định sẵn có. Có thể kết hợp với Reducer và bất kỳ Redux middleware nào bao gồm cả Redux DevTools và redux-thunk.

createReducer(): giúp việc tra cứu các Reducer dễ dàng hơn, thay vì phải viết các câu lệnh chuyển đổi phức tạp. Ngoài ra nó còn sử dụng thư viện Immer giúp cho việc mutate trực tiếp sửa ở State hiện tại không cần trả về State mới. Ví dụ `state.todos[2].isStatus = true`.

createAction(): tạo ra Action, dạng như một hàm trả về Object.

createSlice(): giúp thay thế cho cả Reducer và Action [8].

...

Cài đặt

```
npm install @reduxjs/toolkit
```

Ví dụ dùng ReduxToolkit.

Tạo ra Slice lưu State ngôn ngữ

```
import { createSlice } from "@reduxjs/toolkit";

const language = createSlice({
  name: "ui",
  initialState: { currentLanguage: "vn" },
  reducers: {
    changeLanguage(state, action) {
      state.currentLanguage = action.payload;
    },
  },
});

export const languageActions = language.actions;
```



```
export default language;
```

Thêm Slice đó vào Store

```
import { configureStore } from "@reduxjs/toolkit";

import language from "../language";
import poSlice from "../poSlice";

const store = configureStore({
  reducer: {
    currentLanguage: language.reducer,
  },
});

export default store;
```

Gắn Redux Provider vào ứng dụng

```
import { Provider } from "react-redux";

import store from "../store";
import App from "../App";

function Main() {
  return (
    <Provider store={store}>
      <App />
    </Provider>
  );
}
```

Khi sử dụng dùng các Hook `useDispatch` để gọi action , `useSelector` để lấy State.

```
import { useDispatch, useSelector } from "react-redux";
```

```
const currentLanguage = useSelector(
```

```
(state) => state.currentLanguage.currentLanguage,  
);
```

```
const dispatch = useDispatch();  
  
function changeLang(params) {  
  dispatch(languageActions.changeLanguage(params));  
}
```

Chương 3. Lý thuyết Python Django

3.1. Giới thiệu Python Django

Django là một trong những high-level web framework mã nguồn mở của Python mạnh mẽ và phổ biến nhất hiện nay. Nhờ các thư viện module có sẵn, cũng như hỗ trợ rất mạnh các tính năng cơ bản của một trang web như chứng thực, phân quyền, những thao tác cơ bản trên trang admin là thêm, sửa, xóa cũng được django hỗ trợ từ đầu đến cuối. Lập trình viên chỉ việc đưa ra các ý tưởng và hiện thực hóa ý tưởng đó theo cách của riêng mình.

Cài đặt môi trường:

Bước 1: cài đặt Python SDK: <https://www.python.org/downloads/>

Để kiểm tra Python SDK đã cài đặt thành công hay chưa, ta vào Start > Command Prompt, sau đó, ta thực hiện gõ lệnh sau:

```
python -version
```

và đây là kết quả sau khi cài thành công python SDK:

```
C:\Users\lehuu>python --version  
Python 3.9.6
```

Cài đặt ứng dụng Django:

Tạo một project rỗng và vào terminal ta thực hiện các lệnh sau:

Cài đặt môi trường Django:

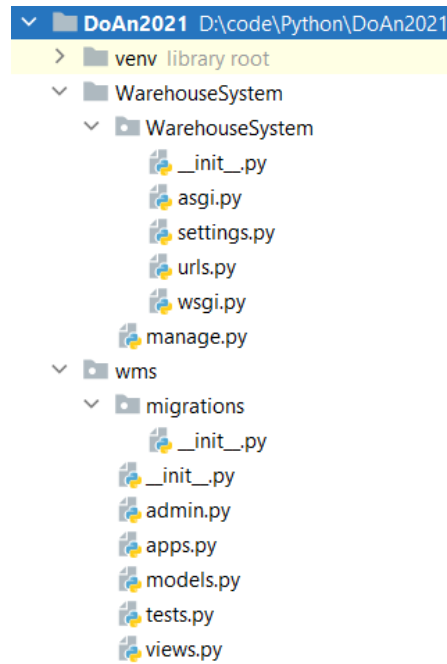
```
pip install django
```

Khởi tạo project:

```
django-admin startproject <project-name>
```

Trong đó: <app-name> là tên app mà bạn muốn đặt trong project của bạn.

Sau khi hoàn thành các câu lệnh trên, ta sẽ có được một cấu trúc project Django như sau:



Hình 3.1 Cấu trúc project Django.

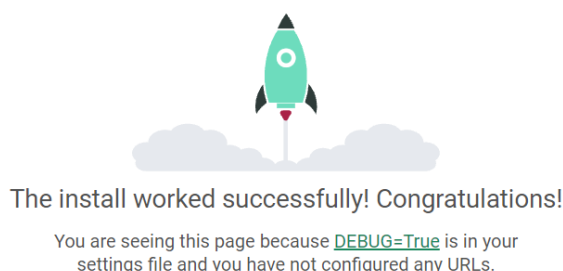
Trong đó, ta có các file thường sử dụng đó là:

- settings.py: Chứa các biến cấu hình và thông tin về cấu hình của project
- url.py: Đây là file chứa tất cả các đường dẫn của project, ta có thể khai báo url project của ta ở đây.
- manage.py: đây là một tiện ích giúp ta có thể tương tác dễ dàng với project theo nhiều cách khác nhau thông qua dòng lệnh.

Chạy ứng dụng ta gõ lệnh sau:

```
python manage.py runserver
```

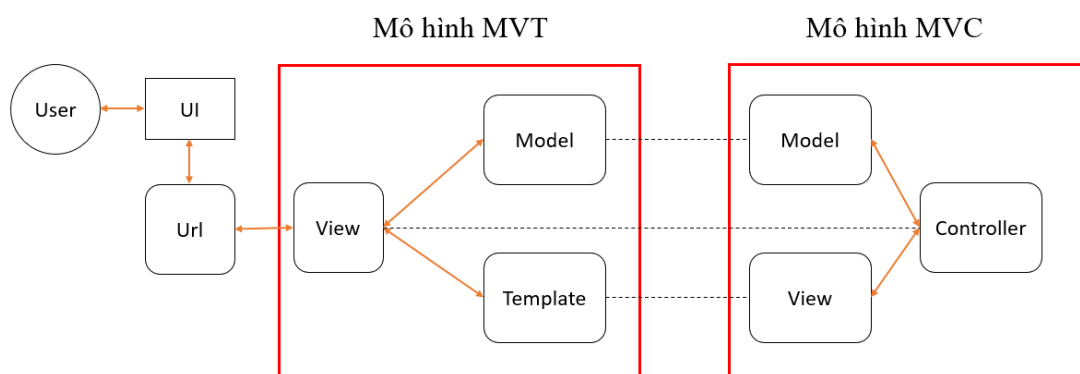
Bước 2: truy cập vào đường dẫn: <http://127.0.0.1:8000/> để xem kết quả.



Hình 3.2 Giao diện sau khi chạy thành công.

Django chia ứng dụng thành ba phần, sử dụng mô hình Model-View-Template(MVT) là một biến dị của mô hình ta thường thấy, đó là Model-View-Controller(MVC) để hoạt động.

Trong mô hình MVT, thì View ở đây đóng vai trò như một Controller để xử lý logic các chức năng của project, Template đóng vai trò là tiếp xúc và hiển thị với giao diện và cuối cùng là Model là lớp đóng vai trò tương tác, giao tiếp với cơ sở dữ liệu.



Hình 3.3 Mô hình hoạt động của Django.

3.2. Tìm hiểu về model, meta options và migrations

3.2.1.1. Model

Một model là nơi duy nhất tương tác với cơ sở dữ liệu trong project Django. Mỗi class trong model là thể hiện của một bảng được ánh xạ xuống dưới cơ sở dữ liệu khi kế thừa class `django.db.models.Model`.

Mỗi trường được khai báo trên model là thể hiện của một trường của table đó dưới cơ sở dữ liệu. Cùng với việc khai báo thêm hay thay đổi các hành vi ngay trên code giúp việc kiểm soát dữ liệu được chặt chẽ và dễ dàng hơn cho lập trình viên. Trong class Model, các trường dữ liệu là thể hiện lớp Field, đó là một lớp trừu tượng chứa các lớp con đưa ra những kiểu dữ liệu tương ứng với kiểu dữ liệu dưới cơ sở dữ liệu: CharField, IntegerField, DateTimeField, BooleanField... Ngoài ra, ta còn có thể thiết lập khóa ngoại ngay trên code thông qua Field “ForeignKey” hay các quan hệ giữa các bảng thông qua Field: ManyToMany, OneToOne, ManyToOne, giúp cho việc kiểm soát cơ sở dữ liệu trở nên dễ dàng hơn rất nhiều. Ví dụ:

```
class Product(models.Model):  
  
    category = models.ForeignKey(Category, on_delete=models.CASCADE)  
    name = models.CharField(max_length=100, null=False)  
    unit = models.IntegerField()  
    status = models.BooleanField(default=True)  
  
class Category(models.Model):  
    name = models.CharField(max_length=100, null=False)  
    status = models.BooleanField(default=True)
```

3.2.1.2. Meta option trong model

Trong cơ sở dữ liệu, việc ràng buộc dữ liệu là rất cần thiết. Nó giúp cho việc tương tác với cơ sở dữ liệu được chặt chẽ hơn, thì với Django ta có thể ràng buộc cơ sở dữ liệu thông qua các meta options trong model. Meta class là lớp chứa những thông tin thể hiện ràng buộc đó.

Trong đó, việc sử dụng meta class còn giúp ta có thể trực quan hóa cách nhìn về model khi viết code theo tư duy về hướng đối tượng thông qua thuộc tính “abstract”. Ví dụ:

```
class BaseModel(models.Model):  
    name = models.CharField(max_length=100, null=False)  
    status = models.BooleanField(default=True)  
  
    class Meta:  
        abstract = True  
        ordering = ['-id']
```

```
class Product(BaseModel):  
    category = models.ForeignKey(Category, on_delete=models.CASCADE)  
    unit = models.IntegerField(default=1)  
  
class Category(BaseModel):  
    description = models.TextField(null=True, blank=True)
```

Mặc định các các model con kế thừa từ model cha thì sẽ kế thừa luôn meta option của model cha.

Ngoài ra, Django còn có các kiểu kế thừa model khác, đó là:

- Multi-table inheritance
- Proxy model

3.2.1.3. Khái niệm Migration

Migration là tính năng cho phép ta chuyển những thay đổi trong cấu trúc cơ sở dữ liệu thành kịch bản, và từ đó xuất ra các script sql để thực thi thay đổi xuống cơ sở dữ liệu. Và nhờ đó, việc điều chỉnh cấu trúc cơ sở dữ liệu của project Django trở nên tiện lợi, linh hoạt hơn rất nhiều.

Để thực thi cơ sở dữ liệu từ model ta cần phải trải qua các bước sau:

Bước 1: tạo những kịch bản thay đổi:

```
python manage.py make migrations <app_name>
```

Lệnh này sẽ sinh ra kịch bản nếu có sự thay đổi trên code model so với dưới cơ sở dữ liệu và chỉ có hiệu lực khi ta đã khai báo tên app của mình trong biến `INSTALLED_APPS` trong `setting.py`.

Bước 2: thực thi thay đổi xuống cơ sở dữ liệu

```
python manage.py migrate
```

Lệnh này sẽ sử dụng những migrations vừa được tạo ra mà theo đó mapping những thay đổi xuống lược đồ cơ sở dữ liệu và đều lưu lại lịch sử mỗi lần migrate.

Tên bảng dưới lược đồ cơ sở dữ liệu sẽ là : <app_name>_<table_name>. Đặc biệt chú ý, khi trong mô hình code first như thế này, những thay đổi cấu trúc cần được sửa trên code, hạn chế tự sửa trực tiếp thay đổi dưới cơ sở dữ liệu. Vì khi đó ta migrate thì chương trình sẽ không biết là chương trình đã migrate tới đâu rồi, từ đó dễ dẫn đến xung đột cơ sở dữ liệu.

3.3. Url Dispatcher

Là module chứa các định nghĩa cấu hình url, ta có thể tạo riêng một python module cho từng app.

Django xác định module chứa URLconf gốc dựa trên biến ROOT_URLCONF để từ đó tìm các url được cấu hình trong urlpatterns của python module. Nếu trong thành phần HttpRequest có urlconf thì sẽ ưu tiên hơn ROOT_URLCONF.

Django có hỗ trợ áp dụng biểu thức chính quy vào url, điều đó làm cho url trở nên sạch sẽ hơn, nhiều ràng buộc rõ ràng hơn, chặt chẽ hơn, và chuyên nghiệp hơn trong một ứng dụng web chất lượng. Ví dụ:

```
from django.urls import re_path

from . import views

urlpatterns = [

    path('category/', views.Category),
    re_path(r'^product/(?P<id>[0-9]+)/$', views.product)

]
```

3.4. View

Đây là nơi tiếp nhận những request để xử lý logic các vấn đề cần thiết và trả về response cho người dùng. Response có thể là thành phần HTML, error status hay bất cứ thứ gì mà người dùng yêu cầu. Ta có thể sử dụng views ở bất cứ đâu trong project Django.

Có hai cách tiếp cận view trong Django:

- Tiếp cận bằng dạng hàm
- Tiếp cận bằng dạng class

Khi thực thi chương trình, ta phải ánh xạ Url vào View như sau:


```
from . import views

urlpatterns = [

    path('category/', views.Category),

]
```

3.5. Authentication và Authorization

Authentication trong Django được hỗ trợ rất mạnh, không những thế Django còn hỗ trợ thêm cả Authorization. Các yêu cầu cấu hình được thiết lập sẵn trong setting từ khi ta khởi tạo project. Nó bao gồm hai thành phần trong biến `INSTALLED_APPS`, đó là:

- `django.contrib.auth`
- `django.contrib.contenttypes`
- và hai thành phần trong biến `MIDDLEWARE`
- `SessionMiddleware`
- `AuthenticationMiddleware`

3.5.1.1. Authentication

Model `User` là đại diện duy nhất của người dùng trong hệ thống của Django. Khi người dùng thao tác, hệ thống sẽ lưu thông tin người dùng qua các trường dữ liệu có sẵn của model `User`, đó là: `first_name`, `last_name`, `username`, `password`, `email`, `is_staff`, `is_superuser`, `is_active`.

Với nhu cầu mở rộng model `User` để dễ dàng hơn trong việc bổ sung thêm thông tin user trong tương lai, ta có 2 cách:

- Tạo model `User` kế thừa lớp `AbstractUser`: Với điều này, ta cần tạo model `User` từ lúc mới khởi tạo project và cấu hình lại biến `AUTH_USER_MODEL` của Django

```
AUTH_USER_MODEL = '<app_name>.User'
```

- Tạo model `User` chứa các thông tin bổ sung và sử dụng mối quan hệ One to One tham chiếu đến model `User` mặc định

Request.user là thuộc tính thể hiện cho user hiện tại đã chứng thực. Nếu user chưa được chứng thực, thì thể hiện của nó là AnonymousUser. Ta có thể kiểm tra user hiện tại đã được chứng thực hay chưa bằng câu lệnh:

```
request.user.is_authenticated()
```

hoặc kiểm tra user có phải là Anonymous hay không như sau:

```
request.user.is_anonymous()
```

3.5.1.2. Authorization

Django cung cấp sẵn cho lập trình viên các cơ chế phân quyền cho từng user hoặc một nhóm user rất chặt chẽ. Model User có hai quan hệ Many To Many với các model: groups và user_permission.

Với model group và user_permission, ta có các quyền cơ bản của một model như thêm, sửa, xóa và ta có thể chứng thực một hay một nhóm user với các phân quyền được gán cho nhóm đó. Nhờ đó, admin dễ dàng kiểm soát và phân quyền người dùng hơn.

Add group

Name:

Permissions:

Available permissions ?

- auth | group | Can add group
- auth | group | Can change group
- auth | group | Can delete group
- auth | group | Can view group
- auth | permission | Can add permission
- auth | permission | Can change permission
- auth | permission | Can delete permission
- auth | permission | Can view permission
- contenttypes | content type | Can add content type
- contenttypes | content type | Can change content type
- contenttypes | content type | Can delete content type
- contenttypes | content type | Can view content type
- corsheaders | cors model | Can add cors model

Choose all ?

Chosen permissions ?

- admin | log entry | Can change log entry
- admin | log entry | Can delete log entry
- admin | log entry | Can add log entry
- admin | log entry | Can view log entry

Remove all

Hold down "Control", or "Command" on a Mac, to select more than one.

Hình 3.4 Group và Permission trang Admin

Django cho phép ta kiểm tra xem người dùng có permission đó không thông qua câu lệnh:

- has_perm(self, perm, obj=None): kiểm tra với 1 permission.
- has_perms(self, perm_list, obj=None): kiểm tra user có nhiều permission.

3.6. Query

ORM(Object Relational Mapping) là một kỹ thuật giúp ta có thể dễ dàng thực hiện truy vấn, tương tác với dữ liệu dưới cơ sở dữ liệu ngay trên code.

Về bản chất, Django cũng là một ORM, ta có thể dễ dàng truy vấn dữ liệu thông qua các câu QuerySet, ví dụ:

```
user = User.objects.get(pk=1)
```

Các câu QuerySet này sẽ thông qua một Manager của lớp model đại diện là “objects” mà thực hiện truy vấn đến bảng tương ứng với lớp model đó.

Vì thế, ta có thể thực hiện tạo ra các QuerySet để truy vấn như lọc, thống kê, join bảng... một cách dễ dàng. Các Queryset này sẽ không thực thi xuống cơ sở dữ liệu cho đến khi có một câu lệnh nào đó được thực hiện.

3.7. Giới thiệu Django Rest Framework

Với nhu cầu tạo ra các API nhanh chóng và linh hoạt, Django cung cấp cho ta một framework để đáp ứng nhu cầu đó, đó là Django Rest Framework.

Django Rest Framework (DRF) là một bộ công cụ giúp cho lập trình viên có thể tạo ra những Restful API một cách nhanh chóng, an toàn và bảo mật để cung cấp cho Client sử dụng, do có hỗ trợ nhiều authentication policies cũng như giao diện duyệt API hiệu quả, kèm theo đó là tài liệu học tập phong phú, cũng như khả năng khai thác thông tin học tập từ các nguồn cộng đồng IT lớn.

Cài đặt môi trường:

Bước 1: Mở terminal và gõ lệnh:

```
pip install djangorestframework
```

Bước 2: Vào biến INSTALLED_APPS, khai báo:

```
INSTALLED_APPS = [  
    'rest_framework',  
]
```

3.8. View

Lớp `APIView` là lớp con duy nhất kế thừa từ lớp `View` trong Django. Ta có thể viết các API mà không cần dựa trên model, được sử dụng dễ dàng như khi sử dụng `View`. Các đặc điểm chính làm cho lớp `APIView` khác với những lớp `View` thông thường khác là:

- Đối tượng `Request` được gửi tới xử lý là một thể hiện của `Rest Framework` chứ không phải là thể hiện của `Django Request (HttpRequest)`
- Những dữ liệu trả về phương thức response là thể hiện của `Response` của `Rest Framework`
- Những ngoại lệ bắt được sẽ là `APIException`
- Ta có thể tiếp cận `APIView` bằng class hay function đều được.
- Ví dụ tiếp cận bằng class ta làm như sau:

```
from rest_framework.views import APIView

class UserView(APIView):
    def get(self, request, format=None):
        users = User.objects.filter(is_active=True)
        return Response(users)
```

Đây là cách tiếp cận bằng hàm

```
from rest_framework.decorators import api_view

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def user_view(request):
    if request.method == 'GET':
        users = User.objects.filter(is_active=True)
```

3.9. ViewSet

`ViewSet` là lớp con được kế thừa từ `APIView`, được hình thành từ việc kết hợp logic các view liên quan lại với nhau.

Các lớp `ViewSet` này sẽ không thực hiện sẵn cho lập trình viên các action như `Get`, `Post`... Nhưng thay vào đó là các phương thức như `list`, `retrieve`, `update`, `destroy`, `create`. Ví dụ:

```

from .models import Item
from .serializers import ItemSerializer
from rest_framework import generics

class ItemsViewSet(viewsets.ViewSet):
    queryset = Item.objects.all()
    serializer_class = ItemSerializer

    def list(self, request):
        items = Item.objects.all()

        serializer = ItemSerializer(items, many=True)
        return Response(data=serializer.data)

    def create(self, request):
        data = request.data

        item = Item.objects.create(name=data['subject'],
        quantity=data['quantity'],

        supplier=data['supplier_id'])
        serializer = ItemSerializer(item)
        return Response(serializer.data,
        status=status.HTTP_201_CREATED)

```

3.10. ViewSet

GenericView là lớp con được kế thừa từ APIView nhằm giúp phát triển nhanh các API có xu hướng lặp đi lặp lại. Ngoài ra, GenericView còn bổ sung thêm các hành vi như list, detail. Ví dụ:

```

from .models import Item
from .serializers import ItemSerializer
from rest_framework import generics

class ItemList(generics.ListAPIView, generics.RetrieveAPIView):
    queryset = Item.objects.all()
    serializer_class = ItemSerializer
    pagination_class = None

```

Ở ví dụ này, DRF sẽ thực hiện sẵn cho ta hai API đó là lấy danh sách các Item và lấy chi tiết một Item:

item			▼
GET	/item/	item_list	🔒
GET	/item/{id}/	item_read	🔒

Hình 3.5 Các APIs đã tạo.

Khi sử dụng `GenericView`, ta cần cung cấp cho nó các thuộc tính như sau:

- `queryset`: trả về các đối tượng từ view, đặc biệt, trong một vài trường hợp thì ta có thể ghi đè phương thức: `get_queryset()` để lấy ra queryset mà mình muốn.
- `serializer_classes`: lớp serializer nhằm kiểm tra dữ liệu đầu vào có đúng với những điều kiện được ta thiết lập hay không hay deserializer dữ liệu đó. Lớp này cũng có thể thay đổi trong một vài trường hợp, ta ghi đè phương thức: `get_serializer()`.
- `get_object()`: đây là phương thức giúp ta có thể lấy được detail objects.
- `lookup_field`: chỉ định các field để lọc dữ liệu của model, mặc định là pk.
- `lookup_url_kwarg`: chỉ định các đối số url argument.
- `pagination_class`: ta có thể dùng lớp này để phân trang nếu không muốn dùng mặc định cấu hình phân trang của biến `DEFAULT_PAGINATION_CLASS` trong setting. Nếu `pagination_class=None` thì sẽ không phân trang.

3.11. Routers

Django Rest Framework hỗ trợ xử lý tự động các URL trong Django đơn giản, nhanh chóng và nhất quán giúp ta có thể set URL cho project một cách logic, dễ dàng hơn.

```
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register('po', views.CategoryViewSet, basename='Category')
router.register('so', views.ProductViewSet)
router.register('user', views.UserViewSet)
```

Trong method `register()` ta sẽ buộc phải truyền hai đối số sau:

- `prefix`: tiền tố URL sử dụng cho những url dạng `{prefix}/`
 - `viewset`: lớp `ViewSet` sử dụng URL này
- Ngoài ra ta còn có đối số tùy chọn:
- `basename`: tên URL được tạo ra, sẽ tự động tạo dựa trên thuộc tính `QuerySet` của viewset, nếu viewset ko bao gồm queryset thì ta phải khai báo trong `register()`.

Ta có thể sử dụng “include” cùng với thuộc tính “urls” routers, vì các thuộc tính trên routers chỉ đơn giản là một danh sách chuẩn của URL patterns.

```
urlpatterns = [
    path('', include(router.urls)),
]
```

3.12. Request và Response

3.12.1. Request

Request của Rest Framework được kế thừa từ `HttpRequest` chuẩn, nhưng linh hoạt hơn trong việc xử lý các request hay trong việc chứng thực. Cho phép ta xử lý các request dưới dạng JSON hay các dạng dữ liệu khác trong cùng cách thức làm việc như khi ta thường xử lý với dữ liệu form.

Ta có các thuộc tính sau trong request:

- `Request.user`: là thể hiện của user, nếu đã chứng thực thì là thể hiện của “`django.contrib.auth.models.User`”, còn chưa chứng thực là thể hiện của “`django.contrib.auth.models.AnonymousUser`”.
- `Request.data`: chứa những thông tin dữ liệu được gửi lên từ body.
- `Request.method`: trả về phương thức dùng để thực hiện request.
- `Request.query_params`: chứa dữ liệu từ các tham số truyền vào request từ phương thức Get.
- `Request.auth`: chứa thông tin chứng thực, thường là thông tin token của request. Giá trị mặc định nếu chưa được chứng thực là `None`.
- `Request.authenticators`: chứa thông tin về các chính sách chứng thực trong các lớp `APIView` hay `@api_view` được thiết lập thông qua thuộc tính `authentication_classes` hay mặc định sẽ lấy trong setting ở biến cấu hình `DEFAULT_AUTHENTICATORS`.
- `Request.content_type`: trả về chuỗi đại diện media type của HTTP request body hoặc là chuỗi rỗng nếu không có media type nào được cung cấp.

3.12.2. Response

Là lớp trả về dữ liệu nội dung có thể render ra thành nhiều dạng kiểu dữ liệu khác nhau. Đây là lớp con của `Django SimpleTemplateResponse`.

Các thuộc tính của lớp này là:

- `Data`: là dữ liệu đã được serializer.

- Status: status code được trả về cho response, mặc định là HTTP_200_OK.
- Headers: là một từ điển của HTTP headers để sử dụng trong response.
- Content_type: kiểu nội dung được trả về cho response, được thiết lập tự động nhưng trong vài trường hợp, ta có thể thiết lập dễ dàng.
- Template_name: template được chọn sử dụng.

3.13. Serializer

Serializer là cơ chế chuyển các kiểu dữ liệu phức tạp như QuerySet, hay thể hiện của một model thành các kiểu dữ liệu python đơn giản hơn như JSON, XML... Ngược lại thì Deserializer là cơ chế chuyển các kiểu dữ liệu python đơn giản trở về kiểu dữ liệu ban đầu sau khi đã validate dữ liệu.

Lớp Serializer cung cấp các phương thức chung giúp cho việc xử lý dữ liệu output trong Response.

Khi kiểm tra dữ liệu đầu vào, nếu trường được khai báo là required là False thì việc kiểm tra sẽ không được thực hiện trên trường đó. Ta có thể kiểm tra dữ liệu đầu vào bằng hai phương thức:

- validate(): phương thức này chứa giá trị duy nhất các trường dưới dạng từ điển.
- validate_<field_name>: phương thức này kiểm tra giá trị của trường tương ứng và ném ra ngoại lệ là “serializers.ValidationError”.

Lớp ModelSerializer: là lớp giúp cho ta nhanh chóng tạo ra một serializer với các trường tương ứng với model được khai báo trong “class Meta”. Khi đó, nó sẽ tự tạo cho ta:

- Các trường giống với model mà được ta khai báo.
- Những validator cho serializer: đó là những validate mà ta khai báo khi tạo model. Ví dụ: null là False.
- Các phương thức mặc định cơ bản của một model là thêm, sửa, xóa.

Lớp HyperLinkedModelSerializer: là lớp tương tự như lớp ModelSerializer, nhưng sử dụng những siêu liên kết đại diện cho các mối quan hệ.

3.13.1.1. ModelSerializer

Trong lớp ModelSerializer, ta có các thuộc tính quan trọng thường dùng trong meta class:

- Model: đây là lớp model được liên kết với serializer.
- Fields: các trường trong model hoặc các trường được khai báo trong serializer.
- read_only_fields: Các trường được cho phép chỉ đọc.
- exclude: Các trường mà ta không muốn serializer thì khai báo ở đây.
- extra_kwargs: các thông tin mà ta muốn ràng buộc thêm trên fields trong serializer.

3.14. Authentication

Authentication là cơ chế kết hợp những thông tin trong request với các tập các thông tin chứng thực. Các chính sách permission và throttling sau đó có thể dùng những thông tin chứng thực này xem request có được phép hay là không, vì thế, Authentication luôn chạy trước view và trước permission và throttling.

Ta có hai thuộc tính thường dùng, đó là:

- Request.user: là thể hiện của gói contrib.auth.
- Request.auth: chứa những thông tin chứng thực của request, ví dụ như lưu token.

Authentication schemes được định nghĩa là một danh sách các lớp python. Rest Framework sẽ tham gia vào chứng thực từng lớp python trong danh sách và sẽ thiết lập giá trị trả về cho lớp được chứng thực thành công đầu tiên, giá trị được thiết lập đó là request.user và request.auth.

Ta cấu hình authentication schemes bằng cách chỉ định các lớp mặc định chứng thực trong biến REST_FRAMEWORK trong setting:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication',
        'rest_framework.authentication.SessionAuthentication'),
}
```

hoặc có thể cấu hình bằng thuộc tính `authentication_classes` trong các lớp view.

Nếu một request không được chứng thực thì sẽ có hai status code phù hợp sau: HTTP 401 – Unauthorized và HTTP 403 – Permission Denied.

3.15. Giới thiệu OAuth2

OAuth2 là phiên bản hiện tại mới nhất của OAuth (Open Authentication), được phát triển và được sử dụng phổ biến và rộng rãi bởi Facebook, google,... Là một giao thức mở đường để chứng thực với các dịch vụ liên quan khác.

OAuth hoạt động bằng cách ủy thác xác thực cho một dịch vụ để truy cập vào các tài nguyên mà không cần đến username, password.

3.15.1. OAuth2

Muốn sử dụng OAuth2 ta cần phải đăng ký một application với OAuth2 Provider.

Khi kết thúc quá trình đăng ký, ta sẽ nhận được những thông tin sau:

- Client ID: là thông tin công khai ứng dụng.
- Client secret: là thông tin riêng tư gửi lên server với Client ID.
- Authorization server URL: URL để client sử dụng.
- Access token URL: URL để lấy access_token, refresh_token,...
- Resource server URL: URL tài nguyên được bảo vệ mà cho phép client sử dụng.

Để giao tiếp với server từ client thông qua OAuth2, ta cần phải trải qua ba bước:

- Authorized: chứng thực quyền truy cập tài nguyên được bảo vệ của user.
- Yêu cầu lấy access token.
- Truy cập vào những tài nguyên được cho phép.

3.15.2. Django OAuth Toolkit

Gói Django OAuth Toolkit cung cấp các thông tin dữ liệu, các endpoint và logic để thực hiện OAuth trong Django. Nó hỗ trợ OAuth2 cùng với python từ phiên bản 3.4 trở lên.

Các bước cài đặt và sử dụng:

Bước 1: cài đặt gói Django-oauth-toolkit vào project:

```
pip install django-oauth-toolkit
```

Bước 2: Khai báo OAuth2 provider trong biến INSTALLED_APP trong setting:

```
INSTALLED_APPS = (  
    ...  
    'oauth2_provider',  
)
```

Bước 3: Thêm cấu hình cho biến REST_FRAMEWORK:

```
REST_FRAMEWORK = {  
    ...  
    'DEFAULT_AUTHENTICATION_CLASSES': (  
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication'  
,  
    )  
}
```

Bước 4: Thêm urls cho URLConf:

```
from django.urls import include, path  
urlpatterns = [  
    ...  
    path('o/', include('oauth2_provider.urls',  
        namespace='oauth2_provider')),  
]
```

Bước 5: Thực thi migrate xuống cơ sở dữ liệu.

Sau khi xong các bước trên, ta truy cập vào: <http://127.0.0.1:8000/o/applications/> để tạo app.

Register a new application

Name

Client id

Client secret

Client type

Authorization grant type

Redirect uris

Algorithm

Hình 3.6 Giao diện tạo application chứng thực.

Sau khi tạo app xong ta sẽ được những thông tin như sau:

WMSPY

Client id

Client secret

Client type
confidential

Authorization Grant Type
password

Redirect Uris

Hình 3.7 Giao diện tạo thành công.

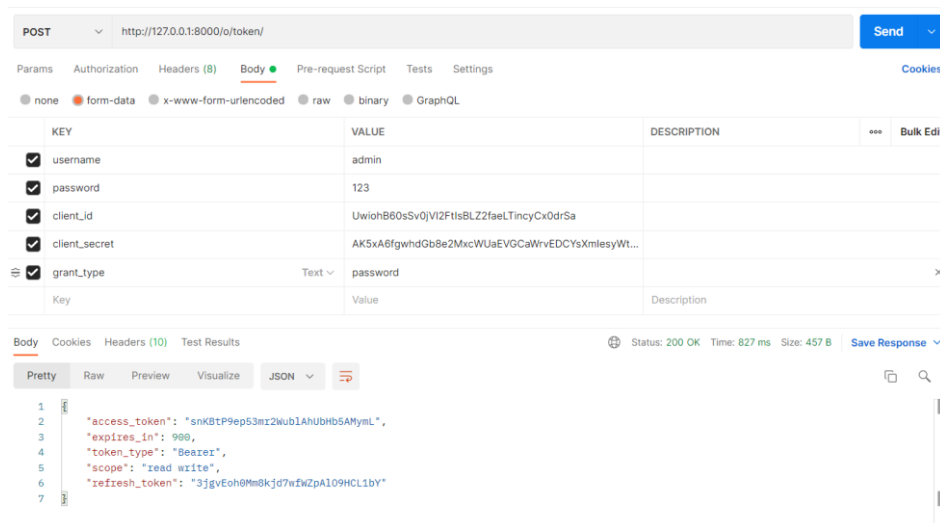
Ta truy cập vào Url: /o/token/, method POST cùng với body data như sau để lấy access token:

```
{
  "grant_type": "password",
  "username": "<your_username>",
  "password": "<your_password>",
  "client_id": "<your_client-id>",
```

```

"client_serect": "<your_client-serect>"
}

```



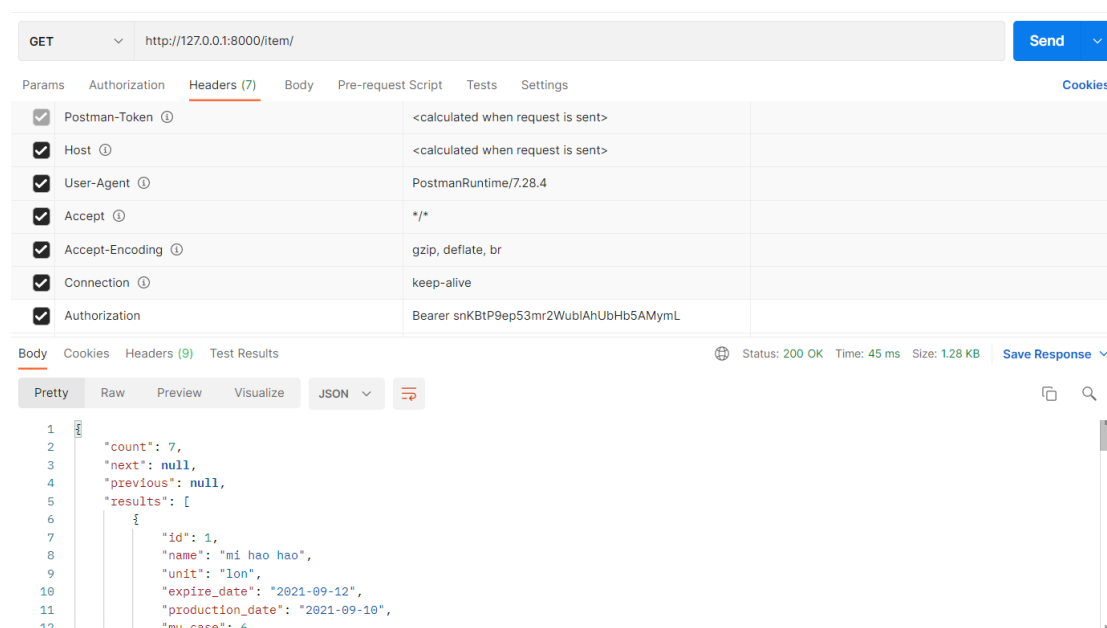
Hình 3.8 Lấy token thành công.

Ta sử dụng access token này trong phần HTTP header của mỗi request ta gửi lên phải có thêm Authorization với giá trị của nó có thể là Token hoặc Bearer, với cú pháp như sau:

```

Authorization: Bearer <access_token>

```



Hình 3.9 Sử dụng token vừa tạo để thực hiện API.

3.16. Cors

Cors là viết tắt của Cross-origin resource sharing. Đó là một cơ chế dựa trên HTTP header cho phép giới hạn những yêu cầu truy cập tài nguyên từ domain, schemes hay port khác.

Cors còn cho phép ta xác định được request cross-origin nào có an toàn hay không.

Cách cài đặt:

Bước 1: cài đặt cors middleware

```
pip install django-cors-middleware
```

Bước 2: Bổ sung vào INSTALLED_APPS trong setting:

```
INSTALLED_APP = [  
    ...,  
    'corsheaders'  
]
```

Bước 3: Ta cập nhật biến MIDDLEWARE trong setting, giá trị sau đây nên đặt trước để có thể tạo ra response.

```
MIDDLEWARE = [  
    'corsheaders.middleware.CorsMiddleware',  
    ...  
]
```

Bước 4: Thêm biến CORS_ORIGIN_ALLOW_ALL với giá trị là True trong setting, hoặc ta có thể chỉ định rõ những domain nào được phép truy cập.

```
CORS_ORIGIN_ALLOW_ALL = True
```

3.17. Tích hợp Swagger

Swagger là bộ công cụ mã nguồn mở OpenAPI specifications giúp việc xây dựng, thiết kế tài liệu và quản lý Restful API được trực quan, rõ ràng và hiệu quả.

Cách cài đặt:

Bước 1: Cài đặt drf-yasg

```
pip install django-cors-middleware
```

Bước 2: Cập nhật biến `INSTALL_APPS` trong setting:

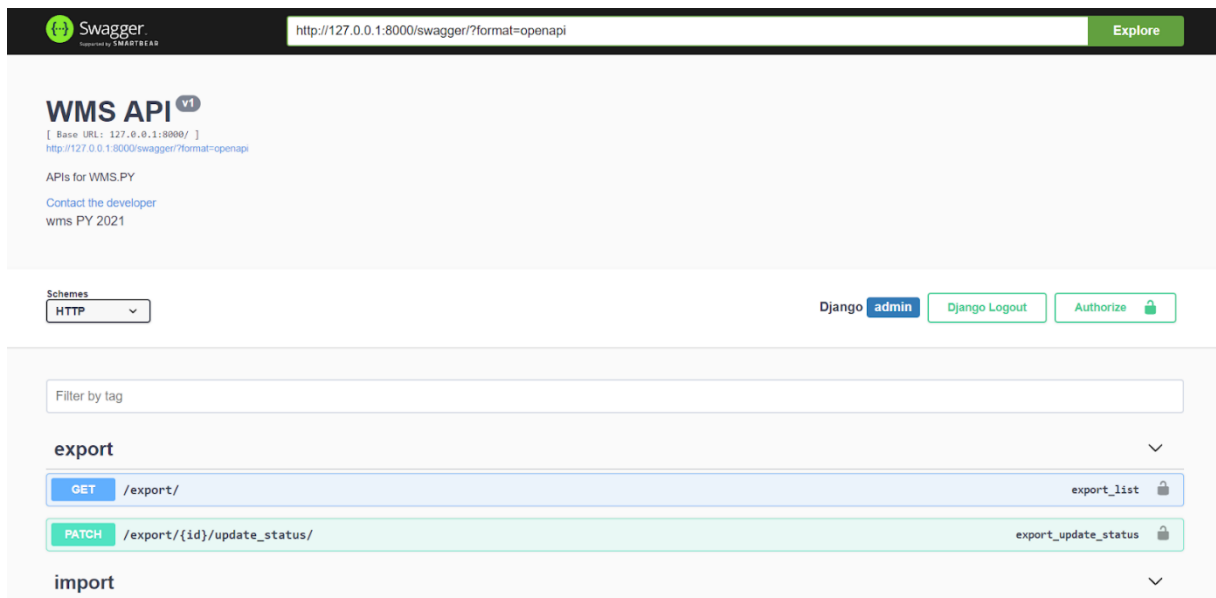
```
INSTALLED_APPS = [  
    ...  
    'django.contrib.staticfiles',  
    'drf_yasg',  
]
```

Bước 3: Cập nhật url cho URLConf

```
from rest_framework import permissions  
from drf_yasg.views import get_schema_view  
from drf_yasg import openapi  
  
schema_view = get_schema_view(  
    openapi.Info(  
        title="WMS API",  
        default_version='v1',  
        description="APIs for WMS.PY",  
        contact=openapi.Contact(email="1851050120phuoc@ou.edu.vn"),  
        license=openapi.License(name="wms PY 2021"),  
    ),  
    public=True,  
    permission_classes=(permissions.AllowAny,),  
)
```

```
urlpatterns = [  
    ...  
    re_path(r'^swagger(?P<format>\.json|\.yaml)$',  
        schema_view.without_ui(cache_timeout=0), name='schema-json'),  
    re_path(r'^swagger/$', schema_view.with_ui('swagger',  
        cache_timeout=0), name='schema-swagger-ui'),  
    re_path(r'^redoc/$', schema_view.with_ui('redoc',  
        cache_timeout=0), name='schema-redoc'),  
]
```

Sau khi hoàn tất các bước trên ta chạy project và truy cập vào Url `/swagger/` ta sẽ thấy được như sau:



Hình 3.10 Giao diện Swagger.

3.18. Cloudinary storage

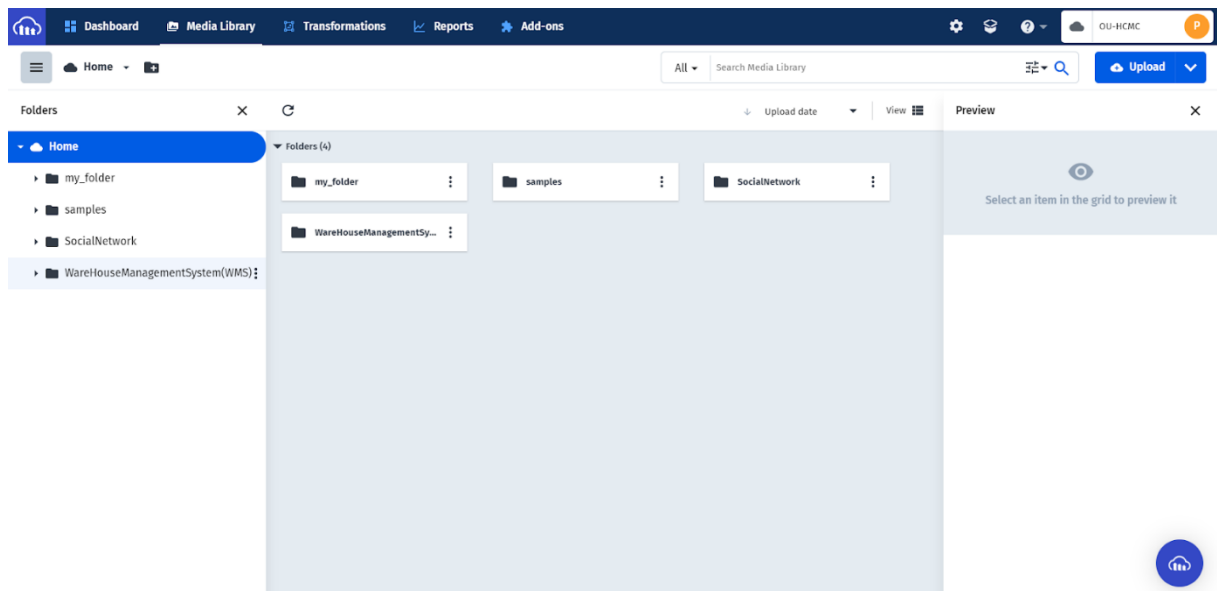
Cloudinary là một cloud-based service, cho phép ta quản lý các static file như hình ảnh, video một cách dễ dàng. Cloud có thể sử dụng với hai phiên bản miễn phí và mất phí.

Cài đặt

Bước 1: tạo một tài khoản Cloudinary trên <https://cloudinary.com/> để nhận các thông tin sau:

- API key
- API secret
- Cloud name

Bước 2: tạo một thư mục muốn lưu trữ hình ảnh của project trong tab MEDIA LIBRARY. Ví dụ ta tạo thư mục có tên WarehouseManagementSystem (WMS)



Hình 3.11 Giao diện MEDIA LIBRARY trên Cloudinary.

Bước 3: Cài đặt trong project python, ta mở terminal và gõ lệnh sau:

```
pip install django-cloudinary-storage
```

Bước 4: Trong setting.py ta khai báo các biến sau:

```
MEDIA_URL = '/WareHouseManagementSystem(WMS) /'

DEFAULT_FILE_STORAGE =
'cloudinary_storage.storage.MediaCloudinaryStorage'
CLOUDINARY_STORAGE = {
    'CLOUD_NAME': 'your cloud name',
    'API_KEY': 'your API key',
    'API_SECRET': 'your API secret'
}
INSTALLED_APPS = [
    # ...
    'cloudinary_storage',
    'django.contrib.staticfiles',
    'cloudinary',
    # ...
]
```

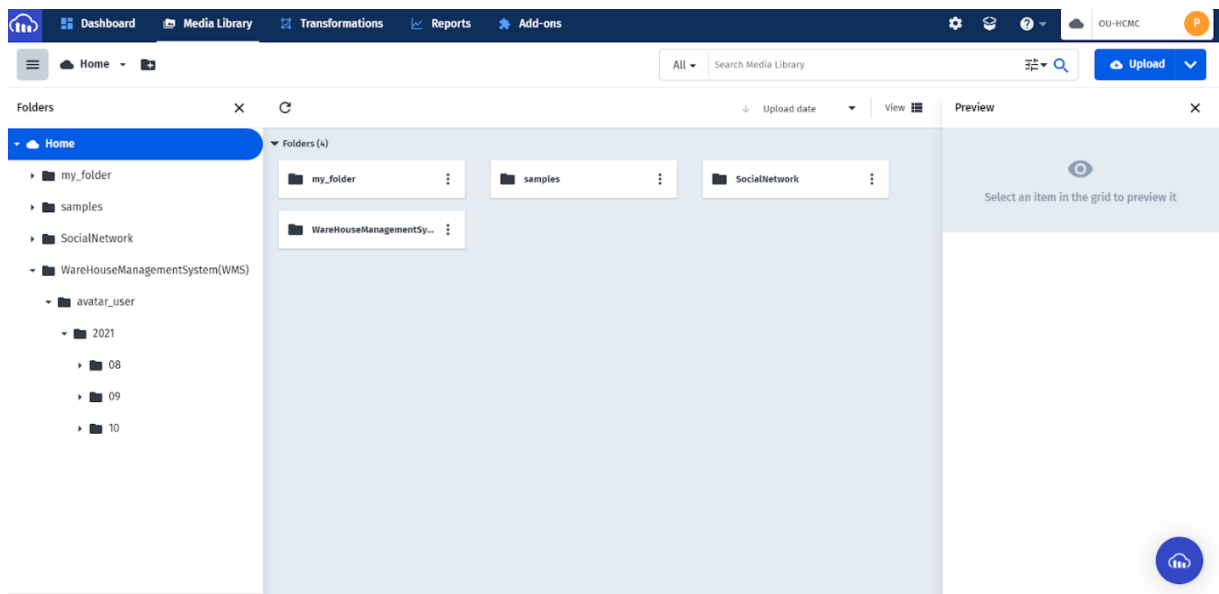
- MEDIA_URL: ta chỉ tới URL thư mục ta vừa tạo trên Cloudinary ở đây là ‘/WarehouseManagementSystem(WMS)’.
- DEFAULT_FILE_STORAGE: lưu thông tin kho lưu trữ.
 - Ảnh: MediaCloudinaryStorage.
 - Video: VideoMediaCloudinaryStorage.

- Các file như .txt, .pdf, ...: RawMediaCloudinaryStorage.

Bước 5: Sử dụng trong model thông qua ImageField của Django, ví dụ:

```
class User(AbstractUser):
    avatar = models.ImageField(upload_to='avatar_user/%Y/%m',
                               blank=True)
```

Như vậy ta sẽ có một cây thư mục lưu trữ hình ảnh trên Cloudinary như sau:



Hình 3.12 Cây thư mục lưu avatar user vừa tạo

Chương 4. Phân tích hệ thống WMS.PY

4.1. Xác định yêu cầu

Dựa vào những thông tin khảo sát nghiệp vụ, từ internet trong những năm gần đây đặc biệt là trong thời gian đại dịch Covid-19.

Nhằm đưa ra giải pháp giải quyết những mục đích sau:

- Giảm tải hao tổn sức lực không đáng có tại các doanh nghiệp lĩnh vực ngành công nghiệp Logistic.
- Xử lý hàng tồn kho.
- Giảm thiểu sai sót do con người.
- Tiết kiệm thời gian xây dựng phát triển ứng dụng web thương mại lớn, nhanh chóng đưa vào sử dụng cho doanh nghiệp.
- Từ đó, xác định những yêu cầu cần có của một hệ thống quản lý kho hàng:

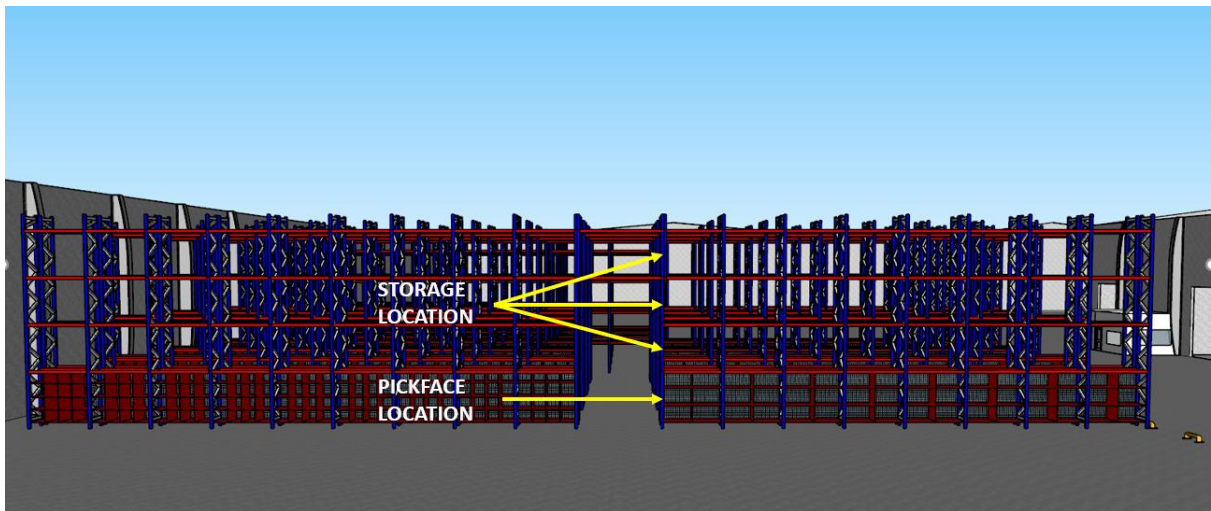
- Đăng nhập, đăng xuất.
- Quản lý đơn hàng nhập/xuất hàng từ nhà cung cấp.
- Quản lý biên nhận nhập/xuất mỗi lần vận chuyển.
- Phân bổ vị trí tác vụ nhập hàng vào kho hàng cho đơn nhập.
- Phân bổ vị trí tác vụ rút hàng, lấy hàng và sắp xếp hàng cho đơn xuất.
- Quản lý tài khoản nhà cung cấp, nhân viên.
- Theo dõi, cập nhật tình trạng nhập hàng vào kho.
- Theo dõi, cập nhật tình trạng rút hàng, pick hàng và sắp xếp hàng khi xuất hàng.
- Thống kê.

4.2. Tổng quan nghiệp vụ hệ thống

Quản lý kho hàng là hệ thống lưu trữ, quản lý hàng hóa, đơn hàng, hoạt động điều vận trong kho cũng như giám sát vận chuyển hàng hóa nhằm phục vụ cho các kho hàng, các đơn vị tiếp vận ngành Logistic.

Các thuật ngữ trong nghiệp vụ:

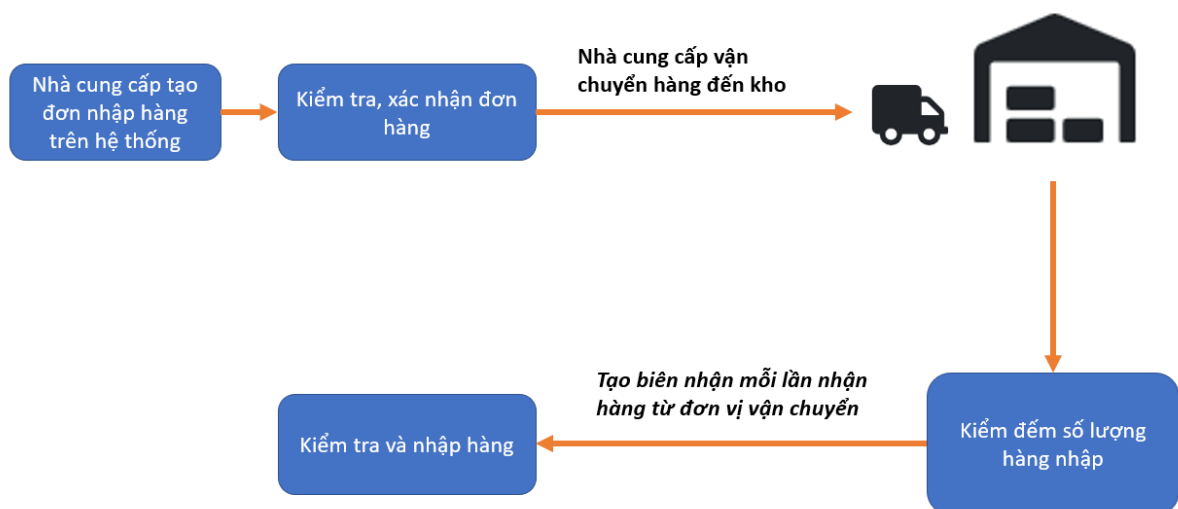
- Vị trí (Location): Nơi lưu trữ sản phẩm tồn kho. Vị trí được phân thành 2 loại sau:
 - Vị trí lưu kho (Storage): Nơi lưu trữ sản phẩm tồn kho, mỗi vị trí chỉ được lưu 1 loại sản phẩm và tối đa số lượng sản phẩm là 20 thùng carton.
 - Vị trí lấy hàng (PickFace): hay còn gọi là Fast pick hoặc Moving. Nơi lưu trữ những sản phẩm sau khi rút hàng và chuẩn bị cho người lao động đi lấy hàng.
 - Định danh của mỗi vị trí được cấu thành từ tên dãy + tên cột + tên tầng.



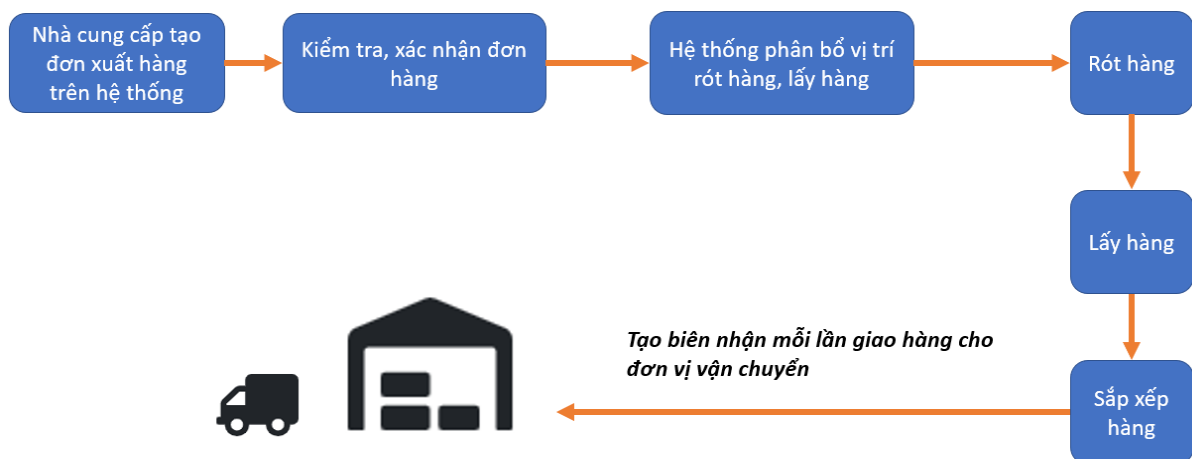
Hình 4.1 Các loại Location trong kho hàng

- Đơn nhập hàng(PO - Purchase Order): Đơn này nhà cung cấp tạo mỗi khi muốn nhập hàng vào kho hàng.
- Đơn xuất hàng(SO - Shipment Order): Đơn này nhà cung cấp tạo mỗi khi muốn xuất hàng giao cho khách hàng.
- Biên nhận nhập hàng(Receipt): Đơn này được nhân viên tạo mỗi lần nhận hàng từ đơn vị vận chuyển.
- Biên nhận xuất hàng(Order): Đơn này được nhân viên tạo mỗi lần xuất hàng khỏi kho giao cho đơn vị vận chuyển.
- Rót hàng(Allocate): Thao tác người lao động sử dụng xe nâng lấy hàng từ vị trí lưu kho sau đó đổ đầy vào các vị trí lấy hàng.
- Lấy hàng(Picking): Thao tác người lao động đi lấy hàng từ mỗi vị trí lấy hàng.
- Sắp xếp hàng(Sorting): Thao tác người lao động sắp xếp hàng hóa ra thành từng loại để gói hàng sau khi lấy hàng.
- Các trạng thái thường gặp trong hệ thống:
 - PENDING: đơn hàng mới được tạo và chờ nhân viên xác nhận kiểm tra đơn hàng.
 - FAILED: đơn hàng đã được kiểm tra và bị lỗi.
 - ACCEPTED: đơn hàng đã được kiểm tra và được chấp nhận
 - DONE
 - SO: đơn hàng đã hoàn thành.
 - PO: đơn hàng đã hoàn thành và chờ được nhập hàng lên kệ

- IMPORTED: đơn hàng đã được nhập lên kệ
- EXPORTED: đơn hàng đã được sắp xếp và chờ xuất hàng cho đơn vị vận chuyển.
- ALLOCATED: đơn hàng đã được phân bổ rút hàng.
- PICKED: đơn hàng đã được người lao động đi lấy hàng.
- SORTED: đơn hàng đã được người lao động sắp xếp.



Hình 4.2 Quy trình nhập hàng.



Hình 4.3 Quy trình xuất hàng.

4.3. Phân tích yêu cầu

Trong hệ thống chức năng đăng nhập để sử dụng hệ thống được phân thành 3 vai trò: Admin, User, Supplier. Trong đó Admin có toàn quyền quản lý tất cả trong hệ thống.

Các chức năng trong hệ thống:

- **Tạo đơn nhập/xuất:** chỉ nhà cung cấp hay admin có thể tạo.
 - Đối với đơn nhập: khách hàng sẽ chọn hoặc **tạo mới sản phẩm** muốn nhập, việc tạo mới sản phẩm sẽ phải được thực hiện trước khi tạo đơn nhập.
 - Đối với đơn xuất: khách hàng chỉ việc chọn sản phẩm muốn xuất kho, sau đó hệ thống sẽ tìm những sản phẩm đó theo hạn sử dụng. Sản phẩm nào trùng tên được chọn và có hạn sử dụng hết hạn trước thì sẽ được chọn để tạo đơn xuất cho nhà cung cấp. Hệ thống sẽ tự lấy sản phẩm cùng tên khác nếu không đủ số lượng yêu cầu. Khách hàng sẽ không thể tạo đơn xuất nếu số lượng yêu cầu xuất kho lớn hơn số lượng tồn kho.
- **Cập nhật trạng thái đơn nhập/xuất:** ghi lại thời gian cập nhật cũng như nhân viên cập nhật, chỉ nhân viên/admin có thể cập nhật. Nhân viên xác nhận lần đầu sẽ ghi lại và không thể chỉnh sửa hay xóa. Các nhân viên cập nhật trạng thái đơn nhập/xuất sau sẽ được lưu lại.
- **Xóa đơn nhập/xuất:** nhà cung cấp hoặc admin có thể xóa. Nhà cung cấp chỉ được xóa khi đơn nhập/xuất chưa được nhân viên xác nhận tức trạng thái là ACCEPT.
- **Tạo biên lai nhập/xuất:** nhân viên/admin sử dụng. Được tạo mỗi lần nhận/giao hàng cho đơn vị vận chuyển.
 - Đối với biên lai nhập: tạo khi đơn nhập có trạng thái là ACCEPT, nếu số lượng hàng nhận được đầy đủ so với số lượng hàng trong đơn nhập thì cập nhật trạng thái đơn nhập đại diện cho các biên lai nhập đó thành DONE. đồng thời thêm các sản phẩm trong đơn lên hệ thống với số lượng tương ứng trong đơn nhập.
 - Đối với biên lai xuất: tạo khi đơn xuất có trạng thái là EXPORTED, nếu số lượng hàng xuất kho đầy đủ so với số lượng hàng trong đơn xuất thì

cập nhật trạng thái đơn xuất đại diện cho các biên lai xuất đó thành DONE.

- **Cập nhật biên lai nhập/xuất:** nhân viên/admin sử dụng. Cập nhật trạng thái các đơn hàng khi có sửa đổi hoặc hoàn thành tác vụ nhận/giao hàng cho đơn vị vận chuyển.
 - Nếu đã đã nhận/giao đầy đủ số lượng hàng so với số lượng hàng trong đơn nhập/xuất thì sẽ cập nhật trạng thái đơn nhập/xuất đại diện cho các biên lai nhập/xuất đó thành DONE. Đồng thời thêm/bớt các sản phẩm trong đơn lên hệ thống với số lượng tương ứng trong đơn nhập/xuất.
 - Một khi đơn nhập/xuất đã chuyển sang trạng thái DONE thì không thể tạo, cập nhật hay xóa biên lai nữa.
- **Xóa biên lai nhập/xuất:** nhân viên/admin sử dụng. Cập nhật trạng thái đó thành False. Đối với admin thì có thể xóa hẳn biên lai.
- **Nhập hàng vào kho:** cho nhân viên/admin sử dụng.
 - Hệ thống sẽ kiểm tra các điều kiện trước khi bắt đầu xử lý thông đơn nhập cho việc nhập hàng, đó là: Kiểm tra có đủ vị trí storage trống cho việc nhập hàng, mỗi vị trí chỉ chứa một loại sản phẩm với cùng hạn sử dụng. Đơn nhập có trạng thái là IMPORTED thì sẽ không thể nhập hàng được nữa. Đơn nhập đã được phân vị trí nhập hàng thì sẽ lấy kết quả từ lần phân vị trí đầu tiên, không phân lại nữa.
- **Phân vị trí nhập hàng:** hệ thống phân vị trí lần lượt cho các sản phẩm trong đơn nhập. Phân vào các vị trí STORAGE có trạng thái trống hoặc cùng thông tin sản phẩm. Nếu vị trí đủ số lượng mà chưa phân đủ số lượng yêu cầu thì hệ thống sẽ chuyển qua vị trí hợp lệ khác phân hàng và lần lượt đến khi phân đủ số lượng sản phẩm đó.
 - Thông tin phân vị trí nhập hàng sẽ bị xóa khỏi hệ thống trong vòng 7 ngày.
- **Xuất hàng khỏi kho:** cho nhân viên/admin sử dụng. Việc này gồm có các thao tác chính sau: Rót hàng, lấy hàng và sắp xếp hàng.
 - Hệ thống sẽ kiểm tra các điều kiện trước khi bắt đầu xử lý thông đơn xuất cho việc xuất hàng, đó là: kiểm tra có đủ vị trí PICKFACE trống cho việc rót hàng, mỗi vị trí chứa một hoặc nhiều loại sản phẩm loại sản

phẩm. Đơn xuất có trạng thái là EXPORTED hay DONE thì sẽ không thể rút hàng được nữa. Đơn xuất đã được phân vị trí rút hàng thì sẽ lấy kết quả từ lần phân vị trí đầu tiên, không phân lại nữa.

- Việc xuất hàng sẽ lần lượt trải qua 3 giai đoạn tương ứng với 3 trạng thái: rút hàng - ALLOCATED, lấy hàng - PICKED, sắp xếp hàng - SORTED.
- Cập nhật trạng thái sẽ phải trải qua đủ các giai đoạn
- Thông tin phân vị trí rút hàng sẽ bị xóa khỏi hệ thống trong vòng 7 ngày.
- **Thông kê:** thống kê theo các đơn hàng xuất và nhập của các tháng theo năm, thống kê tổng số đơn hàng nhập, đơn hàng xuất, các đơn hàng thành công, tổng số sản phẩm... Ở chức năng này thống kê theo quyền, nghĩa nếu là nhà cung cấp thì thống kê các thông tin trên theo nhà cung cấp còn là Admin và nhân viên thì thống kê theo toàn bộ hệ thống.

4.4. Thiết kế cơ sở dữ liệu

4.4.1. Bảng tài khoản người dùng – User

Mô tả: lưu tất cả tài khoản người dùng trong hệ thống, được chia thành 3 vai trò: Admin, User và Supplier

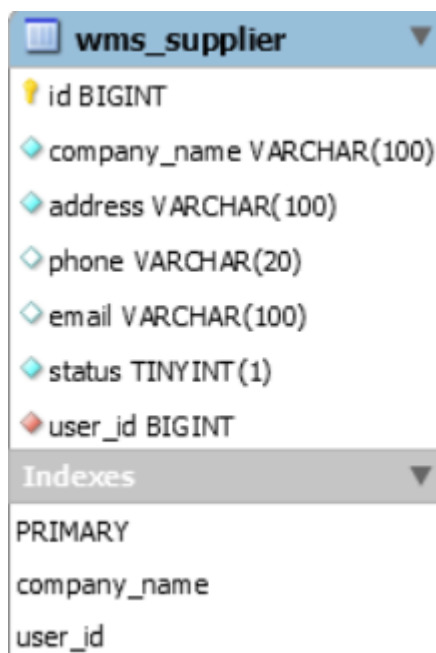


wms_user	
id	BIGINT
password	VARCHAR(128)
last_login	DATETIME(6)
is_superuser	TINYINT(1)
username	VARCHAR(150)
first_name	VARCHAR(150)
last_name	VARCHAR(150)
email	VARCHAR(254)
is_staff	TINYINT(1)
is_active	TINYINT(1)
date_joined	DATETIME(6)
avatar	VARCHAR(100)
address	VARCHAR(255)
phone_number	VARCHAR(10)
role	SMALLINT
Indexes	
PRIMARY	
username	

Hình 4.4 Bảng User.

4.4.2. Bảng nhà cung cấp – Supplier

Mô tả: lưu thông tin nhà cung cấp, đối tác của kho hàng. Mỗi nhà cung cấp đều có một tài khoản truy cập để sử dụng hệ thống.

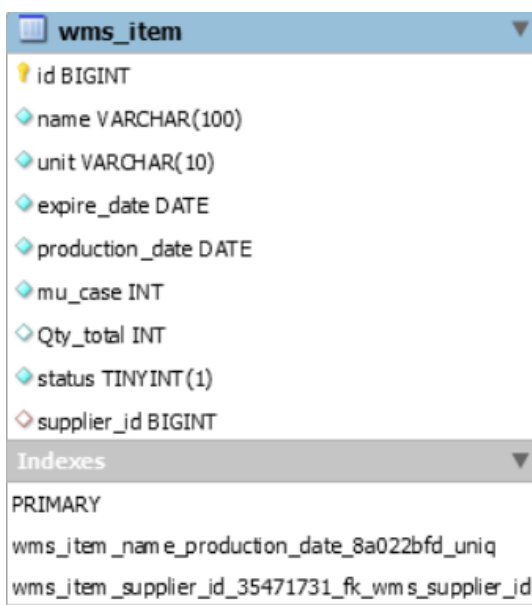


wms_supplier	
id	BIGINT
company_name	VARCHAR(100)
address	VARCHAR(100)
phone	VARCHAR(20)
email	VARCHAR(100)
status	TINYINT(1)
user_id	BIGINT
Indexes	
PRIMARY	
company_name	
user_id	

Hình 4.5 Bảng Supplier.

4.4.3. Bảng sản phẩm – Item

Mô tả: lưu thông tin sản phẩm và số lượng sản phẩm có trong kho. Mỗi sản phẩm đều có một nhà cung cấp riêng.

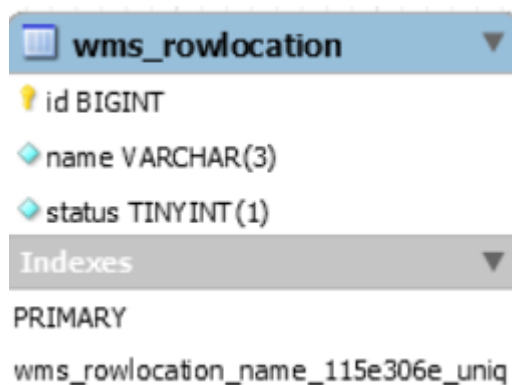


wms_item	
id	BIGINT
name	VARCHAR(100)
unit	VARCHAR(10)
expire_date	DATE
production_date	DATE
mu_case	INT
Qty_total	INT
status	TINYINT(1)
supplier_id	BIGINT
Indexes	
PRIMARY	
wms_item_name_production_date_8a022bfd_uniq	
wms_item_supplier_id_35471731_fk_wms_supplier_id	

Hình 4.6 Bảng Item.

4.4.4. Bảng dãy kệ hàng – Row Location

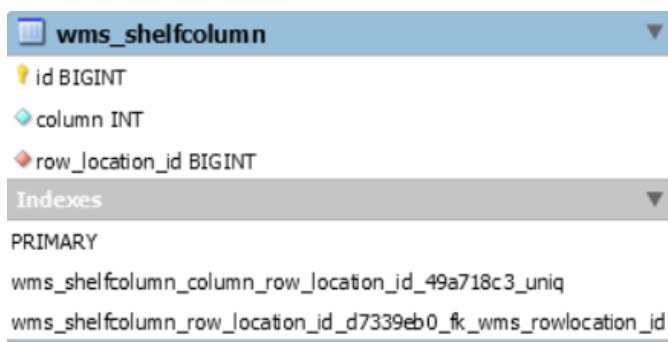
Mô tả: lưu thông tin các dãy kệ chứa hàng trong kho. Trong đó, thông tin tên kệ không trùng nhau.



Hình 4.7 Bảng RowLocation

4.4.5. Bảng cột kệ hàng – Shelf Column

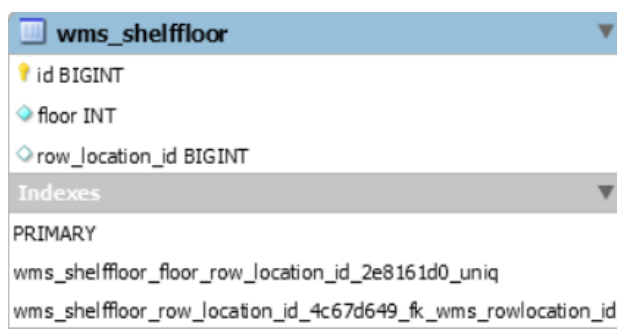
Mô tả: lưu thông tin các cột của kệ hàng. Thông tin tên cột không trùng nhau.



Hình 4.8 Bảng ShelfColumn.

4.4.6. Bảng tầng kệ hàng – Floor Column

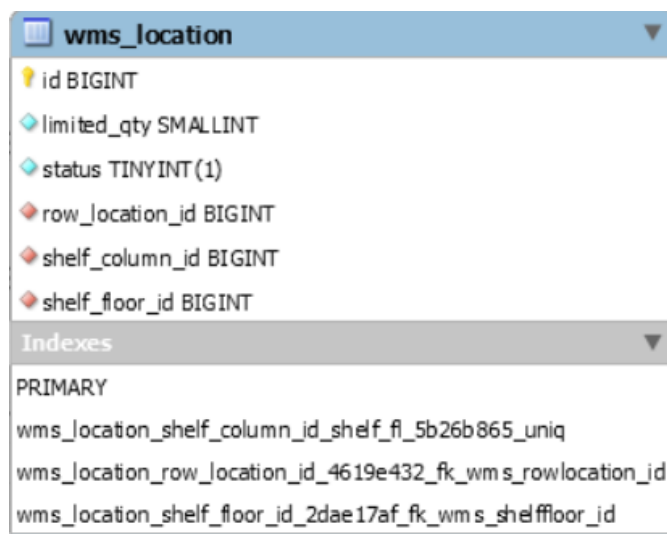
Mô tả: lưu thông tin các tầng của kệ hàng. Thông tin tầng của kệ hàng không trùng nhau.



Hình 4.9 Bảng ShelfFloor.

4.4.7. Bảng vị trí – Location

Mô tả: lưu thông tin từng vị trí lưu hàng trong kho. Mỗi thông tin này đều không được trùng nhau.

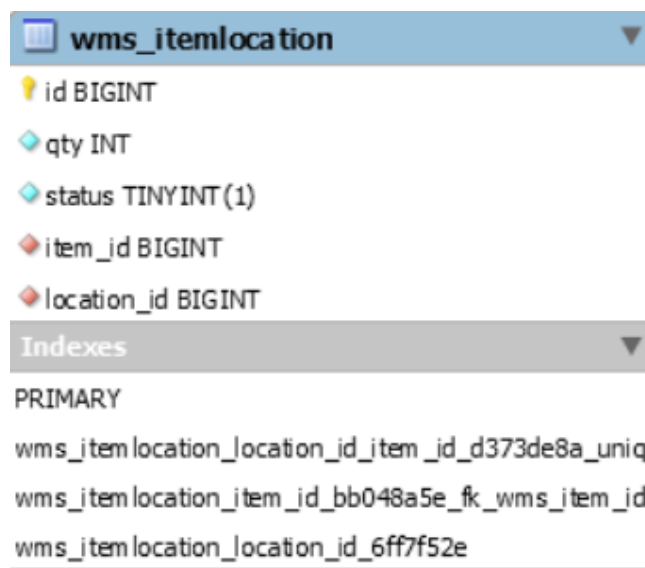


wms_location	
id	BIGINT
limited_qty	SMALLINT
status	TINYINT(1)
row_location_id	BIGINT
shelf_column_id	BIGINT
shelf_floor_id	BIGINT
Indexes	
PRIMARY	
wms_location_shelf_column_id_shelf_fl_5b26b865_uniq	
wms_location_row_location_id_4619e432_fk_wms_rowlocation_id	
wms_location_shelf_floor_id_2dae17af_fk_wms_shelffloor_id	

Hình 4.10 Bảng Location.

4.4.8. Bảng sản phẩm tại vị trí – Item Location

Mô tả: lưu thông tin những sản phẩm được lưu kho ở các vị trí STORAGE.

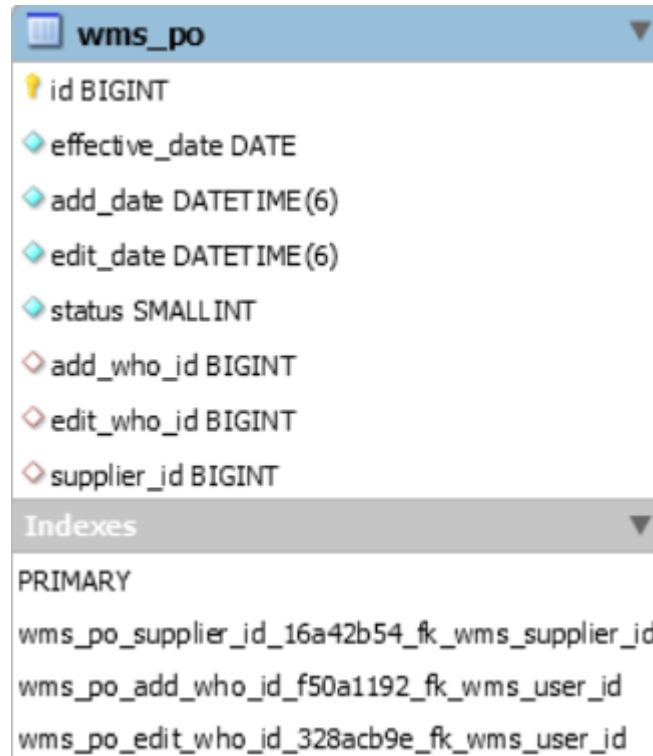


wms_itemlocation	
id	BIGINT
qty	INT
status	TINYINT(1)
item_id	BIGINT
location_id	BIGINT
Indexes	
PRIMARY	
wms_itemlocation_location_id_item_id_d373de8a_uniq	
wms_itemlocation_item_id_bb048a5e_fk_wms_item_id	
wms_itemlocation_location_id_6ff7f52e	

Hình 4.11 Bảng ItemLocation.

4.4.9. Bảng đơn nhập – PO

Mô tả: lưu thông tin tất cả đơn nhập hàng của nhà cung cấp. Mỗi đơn nhập hàng có trạng thái sau khi nhân viên xác nhận là ACCEPT thì trạng thái đơn đó sẽ luôn thay đổi ở mỗi tác vụ. Trạng thái cuối cùng là IMPORTED.

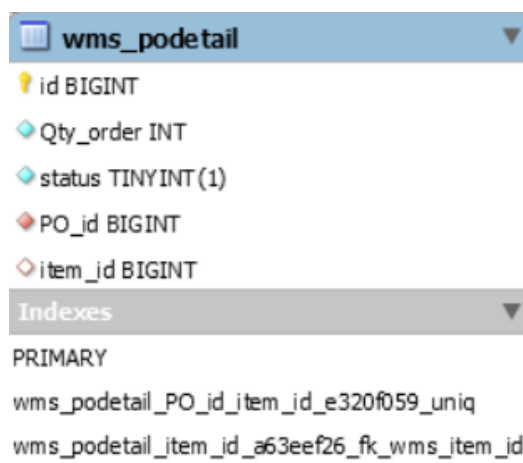


wms_po	
id	BIGINT
effective_date	DATE
add_date	DATETIME(6)
edit_date	DATETIME(6)
status	SMALLINT
add_who_id	BIGINT
edit_who_id	BIGINT
supplier_id	BIGINT
Indexes	
PRIMARY	
wms_po_supplier_id_16a42b54_fk_wms_supplier_id	
wms_po_add_who_id_f50a1192_fk_wms_user_id	
wms_po_edit_who_id_328acb9e_fk_wms_user_id	

Hình 4.12 Bảng PO.

4.4.10. Bảng chi tiết đơn nhập – PO Detail

Mô tả: lưu thông tin chi tiết đơn nhập hàng, bao gồm sản phẩm và số lượng yêu cầu nhập kho của sản phẩm đó.

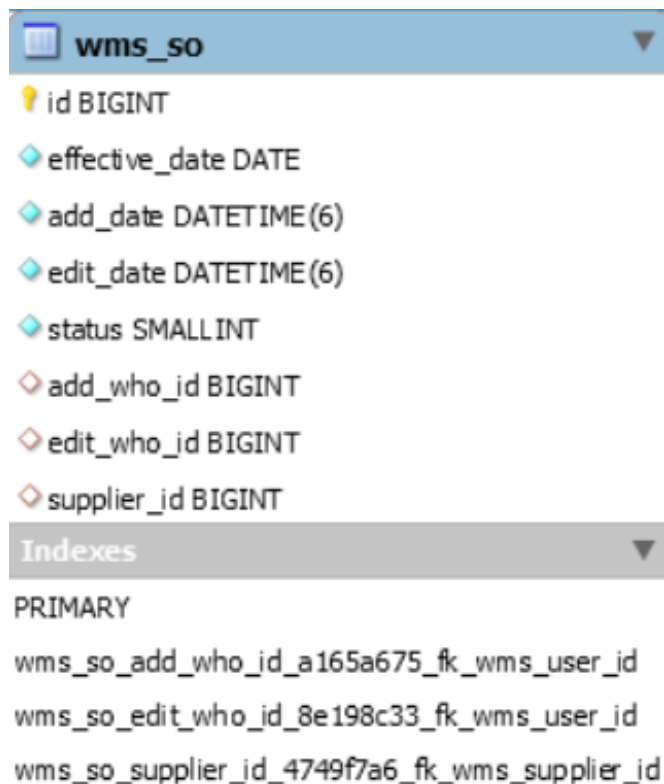


wms_podetail	
id	BIGINT
Qty_order	INT
status	TINYINT(1)
PO_id	BIGINT
item_id	BIGINT
Indexes	
PRIMARY	
wms_podetail_PO_id_item_id_e320f059_uniq	
wms_podetail_item_id_a63eef26_fk_wms_item_id	

Hình 4.13 Bảng PODetail.

4.4.11. Bảng đơn xuất – SO

Mô tả: lưu thông tin đơn xuất hàng của nhà cung cấp. Mỗi đơn xuất hàng có trạng thái sau khi nhân viên xác nhận là ACCEPT thì trạng thái đơn đó sẽ luôn thay đổi ở mỗi tác vụ. Trạng thái cuối cùng là DONE.

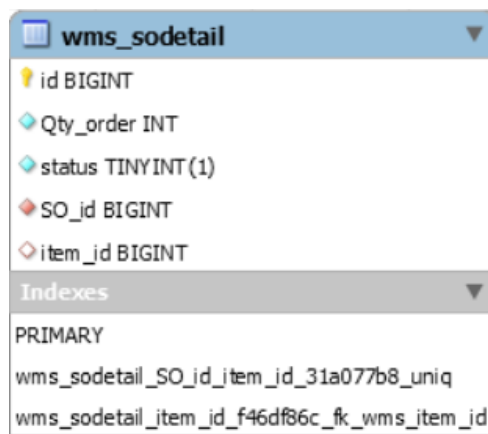


wms_so	
id	BIGINT
effective_date	DATE
add_date	DATETIME(6)
edit_date	DATETIME(6)
status	SMALLINT
add_who_id	BIGINT
edit_who_id	BIGINT
supplier_id	BIGINT
Indexes	
PRIMARY	
wms_so_add_who_id_a165a675_fk_wms_user_id	
wms_so_edit_who_id_8e198c33_fk_wms_user_id	
wms_so_supplier_id_4749f7a6_fk_wms_supplier_id	

Hình 4.14 Bảng SO.

4.4.12. Bảng chi tiết đơn xuất – SO Detail

Mô tả: lưu thông tin chi tiết đơn xuất hàng, bao gồm sản phẩm và số lượng yêu cầu xuất kho của sản phẩm đó.

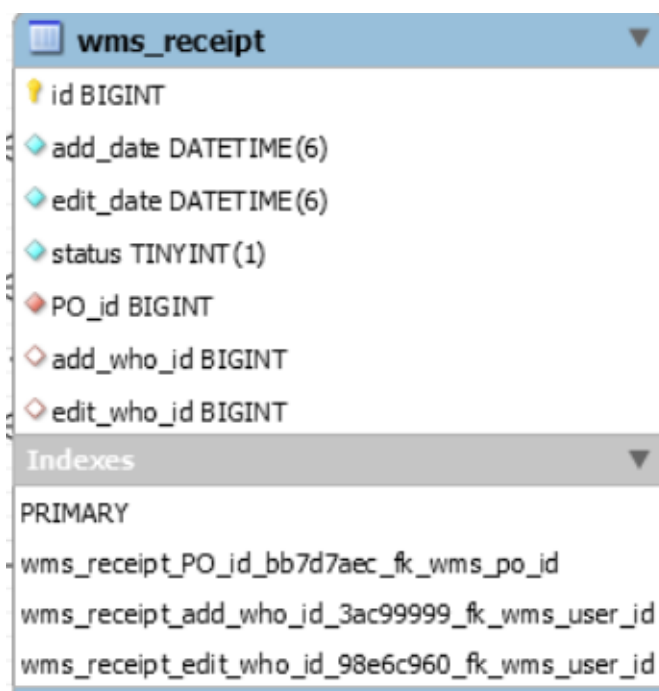


wms_sodetail	
id	BIGINT
Qty_order	INT
status	TINYINT(1)
SO_id	BIGINT
item_id	BIGINT
Indexes	
PRIMARY	
wms_sodetail_SO_id_item_id_31a077b8_uniq	
wms_sodetail_item_id_f46df86c_fk_wms_item_id	

Hình 4.15 Bảng SODetail.

4.4.13. Bảng biên lai nhập – Receipt

Mô tả: lưu thông tin biên lai nhập hàng mỗi lần nhận hàng từ đơn vị vận chuyển.

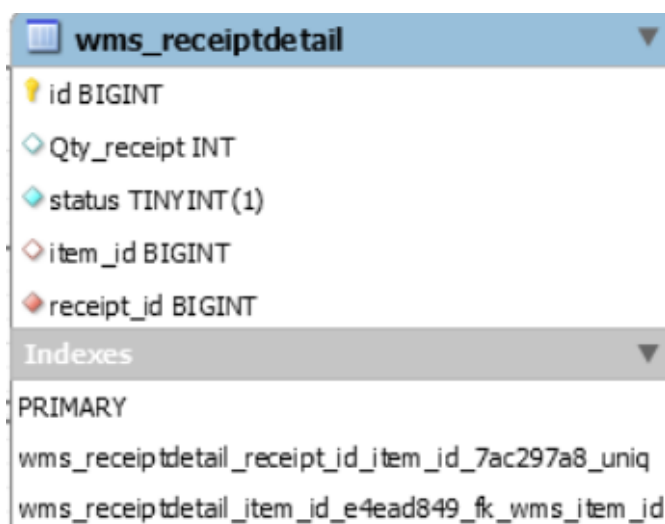


wms_receipt	
id	BIGINT
add_date	DATETIME(6)
edit_date	DATETIME(6)
status	TINYINT(1)
PO_id	BIGINT
add_who_id	BIGINT
edit_who_id	BIGINT
Indexes	
PRIMARY	
wms_receipt_PO_id_bb7d7aec_fk_wms_po_id	
wms_receipt_add_who_id_3ac99999_fk_wms_user_id	
wms_receipt_edit_who_id_98e6c960_fk_wms_user_id	

Hình 4.16 Bảng Receipt.

4.4.14. Bảng chi tiết biên lai nhập – Receipt Detail

Mô tả: lưu thông tin chi tiết biên lai nhập hàng, bao gồm sản phẩm và số lượng nhập của sản phẩm đó.

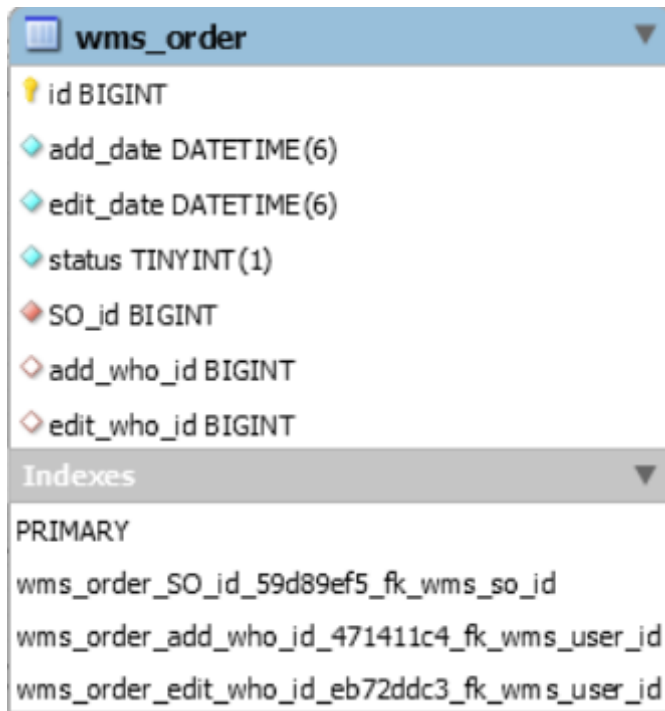


wms_receiptdetail	
id	BIGINT
Qty_receipt	INT
status	TINYINT(1)
item_id	BIGINT
receipt_id	BIGINT
Indexes	
PRIMARY	
wms_receiptdetail_receipt_id_item_id_7ac297a8_uniq	
wms_receiptdetail_item_id_e4ead849_fk_wms_item_id	

Hình 4.17 Bảng ReceiptDetail.

4.4.15. Bảng biên lai xuất – Order

Mô tả: lưu thông tin biên lai xuất hàng mỗi lần giao hàng cho đơn vị vận chuyển.

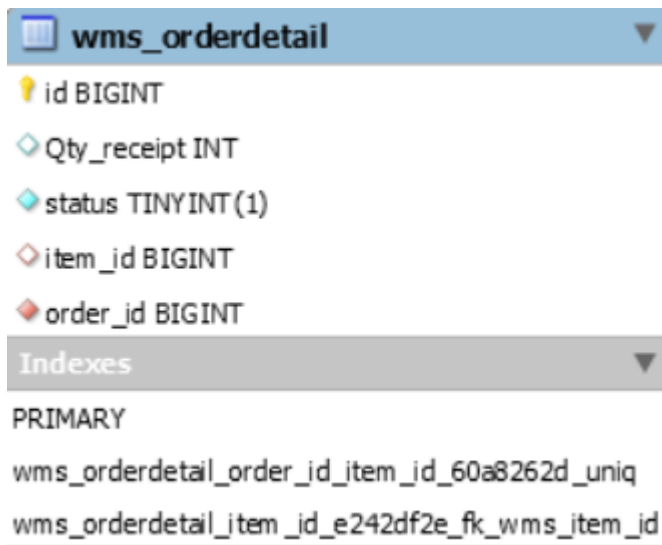


wms_order	
id	BIGINT
add_date	DATETIME(6)
edit_date	DATETIME(6)
status	TINYINT(1)
SO_id	BIGINT
add_who_id	BIGINT
edit_who_id	BIGINT
Indexes	
PRIMARY	
wms_order_SO_id_59d89ef5_fk_wms_so_id	
wms_order_add_who_id_471411c4_fk_wms_user_id	
wms_order_edit_who_id_eb72ddc3_fk_wms_user_id	

Hình 4.18 Bảng Order.

4.4.16. Bảng chi tiết biên lai xuất – Order Detail

Mô tả: lưu thông tin chi tiết biên lai xuất hàng, bao gồm sản phẩm và số lượng xuất của sản phẩm đó.

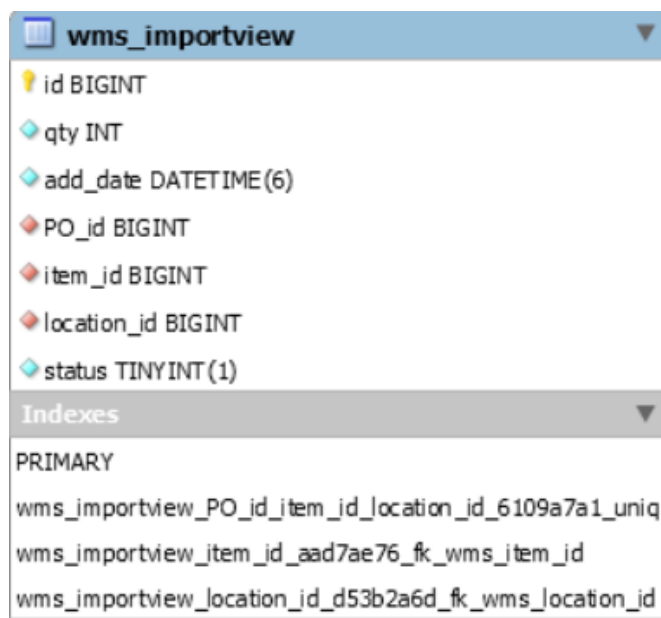


wms_orderdetail	
id	BIGINT
Qty_receipt	INT
status	TINYINT(1)
item_id	BIGINT
order_id	BIGINT
Indexes	
PRIMARY	
wms_orderdetail_order_id_item_id_60a8262d_uniq	
wms_orderdetail_item_id_e242df2e_fk_wms_item_id	

Hình 4.19 Bảng OrderDetail.

4.4.17. Bảng lưu kho tại vị trí – ImportView

Mô tả: lưu thông tin từng sản phẩm của đơn nhập được nhập vào kho với số lượng cùng vị trí nhập.

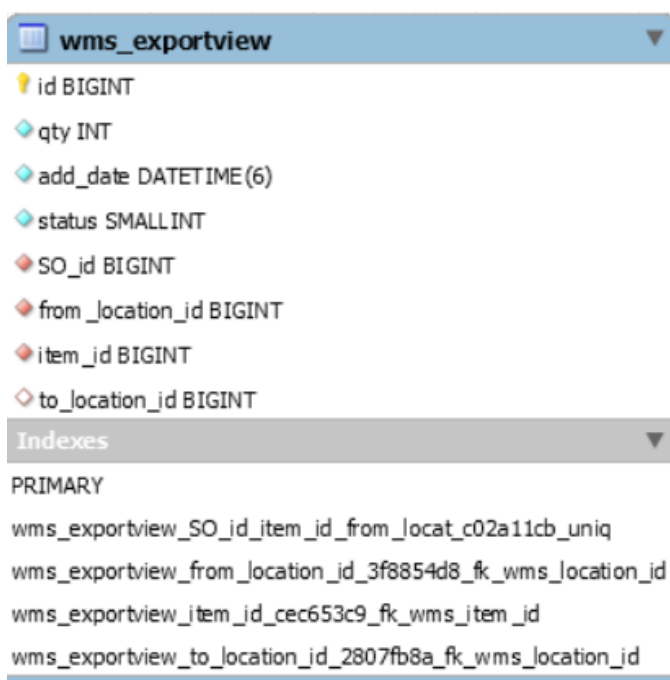


wms_importview	
id	BIGINT
qty	INT
add_date	DATETIME(6)
PO_id	BIGINT
item_id	BIGINT
location_id	BIGINT
status	TINYINT(1)
Indexes	
PRIMARY	
wms_importview_PO_id_item_id_location_id_6109a7a1_uniq	
wms_importview_item_id_aad7ae76_fk_wms_item_id	
wms_importview_location_id_d53b2a6d_fk_wms_location_id	

Hình 4.20 Bảng ImportView.

4.4.18. Bảng xuất kho tại vị trí – ExportView

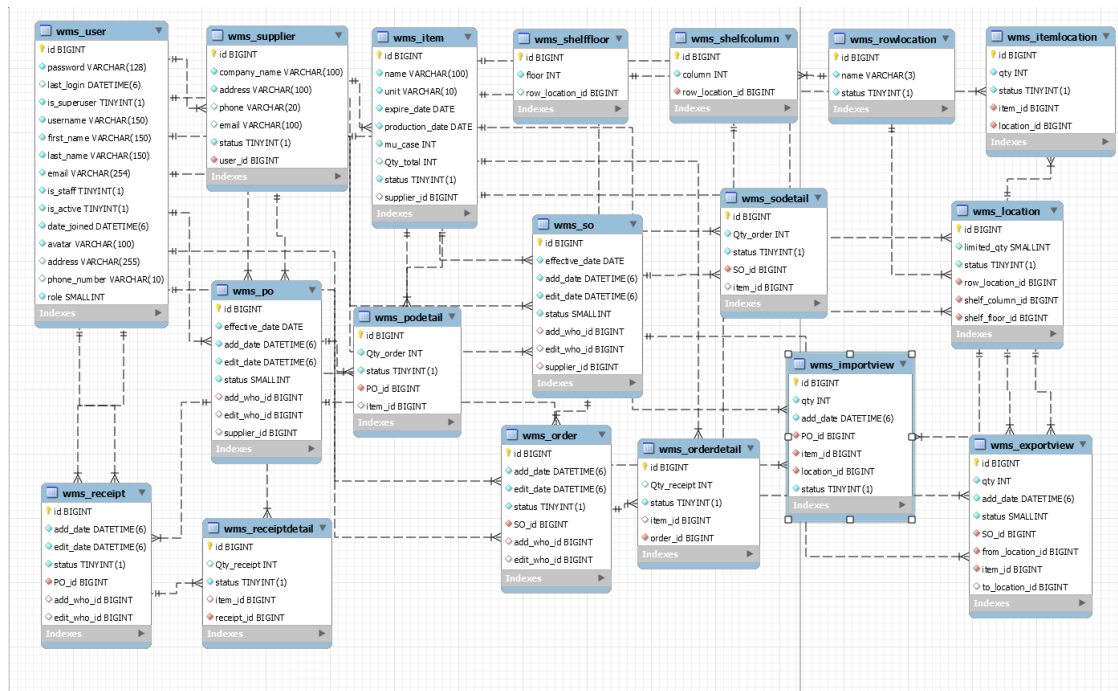
Mô tả: lưu thông tin xuất kho của từng sản phẩm của đơn xuất với số lượng và vị trí lấy hàng, rót hàng.



wms_exportview	
id	BIGINT
qty	INT
add_date	DATETIME(6)
status	SMALLINT
SO_id	BIGINT
from_location_id	BIGINT
item_id	BIGINT
to_location_id	BIGINT
Indexes	
PRIMARY	
wms_exportview_SO_id_item_id_from_locat_c02a11cb_uniq	
wms_exportview_from_location_id_3f8854d8_fk_wms_location_id	
wms_exportview_item_id_cec653c9_fk_wms_item_id	
wms_exportview_to_location_id_2807fb8a_fk_wms_location_id	

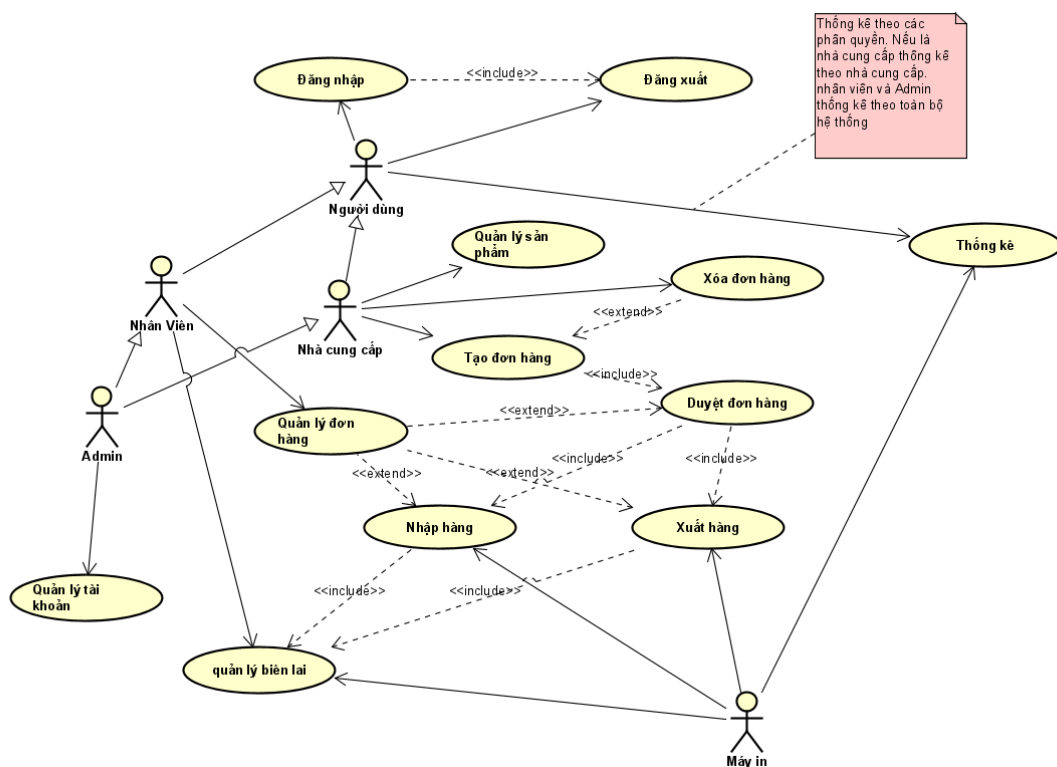
Hình 4.21 Bảng ExportView.

4.5. Lược đồ cơ sở dữ liệu



Hình 4.22 Lược đồ cơ sở dữ liệu của hệ thống.

4.6. Lược đồ usecase



Hình 4.23 Lược đồ use case của toàn hệ thống.

Mô tả use case có các actor là:

- Admin
- Nhà cung cấp
- Nhân viên
- Máy in

Mô tả chung

Mã use case	Tên use case
1	Đăng nhập
2	Đăng xuất
3	Quản lý tài khoản
4	Thống kê
5	Quản lý sản phẩm
6	Tạo đơn hàng
7	Duyệt đơn hàng
8	Nhập hàng
8	Xuất hàng
9	Quản lý biên lai

Bảng 4.1 Mô tả chi tiết use case của hệ thống.

4.6.1. Đặc tả use case “đăng nhập”

Mô tả	Use case cho phép người dùng đăng nhập vào hệ thống để thực hiện chức năng của mình.
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có tài khoản trên hệ thống

Hậu điều kiện	Nếu người dùng đăng nhập thành công thì được thực hiện các chức năng của hệ thống, ngược lại sẽ tiến hành xử lý lỗi đăng nhập.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Hệ thống hiển thị đăng nhập. 2. Người dùng nhập tên đăng nhập và mật khẩu. 3. Hệ thống kiểm tra tên đăng nhập và mật khẩu. 4. Nếu thành công thì hiển thị trang chủ, ngược lại hiển thị thông báo lỗi và quay lại bước 2. 5. Kết thúc use case.
Luồng ngoại lệ	<p>I - Mật khẩu không hợp lệ:</p> <p>Khi người dùng nhập sai tên đăng nhập và mật khẩu:</p> <ol style="list-style-type: none"> 1. Hệ thống báo lỗi, người dùng liên lạc với admin để lấy lại mật khẩu. 2. Quay lại bước 2 trong luồng sự kiện chính.

Bảng 4.2 Đặc tả use case đăng nhập

Đặc use Quản lý tài khoản

Mô tả	Admin quản lý tài khoản của nhân viên và nhà cung cấp gồm các chức năng thêm, sửa, xóa...
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin
Hậu điều kiện	Nếu use case thành công, thông tin của nhân viên hoặc nhà cung cấp được thêm, cập nhật hoặc xóa khỏi hệ thống. Ngược lại, trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn xem tùy chọn như thêm, sửa, xóa... 2. Hệ thống hiển thị theo các tùy chọn <ul style="list-style-type: none"> - Thêm – hiển một biểu mẫu gồm các trường là các thông tin của một người dùng trên hệ thống để người dùng nhập. - Sửa – hiển một biểu mẫu gồm các trường là các thông tin của người dùng cần sửa để người dùng nhập.

	<ul style="list-style-type: none"> - Xóa – hiển thị danh sách người dùng cần xóa khỏi hệ thống, người dùng chỉ cần nhấn nút xóa. <p>3. Hệ thống kiểm tra các thông tin của các hành động trên</p> <p>4. Nếu thành công thì hiện thông báo thành công.</p> <p>5. Kết thúc use case.</p>
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu người dùng nhập sai thông tin hoặc thiếu thông tin yêu cầu.

Bảng 4.3 Đặc tả use case quản lý tài khoản

4.6.2. Đặc tả use case “thống kê”

Mô tả	Thống kê các đơn hàng xuất và nhập của các tháng theo năm, thống kê tổng số đơn hàng nhập, đơn hàng xuất, các đơn hàng thành công, tổng số sản phẩm... Ở chức năng này thống kê theo quyền, nghĩa nếu là nhà cung cấp thì thống kê các thông tin trên theo nhà cung cấp còn là Admin và nhân viên thì thống kê theo toàn bộ hệ thống.
Actor chính	Người dùng
Actor phụ	Máy in
Tiền điều kiện	Người dùng phải đăng nhập vào hệ thống
Hậu điều kiện	Nếu Use case được thực hiện thành công sẽ hiển thị báo cáo cho người dùng. Nếu không trạng thái hệ thống vẫn giữ nguyên không đổi.
Luồng hoạt động	<p>1. Người dùng chọn xem thống kê.</p> <p>2. Hệ thống sẽ yêu cầu người dùng chọn thống kê theo năm.</p> <p>3. Hệ thống hiển thị thông tin thống kê báo cáo theo tháng và năm mà người dùng chọn.</p> <p>4. Nếu người dùng gửi yêu cầu in báo cáo thống kê, máy in tiến hành in bảng báo cáo thống kê.</p> <p>5. Kết thúc use case</p>

Bảng 4.4 Đặc tả use case thống kê.

4.6.3. Đặc tả use case “quản lý sản phẩm”

Mô tả	Nhà cung cấp và nhân viên được quyền quản lý sản phẩm bao gồm các chức năng thêm, sửa hoặc xóa sản phẩm.
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhà cung cấp.
Hậu điều kiện	Nếu use case thành công thì thông tin của sản phẩm của cung cấp được thêm, cập nhật hoặc xóa khỏi hệ thống. Ngược lại, trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn xem tùy chọn như thêm, sửa, xóa... 2. Hệ thống hiển thị theo các tùy chọn <ul style="list-style-type: none"> - Thêm – hiển một biểu mẫu gồm các trường là các thông tin của sản phẩm để người dùng nhập. - Sửa – hiển một biểu mẫu gồm các trường là các thông tin của sản phẩm cần sửa để người dùng nhập. - Xóa – hiển thị thông tin sản phẩm cần xóa khỏi hệ thống, người dùng chỉ cần nhấn nút xóa. 3. Hệ thống kiểm tra các thông tin của các hành động trên và thực hiện. 4. Nếu thành công thì hiện thông báo thành công. 5. Kết thúc use case
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu người dùng nhập sai thông tin sản phẩm ví dụ này sản xuất lớn hơn ngày hiện tại hay ngày sử dụng hết hạn...

Bảng 4.5 Đặc tả use case quản lý sản phẩm

4.6.4. Đặc tả use case tạo đơn hàng.

Mô tả	Admin và nhà cung cấp được tạo đơn hàng nhập hoặc xuất.
Actor chính	Người dùng
Actor phụ	Không có

Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhà cung cấp.
Hậu điều kiện	Nếu use case thành công thì thông tin của đơn hàng nhập hoặc xuất của cung cấp được thêm vào hệ thống. Ngược lại, trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn thêm đơn nhập hàng. 2. Hệ thống hiển thị biểu mẫu để người dùng nhập các thông tin của đơn hàng. Gồm các thông tin như ngày nhập hoặc xuất kho, danh sách sản phẩm cho đơn hàng và số lượng. 3. Hệ thống kiểm tra các thông tin của đơn hàng. 4. Nếu thành công thì hiện thông báo thành công. 5. Kết thúc use case
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu người dùng nhập sai thông tin đơn hàng.

Bảng 4.6 Đặc tả use case tạo đơn hàng.

4.6.5. Đặc tả use case “xóa đơn hàng”

Mô tả	Admin và nhà cung cấp được xóa đơn hàng nhập hoặc xuất. Nếu đơn hàng này bị lỗi, không được chấp nhận...
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhà cung cấp.
Hậu điều kiện	Nếu use case thành công thì thông tin của đơn hàng nhập hoặc xuất của cung cấp được xóa ra khỏi hệ thống. Ngược lại, trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn đơn nhập hàng cần xóa. 2. Hệ thống hiển thị nút bấm để người dùng xóa đơn hàng. 3. Hệ thống kiểm tra đơn hàng cần xóa. 4. Nếu thành công thì hiện thông báo thành công. 5. Kết thúc use case.
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu người dùng chọn xóa đơn hàng không được phép xóa.

Bảng 4.7 Đặc tả use case xóa đơn hàng

4.6.6. Đặc tả use case “duyet đơn hàng”.

Mô tả	Admin và nhân được duyệt đơn hàng nhập hoặc xuất nếu đơn hàng đáp ứng các điều kiện.
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhân viên.
Hậu điều kiện	Nếu use case thành công thì trạng thái đơn hàng được cập nhật, đơn hàng đó được chuyển sang các bước tiếp theo. Ngược lại, đơn hàng không được duyệt, đơn hàng đó sẽ được xóa tự động.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn đơn hàng cần duyệt. 2. Hệ thống hiển thị giao diện cập nhật đơn hàng. 3. Hệ thống kiểm tra thông tin đơn hàng. 4. Nếu thành công thì hiện thông báo thành công. 5. Kết thúc use case.
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu có lỗi xảy ra.

4.6.7. Đặc tả use case nhập hàng.

Mô tả	Chức năng này cho Admin hoặc nhân viên nhập hàng vô kho được xem thông tin chi tiết một đơn hàng, phân bổ vị trí đơn hàng vào kho. Trong các chức năng này được phép dùng máy in nếu có nhu cầu.
Actor chính	Người dùng
Actor phụ	Máy In
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhân viên.
Hậu điều kiện	Nếu use case thành công thì đơn hàng được nhập vào kho . Ngược lại trạng thái của hệ thống không thay đổi.

Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng xem thông tin đơn hàng hoặc, nhập hàng phân vị trí hàng vào kho. 2. Hệ thống hiển thị giao diện theo người dùng chọn. <ul style="list-style-type: none"> - Xem thông tin đơn hàng – hiển thị thông tin chi tiết của một đơn hàng. - Nhập hàng – hiển thị một biểu mẫu xác nhận người dùng yêu cầu nhập đơn hàng vào kho. 3. Hệ thống kiểm tra thực hiện theo yêu cầu. 4. Nếu thành công thì hiện thông báo thành công, vị trí của sản phẩm trong kho. 5. Kết thúc use case.
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu có lỗi xảy ra.

Bảng 4.8 Đặc tả use case nhập hàng

4.6.8. Đặc tả use case “xuất hàng”.

Mô tả	Chức năng này cho Admin hoặc nhân viên được xuất hàng ra khỏi kho, xem thông tin chi tiết một đơn hàng, rút hàng, lấy hàng và sắp xếp hàng. Trong các chức năng này được phép sử dụng máy in nếu có nhu cầu.
Actor chính	Người dùng
Actor phụ	Máy In
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhân viên.
Hậu điều kiện	Nếu use case thành công thì đơn hàng được nhập vào kho . Ngược lại trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<ol style="list-style-type: none"> 1. Người dùng chọn chức năng xem thông tin đơn hàng hoặc, rút hàng, lấy hàng và sắp xếp hàng. 2. Hệ thống hiển thị giao diện theo người dùng chọn. <ul style="list-style-type: none"> - Xem thông tin đơn hàng – hiển thị thông tin chi tiết của một đơn hàng. - Rút hàng – hiển thị một biểu mẫu xác nhận người dùng yêu cầu nhập đơn hàng vào kho. - Lấy hàng – hiển thị danh sách sản phẩm được lấy khi người dùng nhấn nút bấm.

	<ul style="list-style-type: none"> - Sắp xếp hàng – hiển thị danh sách sản phẩm đã được sắp xếp khi người dùng bấm nút. <p>3. Hệ thống kiểm tra thực hiện theo yêu cầu.</p> <p>4. Nếu thành công thì hiện thông báo thành công kèm theo thông tin yêu cầu.</p> <p>5. Kết thúc use case.</p>
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu có lỗi xảy ra.

Bảng 4.9 Đặc tả use case xuất hàng.

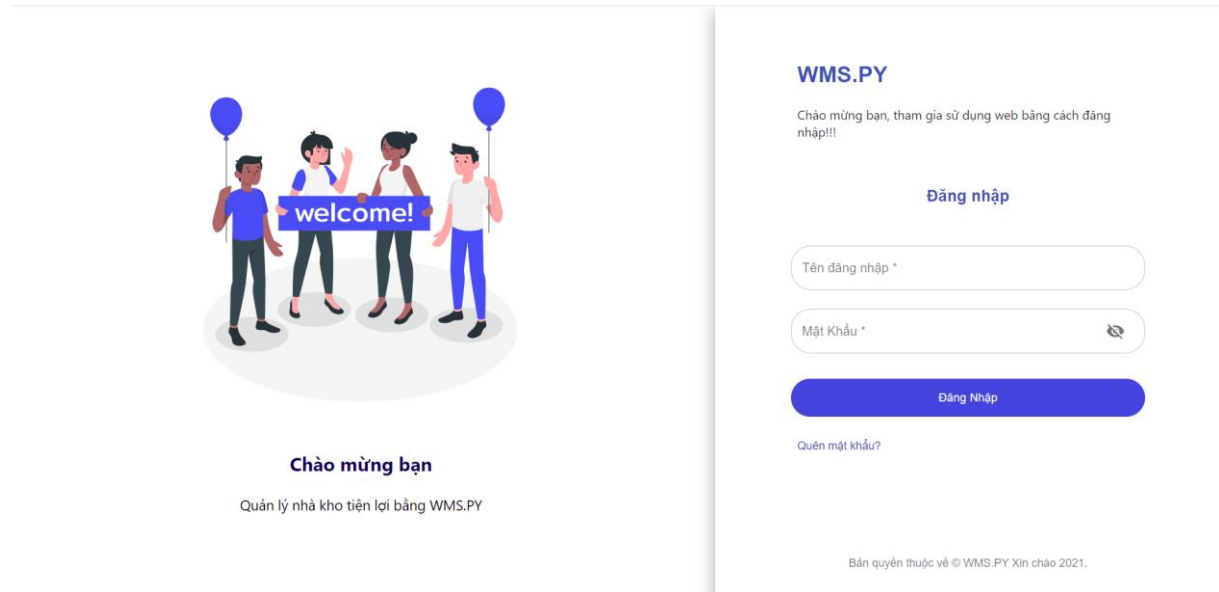
4.6.9. Đặc tả use case “quản lý biên lai”

Mô tả	Quản lý biên lai bao gồm thêm, sửa, xóa biên lai nhập hoặc xuất, chỉ có admin và nhân viên được thực hiện các chức năng này.
Actor chính	Người dùng
Actor phụ	Không có
Tiền điều kiện	Người dùng phải có trên hệ thống và có quyền là Admin hoặc nhà cung cấp.
Hậu điều kiện	Nếu use case thành công thì thông tin của biên lai của đơn hàng được thêm, cập nhật hoặc xóa khỏi hệ thống. Ngược lại, trạng thái của hệ thống không thay đổi.
Luồng hoạt động	<p>1. Người dùng chọn xem tùy chọn như thêm, sửa xóa...</p> <p>2. Hệ thống hiển thị theo các tùy chọn</p> <ul style="list-style-type: none"> - Thêm – hiển một biểu mẫu gồm các trường là các thông tin của một biên lai để người dùng nhập. - Sửa – hiển một biểu mẫu để cập nhật trạng thái của biên lai khi sửa đổi hoặc cập nhật. - Xóa – hiển thị thông tin biên lai cần xóa khỏi hệ thống, người dùng chỉ cần nhấn nút xóa. <p>3. Hệ thống kiểm tra các thông tin của các hành động trên và thực hiện.</p> <p>4. Nếu thành công thì hiện thông báo thành công.</p> <p>5. Kết thúc use case.</p>
Luồng ngoại lệ	ở bước 3 hệ thống báo lỗi nếu người dùng nhập sai thông tin.

4.7. Xây dựng giao diện hệ thống

4.7.1. Thiết kế giao diện

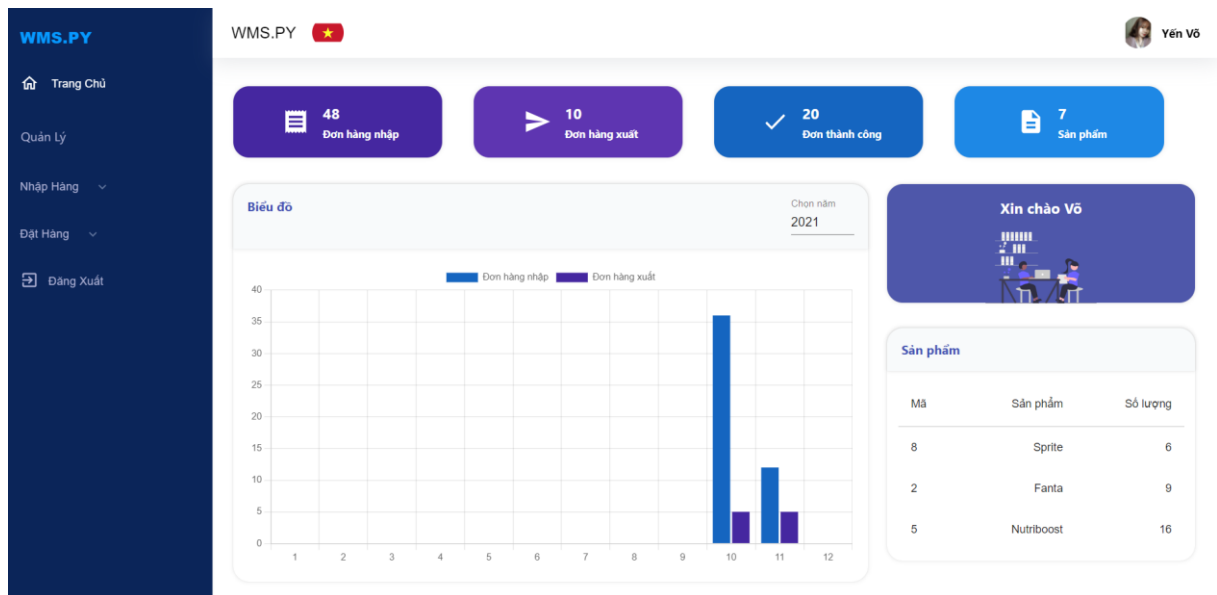
Khi người dùng truy cập vào một trang website WMSPY hệ thống hiện thị giao diện để người dùng đăng nhập. Trong giao diện đăng nhập ngoài biểu mẫu để đăng nhập còn có hiển thị vài thông tin giới thiệu sơ qua về WMSPY.



Hình 4.24 Giao diện đăng nhập.

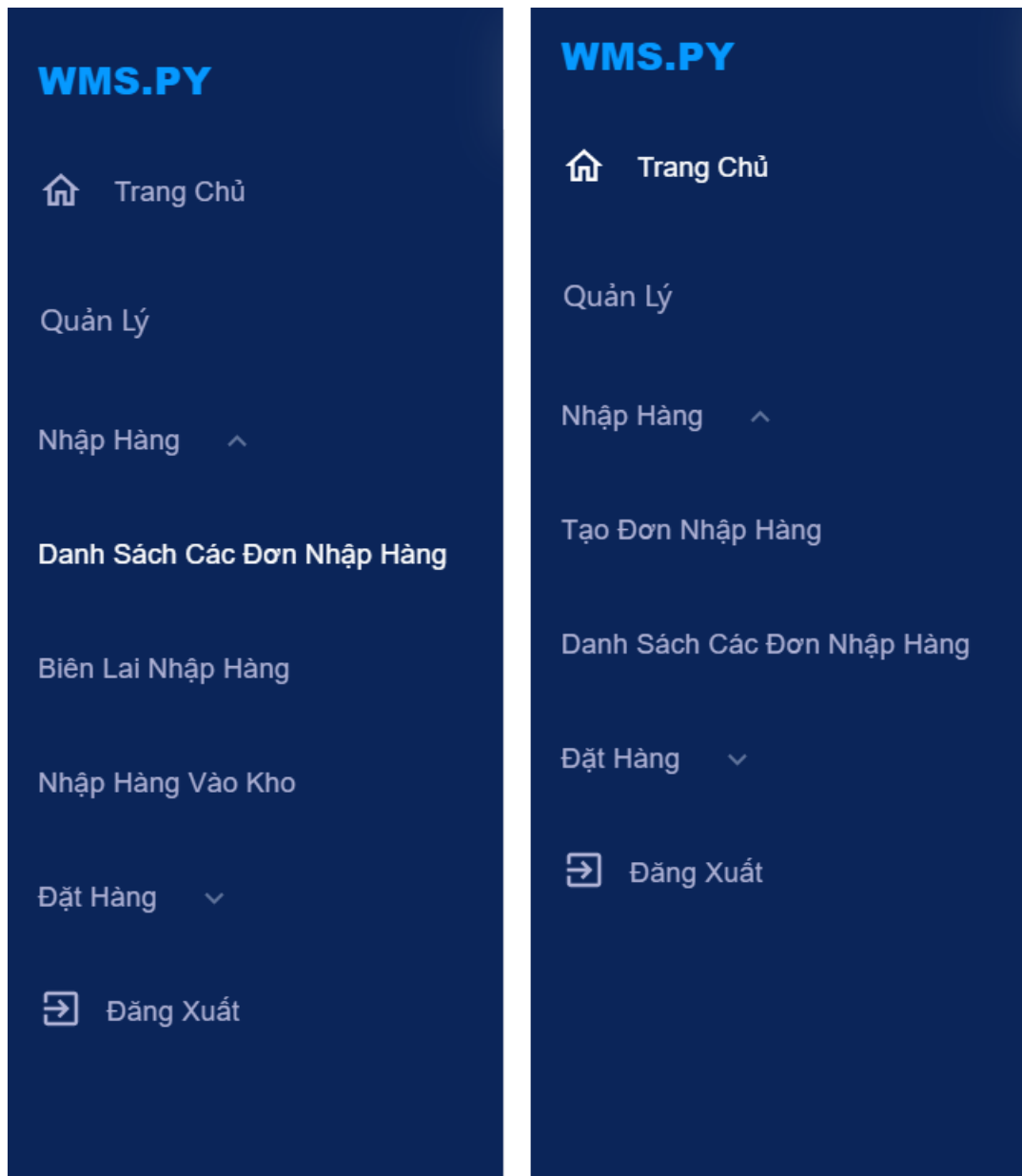
Sau khi người dùng nhập đúng thông tin tài khoản hệ thống sẽ hiện thị màn hình làm việc chính cho người dùng. Trong hệ thống có phân quyền cho nhà cung cấp, nhân viên, khi người dùng đăng nhập với tài khoản quyền nào hệ thống sẽ chuyển sang trang chủ với quyền như vậy.

Giao diện trang chủ gồm các thông tin thống kê về tổng đơn hàng nhập hoặc xuất, tổng số đơn hàng thành công, tổng sản phẩm, tổng số đơn hàng nhập hoặc xuất của tháng theo năm người dùng chọn mặc định là năm hiện tại. Ngoài ra hệ thống cho phép người dùng chuyển ngôn ngữ để thuận tiện trong công việc gồm tiếng việt và tiếng anh.



Hình 4.25 Giao diện trang chủ.

Người dùng có thể tương tác được với các giao diện khác trong hệ thống qua thanh menu. Để di chuyển tới một giao diện khác người dùng chỉ cần chọn chỉ mục menu tương ứng của giao diện đó. Menu trang website đang ở giao diện nào thì chỉ mục của màn hình đó sẽ có màu khác với các chỉ mục còn lại. Người dùng là nhân viên sẽ có menu khác so với nhà cung cấp vì sẽ có những chức năng phù hợp với vai trò mới dùng được.



Hình 4.26 Giao diện menu.

Giao diện tạo yêu cầu nhập hàng vào kho cho nhà cung cấp khi có yêu cầu. Ở giao diện này nhà cung cấp được tạo thêm sản phẩm mới hoặc tạo yêu cầu nhập kho, chỉ với các thao tác đơn giản chọn và tìm sản phẩm cần nhập sau đó nhập số lượng mong muốn. Chúng em có ràng buộc các điều kiện và hiện thì thông báo thành công hoặc lỗi nếu xảy ra mang lại trải nghiệm tốt nhất cho người dùng.

+

Thêm Sản Phẩm Mới

+

Tạo Yêu Cầu Nhập Hàng

Tạo yêu cầu nhập hàng

Nhà cung cấp:

+

Tra đao

Thêm sản phẩm

+

Ngày tạo:

28/10/2021

Ngày nhập:

29/10/2021

Danh sách sản phẩm

1	Sản phẩm *	kẹo 4	Ngày sản xuất	2021-09-03	Ngày hết hạn	2021-09-30	Số lượng	3	×
2	Sản phẩm *	Bánh Quy	Ngày sản xuất	2021-09-01	Ngày hết hạn	2021-09-21	Số lượng	3	×
3	Sản phẩm *	kẹo	Ngày sản xuất	2021-09-03	Ngày hết hạn	2021-09-25	Số lượng	6	×
4	Sản phẩm *	bánh da lợn	Ngày sản xuất	2021-09-22	Ngày hết hạn	2021-09-30	Số lượng		×

vui lòng nhập đồng này

×

Hủy

➤

Gửi Yêu Cầu

Hình 4.27 Giao diện tạo yêu cầu nhập kho.

Giao diện xem danh sách các đơn hàng của nhân viên, người dùng có thể xem toàn bộ danh sách xuất file excel về máy, xem chi tiết một đơn hàng, cập nhật trạng thái của đơn hàng. Chúng em sử dụng thư viện Material Ui hỗ trợ nên phần bảng danh sách đơn hàng này rất thuận tiện, có thể tìm kiếm tùy theo các trường kéo thả các cột theo ý thích.

WMS.PY

🇻🇳

Yến Võ

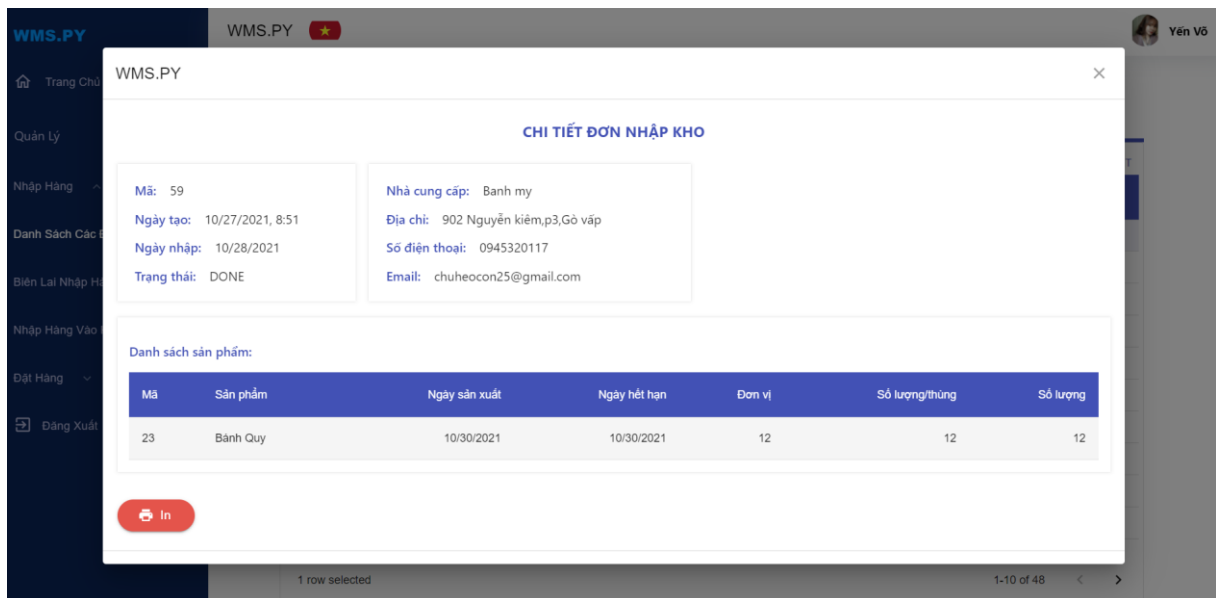
DANH SÁCH CÁC ĐƠN NHẬP HÀNG

Mã	Ngày tạo	Ngày nhập	Nhân viên xác nhận	Nhân viên sửa	Trạng thái	Hành động
59	10/27/2021, 8:51	10/28/2021	nhanvien	nhanvien	✓ DONE	👁️ ✎️ 🗑️
58	10/27/2021, 8:51	10/29/2021	nhanvien	nhanvien	✓ DONE	👁️ ✎️ 🗑️
57	10/25/2021, 12:...	10/27/2021	nhanvien	nhanvien	ⓘ ACCEPTED	👁️ ✎️ 🗑️
56	10/25/2021, 12:...	10/28/2021	nhanvien	nhanvien	✓ IMPORTED	👁️ ✎️ 🗑️
55	10/24/2021, 10:...	10/28/2021	nhanvien	nhanvien	ⓘ ACCEPTED	👁️ ✎️ 🗑️
54	10/24/2021, 10:...	10/28/2021	nhanvien	nhanvien	✓ IMPORTED	👁️ ✎️ 🗑️
53	10/24/2021, 10:...	10/29/2021	nhanvien	nhanvien	✓ IMPORTED	👁️ ✎️ 🗑️
52	10/18/2021, 10:...	10/29/2021	nhanvien	nhanvien	✓ IMPORTED	👁️ ✎️ 🗑️
51	10/09/2021, 1:01	10/30/2021	nhanvien	nhanvien	✓ DONE	👁️ ✎️ 🗑️
50	10/09/2021, 12:...	10/30/2021	nhanvien	nhanvien	✓ DONE	👁️ ✎️ 🗑️

1-10 of 48

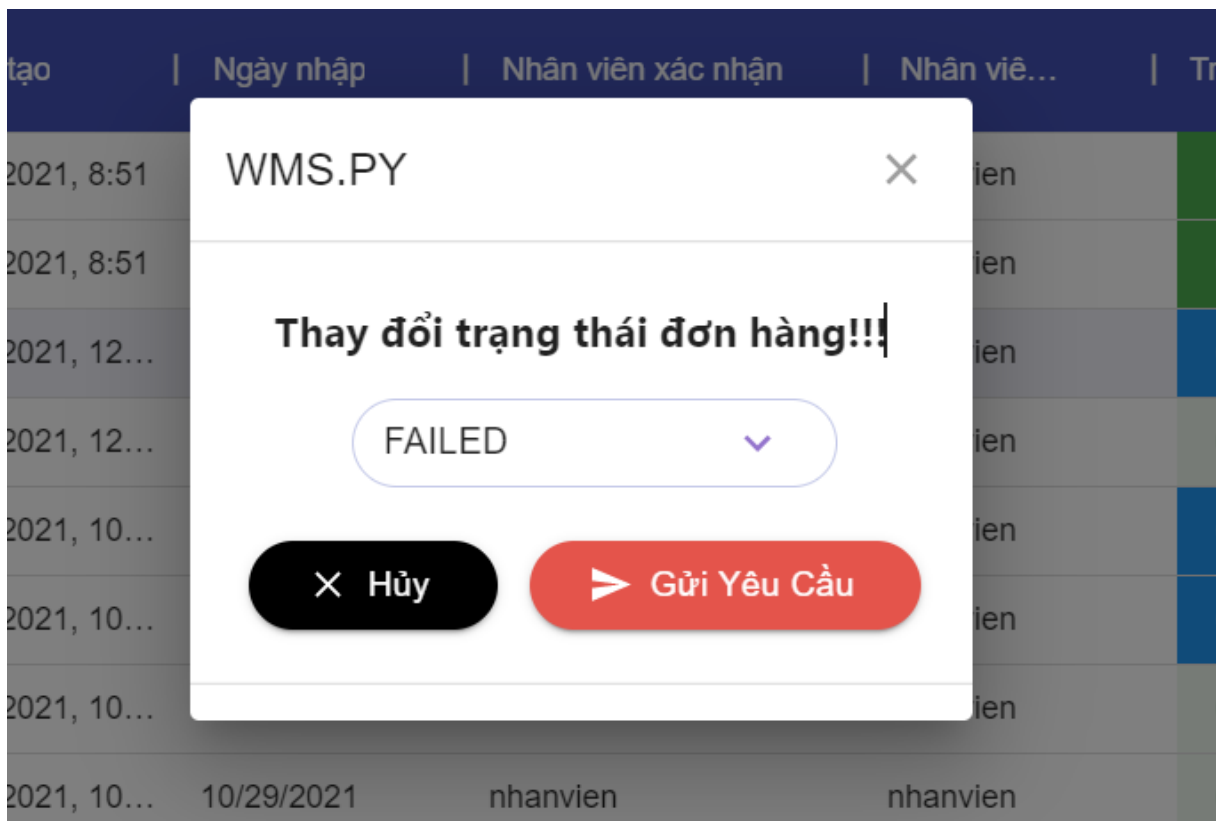
Hình 4.28 Giao diện xem danh sách đơn hàng nhập kho

Khi người dùng chọn hành động xem thông tin chi tiết của một đơn hàng bằng bấm nút bấm có hình icon con mắt. Hệ thống hiển thị modal có thông tin chi tiết của một đơn hàng. Người dùng được phép in đơn hàng nếu có nhu cầu.



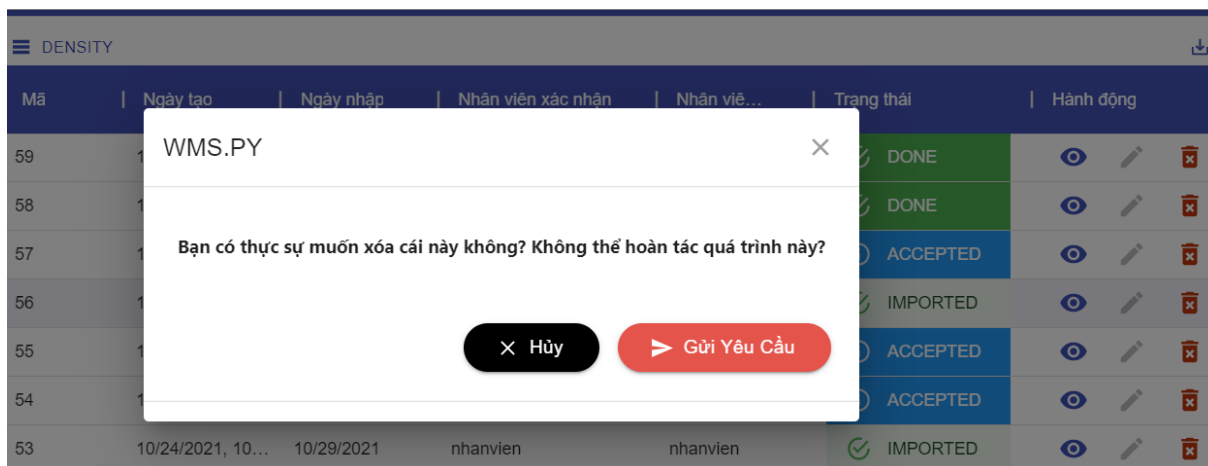
Hình 4.29 Giao diện xem chi tiết đơn hàng nhập kho

Khi người dùng muốn cập nhật đơn hàng chấp nhận đơn hàng hoặc không. Người dùng chỉ cần chọn đơn hàng và bấm nút cập nhật sau đó chọn trạng thái mà mình mong muốn. Hệ thống hiển thị thông báo thành công hoặc lỗi nếu có lỗi xảy ra.



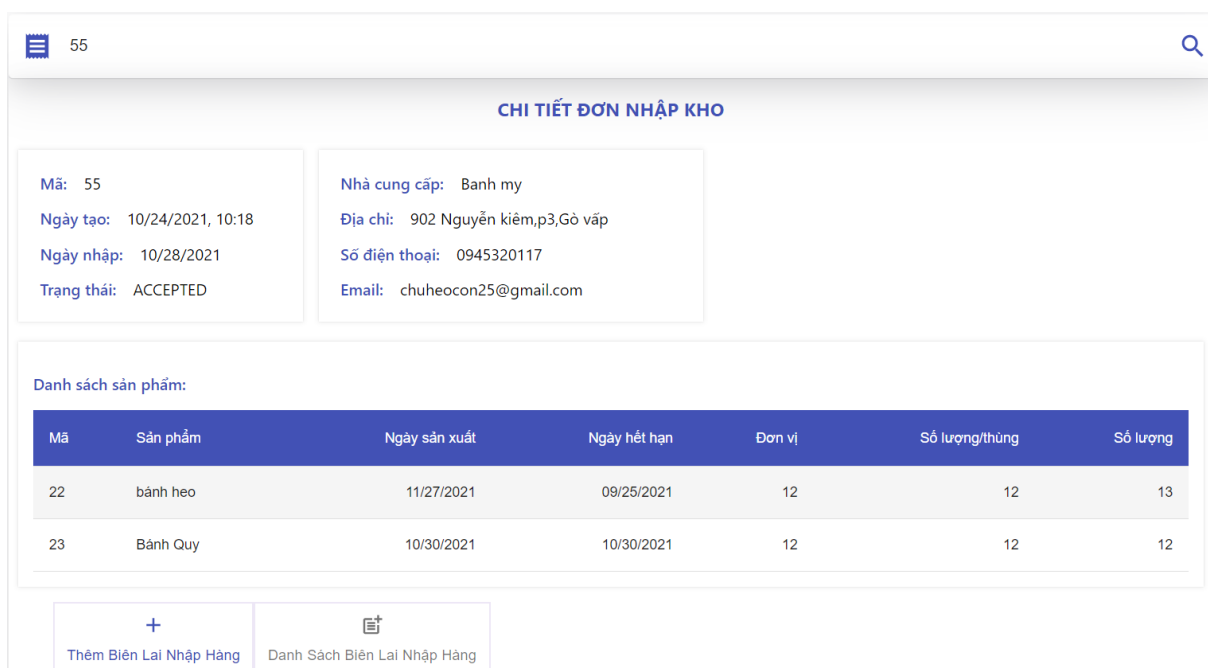
Hình 4.30 Giao diện thay đổi trạng thái đơn hàng.

Tương tự như cập nhật trạng thái đơn hàng, xóa đơn hàng cũng nhập đơn hàng hệ thống hiển thị một modal xác nhận lại xóa đơn hàng.



Hình 4.31 Giao diện xác nhận xóa đơn hàng.

Khi người dùng có yêu cầu tạo biên lai cho đơn nhập hoặc đơn xuất chỉ cần nhập mã đơn hàng muốn tạo biên lai và tiến hành tạo các biên lai. Ở giao diện này ngoài hiện thông tin của một đơn hàng cho người dùng để theo dõi còn hiển thị hai tab tạo và danh sách các biên lai mang lại sự thuận tiện trong quá trình làm việc.



Hình 4.32 Giao diện thêm biên lai của đơn nhập

Giao diện tạo biên lai là một biểu mẫu hiển thị các thông tin cần thiết, người dùng chỉ cần chọn sản phẩm muốn thêm vào biên lai và nhập số lượng.

+

Thêm Biên Lai Nhập Hàng

📋

Danh Sách Biên Lai Nhập Hàng

Thêm sản phẩm

+

Danh sách sản phẩm

1

Sản phẩm *

Bánh Quy

Số lượng đặt hàng

12

Số lượng nhập kho

12

Số lượng

3

×

2

Sản phẩm *

bánh heo

Số lượng đặt hàng

13

Số lượng nhập kho

0

Số lượng

×

× Hủy

Gửi Yêu Cầu

Hình 4.33 Giao diện biểu mẫu tạo biên lai

Giao diện danh sách biên lai hiển thị các biên lai cho xem thông tin chi tiết của một biên lai, chỉnh sửa biên lai, xóa biên lai.

+

Thêm Biên Lai Nhập Hàng

📋

Danh Sách Biên Lai Nhập Hàng

Search...

Mã	Ngày tạo	Ngày chín...	Nhân viên xác nhận	Nhân viên sửa	Mã đơn nhập	Hành động
49	7 days ago	10/24/2021	nhanvien	nhanvien	55	👁️ ✎️ 🗑️

1-1 of 1 < >

Hình 4.34 Giao diện danh sách các biên lai được tạo.

WMS.PY

Trang Chủ

Quản Lý

Nhập Hàng

Danh Sách Các Đơn Nhập Hàng

Biên Lai Nhập Hàng

Nhập Hàng Vào Kho

Đặt Hàng

Danh Sách Các Đơn Xuất Hàng

Biên Lai Xuất Hàng

Xuất Hàng Ra Kho

Đăng Xuất

WMS.PY

Ngày nhập: 10/26/2021

Số điện thoại: 0943320117

Trạng thái: ACCEPTED

Email: chuibaon25@gmail.com

Yên Võ

CHI TIẾT BIÊN LAI

Mã: 49

Mã đơn nhập: 55

Ngày tạo: 10/24/2021, 10:20

Ngày chỉnh sửa: 10/24/2021, 10:20

Danh sách sản phẩm

Mã

Sản phẩm

Số lượng đặt hàng

Số lượng

23

Bánh Quy

12

7

× Hủy

Gửi Yêu Cầu

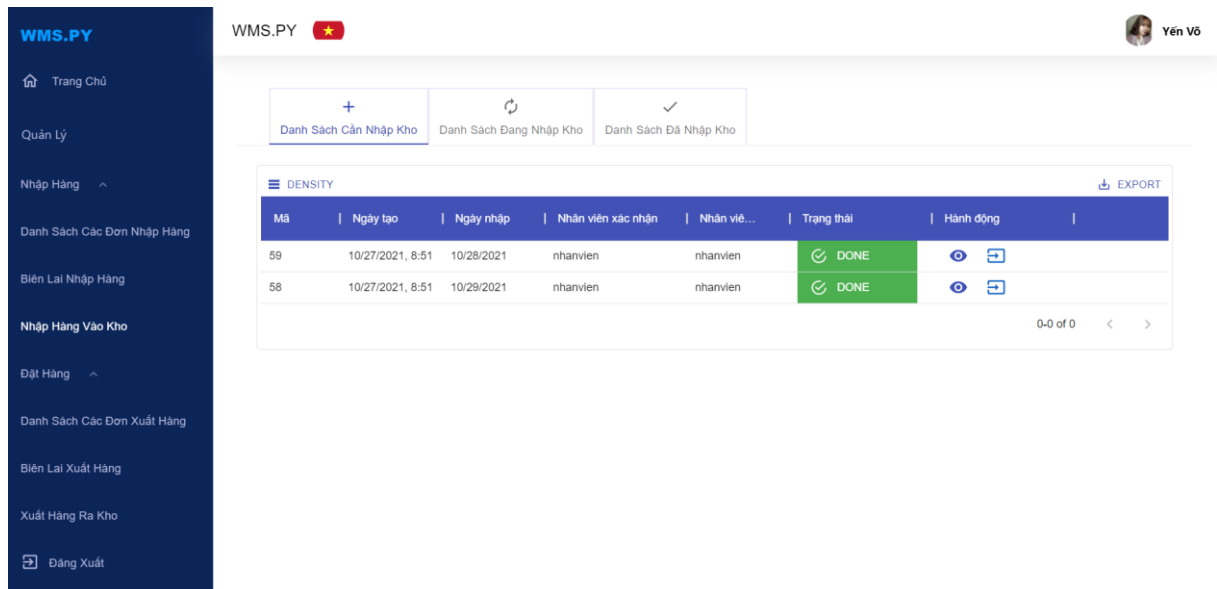
1 row selected

1-1 of 1

Thành công

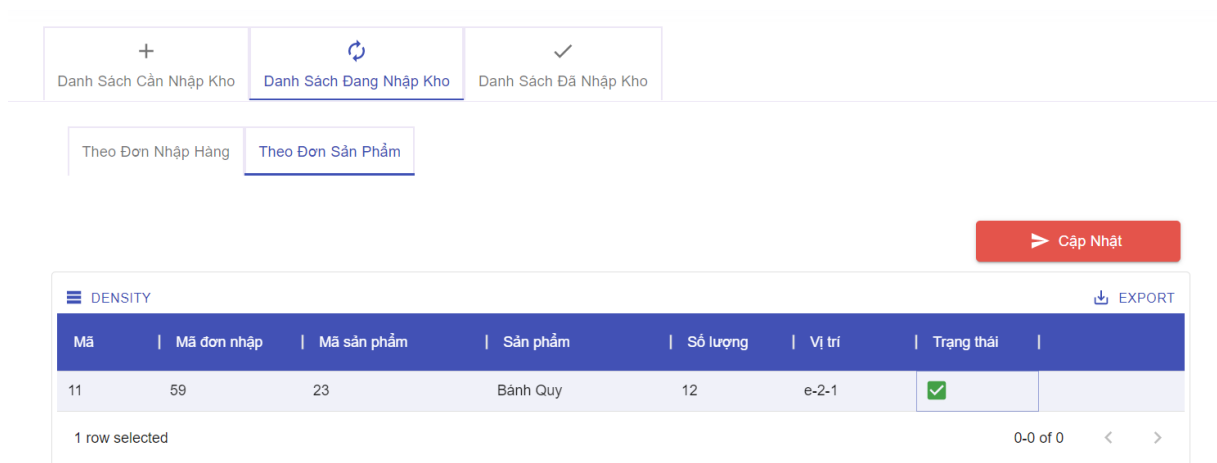
Hình 4.34 Giao diện chỉnh sửa một biên lai.

Nhập hàng vô kho gồm các tab để người dùng thực hiện các thao tác, các tab là cái giai đoạn nhập hàng trong các tab là bảng hiện thị thông tin và có thể xuất file excel. Đầu tiên tab danh sách cần nhập là người dùng sẽ chọn những đơn hàng nhập cần nhập vào kho chọn hành động nhập kho để yêu cầu nhập đơn hàng này vào kho.



Hình 4.35 Giao diện nhập kho.

Tab tiếp theo là danh sách các đơn hàng đang được nhập kho, khi nhân viên sắp xếp theo đúng thứ tự hệ thống yêu cầu, ở tab này nhân viên sẽ cập nhập những sản phẩm mà nhân viên đã nhập vào kho. Sản phẩm được phân theo đơn hàng hoặc toàn bộ sản phẩm đang được đưa vào kho.



Hình 4.36 Giao diện cập nhập sản phẩm đang nhập kho.

Tab cuối cùng là tab hiển thị danh sách các đơn hàng đã nhập kho thành công cho người dụng tiện thể kiểm tra lại.

+	↺	✓
Danh Sách Cần Nhập Kho	Danh Sách Đang Nhập Kho	Danh Sách Đã Nhập Kho

DENSITY							EXPORT
Mã	Ngày tạo	Ngày nhập	Nhân viên xác nhận	Nhân viên...	Trạng thái	Hành động	
56	10/25/2021, 12...	10/28/2021	nhanvien	nhanvien	IMPORTED		
53	10/24/2021, 10...	10/29/2021	nhanvien	nhanvien	IMPORTED		
52	10/18/2021, 10...	10/29/2021	nhanvien	nhanvien	IMPORTED		
1 row selected							
0-0 of 0							

Hình 4.37 Giao diện hiển thị danh sách đã nhập kho.

WMS.PY

CHI TIẾT ĐƠN NHẬP KHO

Mã đơn nhập: 56

Danh sách sản phẩm nhập vào kho:

Mã	Sản phẩm	Ngày sản xuất	Ngày hết hạn	Đơn vị	Số lượng/thùng	Số lượng	Vị trí	Đã nhập vào kho
21	bánh vệt	11/20/2021	09/03/2021	12	12	12	E-2-1	

In

Hình 4.38 Giao diện của một đơn hàng sau khi nhập kho.

Giao diện xuất hàng giống với giao diện nhập hàng nhưng ở tab danh sách cần xuất kho sẽ trải qua nhiều giai đoạn để xuất kho hơn cần người người cập nhập nhiều giai đoạn.

</

Hình 4.39 Giao diện cập nhật sản phẩm xuất kho.

Chương 5. Kết luận và hướng phát triển

5.1. Kết luận

5.1.1. Ưu điểm:

Hiện tại hệ thống quản lý nhà kho đã xử lý được những vấn đề sau về mặt nghiệp vụ và kỹ thuật:

- Giao diện ứng dụng dễ dùng thuận tiện, thẩm mỹ, có thể xuất file excel, in đơn hàng.
- Ứng dụng xử lý được tình trạng sản phẩm tồn kho.
- Quản lý được từng giai đoạn cụ thể của đơn hàng.
- Quản lý được vấn đề sai sót mỗi lần giao nhận hàng hóa với đơn vị vận chuyển.
- Giảm thiểu sai sót khi nhập hàng lên kệ, có vị trí nhập, số lượng, tên sản phẩm, rõ ràng, cụ thể.
- Giảm thiểu sai sót khi thực hiện các tác vụ xuất hàng, có vị trí lấy hàng, rút hàng với số lượng cụ thể.
- Mở rộng được với nhiều nhà cung cấp, sản phẩm.
- Hiểu hơn về nghiệp vụ thực tế của ngành Logistic.
- Ứng dụng quản lý tài khoản bởi Admin nên rất chặt chẽ. Tài khoản hoạt động được phân quyền theo vai trò cụ thể, rõ ràng.
- Ứng dụng được chứng thực theo token, có hạn chế thời gian hoạt động của token.

5.1.2. Hạn chế:

Trong suốt quá trình làm đồ án, chúng em nhận thấy rằng bản thân còn nhiều thiếu sót cũng như kinh nghiệm lập trình, nên sản phẩm hiện tại còn nhiều hạn chế làm cản trở trải nghiệm của người dùng. Những hạn chế mà chúng em nhận thấy có thể liệt kê là:

- Khả năng ứng dụng mở rộng vai trò user còn hạn chế, chỉ giới hạn được ba vai trò chính.
- Chưa có quản lý khách hàng cho hàng cung cấp. Khách hàng ở đây là những cửa hàng bán sỉ, lẻ trên toàn quốc.

- DashBoard thống kê chưa thực sự hiệu quả, chưa có những thông tin về đơn hàng, sản phẩm trong kho, vị trí lưu kho.
- Việc tìm kiếm đơn hàng còn nhiều hạn chế, chỉ lọc hay tìm kiếm được những dữ liệu hiện có trong bảng đơn hàng hiện tại.
- Ứng dụng còn hạn chế nhiều trải nghiệm cũng như việc quản lý của user với vai trò là người cung cấp, như: Quản lý sản phẩm, quản lý quá trình giao nhận hàng ở kho,...
- Chưa thể phân cửa xuất hàng cho sản phẩm.

5.2. Hướng phát triển

Để có thể áp dụng vào doanh nghiệp, hệ thống hiện tại cần cải thiện những vấn đề sau:

- Tiếp tục hoàn thiện hệ thống, triển khai lên hệ thống web.
- Khắc phục những nhược điểm, hạn chế của hệ thống hiện tại.
- Thêm những chức năng thiết yếu khác cho ứng dụng, ví dụ như: Phân bổ đơn hàng cho các khách hàng của nhà cung cấp, phân cửa mỗi lần xuất kho.
- Mở rộng thêm phạm vi của hệ thống.

Việc cải thiện hệ thống cần có thời gian nghiên cứu tìm tòi thêm cả về kỹ thuật lẫn nghiệp vụ thực tế tại các doanh nghiệp. Hy vọng trong tương lai gần, hệ thống quản lý kho hàng của chúng em có thể được phát triển hơn nữa, đầy đủ tính năng cần thiết cũng như đáp ứng được nhu cầu và trải nghiệm của người dùng, và mong muốn hơn hết luôn là được áp dụng triển khai vào các doanh nghiệp kho vận.

TÀI LIỆU THAM KHẢO

- [1] ‘Học ReactJS trong 15 phút | TopDev’. <https://topdev.vn/blog/hoc-reactjs-trong-15-phut/> (accessed Jun. 27, 2021).
- [2] ‘React là gì? Những điều cơ bản về React dành cho người mới’, *DamMe*, May 07, 2021. <https://damme.io/react-la-gi/> (accessed Oct. 08, 2021).
- [3] ‘React JS - Hiểu về Functional và Class Components’, *Viblo*, Sep. 11, 2019. <https://viblo.asia/p/react-js-hieu-ve-functional-va-class-components-Qbq5QpkRID8> (accessed Oct. 08, 2021).
- [4] ‘Cơ bản về Router trong ReactJs’, *Viblo*, Oct. 14, 2018. <https://viblo.asia/p/co-ban-ve-router-trong-reactjs-07LKXzAEIV4> (accessed Oct. 17, 2021).
- [5] ‘Introducing Hooks – React’. <https://reactjs.org/docs/hooks-intro.html> (accessed Oct. 17, 2021).
- [6] ‘Cookie (tin học)’, *Wikipedia tiếng Việt*. Oct. 10, 2021. Accessed: Oct. 21, 2021. [Online]. Available: [https://vi.wikipedia.org/w/index.php?title=Cookie_\(tin_h%E1%BB%8Dc\)&oldid=66263957](https://vi.wikipedia.org/w/index.php?title=Cookie_(tin_h%E1%BB%8Dc)&oldid=66263957)
- [7] ‘Redux là gì? Hiểu rõ cơ bản cách dùng Redux | TopDev’. <https://topdev.vn/blog/redux-la-gi/> (accessed Oct. 21, 2021).
- [8] ‘Getting Started | Redux Toolkit’. <https://redux-toolkit.js.org/introduction/getting-started> (accessed Oct. 22, 2021).
- [9] ‘The web framework for perfectionists with deadlines | Django’. <https://www.djangoproject.com/> (accessed Nov. 04, 2021).
- [10] Ketan, ‘Django Model View Template (MVT) Overview’, *onlinetutorialspoint*, Mar. 20, 2021. <https://www.onlinetutorialspoint.com/django/django-model-view-template-mvt-overview.html> (accessed Nov. 04, 2021).
- [11] ‘Models | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/db/models/> (accessed Nov. 04, 2021).
- [12] ‘URL dispatcher | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/http/urls/> (accessed Nov. 04, 2021).
- [13] ‘Writing views | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/http/views/> (accessed Nov. 04, 2021).

- [14] ‘User authentication in Django | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/auth/> (accessed Nov. 04, 2021).
- [15] ‘Using the Django authentication system | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/auth/default/> (accessed Nov. 04, 2021).
- [17] ‘Making queries | Django documentation | Django’. <https://docs.djangoproject.com/en/3.2/topics/db/queries/> (accessed Nov. 05, 2021).
- [18] ‘Viewsets - Django REST framework’. Accessed: Nov. 05, 2021. [Online]. Available: <https://www.django-rest-framework.org/api-guide/viewsets/>
- [19] ‘Tìm hiểu về Swagger để viết API’, *Viblo*, Oct. 21, 2017. <https://viblo.asia/p/tim-hieu-ve-swagger-de-viet-api-XL6lAwbAKek> (accessed Nov. 05, 2021).
- [20] ‘Sử dụng Cloudinary để quản lý ảnh cho ứng dụng của bạn’, *Viblo*, Mar. 28, 2016. <https://viblo.asia/p/su-dung-cloudinary-de-quan-ly-anh-cho-ung-dung-cua-ban-E7bGoxggv5e2> (accessed Nov. 04, 2021).
- [21] ‘Routers - Django REST framework’. <https://www.django-rest-framework.org/api-guide/routers/> (accessed Nov. 05, 2021).
- [22] ‘Requests - Django REST framework’. <https://www.django-rest-framework.org/api-guide/requests/> (accessed Nov. 05, 2021).
- [23] ‘Serializers - Django REST framework’. <https://www.django-rest-framework.org/api-guide/serializers/> (accessed Nov. 05, 2021).
- [24] ‘Authentication - Django REST framework’. <https://www.django-rest-framework.org/api-guide/authentication/> (accessed Nov. 05, 2021).
- [25] T. Blog, ‘OAuth2 là gì? Chia sẻ kiến thức cơ bản về OAuth2 TopDev’, *TopDev*, Mar. 26, 2019. <https://topdev.vn/blog/oauth2-la-gi/> (accessed Nov. 05, 2021).
- [26] ‘Home - Django REST framework’. Accessed: Nov. 05, 2021. [Online]. Available: <https://www.django-rest-framework.org/>
- [27] ‘Views - Django REST framework’. <https://www.django-rest-framework.org/api-guide/views/> (accessed Nov. 05, 2021).

PHỤ LỤC

REACT COOKIES

Cookies là một thành phần mà được sử dụng nhiều trong các trang Website ở bất kỳ trình duyệt web nào. Theo wikipedia, chúng là những tập tin của một trang web gửi đến máy của người dùng, được lưu lại thông qua trình duyệt khi người dùng truy cập trang web đó. Cookie được sử dụng với mục đích phổ biến là lưu trữ phiên đăng nhập phục vụ cho mục đích xác thực, duy trì trạng thái đăng nhập. Ngoài ra, cookie còn được dùng để ghi nhớ thông tin trạng thái (ví dụ, ngôn ngữ), ghi nhớ hoạt động người dùng thực hiện trong quá trình truy cập và duyệt một trang web (ví dụ, những nút bấm hay đường liên kết người dùng tương tác). Cookie còn được sử dụng để lưu các thông tin khác khi người dùng nhập hay điền vào trang web như tên, địa chỉ, email, v.v...[6] Trong ReactJS sử dụng gói React Cookies để dễ dàng tương tác với Cookies.

Cài đặt gói React Cookies:

```
npm install react-cookies --save
```

Sử dụng bằng cách import

```
import cookies from 'react-cookies'
```

Một số phương thức quan trọng trong React Cookies

- **cookies.load(name):** lấy một cookie với tham số name tương ứng.
- **cookies.loadAll():** lấy tất cả các cookies.
- **cookies.select([regex]):** tìm và trả về cookie có tên khớp với một biểu thức chính quy.
- **cookies.save(name, value, [options]):** để lưu một cookie.
- **remove(name, [options]):** dùng để xóa một cookie.

Một số lựa chọn quan trọng khi lưu hoặc xóa Cookies

- **path** trả về string: là đường dẫn được phép truy cập cookie.
- **maxAge** trả về number
- **expires** trả về object: là thời điểm hết hạn cookie.
- **domain** trả về kiểu string: tên miền cho cookie.

- **secure** trả về kiểu boolean: thiết lập true nếu muốn cookie chỉ được truy cập qua HTTPS.
- **httpOnly** trả về kiểu boolean: thiết lập true nếu muốn cookie chỉ được phép truy cập trên server.

Ví dụ lưu Token vào Cookies

```
const response = await userApi.login(fromData);  
  
//lưu vào cookie  
  
cookies.save("access-token", response.access_token);
```

Ví dụ lấy Token từ Cookies để xác thực đăng nhập

```
axiosClient.interceptors.request.use(async (config) => {  
  const token = cookies.load("access-token");  
  config.headers.Authorization = token ? `Bearer ${token}` : "";  
  return config;  
});
```