TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI



KHOA CÔNG NGHỆ THÔNG TIN

Icon

Description automatically generated

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**HỆ THỐNG QUẢN LÝ THÁO DỠ LINH KIỆN**

|  |
| --- |
| Sinh viên thực hiện: **Bùi Xuân Tiến**  Mã sinh viên: **0206266**  Lớp quản lý: **66CNPM**  Giảng viên hướng dẫn: **ThS. Nguyễn Đình Anh** |
|  |

**HÀ NỘI 12/2024**

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên:

Họ và tên sinh viên: Bùi Xuân Tiến

Điện thoại liên lạc: 0904850431 Email: [tien0206266@huce.edu.vn](mailto:tien0206266@huce.edu.vn)

Lớp: 66CNPM Hệ đào tạo: Cử nhân Đại học Năm tốt nghiệp: 2025

Đồ án tốt nghiệp được thực hiện tại: Đại học Xây Dựng Hà Nội.

Thời gian làm ĐATN: Từ ngày 05/08/2024 đến 4/12/2024.

2. Mục đích nội dung của ĐATN:

- Xây dựng hệ thống quản lý tháo dỡ linh phụ kiện trong ngành sản xuất

- Cho phép người dùng có thể theo dõi số lượng linh kiện đã và đang được tháo dỡ

- Cho phép người dùng quản lý, theo dõi, xử lý các loại hàng hỏng, hàng cho mượn

3. Các nhiệm vụ cụ thể của ĐATN:

- Tìm hiểu các yêu cầu của một hệ thống quản lý logistic, tháo dỡ linh kiện.

- Đưa ra những tính năng cải thiện, mở rộng thêm chức năng cho người dùng.

- Phân tích yêu cầu bài toán.

- Thực hiện yêu cầu bài toán.

- Cài đặt hệ thống.

4. Lời cam đoan của sinh viên:

Tôi – *Bùi Xuân Tiến* – cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của ThS. Nguyễn Đình Anh.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

*Hà Nội, ngày tháng năm 20*

**Tác giả ĐATN**

**Bùi Xuân Tiến**

5. Xác nhận của giảng viên hướng dẫn về mức độ hoàn thành ĐATN và cho phép bảo vệ:

…………………………………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………………………………

*Hà Nội, ngày tháng năm 20*

**Giảng viên hướng dẫn**

**ThS. Nguyễn Đình Anh**

**TÓM TẮT  
 NỘI DUNG ĐỒ ÁN TỐT NGHIỆP**

**1. Nội dung đồ án**

1.1. Tìm hiểu bài toán, xây dựng ứng dụng

* Tìm hiểu nghiệp vụ hệ thống Unpacking Part thực tế
* Tìm hiểu công nghệ để ứng dụng vào hệ thống
* Tìm hiểu các các hạn chế của hệ thống cũ (nếu có)

1.2. Phân tích thiết kế hệ thống

* Phân tích thiết kế cơ sở dữ liệu
* Phân tích thiết kế chức năng theo nghiệp vụ hệ thống

**2. Nội dung các chương trình bày trong báo cáo**

* Chương 1: Đặt vấn đề - định hướng - giải pháp
* Chương 2: Các kết quả đạt được
* Chương 3: Kết luận

MỤC LỤC

[PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP 1](#_Toc155128567)

[Chương I. Đặt vấn đề - định hướng – giải pháp 1](#_Toc155128568)

[1. Đặt vấn đề: 1](#_Toc155128569)

[2. Nghiệp vụ hệ thống: 1](#_Toc155128570)

[3. Các chức năng chính của hệ thống. 2](#_Toc155128571)

[3.1. Statistics 2](#_Toc155128572)

[3.2. Devaning 2](#_Toc155128573)

[3.3. Unpacking 3](#_Toc155128574)

[3.4. Robing 3](#_Toc155128575)

[4. Cơ sở lý thuyết 3](#_Toc155128576)

[4.1. Tổng quan về Angular 3](#_Toc155128581)

[4.2. Các phiên bản Angular 4](#_Toc155128582)

[4.3. Giới thiệu về Angular 6](#_Toc155128583)

[4.4. ASP .Net Core MVC 9](#_Toc155128587)

[4.5. Tổng quan về LINQ 11](#_Toc155128589)

[Chương II. Các kết quả đạt được 21](#_Toc155128590)

[1. Đặc tả UseCase và sơ đồ phân rã 21](#_Toc155128591)

[1.1. DvnContainer 21](#_Toc155128592)

[1.2. Màn hình dỡ Container (DevaningScreen) 23](#_Toc155128593)

[1.3. UpkLot 25](#_Toc155128594)

[1.4. UpkPartList 26](#_Toc155128595)

[1.5. UpkScreen 27](#_Toc155128596)

[1.6. Robing 29](#_Toc155128597)

[2. Sơ đồ Sequencece 30](#_Toc155128598)

[2.1. Đăng nhập 30](#_Toc155128599)

[2.2. Tìm kiếm 31](#_Toc155128600)

[2.3. Báo cáo 31](#_Toc155128601)

[2.4. Thêm mới DvnContList 32](#_Toc155128602)

[2.5. Cập nhật DvnContList 32](#_Toc155128603)

[2.6. Xóa DvnContList 33](#_Toc155128604)

[2.7. Xuất báo cáo DvnContList 33](#_Toc155128605)

[2.8. Theo dõi tiến trình DvnContList 34](#_Toc155128606)

[2.9. Dỡ Module (Unpacking module) 34](#_Toc155128607)

[2.10. Kết thúc dỡ Module (Unpacking Module Finish) 35](#_Toc155128608)

[2.11. Yêu cầu hàng hỏng (Robing Request) 35](#_Toc155128609)

[2.12. Yêu cầu xử lý hàng hỏng (Robing Request Give Back) 36](#_Toc155128610)

[3. Sơ đồ Activity 36](#_Toc155128611)

[3.1. Thêm mới DvnContainer 36](#_Toc155128612)

[3.2. Chỉnh sửa DvnContainer 37](#_Toc155128613)

[3.3. Xóa DvnContainer 37](#_Toc155128614)

[3.4. Xuất báo cáo DvnContainer 38](#_Toc155128615)

[3.5. Kết thúc dỡ Module 39](#_Toc155128616)

[3.6. Xóa UpkLot 40](#_Toc155128617)

[3.7. Xuất báo cáo UpkLot 41](#_Toc155128618)

[3.8. Xuất báo cáo UpkPartList 42](#_Toc155128619)

[3.9. Yêu cầu xử lý hàng hỏng 43](#_Toc155128620)

[4. Cơ sở dữ liệu 44](#_Toc155128621)

[4.1. Các bảng 44](#_Toc155128622)

[4.2. Cơ sở dữ liệu 46](#_Toc155128623)

[4.3. Sơ đồ Class 47](#_Toc155128624)

[4.4. Sơ đồ Interface 48](#_Toc155128625)

[4.5. Sơ đồ UseCase- Mô tả các chức năng của hệ thống: 48](#_Toc155128626)

[5. Giao diện hệ thống 49](#_Toc155128627)

[5.1. Statistics 49](#_Toc155128628)

[5.2. DvnContainer 5](#_Toc155128629)0

[5.3. DvnScreen 5](#_Toc155128630)1

[5.4. UpkLot 5](#_Toc155128631)2

[5.5. UpkPartList 5](#_Toc155128632)3

[5.6. UpkScreen 5](#_Toc155128633)3

[5.7. Robing 5](#_Toc155128634)4

[5.8. PcStore 5](#_Toc155128635)4

[5.9. PcHome 5](#_Toc155128636)5

[Chương III. PHÂN QUYỀN 5](#_Toc155128637)6

[1. Vai trò người dùng 5](#_Toc155128638)6

[2. Quyền theo màn hình 5](#_Toc155128639)6

[Chương IV. Kết luận 5](#_Toc155128637)7

[1. Đánh giá ưu nhược điểm 5](#_Toc155128638)7

[2. Kết quả thực hiện 5](#_Toc155128639)7

[3. Định hướng phát triển 5](#_Toc155128640)8

[TÀI LIỆU THAM KHẢO 5](#_Toc155128641)9

DANH MỤC ẢNH

[Hình 1: UseCase - UpkLot 21](#_Toc155179632)

[Hình 2: UseCase - DevaningScreen 23](#_Toc155179633)

[Hình 3: UseCase - UpkLot 25](#_Toc155179634)

[Hình 4: UseCase - Export 26](#_Toc155179635)

[Hình 5: UseCase UpkScreen 27](#_Toc155179636)

[Hình 6: UseCase - Robing 29](#_Toc155179637)

[Hình 7: Sơ đồ Sequence - Đăng nhập 30](#_Toc155179638)

[Hình 8: Sơ đồ Sequence - Tìm kiếm 31](#_Toc155179639)

[Hình 9: Sơ đồ Sequence - Báo cáo 31](#_Toc155179640)

[Hình 10: Sơ đồ Sequence Thêm mới DvnContList 32](#_Toc155179641)

[Hình 11: Sơ đồ Sequence cập nhật DvnContList 32](#_Toc155179642)

[Hình 12: Sơ đồ Sequence xóa DvnContList 33](#_Toc155179643)

[Hình 13: Sơ đồ Sequence - Xuất báo cáo DvnContList 33](#_Toc155179644)

[Hình 14: Sơ đô Sequence theo dõi tiến trình DvnContList 34](#_Toc155179645)

[Hình 15: Sơ đồ Sequence Dỡ Module 34](#_Toc155179646)

[Hình 16: Sơ đồ Sequence kết thúc dỡ Module 35](#_Toc155179647)

[Hình 17: Sơ đồ Sequence Yêu cầu hàng hỏng 35](#_Toc155179648)

[Hình 18: Sơ đồ Sequence yêu cầu xử lý hàng hỏng 36](#_Toc155179649)

[Hình 19: Sơ đồ Activity thêm mới DvnContainer 36](#_Toc155179650)

[Hình 20: Sơ đồ Activity chỉnh sửa DvnContainer 37](#_Toc155179651)

[Hình 21: Sơ đồ Activity xóa DvnContainer 37](#_Toc155179652)

[Hình 22: Sơ đồ Activity xuất báo cáo DvnContainer 38](#_Toc155179653)

[Hình 23: Sơ đồ Activity kết thúc dỡ Module 39](#_Toc155179654)

[Hình 24: Sơ đồ Activity xóa UpkLot 40](#_Toc155179655)

[Hình 25: Sơ đồ Activity xuất báo cáo UpkLot 41](#_Toc155179656)

[Hình 26: Sơ đồ Activity xuất báo cáo UpkPartList 42](#_Toc155179657)

[Hình 27: Sơ đồ Activity yêu cầu xử lý hàng hỏng 43](#_Toc155179658)

[Hình 28: Bảng DvnContList 44](#_Toc155179659)

[Hình 29: Bảng LupContModule 44](#_Toc155179660)

[Hình 30: Bảng Part 45](#_Toc155179661)

[Hình 31: Bảng PcStore 45](#_Toc155179662)

[Hình 32: Bảng PcHome 45](#_Toc155179663)

[Hình 33: Bảng Robing 46](#_Toc155179664)

[Hình 34: Sơ đồ class 47](#_Toc155179665)

[Hình 35: Sơ đồ Interface 48](#_Toc155179666)

[Hình 36: Sơ đồ UseCase 48](#_Toc155179667)

[Hình 37: Giao diện Statistics 49](#_Toc155179668)

[Hình 38: Giao diện DvnContainer 49](#_Toc155179669)

[Hình 39: Giao diện DvnScreen 5](#_Toc155179670)2

[Hình 40: Giao diện UpkLot 5](#_Toc155179671)2

[Hình 41: Giao diện UpkPartList 5](#_Toc155179672)3

[Hình 42: Giao diện UpkScreen 5](#_Toc155179673)3

[Hình 43: Giao diện Robing 5](#_Toc155179674)4

[Hình 44: Giao diện PcStore 5](#_Toc155179675)4

[Hình 45: Giao diện PcHome 5](#_Toc155179676)5

Chương I. Đặt vấn đề - định hướng – giải pháp

1. Đặt vấn đề:

Trong ngành công nghiệp cung cấp phụ kiện và linh kiện cho xe ôtô, quá trình đóng gói và Unpacking Part đóng một vai trò quan trọng trong việc đảm bảo chất lượng, an toàn và trải nghiệm tốt nhất cho người tiêu dùng. Tuy nhiên, so với sự chú ý được đặt vào quá trình sản xuất và vận chuyển, Unpacking Part thường ít được nghiên cứu và đánh giá một cách đầy đủ.

Unpacking Part trong ngữ cảnh của ngành cung cấp phụ kiện và linh kiện xe ôtô không chỉ đơn thuần là việc mở gói sản phẩm, mà còn bao gồm quá trình bốc dỡ và sắp xếp các phụ kiện, linh kiện một cách thuận tiện và an toàn. Sự thuận lợi và hiệu quả trong Unpacking Part không chỉ ảnh hưởng đến thời gian và công sức mà còn quan trọng đối với việc bảo vệ sản phẩm khỏi tổn thương và hạn chế lãng phí.

Hiện tại, có nhiều thách thức mà ngành công nghiệp này đang phải đối mặt trong quá trình Unpacking Part, từ việc quản lý lượng thông tin đích thực về sản phẩm đến việc đảm bảo tính tương tác và trải nghiệm tích cực cho người sử dụng cuối cùng. Những thách thức này đặt ra câu hỏi về cách nghiên cứu và cải thiện quá trình Unpacking Part có thể đóng góp vào sự hiệu quả của chuỗi cung ứng và tăng cường hài lòng của khách hàng.

Do đó, nghiên cứu về Unpacking Part trong ngành cung cấp phụ kiện và linh kiện xe ôtô không chỉ hỗ trợ các doanh nghiệp nắm bắt cơ hội thị trường mà còn có thể đưa ra những giải pháp cụ thể để tối ưu hóa quá trình này, từ đơn giản hóa đóng gói đến cải thiện trải nghiệm người sử dụng khi mở gói sản phẩm.

2. Nghiệp vụ hệ thống:

Cách thức hoạt động của hệ thống:

* Quá trình Devaning

 +) Quá trình devanning là quá trình bốc dỡ hàng hóa từ container sau khi chúng đến từ nơi xuất khẩu. Container được mở bằng các thiết bị đặc biệt, sau đó hàng hóa được bốc dỡ bằng xe nâng hoặc cần cẩu. Hàng hóa sau đó được kiểm tra, ghi nhận thông tin và chuyển đến nơi đích cuối cùng như kho lưu trữ hoặc trung tâm phân phối.

* Quá trình Unpacking

 +) Quá trình Unpacking là việc mở gói sản phẩm từ nguồn gốc vận chuyển. Sau khi kiểm tra và ghi nhận thông tin, sản phẩm được bốc dỡ, sắp xếp, và kiểm tra cuối cùng. Các linh kiện, phụ kiện sau khi mở sẽ được đưa vào các xưởng tương ứng để chuẩn bị cho quá trình sản xuất

- Quá trình Robing

 +) Quá trình Robing là việc phân loại hàng hỏng, hàng cho mượn, hàng quá hạn sử dụng trong khi quá trình Unpacking được diễn ra. Các loại hàng đó sẽ được đưa và mục robing để được xử lý cho từng riêng loại hàng

3. Các chức năng chính của hệ thống.

**3.1. Statistics**

Hệ thống có một màn hình để theo dõi các dữ liệu trong suốt quá trình từ Devaning tới Unpacking, và thống kê số lượng hàng lỗi, hỏng (Robing), số lượng hàng đạt tiêu chuẩn.

1. Thống kê số lượng Container
2. Thống kê số lượng hàng Robing
3. Thống kê số lượng Part
4. Thống kê số lượng Module
5. Thống kê số hàng tồn tại trong PC

**3.2. Devaning**

Với Devaning hệ thống gồm 2 màn hình chính và các chức năng như sau:

• Màn hình Devaning Container:

* Theo dõi số lượng Container qua 3 tab: New, Devaning, Devaned.
* Tab New thêm, sửa, xóa Container.
* Tab Devaning thống kê Container đang trong quá trình dỡ.
* Tab Devaned thống kê Container sau khi dỡ xuống xong.
* Xuất báo cáo về số lượng Container cả 3 tab.

• Màn hình Devaning Screen:

* Theo dõi quá trình Devaning (Container đã dỡ và đang dỡ trong ngày)
* Nhưng Container đã được đặt thời gian mở trong ngày sẽ được theo dõi và những Container đã dỡ xong trong ngày sẽ được thống kê.
* Với mỗi Container sẽ được người theo dõi đặt thời gian mở và hết thời gian thì Container đó nếu dỡ thành công theo thời gian đặt trước thì sẽ dỡ Container tiếp theo, nếu không thì sẽ tiếp tục dỡ đến khi có xác nhận thành công bằng cách nhấn (FINISH DVN) hoặc nhấn hoàn thành sau khi thanh tiến trình đạt 100%.
* Kết thúc sớm hoặc muộn hơn quá trình Devaning (FINISH DVN).

**3.3. Unpacking**

Với Unpacking, gồm 3 màn hình chính với các chức năng như sau:

Màn hình UpkLot:

* Theo dõi số lượng, trạng thái của các Module sau quá trình Devaning.
* Thống kê, xuất báo cáo số lượng, chi tiết của các Module.

Màn hình UpkPartList.

* Xem chi tiết các linh kiện, phụ kiện (Part) có trong mỗi Module, từ đó biết được trạng thái, nguồn gốc của các linh kiện, phụ kiện đó.

Màn hình UpkScreen

* Xem số lượng , kế hoạch mở hàng (Unpack) trong 1 ngày : Số Module đã mở, số Module cần mở
* Xem số lượng thực tế của linh kiện đã được mở ra : Số Part đã mở, số Part có trong Module đó
* Phân loại hàng bao gồm : Phân loại vào kho (FINISH) và phân loại hàng hỏng, hàng quá hạn, hàng không sử dụng được, hàng cho mượn,… (Robing)

**3.4. Robing**

- Thống kê, xem chi tiết các loại hàng được phân loại từ quá trình UNPACKING.

- Gửi yêu cầu để xem xét xử lý hàng hỏng (Request Give Back)

4. Cơ sở lý thuyết



**4.1. Tổng quan về Angular**

Angular là một javascript framework do google phát triển để xây dựng các Single Page Application (SPA) bằng JavaScript, HTML và TypeScript. Angular cung cấp các tính năng tích hợp cho animation, http service và có các tính năng như auto-complete, navigation, toolbar, menus, …. Code được viết bằng TypeScript, biên dịch thành JavaScript và hiển thị tương tự trong trình duyệt.

Để học được Angular thì bạn cần biết các kiến thức cơ bản sau:

* HTML
* CSS
* JavaScript
* TypeScript
* Document Object Model (DOM)

**4.2. Các phiên bản Angular**

**Angular js:** Phiên bản đầu tiên của angular là AngularJS được bắt đầu từ năm 2009 và đc ra mắt vào 20/10/2010, do lập trình viên Misko Hevery tại Google viết ra như là một dự án kiểu “viết cho vui”. Lúc đó angular js được viết theo mô hình MVC (Model-View-Controller) trong đó:

* Model là thành phần trung tâm thể hiện hành vi của ứng dụng và quản lí dữ liệu.
* View được tạo ra dựa trên thông tin của Model.
* Controller đóng vai trò trung gian giữa Model và View và để xử lý logic.

**Angular 2:**

Sau phiên bản angular js thì vào tháng 3 năm 2015 phiên bản bản angular 2 ra đời nhằm thay thế Angular Js với các khái niệm mới nhằm đơn giản hóa và tối ưu cho quá trình phát triển sử dụng framework này. Angular 2 thay đổi hoàn toàn so với angular js bằng việc thay Controllers và $scope (Angular js) bằng components và directives. Components = directives + template, tạo lên view của ứng dụng và xử lí các logic trên view. Angular 2 hoàn toàn được viết bằng Typescript. Angular 2 nhanh hơn angular js, hỗ trợ đa nền tảng đa trình duyệt, cấu trúc code được tổ chức đơn giản và dễ sử dụng hơn.

**Angular 4:**

Ra mắt vào tháng 3/2017 đây là một phiên bản nâng cấp từ Angular 2 nên kiến trúc không thay đổi nhiều ngoài việc giảm thiểu code được tạo ra từ đó giảm kích thước tệp được đóng gói xuống 60%, đẩy nhanh quá trình phát triển ứng dụng.

**Angular 5:**

* Đã được phát hành vào ngày 1 tháng 11 năm 2017 với mục tiêu thay đổi về tốc độ và kích thước nên nó nhanh hơn và nhỏ hơn angular 4. Các tính năng mới so với angular 4:
* Sử dụng HTTPClient thay vì sử dụng HTTP: bởi vì nó nhanh, an toàn và hiệu quả hơn.
* Với phiên bản Angular 5 mặc định sử dụng RxJs 5.5
* Multiple export aliases: Một component có thể được xuất bằng nhiều bí danh (aliases) để giảm bớt quá trình di chuyển.
* Internationalized Pipes for Number, Date, and Currency: Các pipe mới được giới thiệu để tiêu chuẩn hóa tốt hơn.
* Tối ưu hóa build production bằng việc sử dụng công cụ build optimizer được tích hợp sẵn vào trong CLI. Công cụ này tối ưu tree shark và loại bỏ code dư thừa.
* Cải thiện tốc độ biên dịch bằng việc dùng TypeScript transforms, giờ đây khi build sẽ sử dụng lệnh “ng serve –aot”. AOT sẽ cải thiện performance khi load page và nó được dùng để deploy app lên production

**Angular 6:**

* Cập nhật CLI, command line interface: thêm 1 số lệnh mới như ng-update để chuyển từ version trước sang version hiện tại; ng-add để thêm các tính năng của ứng dụng để trở thành một ứng dụng web tiến bộ.
* Angular Element: Cho phép các component của Angular được triển khai dưới dạng component web, sau đó có thể được sử dụng trong bất kỳ trang HTML nào một cách dễ dàng.
* Multiple Validators: cho phép nhiều Validators được áp dụng trên form builder.
* Tree-shakeable providers: giúp loại bỏ mã code chết.
* Sử dụng RxJS 6 với syntax thay đổi.

**Angular 7:**

* Được phát hành vào 18 tháng 10 năm 2018 với những thay đổi như :
* ScrollingModule: Để scroll load dữ liệu.
* Drag and Drop: Chúng ta có thể dễ dàng thêm tính năng kéo và thả vào một mục
* Angular 7.0 đã cập nhật RxJS 6.3

**Angular 8:**

* Ra mắt 28 tháng 5 năm 2019 với CLI workflow improvements, Dynamic imports for lazy routes ….

**Angular 9:**

* Ra mắt mới đây 6 tháng 2 năm 2020, Angular 9 di chuyển tất cả các ứng dụng để sử dụng trình biên dịch Ivy và thời gian chạy theo mặc định. Angular đã được cập nhật để hoạt động với TypeScript 3.6 và 3.7

**4.3. Giới thiệu về Angular**



4.3.1Component

Components là một khối code trong app Angular. Nó là sự kết hợp của bộ template html và nhúng kèm code TypeScript (hoặc Javascript). Các components là độc lập với nhau và độc lập với hệ thống. Nó có thể được cài vào hoặc tháo ra khỏi hệ thống dễ dàng. Một component có thể hiểu như một control trên màn hình hiển thị, gồm giao diện html và code logic xử lý sự kiện đi kèm control đó. Một component cũng có thể to lớn như là cả 1 màn hình chứa nhiều control hoặc một nhóm nhiều màn hình. Tức là là một component cũng có thể chứa và gọi được nhiều component khác nối vào.

Text

Description automatically generated

Trong định nghĩa bộ khung html, chúng ta có một số thuộc tính cần chú ý sau đây:

- selector: Là tên được đặt để gọi một component trong code html. Ở ví dụ vừa rồi, từ khóa hello-ng-world được đặt tên cho component này. Khi cần gọi component này ra ở màn hình html cha, ta sẽ gọi bằng html tag <app-category-list></app-category-list>. Gọi như vậy thì component con sẽ được render ra component cha.

- template: Là tự định nghĩa khung html cho component.

- templateUrl: Là đường dẫn url tới file html bên ngoài để load file đó vào làm khung html cho component này.

- styles: Là viết style css luôn vào file component này. Cách này chỉ dùng cho component đơn giản.

- styleUrls: Là đường dẫn url đến file style css độc lập cho component này

4.3.2. Binding

Angular có cách code Binding (kết nối giữa html và data) dữ liệu theo kiểu 2 chiều, nghĩa là html input thay đổi thì biến javascript sẽ ngay lập tức nhận được giá trị trả về và ngược lại, giá trị trong js thay đổi thì ngay lập tức màn hình html thay đổi theo. Để bind một chuỗi ra ngoài màn hình html thì rất đơn giản sử dụng 2 dấu ngoặc nhọn {{TenBien}}.

Text

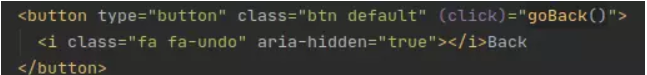
Description automatically generated

- *To-way binding:*

Từ khóa ngModel lúc này không phải là thuộc tính html mà nó là từ khóa của Angular. Khi chúng ta viết [(ngModel)] chúng ta sẽ gắn chặt giá trị của input html với biến title. Dẫn đến người dùng gõ vào ô input thì thẻ h1 sẽ nhận giá trị tương ứng của title.

* + 1. Xử lý sự kiện

Để gắn 1 sự kiện của control html với một hàm javascript, ta viết như sau:



Khi chúng ta muốn (click) tức là muốn bind sự kiện click chuột của người dùng ở control này gọi là goBack().

4.3.4. NgModule

Module là một khái niệm rộng nhất của Angular. Một module có thể bao gồm chứa các components, directives, pipes, v.v.

Module có thể được biên dịch (compile) dưới dạng ahead-of-time (AoT). Nghĩa là biên dịch ra mã thực thi để hiện ra luôn trên trình duyệt không cần vẽ vời gì từ đầu. Hãy tưởng tượng component có html và js viết riêng, khi load trang thì 2 thứ này mới nhào nặn chung để hiển thị html+data lên màn hình. AoT là thứ html+data đã nhào sẵn.

Module cũng có thể gọi module con và bắt tay được với các module khác.

Các thuộc tính của module được định nghĩa như sau:

+ imports: Định nghĩa sự phụ thuộc (Dependency) của module này, module phụ thuộc sẽ được load trước rồi module này mới load.

+ declarations: Định nghĩa tất cả các component sẽ được dùng trong module này. Nếu chưa định nghĩa thì các component trong module sẽ không thể gọi nhau vì không tìm thấy nhau.

+ bootstrap: Mỗi ứng dụng Angular đều cần một module gốc, module này sẽ có một component gốc chứa layout gốc sẽ được render ra ở file index.html.

4.3.5. Router

1. *Router OutLet*

Mỗi một Router sẽ có một URL để load component. Và để biết được là component sẽ render ra chỗ nào thì chúng ta viết đoạn code sau vào khung html cần chèn:



1. *Cài đặt Router cho ứng dụng Angular*

Để cài đặt toàn bộ Router cho một ứng dụng Angular thì chúng ta cần tạo ra một đối tượng JSON chứa các thuộc tính như sau:

* path: Đường dẫn URL của component hiện tại.
* component: Ứng với đường dẫn bên trên thì load component nào.
* redirectTo: Chuyển hướng đến URL này nếu URL ở path không trùng. Ví dụ, khi người dùng gõ URL linh tinh, chúng ta muốn chuyển hướng và load trang Home hoặc trang báo lỗi 404 thì cần ghi rõ URL trang Home hoặc 404 vào redirectTo.
* pathMatch: Cài đặt xem chế độ kiểm tra url là như thế nào. khi giá trị là full thì nghĩa là toàn bộ url từ đầu đến cuối sẽ phải chính xác như trong bộ JSON Router. Còn khi giá trị là prefix thì chuỗi đầu tiên của url (dấu sược đầu tiên) sẽ được kiểm tra. Mặc định nếu không nói gì thì prefix sẽ được chọn. Hãy xem đoạn code ví dụ về Router bên dưới:

Text

Description automatically generated

**4.4. ASP .Net Core MVC**



4.4.1. Mô hình MVC

Mô hình MVC là một kiến trúc thiết kế của một phần mềm công nghệ thông tin, trong đó có cả PHP. Khi một phần mềm sử dụng mô hình này thì nó sẽ phải chia ra thành ba phần chính Model-View-Controller (MVC) với  3 nhiệm vụ khác nhau:

* + - Model: là thành phần chứa tất cả các phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý…
    - View: Là phần đảm nhận việc hiển thị thông tin, tương tác với người dùng. Hoặc chúng ta có thể hiểu nôm na đó là phần giao diện của ứng dụng.
    - Controller: Controller là phần xử lý và điều hướng các hành động của client, từ đó đưa ra các xử lý với database nếu có. Hay nói các khác thì controller là cầu nối giữa view và Model.

4.4.2. ASP.Net Core MVC

ASP.NET Core MVC là một framework “nhẹ”, opensource, giúp tối ưu hóa hiệu năng của ứng dụng với ASP.NET Core.

ASP.NET Core MVC cung cấp các tính năng dựa trên mô hình xây dựng website động cho phép phân chia rõ ràng các khối lệnh. Nó cung cấp cho bạn toàn quyền kiểm soát đánh dấu, hỗ trợ phát triển với TDD-friendly và sử dụng các tiêu chuẩn web mới nhất.

Tính năng của ASP.NET Core MVC bao gồm:

* Routing
* Model binding
* Model validation
* Dependency injection
* Filters
* Areas
* Web APIs
* Testability
* Razor view engine
* Strongly typed views
* Tag Helpers
* View Components

*Routing*

Routing sẽ xác định URL và điều khiển thông tin tương ứng với URL mà người dùng web nhập vào. Tất cả các cấu hình Routing của một ứng dụng ASP.NET MVC được lưu trữ trong RouteTable, nó đóng vai trò định tuyến các URL để xác định các lớp xử lý tương ứng khi có request được gửi đến từ người dùng web.

*Model binding*

Trong ASP.NET Core MVC Model binding chuyển đổi dữ liệu yêu cầu từ phía client (form values, route data, query string parameters, HTTP headers) vào bên trong đối tượng để controller có thể xử lí. Kết quả là Controller của bạn không phải thực hiện công việc tìm ra dữ liệu từ đâu đến. Nó chỉ đơn giản là kiểm tra dữ liệu và tham số từ phương thức action.

*Model validation*

ASP.NET Core MVC hỗ trợ ràng buộc dữ liệu cho các thuộc tính trong model của bạn, các thuộc tính sẽ được kiểm tra ở client xem có hợp lệ không trước khi giá trị của thuộc tính đó được gửi về server. Cũng như trên server trước khi action của controller gọi.

*Dependency injection*

Trong ASP.NET Core MVC controller có thể gửi yêu cầu cần thiết đến các service thông cấu trúc xây dựng của chúng.

*Web APIs*

Ngoài việc là một nền tảng tuyệt vời để xây dựng các trang web. ASP.NET Core MVC hỗ trợ rất nhiều cho việc xây dựng API Web. Bạn có thể xây dựng các dịch vụ tiếp cận nhiều khách hàng bao gồm trình duyệt web và thiết bị di động.

…

**4.5. Tổng quan về LINQ**

a. Khái niệm

Để giảm gánh nặng thao tác trên nhiều ngôn ngữ khác nhau và cải thiện năng suất lập trình, Microsoft đã phát triển giải pháp tích hợp dữ liệu cho .NET Framework có tên gọi là LINQ (Language Integrated Query), đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic.NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu Object, cơ sở dữ liệu và XML.

Điểm mạnh của LINQ là “viết truy vấn cho rất nhiều các đối tượng dữ liệu”. Từ cơ sở dữ liệu, XML, Data Object … thậm chí là viết truy vấn cho một biến mảng đã tạo ra trước đó. Vì thế ta có các khái niệm như là LinQ to SQL, LinQ to XML,….

b. Các thành phần của Linq

*LINQ to Objects*

“LINQ to Objects” ở đây có nghĩa là nói đến cách sử dụng LINQ đối với các đối tượng Collection mà đã được thực thi giao diện IEnumerable hoặc IEnumerable<T> tức những collection có thể “liệt kê” ra được. Đây là trường hợp sử dụng đơn giản nhất của LINQ khi làm việc với dữ liệu.

A screenshot of a computer

Description automatically generated

*LINQ to SQL*

LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework 3.5, nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu dùng LINQ, cũng như cập nhật/thêm/xóa dữ liệu từ đó.

LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP). Nó cũng cung cấp một cách dễ dàng để thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu của bạn.

*LINQ to DataSet*

LINQ to DataSet cho phép người lập trình sử dụng DataSets như một nguồn dữ liệu bình thường bằng các cú pháp truy vấn căn bản của LINQ

4.6. Tổng quan về Docker

**a. Khái niệm**

-Docker là nền tảng phần mềm cho phép bạn dựng, kiểm thử và triển khai ứng dụng một cách nhanh chóng. Docker đóng gói phần mềm vào các đơn vị tiêu chuẩn hóa được gọi là container có mọi thứ mà phần mềm cần để chạy, trong đó có thư viện, công cụ hệ thống, mã và thời gian chạy. Bằng cách sử dụng Docker, bạn có thể nhanh chóng triển khai và thay đổi quy mô ứng dụng vào bất kỳ môi trường nào và biết chắc rằng mã của bạn sẽ chạy được.

Dưới đây là một số khái niệm chính liên quan đến Docker:

-Image (Hình ảnh): Là một gói chứa tất cả các thành phần cần thiết để chạy một ứng dụng, bao gồm cả mã nguồn, thư viện, môi trường và cấu hình. Image là một bản ghi chỉ đọc và không thay đổi.

-Container: Một thực thể chạy của một image. Containers cung cấp môi trường đóng gói để thực thi ứng dụng một cách độc lập với môi trường hệ thống.

-Dockerfile: Một tệp cấu hình văn bản mô tả cách Docker nên xây dựng một image. Nó định nghĩa các bước cụ thể để cài đặt các thành phần cần thiết và cấu hình môi trường.

-Registry: Một nơi lưu trữ chứa các images. Docker Hub là một ví dụ về registry công cộng, trong khi bạn cũng có thể triển khai registry riêng tư.

-Docker Compose: Một công cụ cho phép định nghĩa và quản lý nhiều containers cùng một lúc. Nó cho phép bạn mô tả ứng dụng, các dịch vụ, cấu hình mạng và các yếu tố khác trong một tệp cấu hình đơn.

-Orchestration: Quá trình quản lý và triển khai một số lượng lớn các containers. Kubernetes là một ví dụ phổ biến về orchestration tool.

Volume: Một cách để lưu trữ và quản lý dữ liệu trong container một cách persistent, tức là dữ liệu không bị mất khi container dừng lại.

**b. Docker Container**

Một Docker container là một thực thể chạy của một Docker image. Nó là một môi trường đóng gói, chứa tất cả mọi thứ cần thiết để chạy một ứng dụng, bao gồm cả mã nguồn, thư viện, môi trường thực thi, cấu hình, và các phụ thuộc khác. Containers cung cấp một cách để đóng gói ứng dụng và các thành phần của nó thành một đơn vị đồng nhất, độc lập với môi trường hệ thống.

Dưới đây là một số điểm chính về Docker container:

Lightweight: Containers sử dụng chia sẻ kernel của hệ điều hành host, điều này giúp giảm độ phức tạp và tăng hiệu suất so với máy ảo truyền thống. Containers chỉ chứa các thành phần cần thiết, làm cho chúng nhẹ và nhanh chóng.

Độc lập và Cô lập: Mỗi container chạy độc lập với các container khác và với hệ điều hành host. Điều này đảm bảo tính cô lập giữa các ứng dụng và đồng thời giúp tránh xung đột giữa các phụ thuộc.

Di động và Linh hoạt: Containers có thể chuyển động giữa các môi trường mà không cần sửa đổi mã nguồn hoặc cấu hình. Điều này tạo ra một môi trường phát triển, kiểm thử và triển khai đồng nhất.

Dễ quản lý: Docker cung cấp các công cụ quản lý và giám sát containers, giúp đơn giản hóa quá trình triển khai, cập nhật và mở rộng ứng dụng.

Dockerfile: Để tạo một container, bạn sử dụng Dockerfile, một tệp cấu hình văn bản mô tả cách Docker nên xây dựng image và định cấu hình container.

Docker containers giúp giải quyết các vấn đề liên quan đến khả năng di động, môi trường và sự tin cậy khi triển khai ứng dụng, đồng thời cung cấp một cách tiện lợi để đóng gói và chia sẻ ứng dụng.

**c. Docker Image**

Một Docker image là một gói đóng gói có chứa tất cả các thành phần cần thiết để chạy một ứng dụng, bao gồm cả mã nguồn, thư viện, môi trường thực thi, cấu hình, và các phụ thuộc khác. Image là một bản ghi chỉ đọc và không thay đổi. Containers được tạo ra bằng cách chạy một instance của một image.

Dưới đây là một số điểm chính về Docker image:

Có thể xây dựng từ Dockerfile: Docker image thường được xây dựng từ một tệp cấu hình văn bản gọi là Dockerfile. Dockerfile định nghĩa các bước cụ thể để cài đặt và cấu hình môi trường cần thiết để chạy ứng dụng.

Chia sẻ và Lưu trữ: Docker image có thể được chia sẻ và lưu trữ trong các registry như Docker Hub hoặc registry riêng tư. Registry là nơi lưu trữ và phân phối các images.

Đa phiên bản (Tag): Mỗi image có thể có nhiều phiên bản khác nhau được đánh dấu bằng các tag. Các tag cho phép bạn xác định phiên bản cụ thể của một image.

Chia sẻ môi trường đóng gói: Docker image giúp giải quyết vấn đề "It works on my machine" bằng cách đóng gói toàn bộ môi trường cần thiết để chạy ứng dụng. Điều này giúp đảm bảo tính di động và đồng nhất giữa môi trường phát triển, kiểm thử và triển khai.

Lợi ích về hiệu suất: Images làm cho việc chia sẻ và triển khai ứng dụng trở nên hiệu quả hơn, vì chúng chứa mọi thứ cần thiết và không phải cài đặt lại từ đầu trên mỗi máy chủ hoặc môi trường.

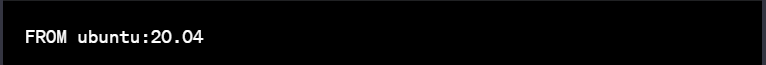
Khi bạn muốn triển khai một ứng dụng, bạn thường sẽ bắt đầu bằng việc tạo một Docker image từ một Dockerfile, sau đó sử dụng image này để tạo và chạy các Docker containers.

**d. Docker file**

Dockerfile là một tệp cấu hình văn bản chứa một loạt các chỉ thị (instructions) mà Docker Engine sử dụng để tự động xây dựng một Docker image. Dockerfile định nghĩa cách cài đặt và cấu hình môi trường cần thiết để chạy một ứng dụng trong một Docker container.

Một Dockerfile thường bao gồm các bước sau:

1. Chọn một base image: Bạn bắt đầu bằng việc chọn một base image từ Docker Hub hoặc một registry khác. Base image là một image có sẵn chứa hệ điều hành cơ bản và một số công cụ cơ bản.



2. Cài đặt các phụ thuộc: Bạn sẽ cài đặt các phần mềm, thư viện, và công cụ cần thiết cho ứng dụng của bạn.

A black screen with white text

Description automatically generated

3. Chuyển đến thư mục làm việc: Điều này làm thay đổi thư mục làm việc mà các lệnh tiếp theo trong Dockerfile sẽ được thực thi.



4. Sao chép mã nguồn vào container: Sao chép các tệp và thư mục từ máy host vào container.



5. Cấu hình môi trường: Đặt các biến môi trường và cấu hình môi trường cho ứng dụng.



6. Mở cổng: Nếu ứng dụng của bạn chạy trên các cổng cụ thể, bạn cần mở chúng để có thể truy cập từ bên ngoài.



7. Chạy lệnh khởi động ứng dụng: Cuối cùng, bạn sẽ chỉ định lệnh hoặc lệnh mặc định để chạy khi container được khởi động.



Mỗi lệnh trong Dockerfile tạo ra một layer trong Docker image, và những layer này có thể được sử dụng lại và chia sẻ, giúp tối ưu hóa quá trình xây dựng, triển khai và quản lý ứng dụng.

**e. Registry**

Trong Docker, một registry là nơi lưu trữ và quản lý các Docker images. Nó là một dịch vụ lưu trữ trực tuyến giúp bạn chia sẻ, quản lý, và triển khai các images Docker. Một registry có thể là một registry công cộng như Docker Hub hoặc một registry riêng tư mà bạn triển khai và quản lý.

Dưới đây là một số điểm quan trọng về registry trong Docker:

Docker Hub: Là một registry công cộng và lớn nhất, Docker Hub cho phép người dùng chia sẻ và tìm kiếm các images Docker công khai. Bạn có thể tìm thấy nhiều images chất lượng từ cộng đồng Docker trên Docker Hub.

Registry Riêng Tư: Nếu bạn muốn giữ các images riêng tư, bạn có thể triển khai một registry riêng tư trên môi trường của mình. Các lựa chọn phổ biến bao gồm Docker Registry, Harbor, và Azure Container Registry, giúp bạn lưu trữ và quản lý images trong môi trường kín.

Quản lý Phiên bản và Tag: Mỗi image được lưu trữ trong registry được đánh dấu bằng một phiên bản và tag. Phiên bản thường là một số nguyên dương (ví dụ: 1, 2, 3), trong khi tag là một nhãn mô tả phiên bản cụ thể (ví dụ: latest).

Tích hợp với Quy trình Phát triển và Triển khai: Registry chơi một vai trò quan trọng trong quy trình phát triển và triển khai với Docker. Images được xây dựng, đánh dấu và lưu trữ trong registry, sau đó có thể được triển khai từ đó trực tiếp lên các môi trường khác nhau.

Quyền truy cập và Bảo mật: Các registry thường cung cấp cơ chế bảo mật để kiểm soát quyền truy cập vào images. Bạn có thể cấu hình quyền truy cập công khai hoặc riêng tư, và đôi khi yêu cầu xác thực để tải và sử dụng các images.

Docker registry đóng vai trò quan trọng trong việc quản lý và phân phối images Docker, đồng thời giúp tối ưu hóa quá trình phát triển và triển khai ứng dụng.

**f. Docker Compose**

Docker Compose là một công cụ trong hệ sinh thái Docker được sử dụng để đơn giản hóa quá trình quản lý và triển khai ứng dụng đa-container. Nó cho phép bạn định nghĩa và quản lý cấu trúc của ứng dụng cùng với các dịch vụ, mạng, và các cài đặt khác trong một tệp cấu hình đơn.

Dưới đây là một số điểm chính về Docker Compose:

YAML Configuration File: Docker Compose sử dụng một tệp cấu hình YAML để định nghĩa cấu trúc của ứng dụng. Trong tệp này, bạn có thể mô tả các dịch vụ, các môi trường, các cài đặt mạng và các yếu tố khác cần thiết để triển khai ứng dụng.

Ví dụ đơn giản của một tệp docker-compose.yml:



Dịch vụ (Services): Mỗi dịch vụ trong Docker Compose đại diện cho một container và các cài đặt liên quan. Điều này giúp bạn đơn giản hóa việc quản lý nhiều container cùng một lúc.

Mạng (Network): Bạn có thể định nghĩa các mạng riêng để kết nối các container trong ứng dụng của mình. Điều này giúp kiểm soát và tối ưu hóa giao tiếp giữa các dịch vụ.

Tích hợp với Docker CLI: Docker Compose tích hợp chặt chẽ với Docker CLI, giúp bạn quản lý container và các tác vụ khác một cách dễ dàng. Bạn có thể sử dụng các lệnh như docker-compose up, docker-compose down, docker-compose ps, và nhiều lệnh khác.

Biến Môi trường và Cấu hình: Bạn có thể đặt các biến môi trường và cấu hình trong tệp cấu hình, giúp bạn linh hoạt khi triển khai ứng dụng trong các môi trường khác nhau.

Quản lý Volumes và Mounts: Docker Compose cung cấp cách dễ dàng để quản lý volumes và mounts, giúp bạn lưu trữ dữ liệu và chia sẻ nó giữa các container.

Docker Compose giúp tối ưu hóa và tự động hóa quá trình triển khai ứng dụng đa-container, đặc biệt là khi có nhiều dịch vụ và các phụ thuộc phức tạp.

**g. Orchestration**

Orchestration trong Docker là quá trình quản lý và điều phối một số lượng lớn các containers để triển khai và chạy các ứng dụng phức tạp. Khi bạn có một hệ thống với nhiều containers tương tác và cần được triển khai và quản lý một cách đồng bộ, orchestration trở nên quan trọng.

Docker không cung cấp một giải pháp orchestration tích hợp mặc định, nhưng có nhiều công cụ nổi tiếng thực hiện công việc này. Một số công cụ orchestration phổ biến bao gồm:

Kubernetes: Là một hệ thống mã nguồn mở được phát triển bởi Google. Kubernetes cung cấp các tính năng mạnh mẽ cho việc triển khai, quản lý, và mở rộng các containers. Nó hỗ trợ cả quy trình triển khai, cân bằng tải, giám sát, và quản lý phiên bản.

Docker Swarm: Là một công cụ orchestration tích hợp của Docker, giúp quản lý một cluster của các máy chủ Docker và tự động triển khai ứng dụng trên chúng. Docker Swarm giúp đơn giản hóa việc triển khai và quản lý các containers.

Amazon ECS (Elastic Container Service): Dịch vụ quản lý containers của Amazon Web Services (AWS) cung cấp khả năng orchestration cho việc triển khai và quản lý các containers trên nền tảng AWS.

OpenShift: Một nền tảng orchestration được xây dựng trên Kubernetes và được hỗ trợ bởi Red Hat. OpenShift cung cấp một số tính năng bổ sung như DevOps và CI/CD tích hợp.

Các công cụ orchestration giúp tự động hóa quá trình triển khai, quản lý tài nguyên, cân bằng tải, và mở rộng ứng dụng. Chúng giúp giảm độ phức tạp và tăng khả năng mở rộng của hệ thống, đồng thời cung cấp các tính năng như phục hồi từ lỗi và môi trường phát triển, thử nghiệm, và triển khai liên tục (CI/CD).

**h. Volume**

Trong Docker, volume là một cách để lưu trữ và quản lý dữ liệu giữa host và container hoặc giữa các container khác nhau. Volume cung cấp một cách để giữ cho dữ liệu tồn tại và không bị mất khi container dừng lại hoặc bị xóa.

Dưới đây là một số điểm quan trọng về volumes trong Docker:

Dữ liệu Persistent: Volume giúp giữ cho dữ liệu tồn tại và persistent, ngay cả khi container của bạn được xóa hoặc khởi động lại.

Chia sẻ Dữ liệu giữa Container: Volume cho phép bạn chia sẻ dữ liệu giữa các container khác nhau. Điều này hữu ích khi bạn muốn một số container cùng chia sẻ dữ liệu, chẳng hạn như dữ liệu cấu hình hoặc dữ liệu cơ sở dữ liệu.

Tích hợp với Local File System: Volume có thể được liên kết với một thư mục trên local file system của host, hoặc chúng có thể được quản lý bởi Docker và lưu trữ ngoài container.

Định nghĩa trong Dockerfile hoặc Docker Compose: Bạn có thể định nghĩa volumes trong Dockerfile để kết nối với các container, hoặc bạn có thể sử dụng Docker Compose để quản lý volumes.

Các Lệnh Docker CLI: Các lệnh Docker CLI như docker volume create, docker volume ls, và docker volume rm cho phép bạn quản lý volumes trực tiếp từ command line.

Dưới đây là một ví dụ cơ bản về cách sử dụng volumes trong Docker:

bash

Copy code

# Tạo một volume



# Chạy container với volume được kết nối



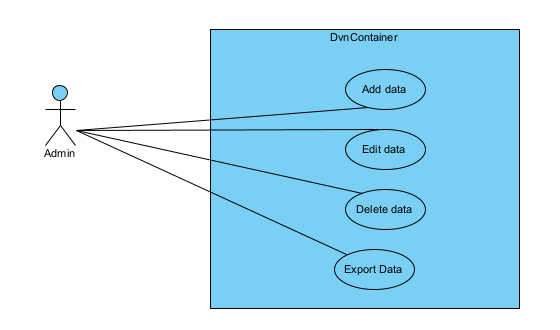
# Khi container dừng lại hoặc bị xóa, dữ liệu vẫn tồn tại trong volume "mydata"

Volumes là một phần quan trọng trong quá trình phát triển và triển khai ứng dụng Docker, đặc biệt là khi bạn cần lưu trữ và chia sẻ dữ liệu giữa các container và giữa container và host.

Chương II. Các kết quả đạt được

1. Đặc tả UseCase và sơ đồ phân rã

**1.1. Devaning Container**



Hình 1: UseCase - UpkLot

|  |  |
| --- | --- |
| UseCase Name | DevaningContainer-Add data |
| **Description** | Chức năng này cho phép người dùng thêm mới dữ liệu của Container |
| Actor | Account admin, Operator |
| Pre-Condition | Tài khoản người dùng được phân quyền  Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Thông báo tạo bản ghi mới thành công và trả về dữ liệu mới nhất |
| Basic Flow | 1. Chọn chức năng ‘Add’ trong màn hình Devaning Container  2. Nhập đủ thông tin và nhấn ‘Save’.  3. Hệ thống ghi nhận dữ liệu và trả lại dữ liệu mới nhất vào tab New |
| Exception Flow | * Thông tin số Container không hợp lệ |

|  |  |
| --- | --- |
| UseCase Name | DevaningContainer – Edit data |
| **Description** | Chức năng này cho phép người dùng sửa dữ liệu của Container |
| Actor | Account Admin, Operator |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | * Thông báo update bản ghi thành công và trả về giá trị mới nhất |
| Basic Flow | 1. Chọn một bản ghi cần cập nhập và dung chuột phải để thực hiện ‘Edit’ bản ghi trong màn hình Devaning Container  2. Nhập thông tin và nhấn ‘Save’.  3. Hệ thống ghi nhận dữ liệu và trả lại dữ liệu mới nhất |
| Exception Flow | * Thông tin số Container không hợp lệ |

|  |  |
| --- | --- |
| UseCase Name | DevaningContainer – Delete data |
| **Description** | Chức năng này cho phép người dùng xóa dữ liệu của Container |
| Actor | Account Admin, Operator |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | * Thông báo xóa bản ghi thành công và trả về giá trị mới nhất |
| Basic Flow | 1. Chọn một bản ghi cần xóa và dung chuột phải để thực hiện ‘Delete’ trong màn hình Devaning Container  2. Xác nhận và chọn ‘OK’  3. Hệ thống ghi nhận dữ liệu và trả lại dữ liệu mới nhất |
| Exception Flow | * Thông tin số container không hợp lệ |

|  |  |
| --- | --- |
| UseCase Name | DevaningContainer – Export data |
| **Description** | Chức năng này cho phép người dùng xuất báo cáo, thống kê container |
| Actor | Account Admin, Operator, Supervisor |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | * Thông báo xuất bản thành công và trả về giá trị mới nhất và trả về file Excel chứa thông tin |
| Basic Flow | 1. Chọn ‘Export’ trên màn hình Devaning Container  2. Hệ thống ghi nhận dữ liệu và trả lại file excel chứa dữ liệu |
| Exception Flow | * Lỗi hệ thống |

**1.2. Màn hình dỡ Container (DevaningScreen)**

A blue square with black text

Description automatically generated

Hình 2: UseCase - DevaningScreen

|  |  |
| --- | --- |
| UseCase Name | DevaningScreen-Update |
| **Description** | Chức năng này cho phép người dùng có thể theo dõi và xử lý được trạng thái của các container |
| Actor | Account Admin, Operator |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | * Cập nhật trạng thái của Container hiện tại và tiếp sau đó |
| Basic Flow | 1. Các Container có Status Ready được lập kế hoạch trong ngày hiện trong bảng và được Submit vào quá trình Devaning và thay đổi Status thành Devaning 2. Container trong quá trình Devaning sẽ được hoàn tất sau khi tiến trình đạt 100% hoặc nhấn vào chức năng Finish   3. Xác nhận đúng Container và đồng ý  4. Hệ thống xác thực cập nhật trạng thái kết thúc quá trình dỡ của container hiện tại và chuyển sang bắt đầu dỡ cho container tiếp theo có thời gian mở đã được câp nhật trước, đối với Container chưa cập nhật thời gian mở thì người dùng sẽ thao tác chọn nút submit để cập nhật bản ghi đó vào quá trình mở |
| Exception Flow | * Thông tin container không hợp lệ |

|  |  |
| --- | --- |
| UseCase Name | DevaningScreen-Tracking |
| **Description** | Chức năng này cho phép người dùng theo dõi được quá trình dỡ hàng, trạng thái của các container |
| Actor | Account Admin, Operator, Supervisor |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition |  |
| Basic Flow |  |
| Exception Flow | Lỗi hệ thống |

**1.3. UpkLot**

A diagram of a data processing process

Description automatically generated

Hình 3: UseCase - UpkLot

|  |  |
| --- | --- |
| UseCase Name | UpkLot-ExportData |
| **Description** | Chức năng này cho phép người dùng xuất báo cáo , thống kê về số lượng module sau khi dỡ (Devan) |
| Actor | Account Admin, Operator, Supervisor |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Thông báo thành công và trả về file Excel chứa thông tin |
| Basic Flow | 1. Chọn chức năng ‘Export’ trên màn hình UpkLot  2. Hệ thống xác thực và trả về kết quả. |
| Exception Flow | * Lỗi hệ thống |

|  |  |
| --- | --- |
| UseCase Name | UpkLot-Delete Data |
| **Description** | Chức năng này cho phép người dùng xóa dữ liệu của module |
| Actor | Account Admin, Operator |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Thông báo thành công và trả về dữ liệu mới sau khi cập nhật |
| Basic Flow | 1. Chọn một bản ghi cần xóa và chọn chức năng ‘Delete trên màn hình UpkLot  2. Hệ thống xác thực và trả về kết quả. |
| Exception Flow | * Giá trị module không hợp lệ |

**1.4. UpkPartList**

A blue rectangular object with a black text

Description automatically generated

Hình 4: UseCase - Export

|  |  |
| --- | --- |
| UseCase Name | UpkPartList – Export |
| **Description** | Chức năng này cho phép người dùng xuất báo báo, thống kê về số lượng linh kiện, phụ kiện |
| Actor | Account Admin, Operator, Supervisor |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Xuất ra file Excel thông tin linh kiện, phụ kiện |
| Basic Flow | 1. Chọn chức năng ‘Export’ trên màn hình UpkPartList  2. Hệ thống xác thực thông tin và xuất ra file Excel phù hợp |
| Exception Flow | * Lỗi hệ thống |

**1.5. UpkScreen**

A blue rectangular object with black text

Description automatically generated

Hình 5: UseCase UpkScreen

|  |  |
| --- | --- |
| UseCase Name | UpkScreen-Add Part to PC |
| **Description** | -Chức năng này cho phép người dùng phân loại Part vào các kho (PC Store và PC Home)  -Dựa vào nhà cung cấp (Supplier) và số Renban để xác định được linh kiện, phụ kiện nào sẽ vào các kho tương ứng |
| Actor | Account Admin, Operator |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Hiển thị thông tin những PT của các đơn hàng đặt cho lệnh đã nhận mà chưa xuất |
| Basic Flow | 1. Chọn ‘Finish’ trên từng bản ghi trên màn hình UpkScreen để phân loại từng loại hàng hoặc có thể chọn ‘Finish’trên góc màn hình để chuyển trạng thái tất cả linh kiện, phụ kiện (Part) có trong Module đó  2. Hệ thống xác thực thông tin phân loại linh kiện, phụ kiện vào các kho sao cho hợp lý |
| Exception Flow | * Linh kiện không tồn tại * Hệ thống lỗi |

|  |  |
| --- | --- |
| UseCase Name | UpkScreen-Add Part to Robing |
| **Description** | -Chức năng này cho phép người dùng phân loại Part vào khu phân loại hàng hỏng (Robing) |
| Actor | Account Admin |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Thông báo thành công |
| Basic Flow | 1. Chọn chức năng ‘Robing’ trên mỗi bản ghi  2. Hệ thống xuất hiện popup, chọn loại hàng để phân loại. ‘FAILED’ được phân loại là hàng hỏng, ‘LOAN’ được phân loại là hàng cho mượn  3. Kiểm tra thông tin và chọn xác nhận ‘OK’  4. Hệ thống kiểm tra và chuyển linh kiện, phụ kiện đó vào khu phân loại hàng hỏng (Robing) |
| Exception Flow | * Thông tin linh kiện, phụ kiện không hợp lệ * Hệ thống lỗi |

**1.6. Robing**

A diagram of export and export

Description automatically generated

Hình 6: UseCase - Robing

|  |  |
| --- | --- |
| UseCase Name | Robing-Export |
| **Description** | Chức năng này cho phép người dùng xuất báo cáo, thống kê về số lượng, chi tiết hàng hỏng, hàng lỗi, hàng không sử dụng được,.. |
| Actor | Account Admin |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Xuất ra file Excel chứa thông tin phù hợp |
| Basic Flow | 1. Chọn chức năng ‘Export’ trên màn hình Robing  2. Hệ thống trả về file Excel phù hợp |
| Exception Flow | * Lỗi hệ thống |

|  |  |
| --- | --- |
| UseCase Name | Robing-Request Give Back |
| **Description** | Chức năng này cho phép người dùng có thể gửi yêu cầu để xác nhận phương pháp xử lý hàng hỏng |
|  |  |
| Actor | Account Admin |
| Pre-Condition | -Tài khoản người dùng được phân quyền  -Thiết bị được kết nối Internet khi thực hiện |
| Post-Condition | -Thông báo gửi yêu cầu thành công |
| Basic Flow | 1. Chọn linh kiện hỏng cần yêu cầu xử lý, chọn ‘Request To Back’  2. Hệ thống gửi yêu cầu lên hệ thống, chờ phê duyệt |
| Exception Flow | * Lỗi hệ thống * Linh kiện không hợp lệ |

## 2. Sơ đồ Sequencece

2.1. Đăng nhập

A diagram of a diagram

Description automatically generated

Hình 7: Sơ đồ Sequence - Đăng nhập

**2.2. Tìm kiếm**

A diagram of a diagram

Description automatically generated

Hình 8: Sơ đồ Sequence - Tìm kiếm

**2.3. Báo cáo**

A diagram of a diagram

Description automatically generated

Hình 9: Sơ đồ Sequence - Báo cáo

**2.4. Thêm mới DvnContList**

A diagram of a diagram

Description automatically generated

Hình 10: Sơ đồ Sequence Thêm mới DvnContList

**2.5. Cập nhật DvnContList**

A diagram of a project

Description automatically generated

Hình 11: Sơ đồ Sequence cập nhật DvnContList

**2.6. Xóa DvnContList**

A diagram of a diagram

Description automatically generated

Hình 12: Sơ đồ Sequence xóa DvnContList

**2.7. Xuất báo cáo DvnContList**

A diagram of a flowchart

Description automatically generated

Hình 13: Sơ đồ Sequence - Xuất báo cáo DvnContList

**2.8. Theo dõi tiến trình DvnContList**

A diagram of a diagram

Description automatically generated

Hình 14: Sơ đô Sequence theo dõi tiến trình DvnContList

**2.9. Dỡ Module (Unpacking module)**

A diagram of a project

Description automatically generated

Hình 15: Sơ đồ Sequence Dỡ Module

**2.10. Kết thúc dỡ Module (Unpacking Module Finish)**

A diagram of a diagram

Description automatically generated

Hình 16: Sơ đồ Sequence kết thúc dỡ Module

**2.11. Yêu cầu hàng hỏng (Robing Request)**

A diagram of a project

Description automatically generated

Hình 17: Sơ đồ Sequence Yêu cầu hàng hỏng

**2.12. Yêu cầu xử lý hàng hỏng (Robing Request Give Back)**

A diagram of a technical diagram

Description automatically generated

Hình 18: Sơ đồ Sequence yêu cầu xử lý hàng hỏng

3. Sơ đồ Activity

**3.1. Thêm mới DvnContainer**

A diagram of a flowchart

Description automatically generated

Hình 19: Sơ đồ Activity thêm mới DvnContainer

**3.2. Chỉnh sửa DvnContainer**

A diagram of a flowchart

Description automatically generated

Hình 20: Sơ đồ Activity chỉnh sửa DvnContainer

**3.3. Xóa DvnContainer**

A diagram of a software system

Description automatically generated with medium confidence

Hình 21: Sơ đồ Activity xóa DvnContainer

**3.4. Xuất báo cáo DvnContainer**

A diagram of a system

Description automatically generated

Hình 22: Sơ đồ Activity xuất báo cáo DvnContainer

**3.5. Kết thúc dỡ Module**

A diagram of a software project

Description automatically generated

Hình 23: Sơ đồ Activity kết thúc dỡ Module

**3.6. Xóa UpkLot**

A diagram of a system

Description automatically generated

Hình 24: Sơ đồ Activity xóa UpkLot

**3.7. Xuất báo cáo UpkLot**

A diagram of a system

Description automatically generated

Hình 25: Sơ đồ Activity xuất báo cáo UpkLot

**3.8. Xuất báo cáo UpkPartList**

A diagram of a system

Description automatically generated

Hình 26: Sơ đồ Activity xuất báo cáo UpkPartList

**3.9. Yêu cầu xử lý hàng hỏng**

A diagram of a software system

Description automatically generated

Hình 27: Sơ đồ Activity yêu cầu xử lý hàng hỏng

4. Cơ sở dữ liệu

**4.1. Các bảng**

A table with text and numbers

Description automatically generated

Hình 28: Bảng DvnContList

A table with text and numbers

Description automatically generated

Hình 29: Bảng LupContModule

A table with text and numbers

Description automatically generated

Hình 30: Bảng Part

A blue and black box with black text

Description automatically generated

Hình 31: Bảng PcStore

A blue and white table with black text

Description automatically generated

Hình 32: Bảng PcHome

A table with text on it

Description automatically generated

Hình 33: Bảng Robing

* Ngoài ra còn có các Bảng AbpTenant (Thông tin đại lý), AbpUser (Thông tin người dùng), AbpRole (Bảng quyền), AbpUserRole (Bảng quyền user)

**4.2. Cơ sở dữ liệu**

*+* Bảng AbpTenant (Thông tin đại lý), AbpUser (Thông tin người dùng), AbpRole (Bảng quyền), AbpUserRole (Bảng quyền user)

* Tất cả các bảng đều có quan hệ với bảng AbpTenant có TenantId là khóa ngoại
* Cơ sở dữ liệu theo các chức năng chính:

**4.3. Sơ đồ Class**

A diagram of a software company

Description automatically generated with medium confidence

Hình 34: Sơ đồ class

**4.4. Sơ đồ Interface**

A diagram of a computer

Description automatically generated

Hình 35: Sơ đồ Interface

**4.5. Sơ đồ UseCase- Mô tả các chức năng của hệ thống:**

A blue screen with many ovals

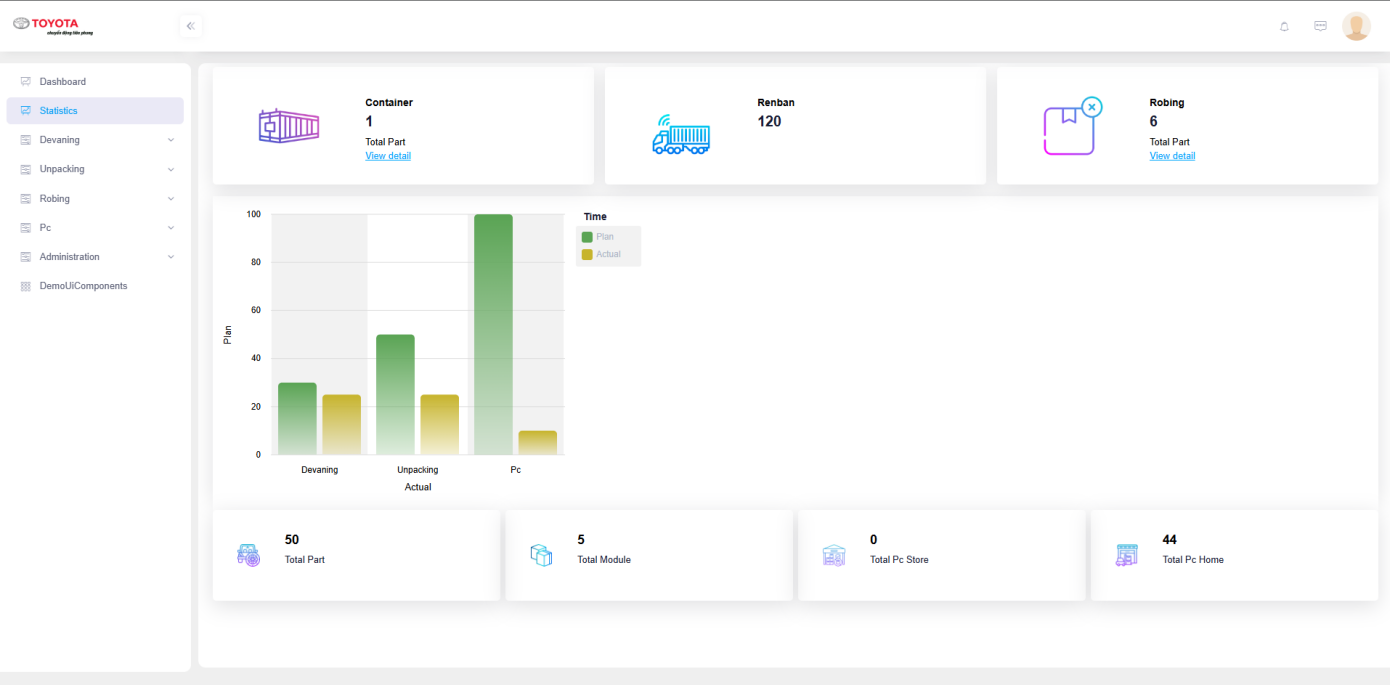
Description automatically generated

Hình 36: Sơ đồ UseCase

5. Giao diện hệ thống

### **5.1. Statistics**

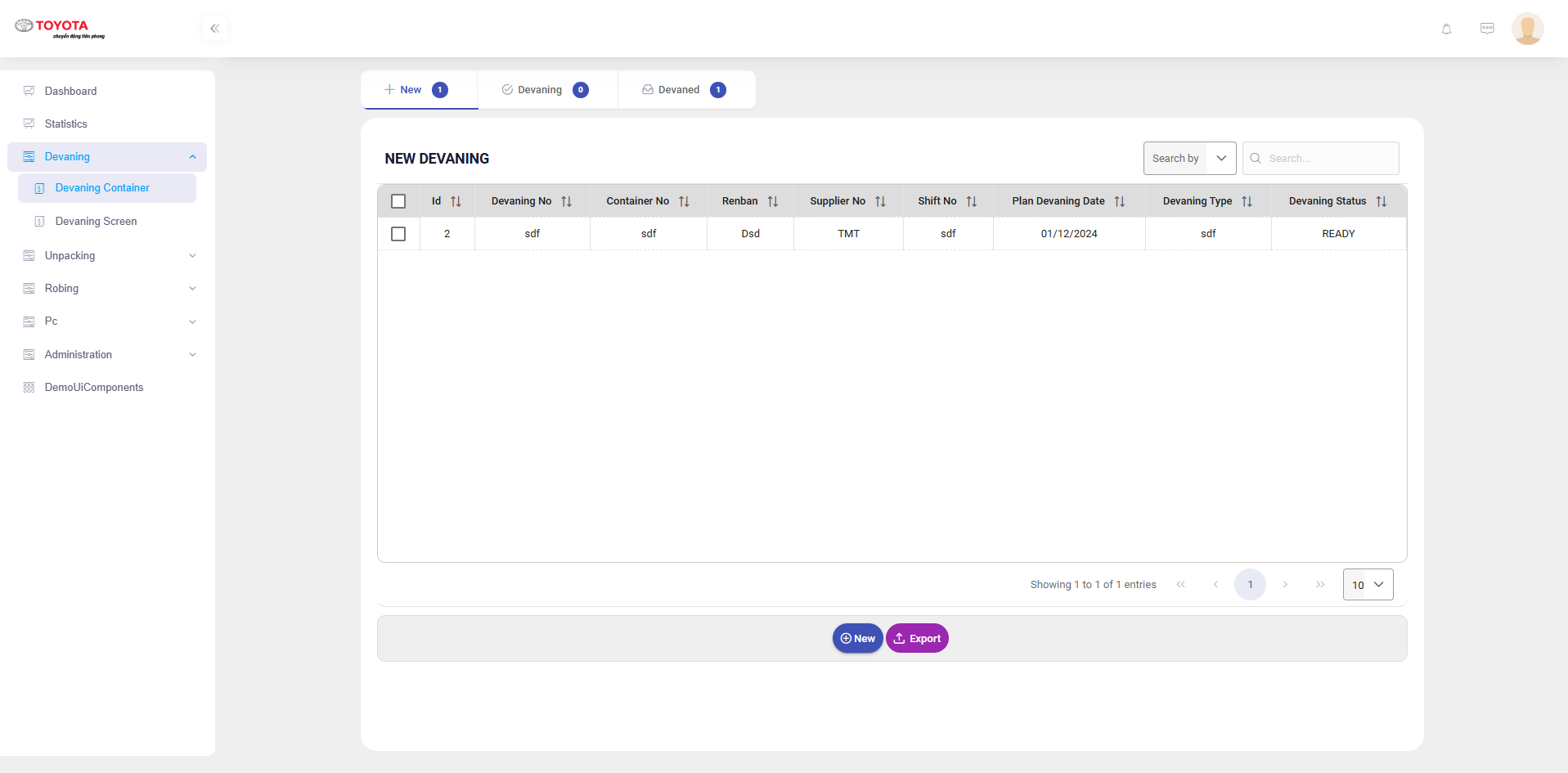
* Chức năng: Hiển thị tất cả các chức năng của hệ thống. Người dung có thể thao tác như thao tác trên các chức năng riêng lẻ.
* Đối tượng sử dụng : Account Admin



Hình 37: Giao diện Statistics

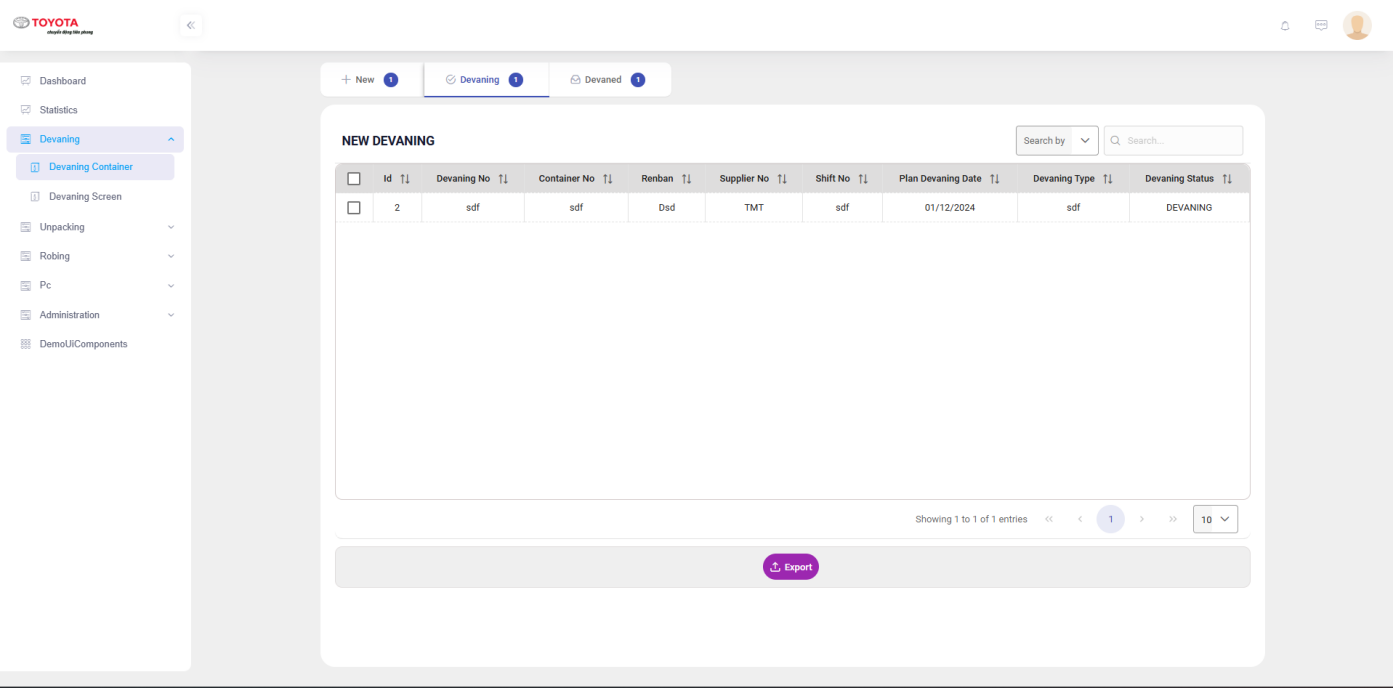
### **5.2. DvnContainer**

5.2.1 Tab New



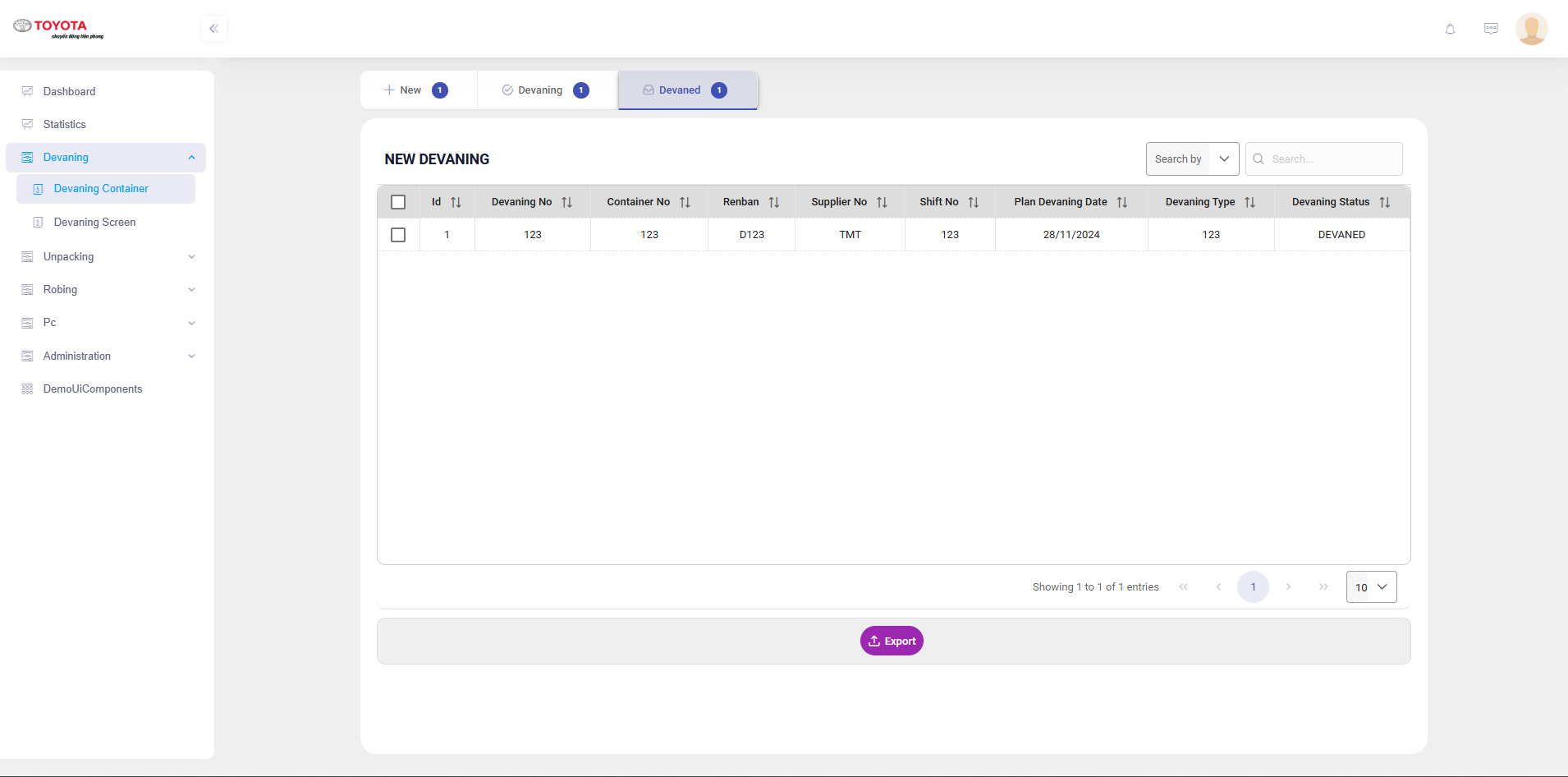
Hình 38: Giao diện DvnContainer - Tab New

5.2.2 Tab Devaning



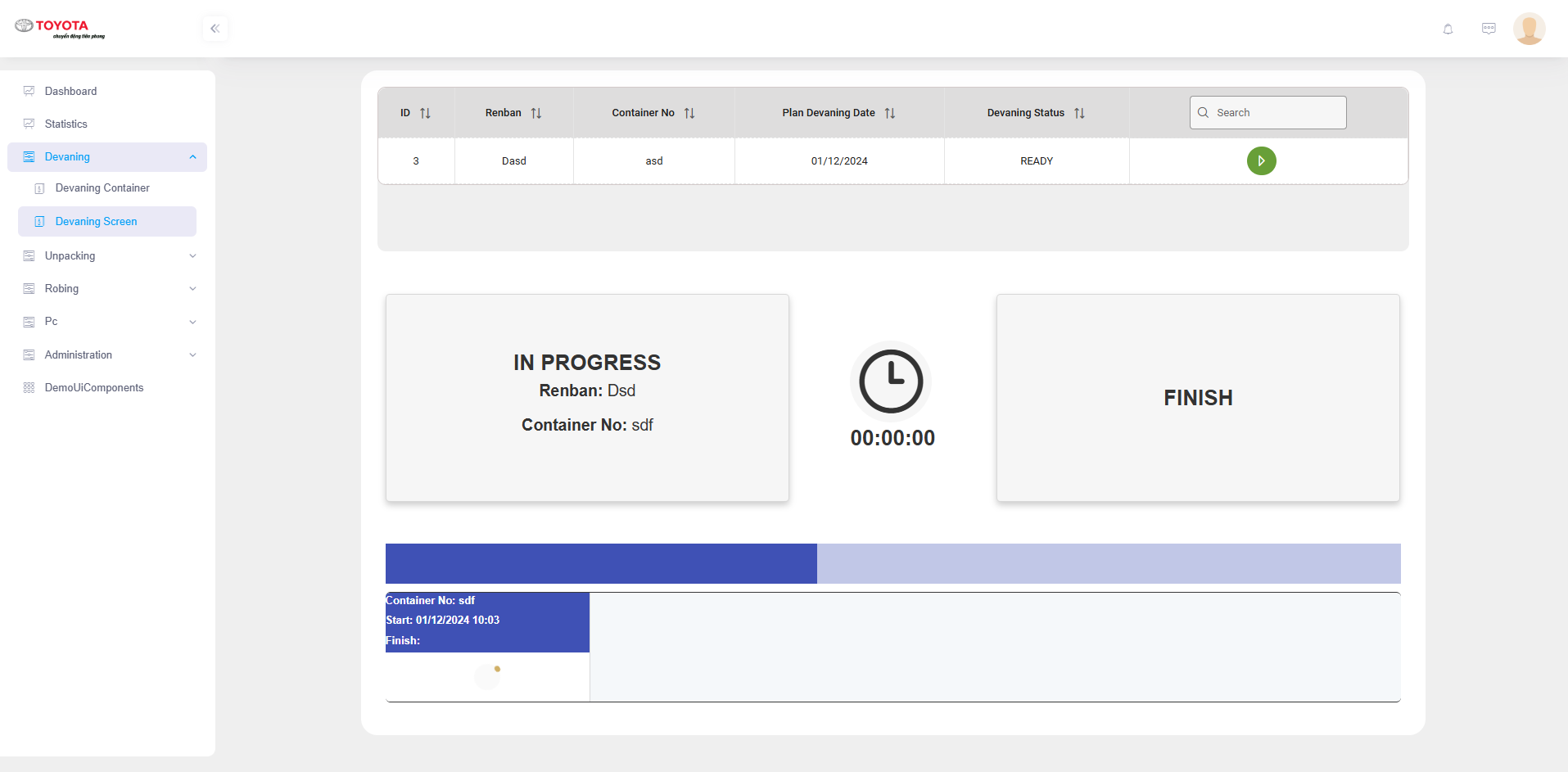
Hình 38.1: Giao diện DvnContainer - Tab Devaning

5.2.3 Tab Devaned



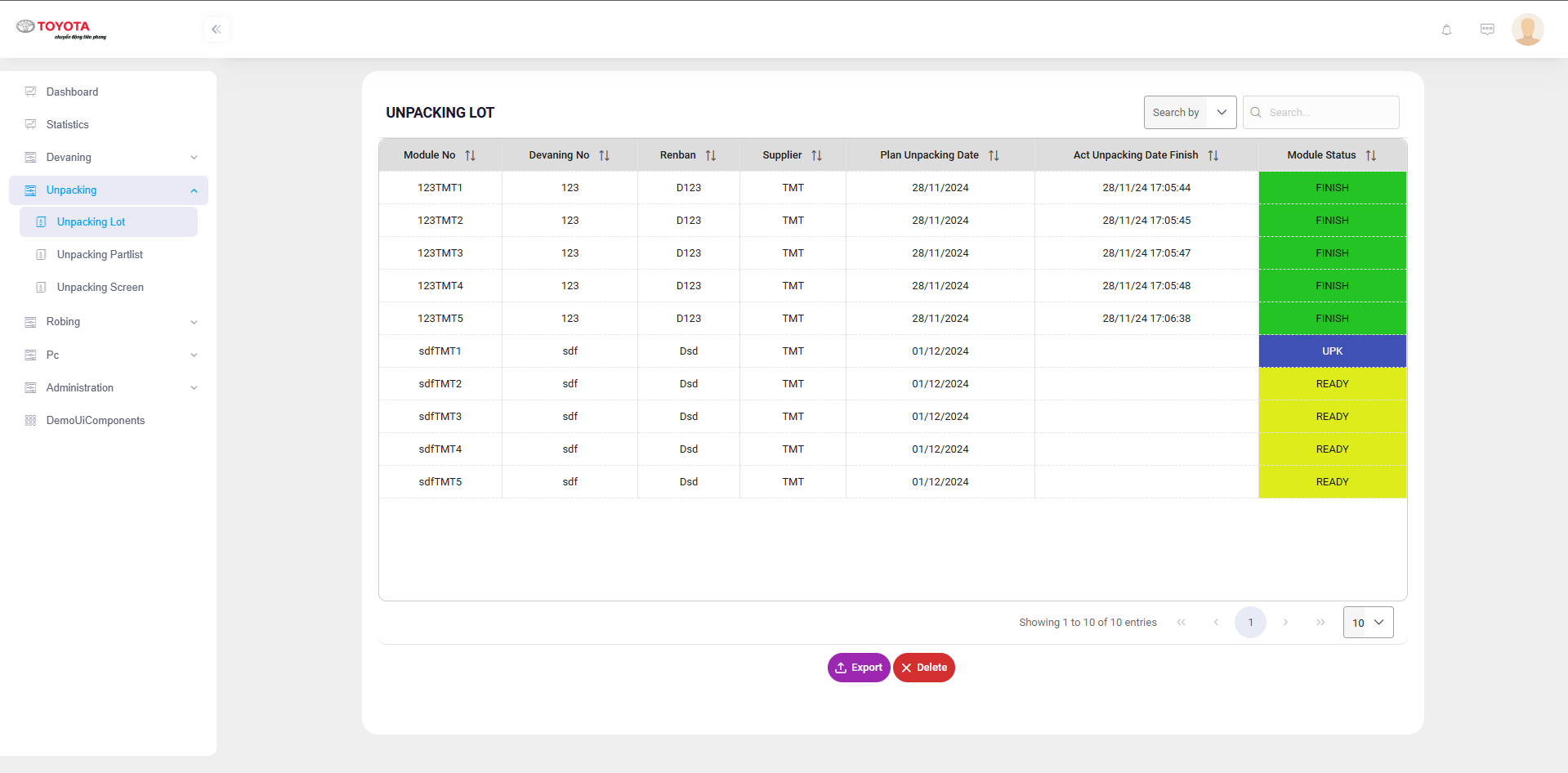
*Hình 38.2: Giao diện DvnContainer - Tab Devaned*

### **5.3. DvnScreen**



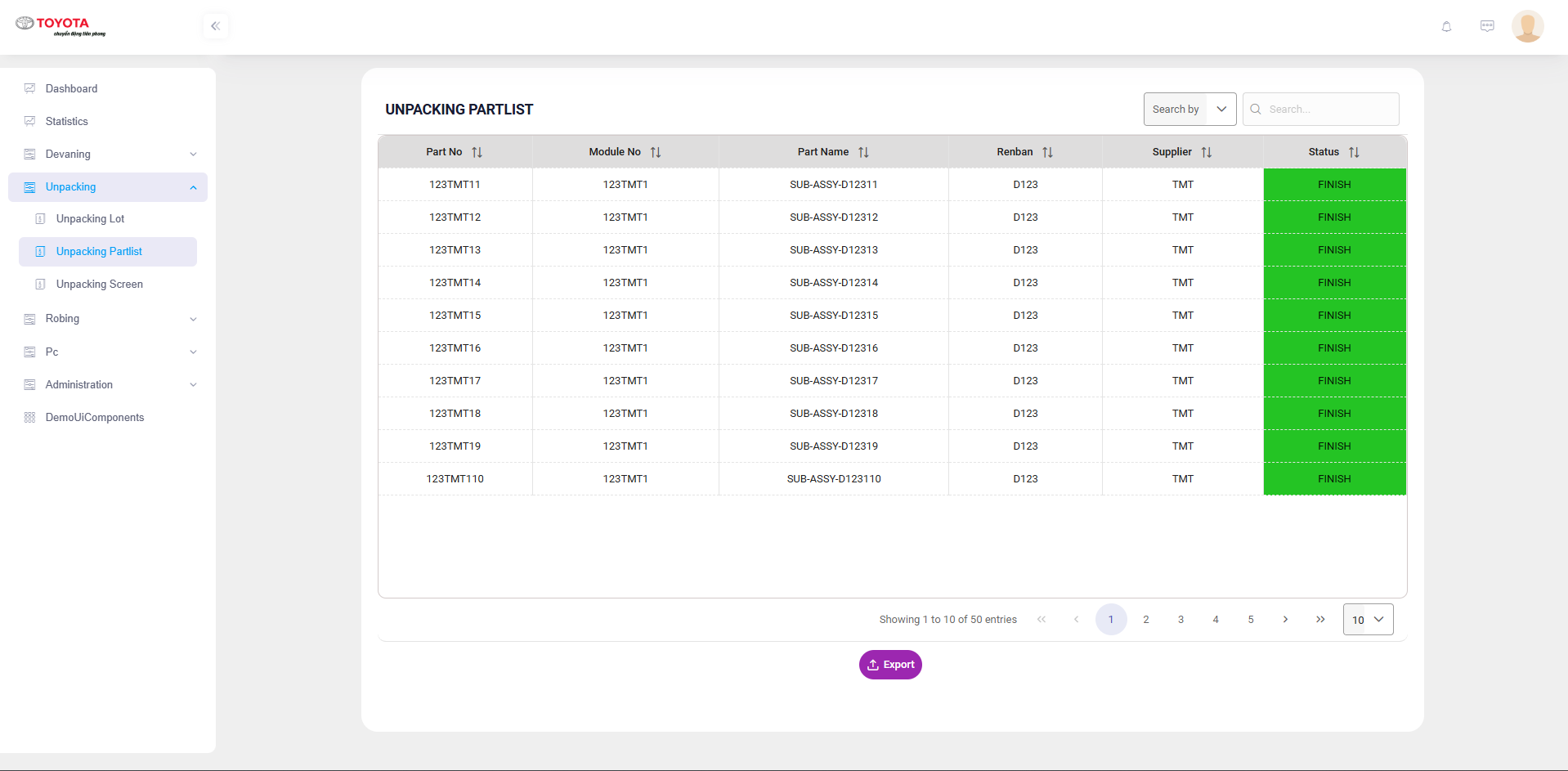
Hình 39: Giao diện DvnScreen

**5.4. UpkLot**



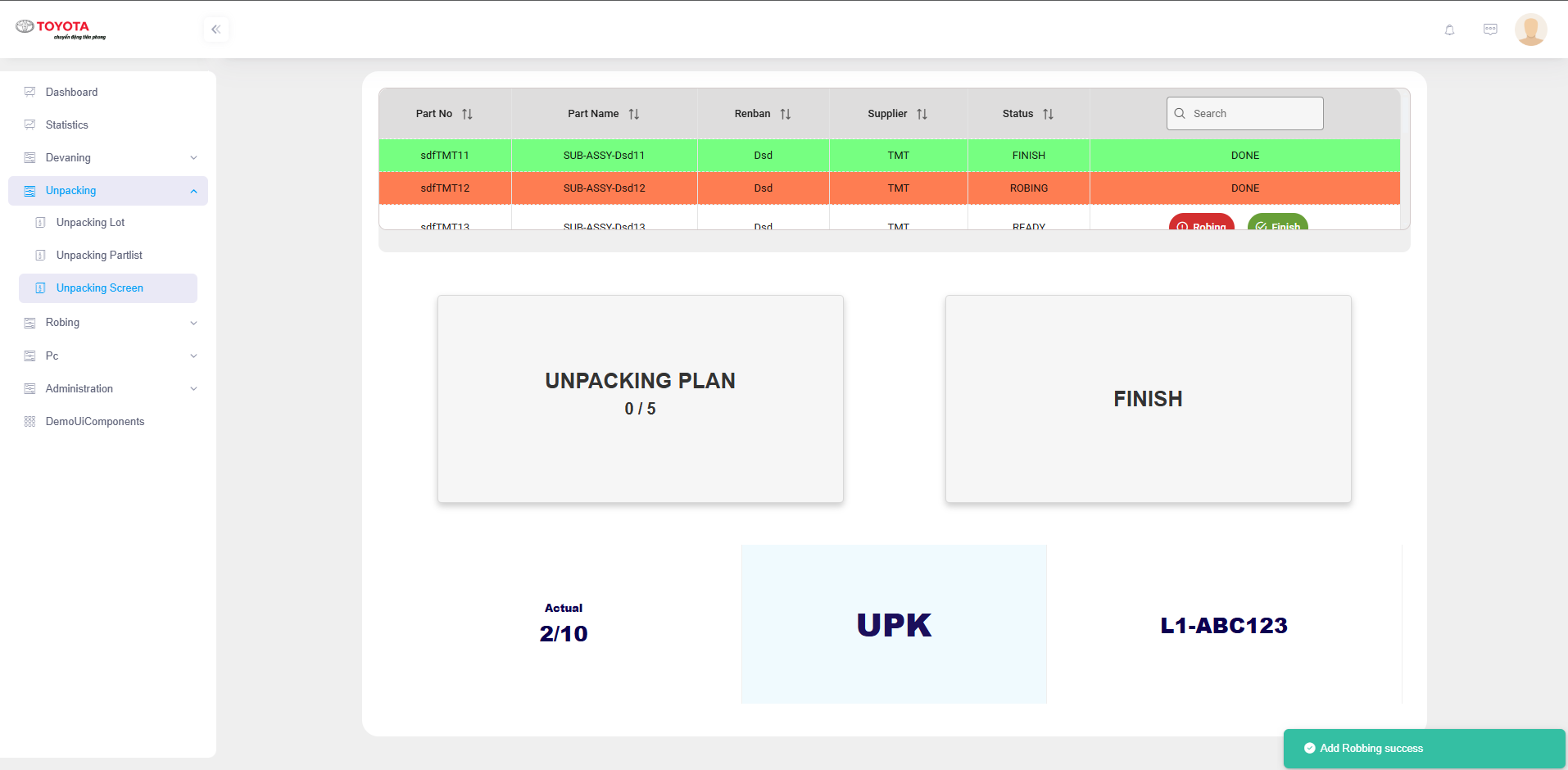
Hình 40: Giao diện UpkLot

**5.5. UpkPartList**



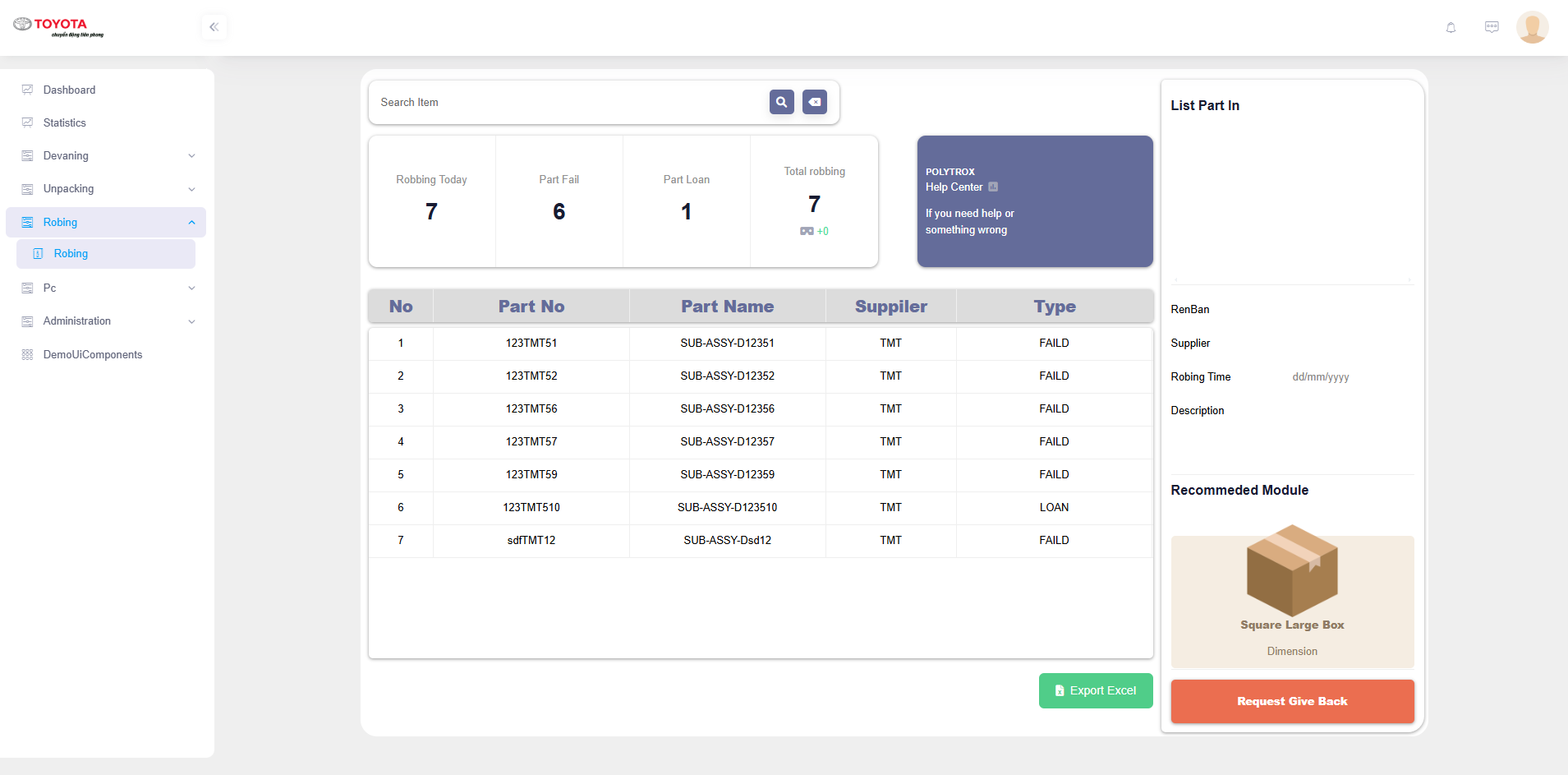
Hình 41: Giao diện UpkPartList

**5.6. UpkScreen**



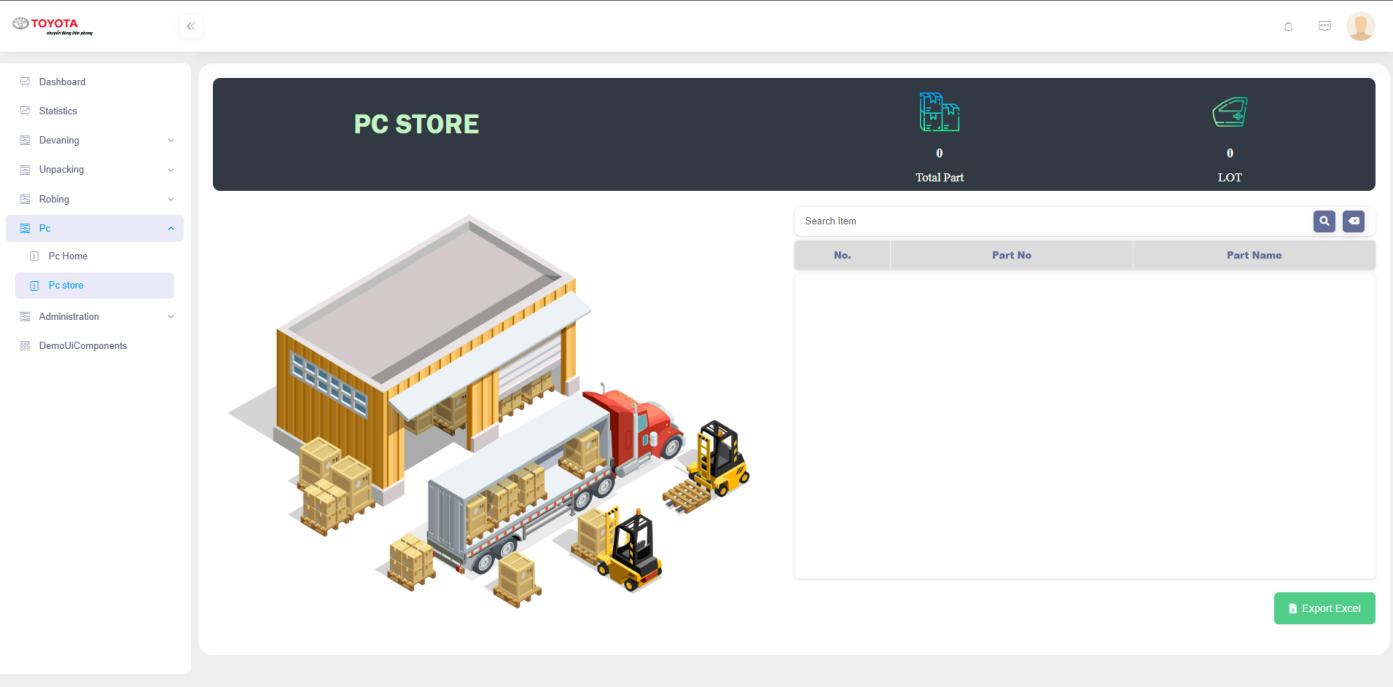
Hình 42: Giao diện UpkScreen

**5.7. Robing**



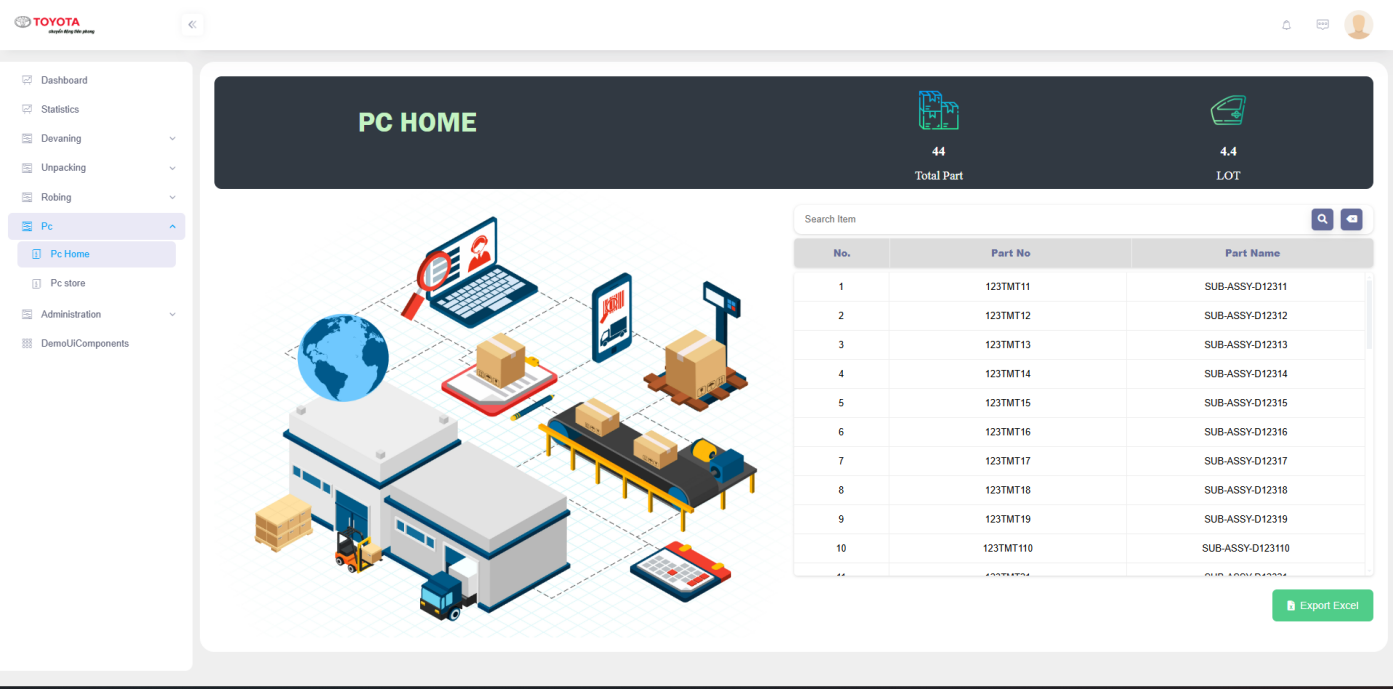
Hình 43: Giao diện Robing

**5.8. PcStore**



Hình 44: Giao diện PcStore

**5.9. PcHome**



Hình 45: Giao diện PcHome

Chương III. Phân quyền

1. Vai trò người dùng
2. Admin

Admin có toàn quyền trong hệ thống (cấu hình, quản lý quyền, giám sát).

1. Operator

Thực hiện các thao tác tại các màn như Devaning, Unpacking, hoặc Robing.

1. Supervisor

Giám sát và xem các màn hình, nhưng không được chỉnh sửa dữ liệu.

1. QC (Quality Control)

Kiểm tra chất lượng, thường tương tác với PC hoặc các báo cáo kiểm tra.

1. Quyền theo màn hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Màn hình | Admin | Supervisor | Operator | QC |
| Devaning | Full Access | View Only | Modify | - |
| Unpacking | Full Access | View Only | Modify | - |
| Robing | Full Access | View Only | Modify | - |
| PC | Full Access | View Only | Modify | Modify/View |

Chương IV. Kết luận

1. Đánh giá ưu nhược điểm

a) Ưu điểm :

+ Hệ thống đã đáp ứng được các yêu cầu nghiệp vụ thực tế

+ Giao diện thân thiện với người dùng

+ Giảm thời gian và công sức cần thiết cho việc tháo dỡ linh kiện

+ Tối ưu hóa quy trình làm việc, giảm thiểu thời gian chờ đợi và tăng năng suất làm việc của nhân viên.

+ Hệ thống giúp đảm bảo rằng mỗi bước tháo dỡ linh kiện được thực hiện đúng theo quy trình.

+ Ngăn chặn sai sót do con người, giảm rủi ro hỏng hóc và tăng cường đồng nhất trong sản xuất.

+ Theo dõi chính xác vị trí và số lượng linh kiện trong kho.

+ Tối ưu hóa lưu trữ và giảm thiểu mất mát linh kiện, đảm bảo sẵn sàng nguồn cung cấp.

**b) Nhược điểm:**

+ Vì thời gian hạn chế, nên có một số nghiệp vụ trong quy trình chưa được hoàn thiện.

+ Mặc dù hệ thống được thiết kế để hoạt động mạnh mẽ, nhưng có nguy cơ gặp sự cố, mất kết nối hoặc lỗi hệ thống, có thể dẫn đến gián đoạn trong quá trình sản

+ Hệ thống chưa tích hợp giá linh phụ kiện, …

+ Hệ thống cần được duy trì định kỳ và nâng cấp để đảm bảo tính ổn định và hiệu suất cao, điều này có thể tạo ra chi phí bảo trì không dự kiến.

2. Kết quả thực hiện

+ Theo tôi đây là một đề tài không mới, nhưng nó đáp ứng được yêu cầu nghiệp vụ của hệ thống quản lý. Với sự hướng dẫn của giảng viên và sự tìm hiểu của bản thân, sau một khoảng thời gian nghiên cứu và thực hiện, tôi đã hoàn thành thiết kế cũng như cài đặt hệ thống, hệ thống hiện tại đã và đang phục vụ yêu cầu thực tế.

+ Tuy nhiên, vì thời gian không cho phép cũng như kiến thức còn hạn hẹp. Ứng dụng được xây dựng vẫn còn nhiều hạn chế như:

+ Chưa sử dụng thanh thông báo

+ Cơ cấu tổ chức chức năng chưa rõ ràng, nhiều hạn chế

+ Chưa tối ưu được hệ thống

3. Định hướng phát triển

Vẫn còn tồn tại những hạn chế, tôi mong rằng trong tương lai có thể cải thiện các chức năng bao gồm:  
1. Học cách tổ chức, cấu trúc code tốt hơn.  
2. Học cách tối ưu hệ thống cho hệ thống mượt mà hơn

1. Phân tích các chức năng rõ ràng hơn
2. Phát triển giao diện người dùng thân thiện, dễ sử dụng và có khả năng tương tác cao để giảm thiểu thời gian đào tạo và tăng cường trải nghiệm người dùng.
3. Tăng cường tính năng quản lý kho, bao gồm theo dõi tồn kho, dự báo nhu cầu và tối ưu hóa vị trí lưu trữ để giảm thiểu lãng phí và tăng khả năng sẵn sàng.

# TÀI LIỆU THAM KHẢO

1. Angular :

<https://viblo.asia/p/gioi-thieu-tong-quan-ve-angular-07LKX9j2ZV4>

<https://material.angular.io/>

2. ASP.Net :

<https://comdy.vn/asp-net-core-mvc/gioi-thieu-asp-net-core/>

<https://www.tutorialspoint.com/asp.net_core/asp.net_core_setup_mvc.htm>

3. LinQ :

https://viblo.asia/p/gioi-thieu-ve-linq-oaORbxymzEx

4. Diễn đàn:

<https://stackoverflow.com/>

<https://viblo.asia/>

<https://medium.com/>

### 5. Docker:

<https://www.docker.com/products/docker-scout/>

<https://nginx.org/en/docs/>