**Trường Đại Học Công Nghiệp TP.HCM**

**Khoa Công Nghệ Thông Tin**

**------**

Icon

Description automatically generated with medium confidence

**Đề tài**: Thiết kế kiến trúc và phần mềm cho hệ thống tái chế thiết bị điện tử

**Báo cáo tiểu luận**

**Môn: Kiến trúc và thiết kế phần mềm**

**Nhóm 10**:

Hồ Thị Mỹ Quỳnh-20053791(Nhóm trưởng)

Ngô Minh Thuận-20053391

Huỳnh Văn Duy-20062071

Hệ thống tái chế thiết bị điện tử

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc167703821)

[**Chương 1: Giới thiệu** 4](#_Toc167703822)

[**Chương 2 Cơ sở lý thuyết** 5](#_Toc167703823)

[**2.1 Node.js** 5](#_Toc167703824)

[**2.2 React.js** 7](#_Toc167703825)

[**2.3 MongoDB** 8](#_Toc167703826)

[**Chương 3. Kiến trúc phần mềm** 9](#_Toc167703827)

[**3.1 Giới thiệu về kiến trúc Microservice và các khái niệm cơ bản.** 9](#_Toc167703828)

[**Containers và Orchestration:** 9](#_Toc167703829)

[**API Gateway:** 10](#_Toc167703830)

[**Service Discovery:** 10](#_Toc167703831)

[**Service Mesh:** 10](#_Toc167703832)

[**Load Balancing:** 10](#_Toc167703833)

[**Circuit Breaker:** 11](#_Toc167703834)

[**Centralized Logging:** 11](#_Toc167703835)

[**Monitoring and Alert:** 11](#_Toc167703836)

[**Messaging System:** 11](#_Toc167703837)

[**Single Page Application (SPA):** 11](#_Toc167703838)

[**Configuration Server:** 11](#_Toc167703839)

[**3.2 Thiết kế kiến trúc cho hệ thống tái chế thiết bị điện tử** 12](#_Toc167703840)

[**3.2.1 User Interface (Giao diện người dùng)** 12](#_Toc167703841)

[**3.2.2 API Gateway** 12](#_Toc167703842)

[**3.2.3 UserProfile Service** 13](#_Toc167703843)

[**3.2.4 Inquiry Service** 13](#_Toc167703844)

[**3.2.5 Assessment Service** 13](#_Toc167703845)

[**3.2.6. Reporting Service** 13](#_Toc167703846)

[**3.2.7. Database** 13](#_Toc167703847)

[**3.3 Ưu và nhược điểm** 14](#_Toc167703848)

[**3.3.1Ưu điểm** 14](#_Toc167703849)

[**3.3.2** **Nhược điểm** 15](#_Toc167703850)

[**Chương 4. Phân tích và thiết kế phần mềm** 16](#_Toc167703851)

[**4.1** **Usecase tổng quát** 16](#_Toc167703852)

[**4.1.1** **Danh sách các tình huống hoạt động (Use cases)** 16](#_Toc167703853)

[**4.2** **Tình huống hoạt động** 18](#_Toc167703854)

[**Chương 5.Hiện thực** 34](#_Toc167703855)

[**5.1 Cấu hình phần cứng, phần mềm** 34](#_Toc167703856)

[**5.2 Giao diện của hệ thống** 35](#_Toc167703857)

[**Chương 6. Kết luận** 39](#_Toc167703858)

[**Chương 7. Tài liệu tham khảo** 40](#_Toc167703859)

# **Chương 1: Giới thiệu**

1. **Tổng quan.**

Trong bối cảnh phát triển hiện nay, các thiết bị điện tử đã trở nên phổ biến trên toàn thế giới. Tuy nhiên, vấn đề tái chế các thiết bị này đang trở thành một thách thức lớn. Do đó, việc tạo ra sự kết nối giữa khách hàng và công ty để thuận tiện trao đổi các thiết bị điện tử cũ, đặc biệt khi mức giá đưa ra là hợp lý, là điều cần thiết.

Dựa trên các tiêu chí này và yêu cầu của đề bài, nhóm chúng tôi đã lựa chọn kiến trúc hợp lý và thiết kế phần mềm cho một website tái chế, nhằm giúp khách hàng và công ty tương tác dễ dàng hơn.

Hệ thống này được xây dựng để tái chế các thiết bị điện tử cũ như điện thoại di động iPhone hoặc Galaxy. Quy trình xử lý tái chế được thực hiện như sau: Đầu tiên, khách hàng yêu cầu công ty, thông qua trang web hoặc kiosk, để biết được số tiền họ có thể nhận được cho thiết bị điện tử cũ của mình (gọi là trích dẫn). Nếu mức giá đề xuất làm khách hàng hài lòng, họ sẽ gửi thiết bị điện tử đến công ty tái chế (gọi là nhận).

Sau khi nhận được thiết bị, công ty sẽ đánh giá tình trạng hoạt động của nó (gọi là đánh giá). Nếu thiết bị còn hoạt động tốt, công ty sẽ gửi cho khách hàng số tiền đã hứa (gọi là kế toán). Trong suốt quá trình này, khách hàng có thể truy cập trang web bất cứ lúc nào để kiểm tra trạng thái của thiết bị (gọi là trạng thái).

Dựa trên kết quả đánh giá, thiết bị sẽ được tái chế bằng cách phá hủy an toàn hoặc bán lại (gọi là tái chế). Cuối cùng, công ty sẽ định kỳ chạy các báo cáo tài chính và hoạt động tái chế (gọi là báo cáo).

Hệ thống này không chỉ giúp quản lý quá trình tái chế hiệu quả mà còn tạo ra một nền tảng tương tác thuận tiện cho cả khách hàng và công ty, góp phần giải quyết vấn đề tái chế thiết bị điện tử một cách bền vững và hiệu quả.

1. **Mục chính của đề tài:**

Mục đích của đề tài này là xây dựng một hệ thống website tái chế thiết bị điện tử nhằm:

Tạo ra một nền tảng kết nối thuận tiện giữa khách hàng và công ty tái chế: Giúp khách hàng dễ dàng trao đổi các thiết bị điện tử cũ và nhận được giá trị hợp lý cho các thiết bị đó.

Quản lý hiệu quả quá trình tái chế: Từ việc yêu cầu báo giá, nhận thiết bị, đánh giá tình trạng, đến việc xử lý tái chế hoặc bán lại thiết bị. Hệ thống giúp tổ chức các bước này một cách trơn tru và logic.

Cải thiện trải nghiệm người dùng: Cho phép khách hàng truy cập và theo dõi trạng thái của thiết bị mọi lúc, tạo sự minh bạch và tin cậy trong quá trình tái chế.

Hỗ trợ hoạt động báo cáo: Cung cấp các báo cáo định kỳ về tài chính và hoạt động tái chế, giúp công ty có cái nhìn tổng quan và đưa ra các quyết định chiến lược.

Đóng góp vào bảo vệ môi trường: Khuyến khích việc tái chế thiết bị điện tử, giảm thiểu rác thải điện tử, và bảo vệ tài nguyên thiên nhiên.

Mục tiêu cuối cùng của hệ thống là tạo ra một quy trình tái chế thiết bị điện tử hiệu quả, bền vững, và dễ dàng tiếp cận cho cả khách hàng lẫn công ty tái chế.

# **Chương 2 Cơ sở lý thuyết**

## **2.1 Node.js**



Hình 2- 1 Node.js

a. Tổng quan:

- Nodejs là một **nền tảng** (Platform) phát triển độc lập được xây dựng ở trên

Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

b. Ưu điểm của Nodejs:

- Các ứng dụng Nodejs được viết bằng **javascript**, ngôn ngữ này

là một ngôn ngữ khá thông dụng.

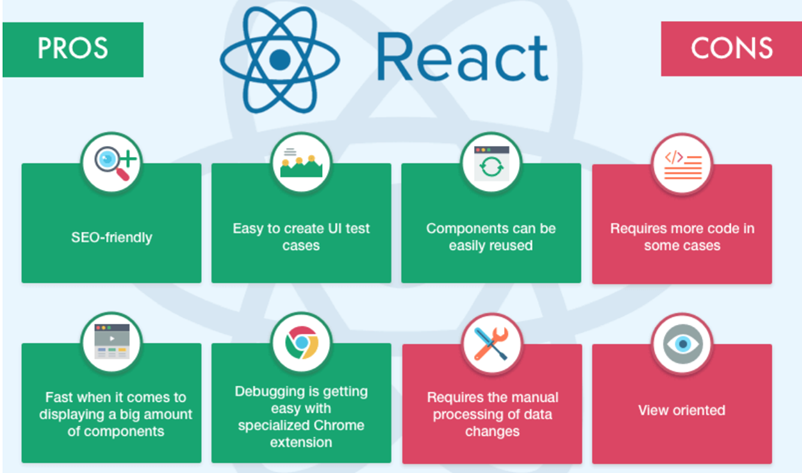
- Nodejs chạy đa nền tảng phía Server, sử dụng kiến trúc hướng sự kiện

Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.

- Có thể chạy ứng dụng Nodejs ở bất kỳ đâu trên máy Mac – Window –

Linux, hơn nữa cộng đồng Nodejs rất lớn và hoàn toàn miễn phí.

## **2.2 React.js**



Hình 2- 2 React.js

a. Tổng quan:

- React là một thư viện để xây dựng giao diện người dùng có thể kết hợp. Nó

khuyến khích việc tạo ra các thành phần giao diện người dùng có thể tái sử dụng, hiển thị dữ liệu thay đổi theo thời gian. Rất nhiều người sử dụng React làm V trong MVC

b. Ưu điểm của reactjs:

- Sử dụng DOM ảo là một đối tượng JavaScript. Điều này sẽ cải thiện hiệu

suất ứng dụng, vì DOM ảo JavaScript nhanh hơn DOM thông thường.

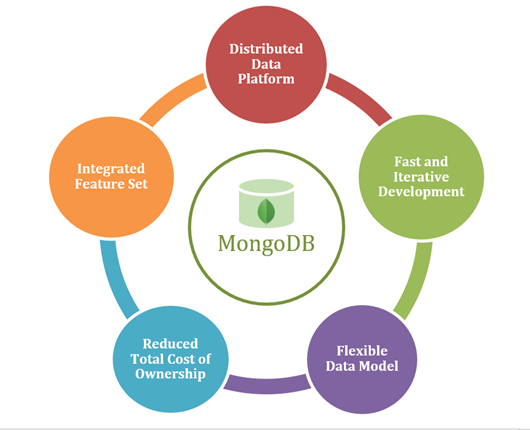
- Có thể được sử dụng trên máy khách và máy chủ cũng như với các khuôn

khổ khác.

- Các mẫu thành phần và dữ liệu cải thiện khả năng đọc, giúp duy trì các ứng

dụng lớn hơn.

## **2.3 MongoDB**



Hình 2- 3 MongoDB

a. Tổng quan:

- MongoDB là một chương trình cơ sở dữ liệu mã nguồn mở được thiết kế theo

kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào.

b. Ưu điểm của MongoDB:

- Hiệu suất cao

- Tính sẵn sàng cao – Nhân rộng

- Khả năng mở rộng cao – Sharding

- Năng động – Không có lược đồ cứng nhắc.

- Linh hoạt – thêm / xóa trường có ít hoặc không ảnh hưởng đến ứng dụng

- Dữ liệu không đồng nhất

- Không joins

- Phân phối được

- Biểu diễn dữ liệu trong JSON hoặc BSON

# **Chương 3. Kiến trúc phần mềm**

## **3.1 Giới thiệu về kiến trúc Microservice và các khái niệm cơ bản.**

Microservice là là một kiểu kiến ​​trúc phần mềm chia các chức năng khác nhau của ứng dụng thành các thành phần nhỏ hơn được gọi là “service”. Mỗi đơn vị này được triển khai độc lập và có thể mở rộng, có thể được viết bằng các ngôn ngữ lập trình khác nhau và kiểm tra riêng biệt. Khả năng này cho phép bảo trì, nâng cấp một cách nhanh chóng, tiết kiệm chi phí mà không ảnh hưởng đến phần còn lại của ứng dụng.

Bên cạnh đó, để các service độc lập của ứng dụng có thể giao tiếp với nhau và với các ứng dụng khác, các nhà lập trình đã sử dụng API. Vì vậy, có thể nói, API được sử dụng để kích hoạt các microservice.

Microservices và API Gateway được ra đời với mục đích cải thiện khả năng mở rộng và tối giản quá trình bảo trì, vận hành của các hệ thống phức tạp.

### **Containers và Orchestration:**

Containers: Containers là các gói nhẹ chứa mọi thứ cần thiết để chạy một ứng dụng, bao gồm mã nguồn, thư viện, và các phụ thuộc khác. Điều này giúp ứng dụng chạy đồng nhất ở mọi môi trường. Docker là một công cụ phổ biến để tạo và quản lý containers.

Orchestration: Đây là quá trình quản lý, điều phối và tự động hóa các container. Kubernetes là một công cụ orchestration phổ biến giúp triển khai, mở rộng và quản lý container tự động, đảm bảo rằng ứng dụng luôn sẵn sàng và hoạt động tốt.

### **API Gateway:**

API Gateway là một điểm truy cập duy nhất mà qua đó các client (như ứng dụng web, di động) tương tác với các microservices. Nó chịu trách nhiệm định tuyến các yêu cầu đến đúng microservice, quản lý xác thực, và thực hiện load balancing. Nginx và Kong là các ví dụ về API Gateway.

### **Service Discovery:**

Service Discovery giúp các microservices tìm thấy và giao tiếp với nhau mà không cần biết trước địa chỉ cụ thể của các dịch vụ khác. Khi một microservice mới khởi động, nó sẽ đăng ký với một service registry (như Consul hoặc Eureka), và các dịch vụ khác có thể tra cứu trong registry này để tìm địa chỉ của nhau.

### **Service Mesh:**

Service Mesh là một lớp hạ tầng chuyên quản lý giao tiếp giữa các microservices. Nó cung cấp các tính năng như load balancing, kiểm soát lưu lượng, bảo mật, và giám sát mà không cần thay đổi mã nguồn của dịch vụ. Istio là một ví dụ về service mesh.

### **Load Balancing:**

Load Balancing là quá trình phân phối lưu lượng truy cập đều giữa các instance của một dịch vụ để đảm bảo hiệu năng và độ tin cậy. Load balancer như HAProxy hoặc Nginx đảm bảo rằng không có instance nào bị quá tải.

### **Circuit Breaker:**

Circuit Breaker là một mẫu thiết kế giúp hệ thống chịu lỗi tốt hơn. Nếu một dịch vụ gặp sự cố hoặc quá tải, circuit breaker sẽ “mở mạch” và ngăn các yêu cầu tới dịch vụ đó, giảm nguy cơ lan truyền lỗi và giúp hệ thống ổn định hơn. Netflix Hystrix là một thư viện cung cấp chức năng circuit breaker.

### **Centralized Logging:**

Centralized Logging tập trung tất cả các log từ các microservices vào một nơi để dễ dàng theo dõi và phân tích. Công cụ phổ biến bao gồm ELK Stack (Elasticsearch, Logstash, Kibana) giúp thu thập, lưu trữ và hiển thị log.

### **Monitoring and Alert:**

Monitoring là quá trình giám sát hiệu năng và trạng thái của các microservices, trong khi alerting là việc tạo ra các cảnh báo khi có sự cố xảy ra. Công cụ như Prometheus và Grafana giúp theo dõi và tạo cảnh báo dựa trên các chỉ số thu thập được.

### **Messaging System:**

Messaging System là hệ thống giúp các microservices giao tiếp với nhau thông qua thông điệp, thay vì gọi trực tiếp. Điều này giúp hệ thống phi đồng bộ và chịu lỗi tốt hơn. Kafka và RabbitMQ là các ví dụ về messaging system.

### **Single Page Application (SPA):**

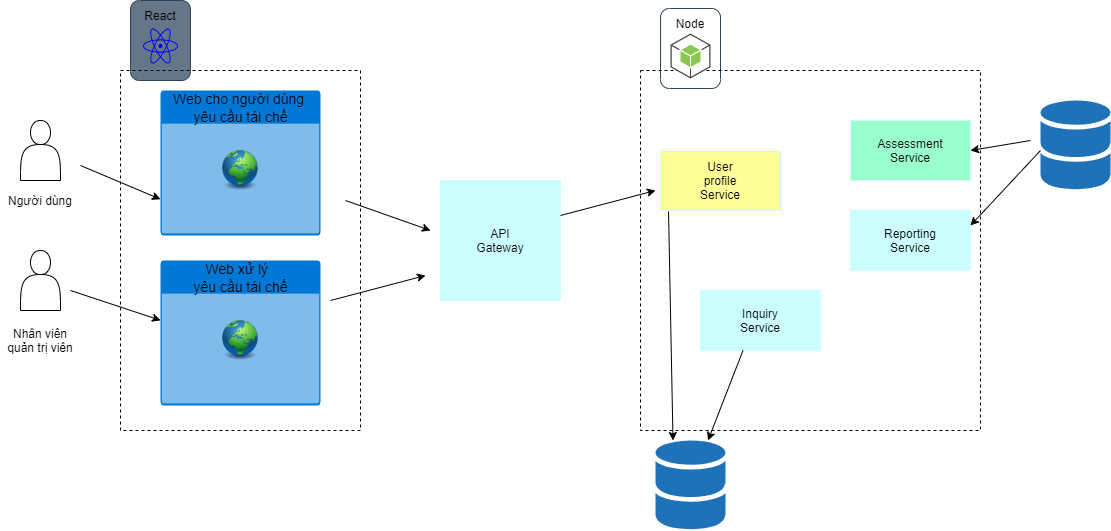
Single Page Application là ứng dụng web mà tải toàn bộ trang web một lần duy nhất và sau đó chỉ cập nhật các phần nội dung khi cần, giúp trải nghiệm người dùng mượt mà hơn. Các framework như React, Angular, và Vue.js thường được sử dụng để xây dựng SPA.

### **Configuration Server:**

Configuration Server quản lý cấu hình của các microservices từ một nơi tập trung. Nó giúp các dịch vụ truy cập cấu hình cần thiết một cách nhất quán và dễ dàng thay đổi mà không cần phải khởi động lại dịch vụ. Spring Cloud Config là một ví dụ về configuration server..

## **3.2 Thiết kế kiến trúc cho hệ thống tái chế thiết bị điện tử**

Từ những nghiên cứu về kiến trúc Microservice, chúng tôi tiến hành thiết kế kiến trúc cho hệ thống Tái chế thiết bị điện tử. Dưới đây là sơ đồ kiến trúc tổng quan



*(Kiến trúc Microsevice dành cho hệ thống tái chế điện tử)*

### **3.2.1 User Interface (Giao diện người dùng)**

Giao diện người dùng được phân chia thành hai phần: một là dành cho khách hàng và hai là dành cho quản trị viên cũng như nhân viên. Trên nền tảng web, cả khách hàng và nhân viên đều có thể dễ dàng tương tác với các biểu mẫu yêu cầu báo giá, kiểm tra trạng thái của thiết bị hoặc xem các báo cáo thống kê.

### **3.2.2 API Gateway**

API Gateway đóng vai trò là điểm nhập duy nhất cho mọi yêu cầu từ giao diện người dùng. Nó thực hiện nhiệm vụ cân bằng tải, xác thực và định tuyến các yêu cầu tới các dịch vụ thích hợp.

### **3.2.3 UserProfile Service**

Userprofile Service đảm nhận vai trò quản lý xác thực và phân quyền người dùng. Tại đây, người dùng có thể đăng ký tài khoản và cũng có thể truy cập để xem hoặc chỉnh sửa thông tin cá nhân của mình.

### **3.2.4 Inquiry Service**

Inquiry Service là dịch vụ cho phép khách hàng tạo yêu cầu thu thiết bị di động. Khách hàng có thể xem lịch sử các yêu cầu, chỉnh sửa thông tin về sản phẩm họ muốn gửi cho công ty. Ngoài ra, dịch vụ cung cấp thông tin về trạng thái hiện tại của thiết bị trong quá trình tái chế, giúp khách hàng theo dõi quy trình một cách dễ dàng.

### **3.2.5 Assessment Service**

Dịch vụ cung cấp các tính năng cho nhân viên. Họ có thể tạo các phiếu để xử lý quy trình đánh giá sản phẩm. Sau khi đánh giá xong, nhân viên có thể cập nhật trạng thái sản phẩm, quyết định liệu nó sẽ được bán lại hay tái chế, cũng như số tiền sẽ được trả lại cho khách hàng.

Thông tin về các sản phẩm sau khi được xem xét kỹ lưỡng cũng sẽ được lưu trong hệ thống. Dịch vụ cho phép nhân viên xem và chỉnh sửa thông tin của các thiết bị điện tử này.

### **3.2.6. Reporting Service**

Tạo báo cáo về hoạt động tái chế và tình hình tài chính. Các thông tin về doanh thu, phiếu thu chi sẽ được xử lý tại service này

### **3.2.7. Database**

Database đóng vai trò quan trọng trong việc lưu trữ dữ liệu liên quan đến thiết bị, báo giá, đánh giá, giao dịch và báo cáo. Để tối ưu hóa quản lý dữ liệu, chúng ta đã phân chia database thành hai cụm: một cụm chứa dữ liệu được xử lý bởi khách hàng và một cụm khác chứa dữ liệu được xử lý bởi nhân viên.

Tuy nhiên, một thách thức nảy sinh là làm thế nào để đảm bảo tính nhất quán và toàn vẹn của dữ liệu khi chia thành các cụm riêng lẻ. Để giải quyết vấn đề này, chúng ta đã thiết lập lịch trình chạy các dịch vụ batch để cập nhật dữ liệu và đồng bộ hóa giữa các cụm database. Các công việc batch này sẽ được kích hoạt vào những khoảng thời gian cụ thể, thường là trong các khoảng thời gian hệ thống không hoạt động chủ yếu, như những khoảng thời gian nghỉ của hệ thống, để tránh ảnh hưởng đến trải nghiệm của người dùng.

## **3.3 Ưu và nhược điểm**

### **3.3.1Ưu điểm**

Tính linh hoạt và khả năng mở rộng: Microservice cho phép mỗi dịch vụ được phát triển, triển khai và mở rộng một cách độc lập. Điều này giúp hệ thống dễ dàng điều chỉnh để đáp ứng nhu cầu thay đổi và tăng trưởng.

Độc lập công nghệ: Mỗi dịch vụ có thể sử dụng ngôn ngữ lập trình, công nghệ và cơ sở dữ liệu phù hợp nhất với nhu cầu cụ thể của nó, không bị ràng buộc bởi các lựa chọn công nghệ của các dịch vụ khác.

Độ tin cậy cao: Nếu một dịch vụ gặp sự cố, nó không làm ảnh hưởng đến toàn bộ hệ thống. Các dịch vụ khác vẫn có thể tiếp tục hoạt động bình thường.

Dễ bảo trì và nâng cấp: Với các dịch vụ được chia nhỏ, việc bảo trì và nâng cấp trở nên dễ dàng hơn, giảm rủi ro khi triển khai các bản cập nhật hoặc sửa lỗi.

Phát triển nhanh chóng: Các đội phát triển có thể làm việc độc lập trên các dịch vụ khác nhau, tăng tốc độ phát triển và triển khai.

Khả năng tái sử dụng: Các dịch vụ microservice có thể được sử dụng lại trong nhiều dự án khác nhau, tiết kiệm thời gian và tài nguyên phát triển.

### **3.3.2 Nhược điểm**

Phức tạp trong quản lý: Số lượng dịch vụ tăng lên dẫn đến việc quản lý và điều phối các dịch vụ trở nên phức tạp hơn, đòi hỏi các công cụ và quy trình quản lý mạnh mẽ.

Giao tiếp giữa các dịch vụ: Việc giao tiếp giữa các dịch vụ qua mạng có thể gây ra độ trễ và phức tạp, đòi hỏi phải có cơ chế quản lý lỗi, xác thực và bảo mật.

Khả năng đồng bộ dữ liệu: Đảm bảo tính nhất quán dữ liệu giữa các dịch vụ có thể là một thách thức lớn, đặc biệt trong các hệ thống phân tán.

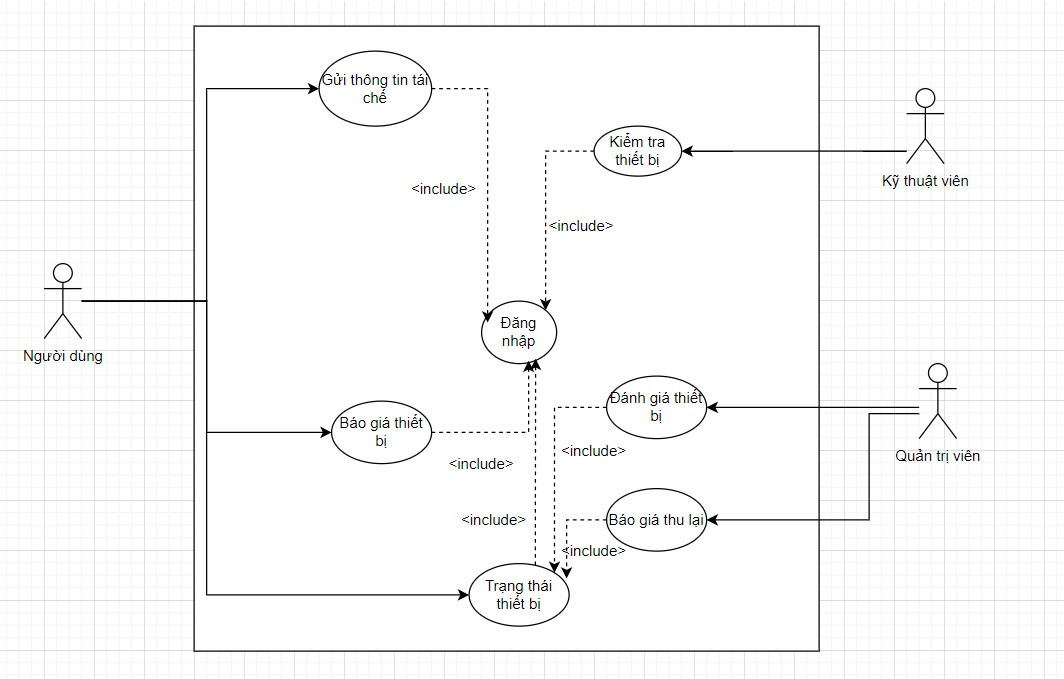
Tài nguyên và chi phí: Việc triển khai và duy trì nhiều dịch vụ độc lập có thể đòi hỏi nhiều tài nguyên hơn và tăng chi phí hạ tầng.

Đòi hỏi kỹ năng chuyên môn: Phát triển và vận hành một hệ thống microservice yêu cầu các kỹ năng và kiến thức chuyên môn cao, có thể gây khó khăn cho đội ngũ phát triển và vận hành.

Giám sát và debug: Theo dõi và xử lý lỗi trong một hệ thống microservice phức tạp đòi hỏi công cụ giám sát và debug mạnh mẽ, có thể khó khăn hơn so với kiến trúc đơn thể.

# **Chương 4. Phân tích và thiết kế phần mềm**

### **Usecase tổng quát**



### **4.1.1 Danh sách các tình huống hoạt động (Use cases)**

|  |  |
| --- | --- |
| **ID use case** | **Tên Use case** |
| **UC01** | Đăng nhập |
| **UC02** | Gửi thông tin thiết bị tái chế |
| **UC03** | Tiếp nhận yêu cầu tái chế |
| **UC04** | Xem thông tin thiết bị đã thu lại |
| **UC05** | Cập nhật thông tin thiết bị |
| **UC06** | Theo dõi trạng thái thiết bị |
| **UC07** | Kiểm tra tình trạng thiết bị |
| **UC08** | Báo giá thu lại |
| **UC09** | Đánh giá thiết bị |

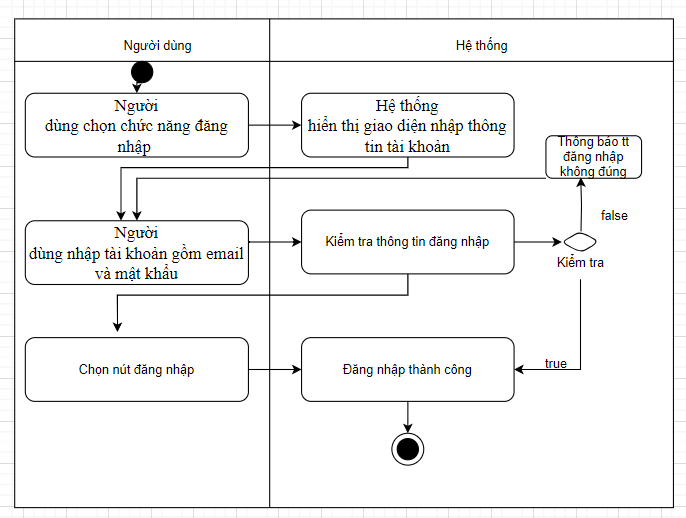
Bảng 3-2 Danh sách các tình huống trong hệ thống

### **Tình huống hoạt động**

#### **UC01\_Đăng nhập**

|  |  |
| --- | --- |
| * **Tên use case**: UC01\_Đăng nhập | |
| * **Mô tả sơ lược**: Người dùng muốn đăng nhập vào hệ thống để sử dụng các dịch vụ | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đã có tài khoản | |
| * **Hậu điều kiện (Post-condition):** | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng đăng nhập |  |
|  | 2. Hệ thống hiển thị giao diện nhập thông tin tài khoản |
| 1. Người dùng nhập tài khoản gồm email và mật khẩu |  |
| 1. Chọn nút đăng nhập |  |
|  | 1. Hệ thống kiểm tra thông tin nhập vào |
|  | 1. Hệ thống chuyển hướng người dùng vào giao diện trang chủ |
|  |  |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 6.1. Hệ thống thông báo thông tin tài khoản đăng nhập không đúng |
|  | 6.2. Hệ thống quay lại bước 3 |

Sơ đồ Activity:



#### Sơ đồ Activity

#### **4.2.2 UC02\_Gửi thông tin thiết bị tái chế**

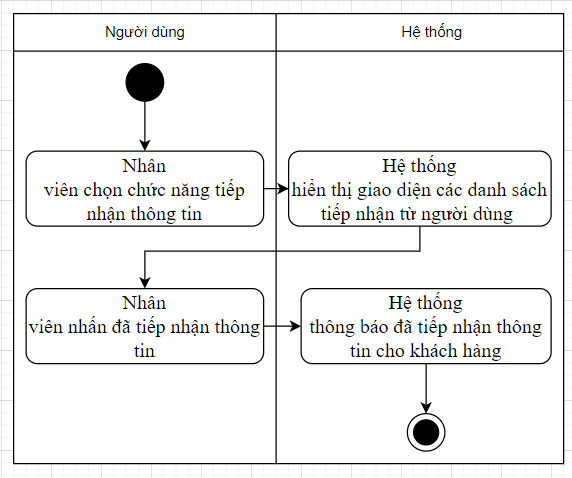
|  |  |
| --- | --- |
| * **Tên use case**: UC05\_Gửi thông tin thiết bị tái chế | |
| * **Mô tả sơ lược**: Giúp người dùng gửi thông tin, tình trạng của thiết bị cần tái chế | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Thông tin, tình trạng thiết bị đã được gửi đi | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng gửi thông tin thiết bị tái chế |  |
|  | 2. Hệ thống hiển thị giao diện gửi thông tin thiết bị tái chế |
| 1. Người dùng nhập các thông tin liên quan về thiết bị tái chế |  |
| 1. Người dùng nhấn gửi thông tin |  |
|  | 1. Hệ thống kiểm tra các thông tin nhập |
|  | 1. Thông báo gửi thông tin thành công |
| * **Luồng sự kiện thay thế (alternate flow):** | |
|  | 5.1. Hệ thống thông báo nhập thiếu thông tin hoặc không hợp lệ |
|  | 5.2. Hệ thống quay lại bước 3 |

Sơ đồ Activity

#### **4.2.3 UC03\_Tiếp nhận yêu cầu tái chế**

|  |  |
| --- | --- |
| * **Tên use case**: UC06\_ Tiếp nhận yêu cầu tái chế | |
| * **Mô tả sơ lược**: Giúp nhân viên tiếp nhận thông tin thiết bị tái chế từ khách hàng | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống thông báo đã tiếp nhận thông tin | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn chức năng tiếp nhận thông tin |  |
|  | 1. Hệ thống hiển thị giao diện các danh sách yêu cầu từ người dùng |
| 1. Nhân viên nhấn vào yêu cầu mình muốn tiếp nhận |  |
|  | 1. Hệ thống hiển thị chi tiết yêu cầu |
| 5. Nhân viên nhấn vào nút “Tiếp nhận yêu cầu” |  |
|  | 6. Hệ thống tạo mới phiếu đánh giá cho yêu cầu mà nhân viên đã tiếp nhận |

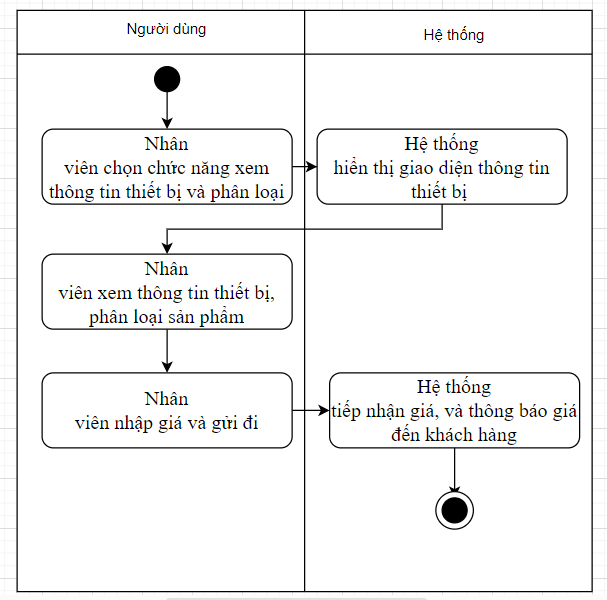
#### Sơ đồ Activity



#### **4.2.7 UC07\_Phân loại thiết bị và báo giá**

|  |  |
| --- | --- |
| * **Tên use case**: UC07\_ Phân loại thiết bị và báo giá | |
| * **Mô tả sơ lược**: Giúp người dùng biết giá thiết bị tái chế | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống thông báo giá thiết bị cho khách hàng | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn chức năng đánh giá thiết bị |  |
|  | 1. Hệ thống hiển thị danh sách các đánh giá |
| 1. Nhân viên chọn vào mẫu đánh giá cần cập nhật trạng thái và báo giá |  |
|  | 1. Hệ thống hiển thị thông tin chi tiết mẫu đánh giá |
| 1. Nhân viên nhập giá và trạng thái thiết bị, sau đó bấm “Cập nhật” |  |
|  | 1. Hệ thống cập nhật thông tin thiết bị và giá thiết bị |
|  | 1. Hệ thống thông báo “Đã cập nhật thông tin thiết bị” |

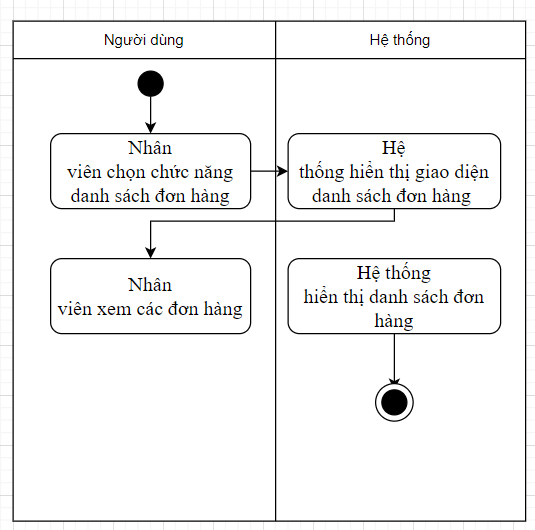
#### Sơ đồ Activity



#### **4.2.4 UC04\_Xem thông tin thiết bị đã thu lại**

|  |  |
| --- | --- |
| * **Tên use case**: UC08\_ Thông tin thiết bị | |
| * **Mô tả sơ lược**: Giúp nhân viên xem danh sách và thông tin chi tiết thiết bị đã thu lại | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống hiển thị thông tin thiết bị | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn chức năng danh sách sản phẩm |  |
|  | 2. Hệ thống hiển thị giao diện danh sách các thiết bị đã thu lại từ khách hàng |
| 1. Nhân viên bấm vào biểu tượng “Xem chi tiết” ở sản phẩm cần xem thông tin |  |
|  | 1. Hệ thống hiển thị chi tiết thiết bị |

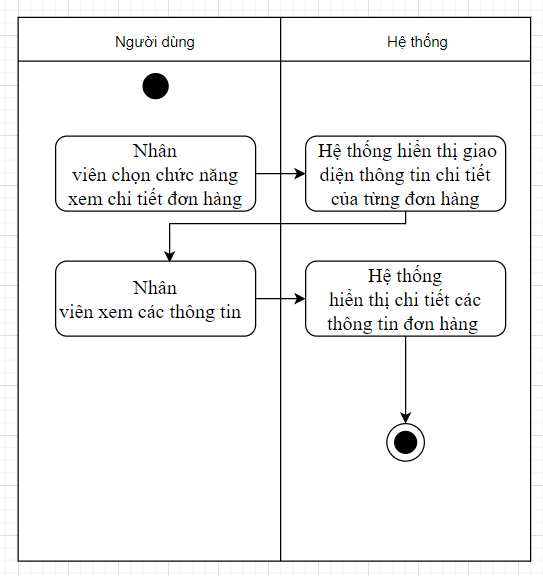
Sơ đồ Activity



#### **4.2.5 UC05\_Cập nhật thông tin thiết bị**

|  |  |
| --- | --- |
| * **Tên use case**: UC09\_Cập nhật thông tin thiết bị | |
| * **Mô tả sơ lược**: Giúp nhân viên cập nhật thông tin thiết bị sau khi đã thu mua lại từ người dùng | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Cập nhật thông tin thiết bị thành công | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên bấm vào mục “Sản phẩm” |  |
|  | 2. Hệ thống hiển thị danh sách các thiết bị dưới dạng bảng |
| 3. Nhân viên chọn thiết bị cần cập nhật thông tin |  |
|  | 4. Hệ thống hiển thị chi tiết sản phẩm và cho phép nhân viên sửa đổi thông tin |
| 5. Nhân viên thay đổi các trường thông tin cần cập nhật và bấm nút “Cập nhật” để lưu lại thông tin thiết bị |  |
|  | 6. Hệ thống kiểm tra dữ liệu nhân viên nhập và tiến hành lưu vào cơ sở dữ liệu |
|  | 7. Hệ thống hiển thị thông báo “Đã cập nhật thành công” |

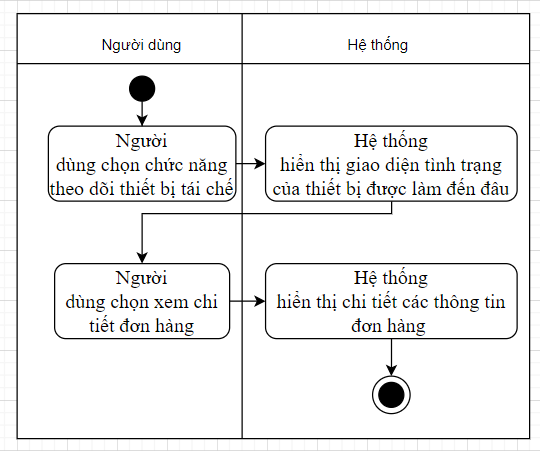
Sơ đồ Activity

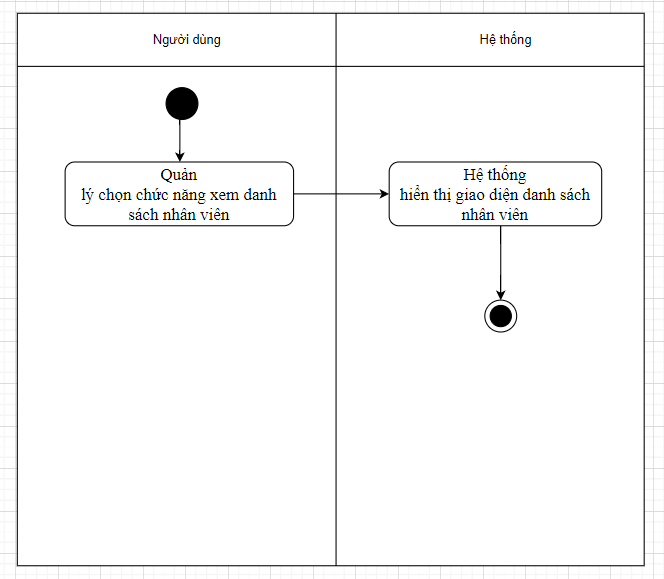


#### **4.2.6 UC06\_ Theo dõi trạng thái thiết bị**

|  |  |
| --- | --- |
| * **Tên use case**: UC10\_Theo dõi trạng thái thiết bị | |
| * **Mô tả sơ lược**: Giúp người dùng xem được tình trạng thiết bị | |
| * **Actor chính**: Người dùng | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Người dùng đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống hiển thị tiến trình theo dõi | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Người dùng chọn chức năng theo dõi thiết bị tái chế |  |
|  | 2. Hệ thống hiển thị danh sách các thiết bị người dùng đã gửi yêu cầu tái chế |
| 1. Người dùng chọn xem chi tiết thiết bị cần theo dõi |  |
|  | 1. Hệ thống hiển thị tình trạng chi tiết của thiết bị |

Sơ đồ Activity





#### **4.2.7 UC07\_Kiểm tra tình trạng thiết bị**

|  |  |
| --- | --- |
| * **Tên use case**: UC17\_ Kiểm tra tình trạng thiết bị | |
| * **Mô tả sơ lược**: Giúp nhân viên kiểm tra tình trạng thiết bị | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống thông báo kiểm tra thành công | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn chức năng kiểm tra tình trạng thiết bị |  |
|  | 2. Hệ thống hiển thị giao diện kiểm tra tình trạng thiết bị |
| 1. Nhân viên kiểm tra thông tin dựa vào thông tin khách hàng đã gửi |  |
| 1. Nhân viên nhập kết quả sau khi đã kiểm tra xong và bấm “Gửi” |  |
|  | 1. Hệ thống kiểm tra các thông tin và thông báo “Kiểm tra thành công” |
|  | 1. Hệ thống gửi phản hồi cho khách hàng |
| * **Luồng sự kiện thay thế (alternate flow):** | |
| 4.1 Nhân viên bấm “Hủy” |  |
|  | 4.2. Hệ thống quay lại bước 3 |

Sơ đồ Activity

#### **4.2.8 UC08\_Báo giá thu lại**

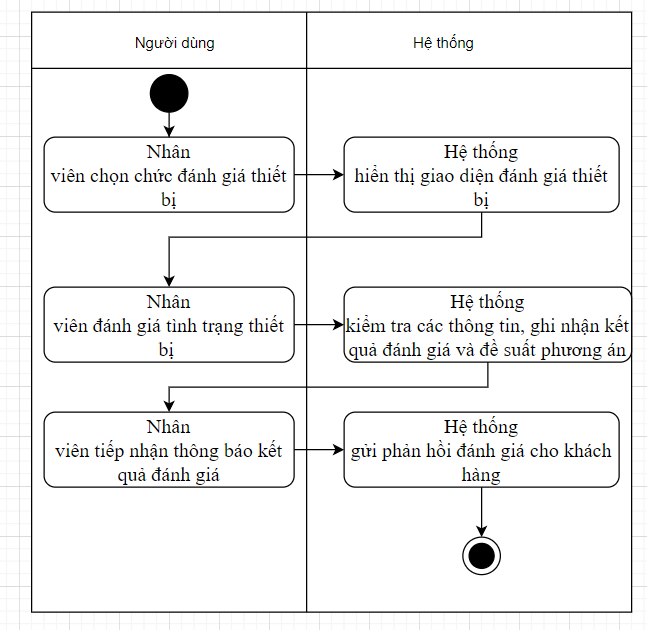
|  |  |
| --- | --- |
| * **Tên use case**: UC18\_Báo giá thu lại | |
| * **Mô tả sơ lược**: Giúp nhân viên cập nhật lại giá lên hệ thống | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống thông báo cập nhật thành công | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn vào mục “Đánh giá thiết bị” |  |
|  | 2. Hệ thống hiển thị giao diện danh sách các mẫu đánh giá |
| 1. Nhân viên chọn mẫu đánh giá cần thực hiện báo giá |  |
|  | 1. Hệ thống hiển thị thông tin chi tiết mẫu đánh giá |
| 1. Nhân viên cập nhật lại giá thiết bị và nhấn nút “Cập nhật” |  |
|  | 1. Hệ thống lưu lại giá thiết bị và thông báo cho người dùng |
| * **Luồng sự kiện thay thế (alternate flow):** | |
| 4.1 Nhân viên bấm “Hủy” |  |
|  | 4.2. Hệ thống quay lại bước 3 |

Sơ đồ Activity

#### **4.2.9 UC09\_Đánh giá thiết bị**

|  |  |
| --- | --- |
| * **Tên use case**: UC19\_ Đánh giá thiết bị | |
| * **Mô tả sơ lược**: Giúp nhân viên đánh giá thiết bị | |
| * **Actor chính**: Nhân viên | |
| * **Actor phụ**: Không. | |
| * **Tiền điều kiện (Pre-condition):** Nhân viên đăng nhập thành công | |
| * **Hậu điều kiện (Post-condition):** Hệ thống thông báo đánh giá thành công | |
| * **Luồng sự kiện chính (main flow):** | |
| **Actor** | **System** |
| 1. Nhân viên chọn vào mục “Đánh giá thiết bị” |  |
|  | 2. Hệ thống hiển thị giao diện danh sách các mẫu đánh giá |
| 1. Nhân viên chọn mẫu đánh giá cần thực hiện báo giá |  |
|  | 1. Hệ thống hiển thị thông tin chi tiết mẫu đánh giá |
| 1. Nhân viên tiến hàng đánh giá thiết bị, sau đó cập nhật tình trạng, kết quả đánh giá |  |
|  | 1. Hệ thống gửi phản hồi đánh giá cho khách hàng |

Sơ đồ activity:



# **Chương 5.Hiện thực**

## **5.1 Cấu hình phần cứng, phần mềm**

· Các dịch vụ MongoDB:

* Lưu trữ ảnh
* Lưu trữ dữ Liệu

Postman:

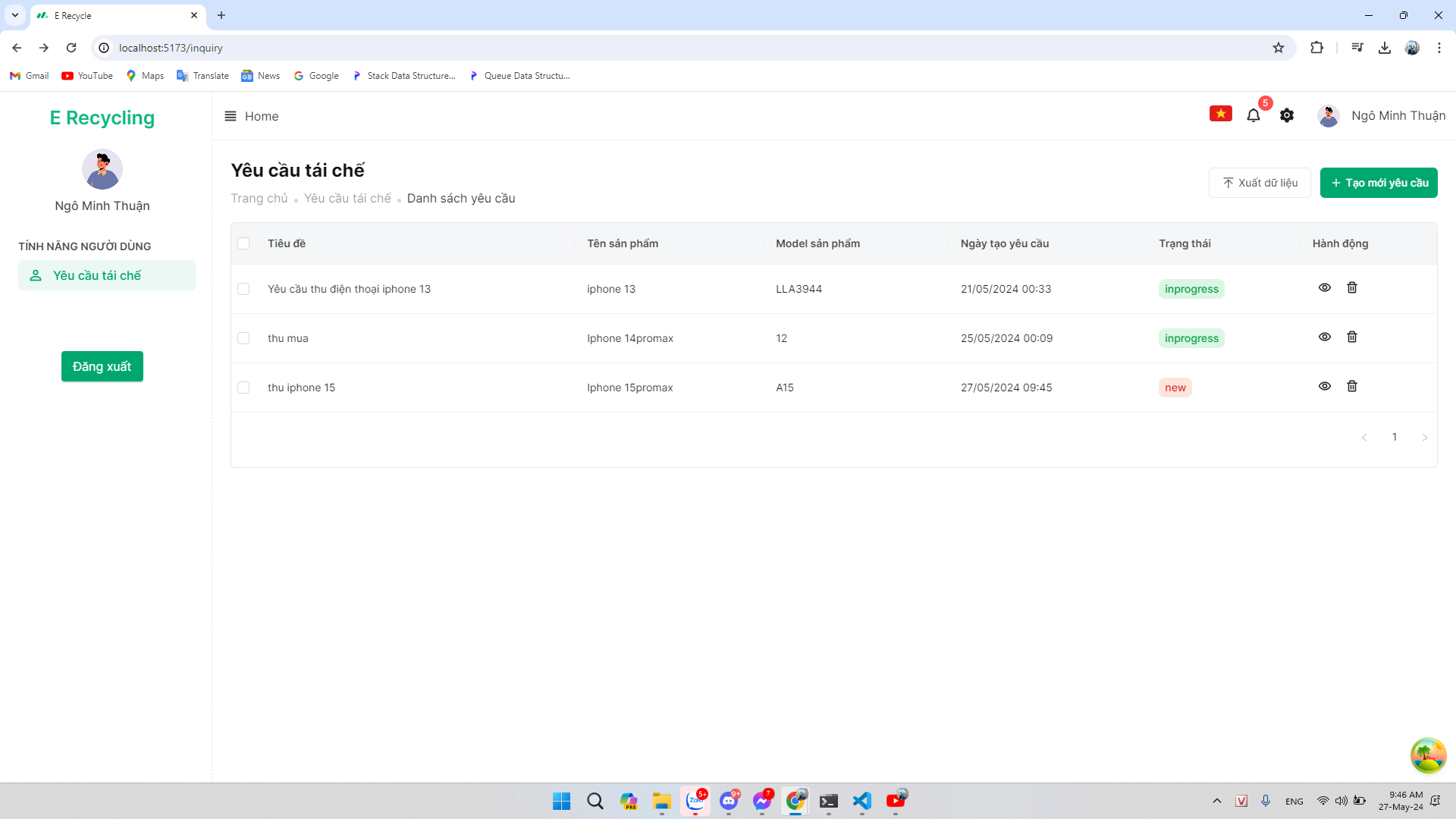
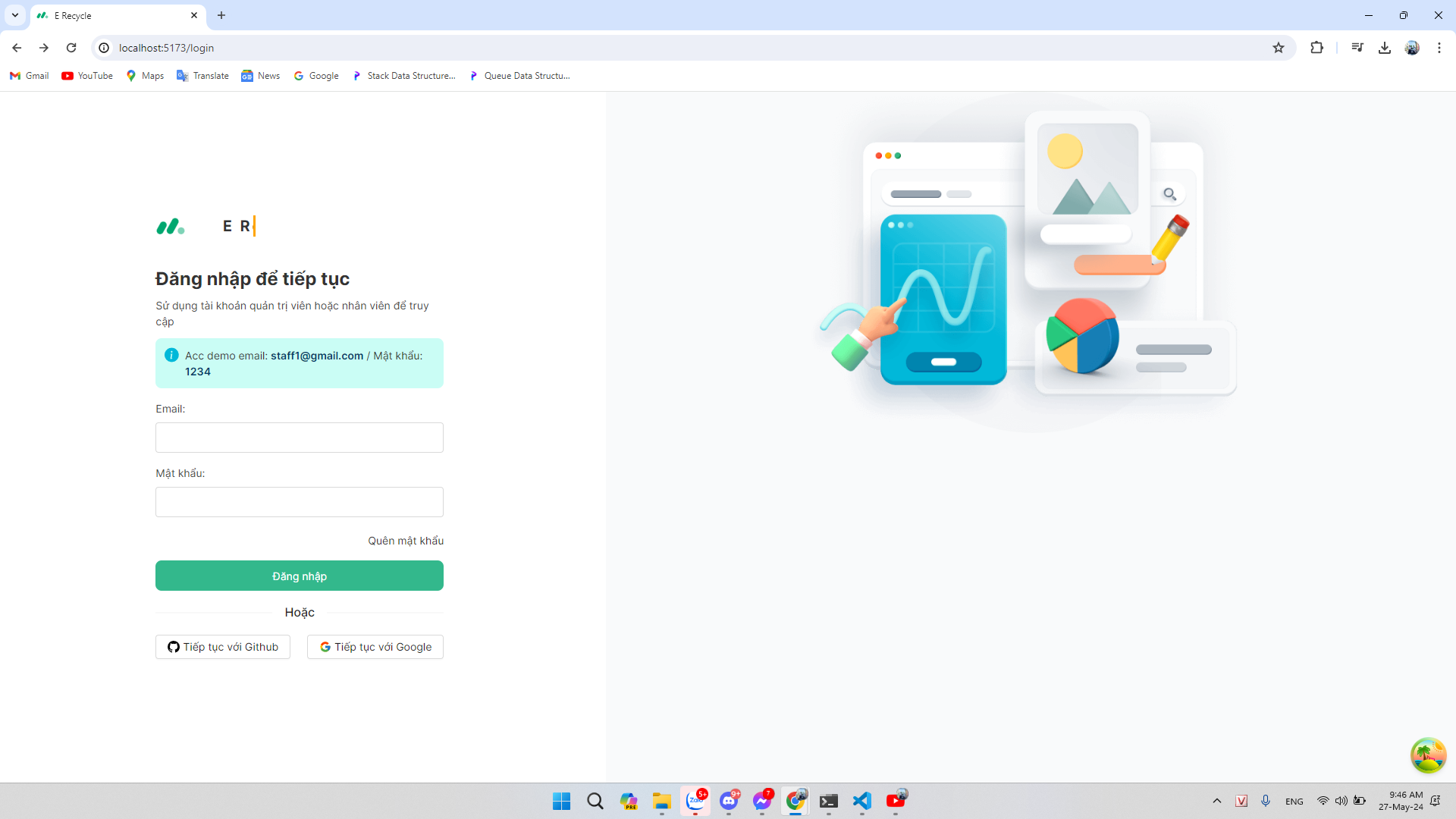
Tool test API

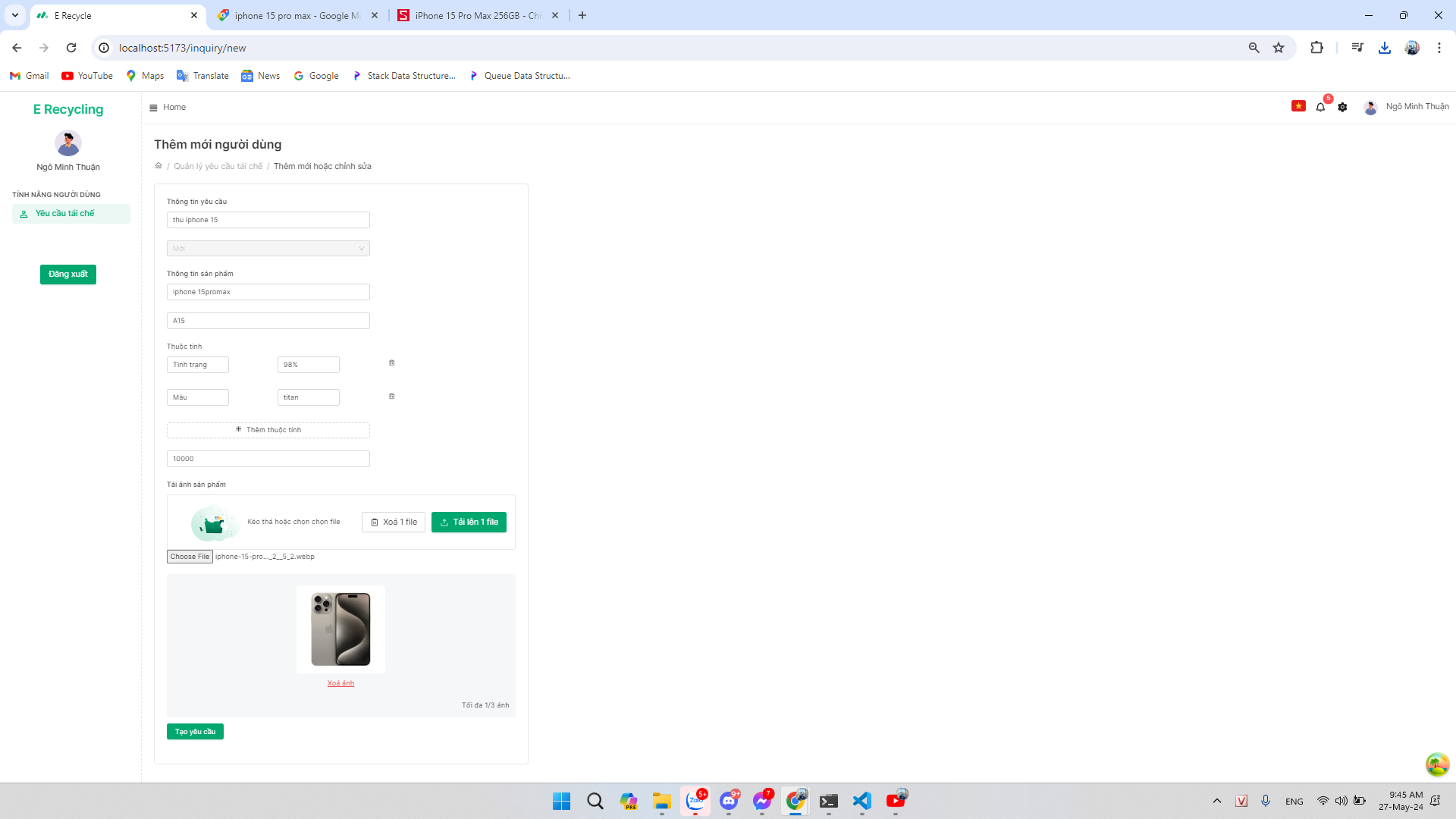
· Cấu hình Sever: Visual code studio

· Cấu hình Client: Visual code studio

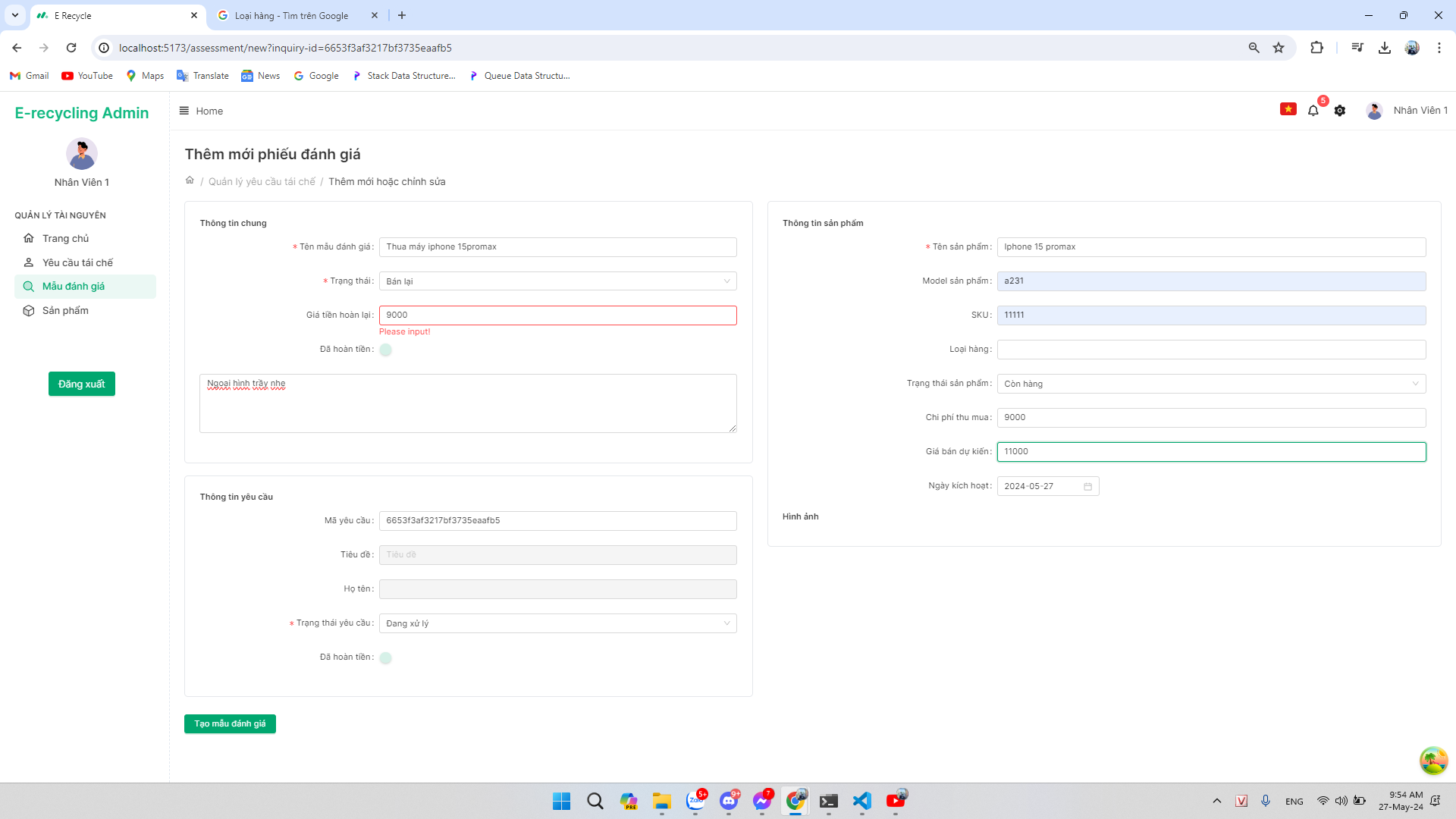
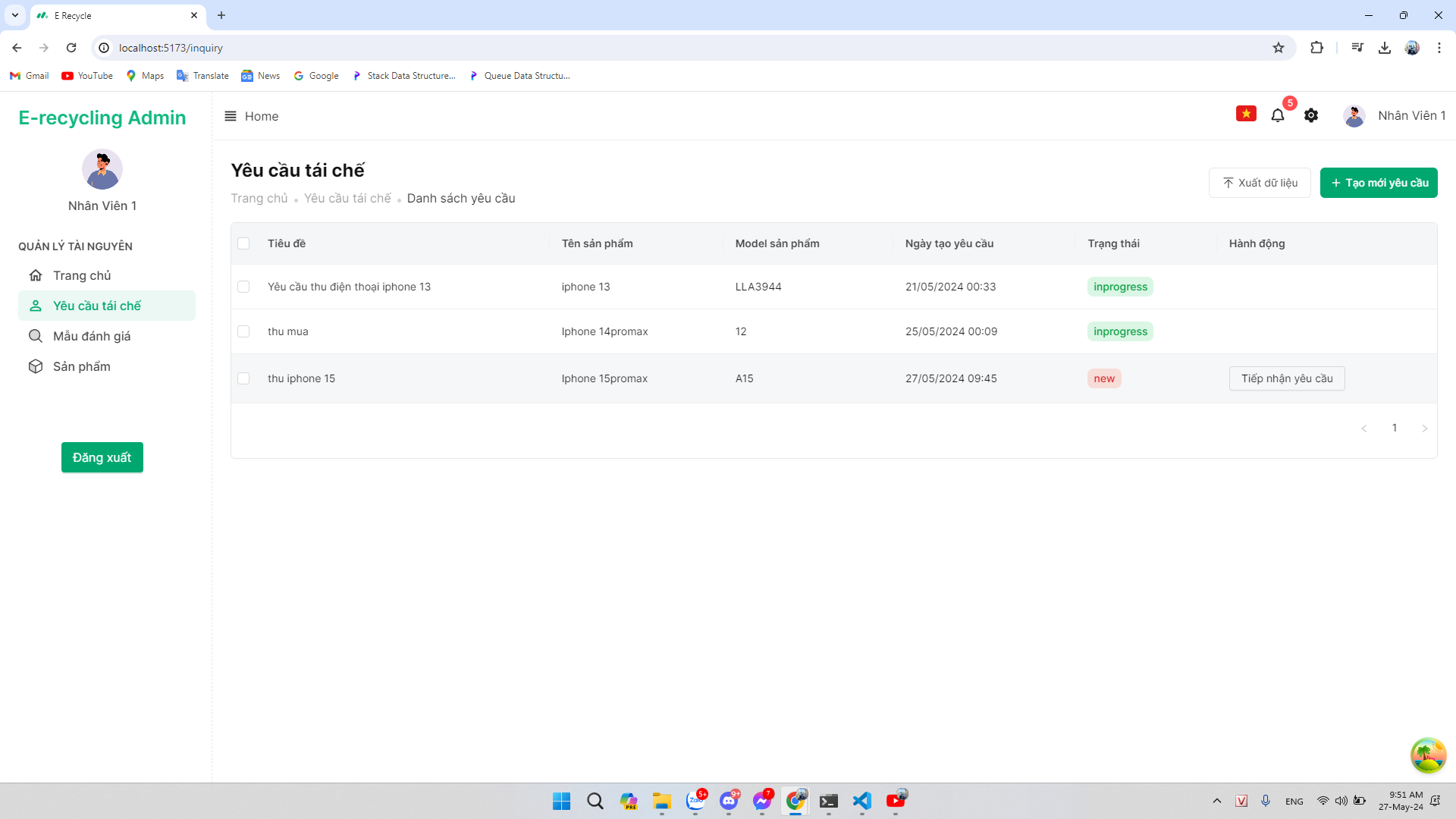
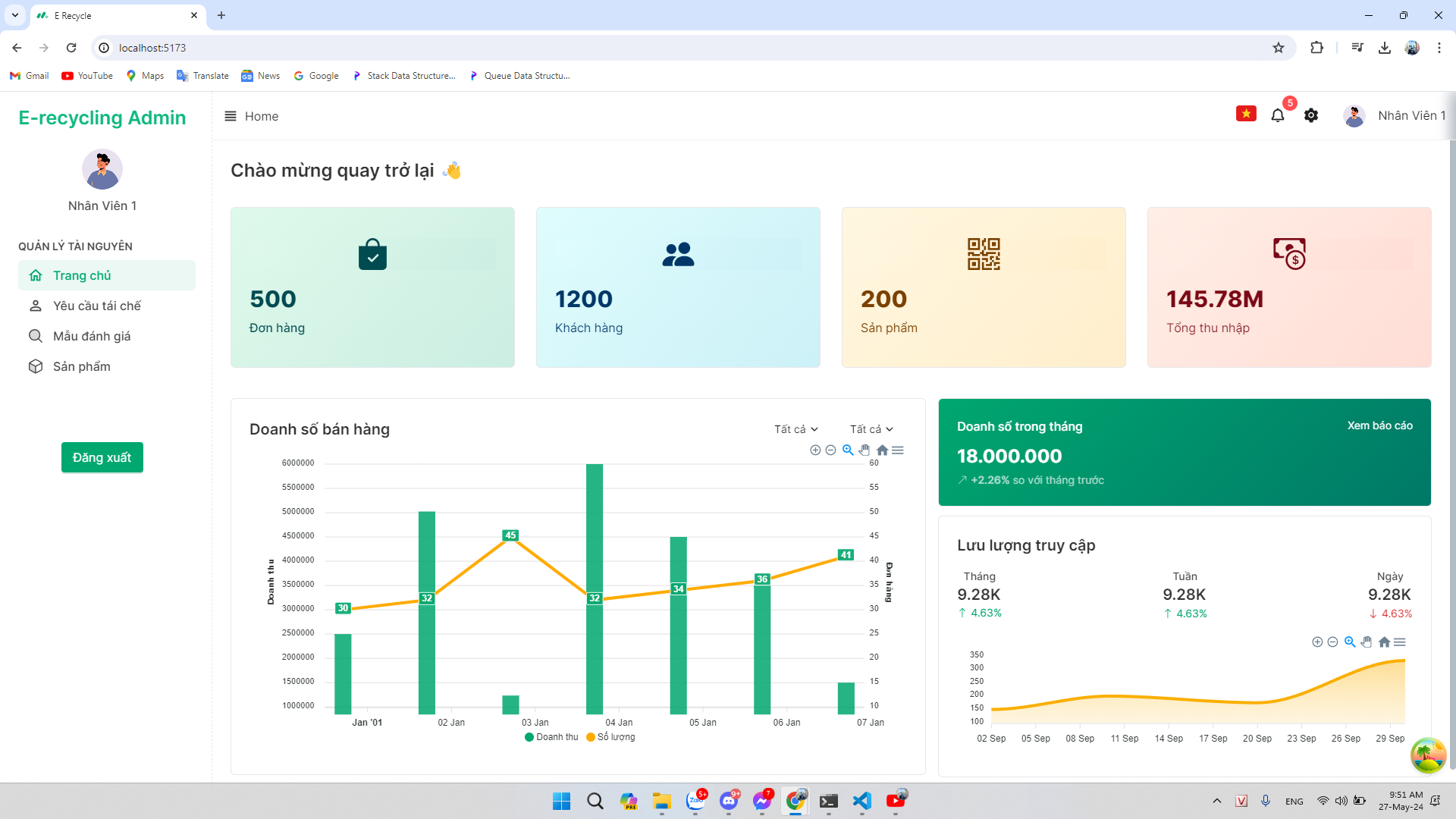
## **5.2 Giao diện của hệ thống**

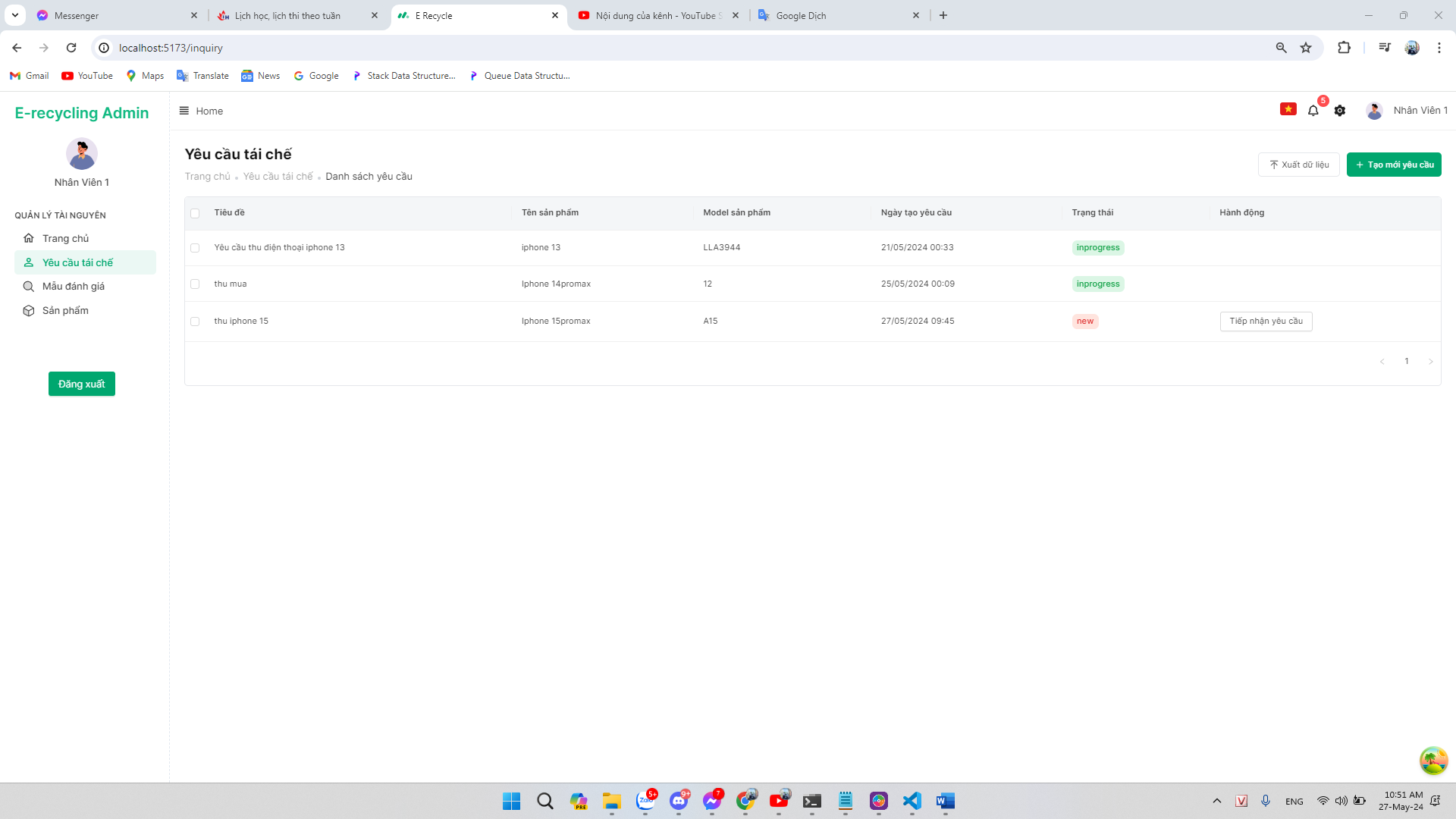
**Giao diện cho người dùng**





**Giao diện cho Admin**

****

****

# **Chương 6. Kết luận**

Việc áp dụng kiến trúc microservice vào mô hình tái chế thiết bị điện tử mang lại nhiều lợi ích vượt trội so với các kiến trúc truyền thống. Kiến trúc microservice cho phép phân tách các chức năng của hệ thống thành các dịch vụ độc lập, dễ dàng quản lý và bảo trì. Điều này giúp cải thiện tính linh hoạt, độ tin cậy và khả năng mở rộng của hệ thống. Các dịch vụ nhỏ có thể được phát triển, triển khai và mở rộng một cách độc lập, giảm thiểu rủi ro và tăng tốc độ phát triển.

Ngoài ra, việc áp dụng microservice giúp tối ưu hóa quy trình tái chế bằng cách tích hợp linh hoạt các công nghệ mới nhất và dễ dàng điều chỉnh theo yêu cầu thực tế. Ví dụ, các dịch vụ có thể được cập nhật để phù hợp với các tiêu chuẩn môi trường mới nhất hoặc để tận dụng các kỹ thuật tái chế tiên tiến mà không ảnh hưởng đến toàn bộ hệ thống.

Tuy nhiên, việc triển khai kiến trúc microservice đòi hỏi một kế hoạch cẩn thận và sự đầu tư đáng kể vào hạ tầng công nghệ cũng như khả năng quản lý. Các thách thức về tích hợp, giám sát và bảo mật cần được giải quyết để đảm bảo sự hoạt động mượt mà và an toàn của hệ thống.

Tóm lại, mô hình tái chế thiết bị điện tử sử dụng kiến trúc microservice không chỉ nâng cao hiệu quả và khả năng mở rộng mà còn tạo điều kiện thuận lợi cho việc cập nhật và cải tiến liên tục. Đây là một hướng đi tiềm năng để đáp ứng những thách thức ngày càng tăng trong việc quản lý và tái chế thiết bị điện tử trong kỷ nguyên số hiện đại.

# **Chương 7. Tài liệu tham khảo**

**Tài liệu:**

[https://vsi-international.com/microservice-api-ung-dung-trong-thuc-te/#:~:text=Hiểu%20một%20cách%20đơn%20giản,và%20kiểm%20tra%20riêng%20biệt.](https://vsi-international.com/microservice-api-ung-dung-trong-thuc-te/#:~:text=Hi%E1%BB%83u%20m%E1%BB%99t%20c%C3%A1ch%20%C4%91%C6%A1n%20gi%E1%BA%A3n,v%C3%A0%20ki%E1%BB%83m%20tra%20ri%C3%AAng%20bi%E1%BB%87t.)

<https://topdev.vn/blog/phat-trien-phan-mem-theo-kien-truc-microservice/>

**Giao diện:** <https://mui.com/store/previews/minimal-dashboard/>

<https://vite-react-ts-admin.netlify.app/#/dashboard/app>

<https://ant.design/>

[https://www.figma.com/design/hjxMnGUJCjY7pX8lQbS7kn/[Preview]-Minimal-Web.v5.4.0?node-id=0-1](https://www.figma.com/design/hjxMnGUJCjY7pX8lQbS7kn/%5BPreview%5D-Minimal-Web.v5.4.0?node-id=0-1)