**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM**

A logo with a red and blue letter and a drop of water

Description automatically generated

**DỊCH VỤ TÁI CHẾ THIẾT BỊ ĐIỆN TỬ CŨ**

***Sinh viên thực hiện***

1. Đỗ Minh Hào - 20086161
2. Bùi Hoàng Giang - 20088591
3. Nguyễn Quang Thạch - 20112881

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU - 2 -](#_Toc167053282)

[1.1 Tổng quan - 2 -](#_Toc167053283)

[1.2 Mục tiêu đề tài - 3 -](#_Toc167053284)

[1.3 Chức năng của hệ thống - 3 -](#_Toc167053285)

[1.3.1 Chức năng người dùng - 4 -](#_Toc167053286)

[1.3.2 Chức năng quản trị viên - 4 -](#_Toc167053287)

[1.4 Mô tả - 4 -](#_Toc167053288)

[1.4.1 Mô tả của hệ thống - 4 -](#_Toc167053289)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT - 5 -](#_Toc167053290)

[2.1 Node.js - 5 -](#_Toc167053291)

[2.2 React.js - 6 -](#_Toc167053292)

[2.3 MYSQL - 7 -](#_Toc167053293)

[2.5 Kiến trúc phân lớp “Layered Architecture” - 9 -](#_Toc167053294)

[Chương 3: PHÂN TÍCH VÀ THIẾT KẾ - 13 -](#_Toc167053295)

[3.1 Thu Thập Yêu Cầu - 13 -](#_Toc167053296)

[3.2 UseCase Tổng Quát - 14 -](#_Toc167053297)

[3.3 Danh sách tác nhân và mô tả - 14 -](#_Toc167053298)

[3.4 Mô tả UseCase và hoạt động - 15 -](#_Toc167053299)

[3.5 Kiến trúc - 17 -](#_Toc167053300)

[Chương 4: HIỆN THỰC - 19 -](#_Toc167053301)

[4.1 Giao Diện - 19 -](#_Toc167053302)

[4.2 Dữ Liệu - 20 -](#_Toc167053303)

[Chương 5: KẾT LUẬN - 21 -](#_Toc167053304)

[5.1 Kết quả đạt được - 21 -](#_Toc167053305)

[5.2 Hạn chế - 21 -](#_Toc167053306)

[5.3 Hướng phát triển - 21 -](#_Toc167053307)

[**Tài Liệu Tham Khảo** - 21 -](#_Toc167053308)

# **CHƯƠNG 1: GIỚI THIỆU**

* 1. **Tổng quan**

Hệ thống dịch vụ Tái chế thiết bị điển tử cũ, mô tả tổng quan đây là hệ thống có khả năng trao đổi thu mua và yêu cầu xử lý các thiết bị điện tử cũ điển hình như (Điện thoại, máy tính,…). Hệ thống được thiết kế cho cả người quản trị và người dùng môi trường trao đổi sản phẩm điện tử cũ. Đối vời người quản trị (người thu mua sản phẩm thiết bị điện tử cũ), đây là hệ thống giúp họ có thể thu mua sản phẩm một cách dễ dàng và xử lý các yều cầu bán các sản phẩm cũ, qua đó họ có thể quyết định sản phẩm sẽ được tái chế hay tiêu huỷ. Về phía của người dùng (người thanh lý các sản phẩm), là một nơi tiện lợi để thanh lý các mặt hàng điện tử cũ của bản thân mà không cần phải khó khăn trong việc di chuyển xa để đến tận nơi các cửa hàng mua bán thiết bị cũ. Đồng thời qua đó người dùng có thể cho người mua thấy được chất lượng của sản phẩm của mình. Tất cả những điều đó tạo ra một hệ thống tốt cho người mua và người bán như một môi trường trao đổi mua bán hàng hoá một cách an toàn và tiện ích.

* 1. **Mục tiêu đề tài**

Mục tiêu của hệ thống dịch vụ tái chế thiết bị điện tử cũ, nhằm tạo ra một hệ thống giống như là một môi trường trao đổi hàng hoá dành cho người mua và người bán về các thiết bị điển tử cũ.

Giúp đỡ người quản trị (người thu mua sản phẩm) quản lý các yêu cầu bán các thiết bị cũ, và quản lý tái chế hay tiêu huỷ sản phảm dựa trên tình trạng.

Giúp người dùng (người bán các sản phẩm diện tử cũ) có thể rao bán sản phẩm cũ của bản thân qua hệ thống web một cách tiện lợi và tiết kiệm thời gian, chi phí di chuyển. Đồng thời hệ thống sẽ đảm bảo an toàn cho mỗi giao dịch và tài khoản của người dùng.

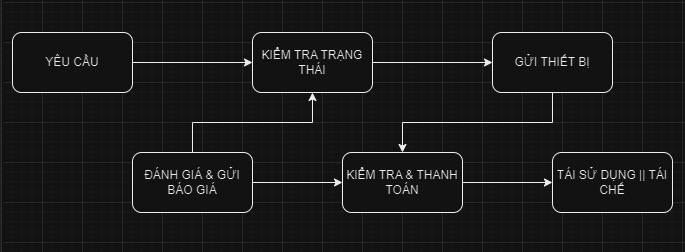
Với nền tảng tảng web có thể:

* Người dùng:

- Đăng yều cầu bán sản phẩm

- Xem phản hồi, xem giá sản phẩm khi bán

- Xem tình trạng đơn hàng trong quá trình xử lý

* Người quản trị:
* Xử lý các yêu cầu của khách hàng
* Cập nhật tình trạng đơn hàng
* Báo giá sản phẩm
* Xem báo cáo thống kê (sản phẩm, tài chình, đơn hàng)
  1. **Chức năng của hệ thống**
     1. Chức năng người dùng
     2. Chức năng quản trị viên
  2. **Mô tả** 
     1. ****Mô tả của hệ thống

*Bảng mô tả quy trình hoạt động hệ thống*

1. **Tạo Yêu Cầu:**

Người dùng truy cập trang web và tạo yêu cầu gửi đến đơn vị bằng cách điền vào biểu mẫu trực tuyến. Thông tin bao gồm chi tiết về thiết bị và các yêu cầu cụ thể từ người dùng.

1. **Đánh Giá Sơ Bộ:**

Đơn vị nhận yêu cầu từ người dùng và tiến hành đánh giá sơ bộ. Quá trình này bao gồm kiểm tra thông tin yêu cầu và tình trạng thiết bị. Sau khi đánh giá, đơn vị gửi lại báo giá kèm nhận xét về yêu cầu đó.

1. **Kiểm Tra Trạng Thái:**

Người dùng có thể kiểm tra trạng thái của yêu cầu bằng cách sử dụng số điện thoại của mình. Khi yêu cầu đã có báo giá, người dùng có thể xác nhận việc gửi thiết bị đến đơn vị.

1. **Nhận Thiết Bị và Thanh Toán:**

Đơn vị nhận thiết bị từ người dùng, tiến hành kiểm tra lại để xác nhận tình trạng và các thông tin liên quan. Sau khi kiểm tra xong, đơn vị sẽ thực hiện thanh toán cho người dùng.

1. **Kiểm Tra Trạng Thái Tiếp Theo:**

Người dùng có thể tiếp tục kiểm tra trạng thái của thiết bị thông qua trang trạng thái trên website. Tại đây, họ có thể biết được tiến trình của thiết bị sau khi gửi.

1. **Tái Sử Dụng hoặc Tiêu Hủy:**

Đơn vị tiến hành các bước tái sử dụng hoặc tiêu hủy thiết bị theo quy trình nội bộ sau khi đã hoàn tất việc kiểm tra và thanh toán.

1. **Thống Kê và Báo Cáo:**

Đơn vị có thể thống kê các giao dịch, tạo báo cáo và xuất dữ liệu ra file Excel để quản lý hiệu quả hơn.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **Node.js**

A logo for a software development company

Description automatically generated

Hình 2.1 Node.js development

* **Tổng quan về Node.js:**

Node.js là một nền tảng phát triển độc lập, được xây dựng dựa trên môi trường thực thi JavaScript của Google Chrome. Nền tảng này cho phép phát triển các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.

* **Ưu điểm của Node.js:**

1. **Đơn ngôn ngữ:** Node.js sử dụng JavaScript, một trong những ngôn ngữ lập trình phổ biến nhất, cho phép phát triển ứng dụng đồng nhất từ client đến server.
2. **Kiến trúc hiệu quả:** Với kiến trúc hướng sự kiện và cơ chế non-blocking I/O, Node.js mang lại hiệu suất cao và khả năng xử lý đồng thời lớn, làm cho nó trở thành lựa chọn lý tưởng cho các ứng dụng cần xử lý nhiều kết nối cùng một lúc.
3. **Đa nền tảng:** Node.js có thể chạy trên nhiều hệ điều hành khác nhau như macOS, Windows và Linux, đảm bảo tính linh hoạt và dễ dàng triển khai trong môi trường phát triển đa dạng.
4. **Cộng đồng mạnh mẽ:** Có một cộng đồng lớn và tích cực hỗ trợ Node.js, với hàng ngàn mô-đun miễn phí có sẵn trên npm (Node Package Manager), giúp tăng tốc độ phát triển và cung cấp các giải pháp sẵn có cho nhiều vấn đề phức tạp.
5. **Miễn phí:** Node.js là một nền tảng mã nguồn mở, cung cấp miễn phí cho các nhà phát triển, giúp giảm chi phí phát triển phần mềm. Nhờ vào những ưu điểm này, Node.js đã trở thành một trong những lựa chọn hàng đầu cho các nhà phát triển web và ứng dụng di động trên toàn thế giới.

## **React.js**



* **Tổng quan về React:**

React là một thư viện JavaScript được sử dụng để xây dựng giao diện người dùng tương tác. Nó khuyến khích việc tạo ra các thành phần giao diện có thể tái sử dụng, và cho phép hiển thị dữ liệu thay đổi theo thời gian. Rất nhiều người sử dụng React làm phần giao diện người dùng (View) trong mô hình MVC (Model-View-Controller).

* **Ưu điểm của React:**

1. **Sử dụng DOM ảo:** React sử dụng một cấu trúc dữ liệu gọi là DOM ảo (Virtual DOM) là một đối tượng JavaScript. Điều này cải thiện hiệu suất ứng dụng, vì DOM ảo JavaScript thường nhanh hơn DOM thực sự và giúp tránh trạng thái đồng bộ hóa tốn kém.
2. **Có thể sử dụng trên máy khách và máy chủ:** React có thể được sử dụng không chỉ trên phía máy khách (client-side) mà còn trên phía máy chủ (server-side), cung cấp khả năng rendering đa nền tảng và tối ưu hóa SEO.
3. **Các mẫu thành phần và dữ liệu cải thiện khả năng đọc:** React thúc đẩy việc sử dụng các mẫu thành phần (component-based) và quản lý dữ liệu một cách hiệu quả, giúp duy trì và phát triển các ứng dụng lớn một cách dễ dàng và hợp lý hơn.

→ Nhờ vào những ưu điểm này, React đã trở thành một trong những công cụ phát triển giao diện người dùng phổ biến nhất hiện nay, được sử dụng rộng rãi trong cả ứng dụng web và di độngTop of Form

## **MYSQL**

A logo for a software company

Description automatically generated

* **Tổng quan về MySQL:**

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (RDBMS) được phát triển bởi công ty MySQL AB và hiện thuộc sở hữu của Oracle Corporation. Nó sử dụng ngôn ngữ SQL (Structured Query Language) để quản lý và thao tác với cơ sở dữ liệu. MySQL được sử dụng rộng rãi trong các ứng dụng web và doanh nghiệp nhờ vào tính hiệu quả, tính ổn định và dễ sử dụng.

* **Ưu điểm của MySQL**
* Mã nguồn mở và miễn phí: Giảm chi phí cho doanh nghiệp.
* Hiệu suất cao: Xử lý nhanh lượng lớn truy vấn đồng thời.
* Khả năng mở rộng: Dễ dàng mở rộng và tối ưu hóa.
* Bảo mật cao: Hỗ trợ SSL, quyền truy cập chi tiết, giao dịch an toàn.
* Hỗ trợ giao dịch: Đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu.
* Cộng đồng và hỗ trợ mạnh mẽ: Tài liệu, diễn đàn, plugin phong phú.
* Tương thích đa nền tảng: Hỗ trợ Windows, Linux, MacOS, Unix.
* **Nhược điểm của MySQL**
* Hạn chế xử lý truy vấn phức tạp: Khó khăn với truy vấn và phân tích dữ liệu lớn.
* Thiếu tính năng tiên tiến: Không hỗ trợ đầy đủ cho dữ liệu không gian và thời gian thực.
* Vấn đề khóa và deadlock: Gặp deadlock trong trường hợp phức tạp.
* Hạn chế về phân tích và báo cáo: Không có công cụ tích hợp mạnh mẽ cho phân tích và báo cáo.

**2.4 Một số công nghệ hõ trợ khác**

* **Ant Design (AntD):** Bộ công cụ giao diện người dùng mạnh mẽ giúp tạo ra các phần tử giao diện nhất quán và chuyên nghiệp.
* **Axios:** Thư viện giúp dễ dàng thực hiện các yêu cầu HTTP từ client tới server, giúp giao tiếp giữa front-end và back-end hiệu quả.
* ***Server:***
* **ExpressJS:** Framework web ứng dụng Node.js nhẹ và mạnh mẽ, giúp tạo ra các API RESTful phục vụ cho giao tiếp giữa front-end và back-end.
* **JWT (JSON Web Token):** Sử dụng để xác thực và bảo mật các thông tin giao tiếp giữa client và server.
* **Crypto:** Thư viện dùng để mã hóa dữ liệu, đảm bảo tính bảo mật.
* **CORS:** Được sử dụng để xử lý các yêu cầu từ các nguồn khác nhau, đảm bảo trang web có thể hoạt động một cách mượt mà mà không gặp vấn đề về bảo mật.
* **Multer:** Thư viện middleware cho Node.js giúp xử lý việc upload file, hỗ trợ việc người dùng gửi thiết bị qua hình ảnh hoặc tài liệu.

## **Kiến trúc phân lớp “Layered Architecture”**

Kiến trúc phân lớp (*Layered architecture* - còn được gọi là *N-tier architecture*) là kiến trúc phổ biến từ xưa cho tới ngày nay. Mỗi lớp trong kiến trúc này có chức năng riêng và tương tác với lớp ngay trên hoặc dưới nó. Các mô hình sử dụng kiến trúc này bao gồm MVC, MVVM,...

Các thành phần (component) trong kiến trúc phân lớp được tổ chức thành các lớp logic ngang, mỗi lớp thực hiện một vai trò cụ thể trong ứng dụng (ví dụ: logic trình bày - User interface hoặc logic nghiệp vụ - Business).

Mặc dù không có con số cụ thể về số lượng và loại lớp phải có, kiến trúc phân lớp thường bao gồm bốn lớp tiêu chuẩn: **presentation**, **business**, **persistence**, và **database**.

A diagram of components on a computer

Description automatically generated

Mỗi layer đóng vai trò như một lớp trừu tượng, che giấu chi tiết triển khai bên dưới và chỉ cung cấp một giao diện hoặc giao thức cho các layer phía trên. Các layer thường được sắp xếp theo một thứ tự nhất định, từ layer thấp nhất (hay gần giao diện vật lý) đến layer cao nhất (gần với người dùng cuối).

Ví dụ, trong một ứng dụng web, bạn có thể có các layer sau:

1. **Layer Dữ liệu (Data Layer):** Layer này chịu trách nhiệm lưu trữ và truy cập dữ liệu. Nó có thể bao gồm cơ sở dữ liệu và các dịch vụ liên quan như ORM (Object-Relational Mapping).
2. **Layer Logic Kinh doanh (Business Logic Layer):** Layer này chứa các quy tắc kinh doanh và logic xử lý chính của ứng dụng. Nó có thể xử lý các yêu cầu từ layer giao diện và tương tác với layer dữ liệu để truy xuất hoặc cập nhật dữ liệu.
3. **Layer Giao diện (Presentation Layer):** Layer này chịu trách nhiệm hiển thị giao diện người dùng và tương tác với người dùng cuối. Nó có thể bao gồm các thành phần như HTML/CSS/JavaScript cho ứng dụng web, hoặc các giao diện người dùng khác như ứng dụng di động.

Mỗi layer có thể tương tác với các layer khác thông qua các giao diện hoặc API, giúp tạo ra một hệ thống có tính tái sử dụng cao và dễ dàng mở rộng.

* **Ưu điểm:**
* **Tính Modularity (Tính mô-đun)**: Kiến trúc layer cho phép hệ thống được phân chia thành các phần nhỏ, gọi là lớp, mỗi lớp đảm nhận một chức năng riêng biệt. Điều này làm cho hệ thống dễ quản lý và dễ mở rộng.
* **Tính Tương tác Thấp (Low Coupling)**: Các lớp không phụ thuộc trực tiếp lẫn nhau, mà thay vào đó chúng chỉ tương tác thông qua giao diện chuẩn hóa. Điều này giúp giảm sự phụ thuộc giữa các thành phần của hệ thống, làm cho việc thay đổi hoặc nâng cấp dễ dàng hơn.
* **Tính Đa dạng (Abstraction)**: Mỗi lớp ẩn đi chi tiết triển khai của các lớp dưới cùng, chỉ cung cấp một giao diện cho các lớp trên cùng. Điều này giúp tạo ra một cách nhìn trừu tượng và dễ hiểu về cách hoạt động của hệ thống.
* **Nhược điểm:**
* **Tăng Overhead (Tăng Chi phí)**: Mỗi lớp thêm vào đều tăng overhead cho hệ thống, từ việc tạo và duy trì các giao diện đến việc truyền thông qua các lớp. Điều này có thể ảnh hưởng đến hiệu suất của hệ thống, đặc biệt là trong các hệ thống có số lượng lớn các lớp.
* **Khó Khắc phục Sự cố (Troubleshooting Difficulty)**: Khi một sự cố xảy ra, việc xác định lớp nào gây ra sự cố và nơi nó xảy ra có thể trở nên phức tạp. Điều này đặc biệt đúng khi mối quan hệ giữa các lớp phức tạp.
* **Một số Khía cạnh Khác:**
* **Tính Bảo mật (Security)**: Kiến trúc layer có thể được sử dụng để cải thiện bảo mật bằng cách tách biệt các lớp và áp dụng các biện pháp bảo mật tại mỗi lớp.
* **Tính Tái sử dụng (Reusability)**: Các lớp có thể được thiết kế để có thể tái sử dụng, giúp giảm thiểu việc viết lại mã và tăng tốc độ phát triển phần mềm.
* **Trường hợp nên dùng kiến trúc phân lớp**

Nếu dự án có những hạn chế nhất định về thời gian và ngân sách, kiến trúc phân lớp là một sự lựa chọn thích hợp vì các lý do chính sau:

* Kiến trúc phân lớp thuộc loại kiến trúc monolithic nên nó không mang sự phức tạp như kiến trúc phân tán.
* Dựa trên mức độ phổ biến của kiến trúc phân lớp, hầu hết các nhà phát triển điều quen thuộc với nó, kiến việc vận hành, phát triển dự án trở nên dễ dàng hơn.
* Kiến trúc phân lớp thuộc nhóm kiến trúc phân vùng kỹ thuật nên việc sử dụng kiến trúc này phù hợp với nhiều tổ chức khác nhau do các nhóm được chia thành nhóm chuyên trách kỹ thuật như nhóm FE, nhóm BE, nhóm database, ...
* **Trường hợp không nên dùng kiến trúc phân lớp**
* Nếu dự án của bạn đòi hỏi khả năng mở rộng, độ đàn hồi, khả năng chịu lỗi và hiệu năng thì kiến trúc phân lớp không phải là sự lựa chọn phù hợp. Nguyên nhân chủ yếu do bản chất của loại kiến trúc monolithic là khó mở rộng và khả năng chịu lỗi thấp. Một component trong hệ thống bị lỗi có thể dẫn tới lỗi cả ứng dụng.
* Nếu dự án của bạn có nhiều sự thay đổi về domain. Ví dụ, ứng dụng thương mại điện tử cần thêm tính năng livestream. Để hoàn thành yêu cầu này, bạn cần sửa nhiều lớp khác nhau như sửa schema của database (lớp database) gồm thêm bảng mới hoặc cột mới hoặc cả hai, sửa SQL trong lớp persistence, sửa logic của lớp business và sửa cả lớp presentation (giả sử cần thêm API mới). Việc thay đổi này có thể kém linh hoạt và mất nhiều thời gian vì scope phải sửa nhiều.
* Nếu team của bạn tổ chức thành các nhóm đa chức năng xử lý chung domain, nghĩa là trong cùng một team sẽ có nhiều thành viên có chuyên môn trên nhiều mảng khác nhau như FE, BE, designer, DA, ... thì việc này không hề phù hợp với bản chất của kiến trúc phân lớp (do thuộc phân vùng kỹ thuật).
* **Khả năng kiểm thử**

Trong kiến thúc này, các thành phần trong thuộc vào một tầng cụ thể, các tầng khác có thể được mô phỏng hoặc khai thác, giúp cho kiến trúc này tương đối dễ kiểm thử. Một nhà phát triển có thể giả lập một thành phần trình bày hoặc màn hình để cô lập thử nghiệm trong một thành phần nghiệp vụ, cũng như mô phỏng tầng nghiệp vụ để kiểm tra chức năng màn hình nhất định.

* **Hiệu năng**

Mặc dù đúng là một số kiến trúc phân tầng có thể hoạt động tốt, nhưng kiến trúc này không phù hợp với các ứng dụng hiệu năng cao do tính không hiệu quả của việc phải đi qua nhiều tầng của kiến trúc để đáp ứng yêu cầu nghiệp vụ.

* **Khả năng mở rộng**

Do xu hướng triển khai nguyên khối và liên kết chặt chẽ của kiến trúc này, các ứng dụng được xây dựng bằng cách sử dụng mẫu kiến trúc này thường khó mở rộng quy mô. Chúng ta có thể mở rộng quy mô kiến trúc phân tầng bằng cách tách các tầng thành các triển khai vật lý riêng biệt hoặc sao chép toàn bộ ứng dụng thành nhiều nút, nhưng nhìn chung mức độ chi tiết quá rộng, khiến việc mở rộng quy mô trở nên tốn kém.

* **Dễ phát triển**

Tính dễ phát triển nhận được điểm tương đối cao, chủ yếu là do mô hình này quá nổi tiếng và không quá phức tạp để thực hiện. Bởi vì hầu hết các công ty phát triển ứng dụng bằng cách tách các bộ kỹ năng theo tầng (trình diễn, nghiệp vụ, cơ sở dữ liệu), kiến trúc này trở thành lựa chọn tự nhiên cho hầu hết việc phát triển ứng dụng kinh doanh. Mối liên hệ giữa cơ cấu tổ chức và truyền thông của công ty với cách thức phát triển phần mềm được vạch ra là cái được gọi là định luật Conway. Bạn có thể Google "Conway’s law" để có thêm thông tin về mối tương quan hấp dẫn này.

# **Chương 3: PHÂN TÍCH VÀ THIẾT KẾ**

## **Thu Thập Yêu Cầu**

|  |  |  |
| --- | --- | --- |
| Câu hỏi | Câu trả lời | Người hỏi |
| Anh chị có cần hệ thống giúp đỡ xử lý mua bán các thiết bị điện tử cũ không ? | Có |  |
| Anh chị cần một hệ thống để sử dụng như thế nào ? | Hệ thống dễ sử dụng |  |
| Anh chị cần giao diện của hệ thống sẽ như thế nào ? | Tối giản |  |
| Anh chị có muốn hệ thống có thể gửi hình ảnh ? | Có |  |
| Anh chị có muốn để lại ghi chú sau khi gửi hình ảnh để mô tả thêm về tình trạng của thiết bị ? | Có |  |
| Anh chị muốn thanh toán như thế nào ? | Chuyển khoản |  |
|  |  |  |

## **UseCase Tổng Quát**

**A diagram of a diagram

Description automatically generated**

## **Danh sách tác nhân và mô tả**

|  |  |
| --- | --- |
| Tác nhân | Mô tả tác nhân |
| User | * Yêu cầu báo gía * Xem báo gía * Xem trạng thái * Xem tình trạng đơn hàng |
| Admin | * Đăng nhập * Báo giá * Thanh toán * Kiểm tra yêu cầu báo giá * Xử lý thiết bị |

## **Mô tả UseCase và hoạt động**

|  |  |
| --- | --- |
| * **Tên use case**: UC\_Đăng nhập | |
| * **Mô tả sơ lược**: | |
| * **Actor chính**: | |
| * **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 | 1. 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 |

A diagram of a project

Description automatically generatedA diagram of a computer program

Description automatically generated

## **Kiến trúc**

Kiến trúc sẽ có 2 phần là Client và Sever, ở trong mỗi phần sẽ có sự phân tầng khác nhau

* **Presentation Layer**

Chức năng:

* Giao diện người dùng: Hiển thị dữ liệu và cung cấp giao diện cho người dùng tương tác với ứng dụng.
* Xử lý đầu vào người dùng: Nhận và xử lý các thao tác từ người dùng như nhấp chuột, nhập liệu, và các sự kiện giao diện khác.
* Chuyển đổi dữ liệu: Chuyển đổi dữ liệu từ tầng business thành dạng có thể hiển thị và ngược lại.

Thành phần chính:

* Client Side (View): Các thành phần giao diện người dùng (UI)
* Server Side (Router): Định tuyến các yêu cầu từ client đến các controller tương ứng.
* **Business Layer (Tầng nghiệp vụ)**

Chức năng:

* Xử lý logic nghiệp vụ: Thực hiện các quy tắc và xử lý logic nghiệp vụ của ứng dụng.
* Điều phối dữ liệu: Điều phối và xử lý dữ liệu giữa presentation layer và persistence layer.
* Thực hiện các tác vụ nghiệp vụ: Thực hiện các tác vụ cụ thể của ứng dụng như tính toán, kiểm tra điều kiện, và các quy trình nghiệp vụ khác.

Thành phần chính:

* Client Side (Service): Xử lý logic nghiệp vụ trên client và gọi API tới server.
* Server Side (Controller, Service): Xử lý yêu cầu từ router, thực hiện logic nghiệp vụ và tương tác với tầng persistence.
* **Persistence Layer (Tầng lưu trữ)**

Chức năng:

* + - Quản lý dữ liệu: Xử lý lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu.
    - Thao tác CRUD: Thực hiện các thao tác tạo, đọc, cập nhật, và xóa dữ liệu.
    - Quản lý kết nối dữ liệu: Quản lý kết nối tới cơ sở dữ liệu và tối ưu hóa truy vấn.

Thành phần chính:

* + - Model: Tương tác với cơ sở dữ liệu và quản lý các thao tác dữ liệu.
* **Database Layer (Tầng cơ sở dữ liệu)**

Chức năng:

* Lưu trữ dữ liệu: Lưu trữ tất cả dữ liệu của ứng dụng.
* Quản lý dữ liệu: Cung cấp các chức năng quản lý và truy xuất dữ liệu một cách hiệu quả.

Thành phần chính:

Database: Cơ sở dữ liệu MySql.

# **Chương 4: HIỆN THỰC**

## **Giao Diện**

Một số hình ảnh về giao diện

A computer chip with green recycle symbol on it

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer chip with a login form

Description automatically generated

## **Dữ Liệu**

* **Sử dụng cơ sở dữ liệu SQL**

Với một số bảng dữ liệu như:

* **Users**(id, username, password).
* **Devices**(device\_id,device\_name,device\_type,image\_Text,notes\_Text).
* **Disposal\_recycle**(disposal\_recycle\_id,reception\_id,disposal\_recycle\_date,disposal\_recycle\_method).
* **Customers(**customer\_id, full\_name, phone\_number, email, address**).**

# **Chương 5: KẾT LUẬN**

1. **Kết quả đạt được**

* Đã tạo được hệ thống dịch vụ tái chế thiết bị điện tử cũ. Hoàn thành các nghiệp vụ cơ bản như: gửi yêu cầu, xem yêu cầu và xem trạng thái yêu cầu (Người dùng)
* Đối với người quản lý đã hoàn thành các chức năng nghiệp vụ của người quản lý như: Xử lý yêu cầu, báo gía, xem yêu cầu và xác nhận tiêu huỷ hoặc tái chế.

1. **Hạn chế**

* Hạn chế trong việc đăng nhập của khách hàng(Khách hàng không cần phải đăn nhập)
* Tài khoản của người quản trị phải tạo riêng ở bên ngoài CSDL
* Nhiều phần còn thiếu sót như đăng xuất của quản trị viên

Ngoài ra còn nhiều mặt hạn chế như việc thanh toán qua thẻ hoặc xử lý thanh toán, đây cũng là một hạn chế nguy hiểm cho việc thanh toán và xác minh người gửi yêu cầu bán hàng thanh lý các sản phẩm điện tử. Nhiều sự hạn chế trong quá trình tạo lập hệ thống.

1. **Hướng phát triển**

Cố gắng khắc phục nhữn hạn chế và sửa đổi hệ thống sao cho phù hợp với việc kinh doanh và quản lý

### **Tài Liệu Tham Khảo**

<https://github.com/>

<https://www.youtube.com/>

<https://viblo.asia\>

https://stackoverflow.blog