

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



**TRẦN HỮU HOÀNG**

**PHÁT TRIỂN HỆ THỐNG WEBSITE ĐẤU GIÁ TRỰC  
TUYẾN THEO THỜI GIAN THỰC**

**Mã số sinh viên: 1951012033**

**ĐỒ ÁN NGÀNH  
NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: ThS. Hồ Hương Thiên**

**TP. HỒ CHÍ MINH, 2023**

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



**TRẦN HỮU HOÀNG**

**PHÁT TRIỂN HỆ THỐNG WEBSITE ĐẤU GIÁ TRỰC  
TUYẾN THEO THỜI GIAN THỰC**

**Mã số sinh viên: 1951012033**

**ĐỒ ÁN NGÀNH  
NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: ThS. Hồ Hương Thiên**

**TP. HỒ CHÍ MINH, 2023**

## **LỜI CẢM ƠN**

Lời đầu tiên cho em xin phép gửi lời cảm ơn đến thầy Hồ Hướng Thiên đã hướng dẫn em trong suốt hơn 2 tháng vừa qua. Nhờ sự tận tâm của thầy và sự nỗ lực của cá nhân bản thân em mà em cũng đã hoàn thành tốt nhiệm vụ được giao. Em cũng chân thành cảm ơn các quý thầy cô khác trường đại học mở thành phố Hồ Chí Minh đã hết lòng hỗ trợ em về mặt kỹ năng cũng như kiến thức vô cùng quý báu để em có thể áp dụng vào chuyên ngành của bản thân trong suốt thời gian em theo học tại trường.

Đối với bản thân em hơn những tháng vừa qua có thể là một thời gian không quá dài nhưng những kiến thức em đã tích lũy được là thực sự đáng trân trọng để em có thể hoàn thành tốt nghiên cứu nhưng do bản thân em vẫn còn thiếu sót về nhiều mặt nên chắc chắn trong quá trình thực hiện sẽ có nhiều thiếu sót nên em mong quý vị thầy cô có thể nhận được những phản hồi và góp ý từ thầy cô để em có thể cải thiện và tốt hơn trong tương lai để có giá trị f trong đời sống thực tiễn.

## **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

## **TÓM TẮT ĐỒ ÁN NGÀNH**

Trong thời đại phát triển hiện đại hiện nay, việc mua sắm trở nên dễ dàng hơn bao giờ hết. Khách hàng không chỉ quan tâm về chất lượng, kiểu dáng sản phẩm mà còn quan tâm đến giá cả của sản phẩm đó. Vậy nên để tạo ra một môi trường vừa có thể mua được giá ưu đãi vừa có tính cạnh tranh thì một trang website đấu giá thời gian thực là một sự lựa chọn vô cùng phù hợp. Hiện nay có rất nhiều ngôn ngữ lập trình và công nghệ mới có thể viết được website nhưng với đánh giá của bản thân thì PHP là ngôn ngữ tương đối thân thiện và dễ sử dụng cho việc phát triển ở phía server. Ở phía giao diện người dùng, để tối ưu hoá quá trình tải trang thì hiện nay có một thuật ngữ rất được ưa chuộng đó là single page application (SPA) nên sau khi cân nhắc em đã chọn VueJS để xây dựng phía giao diện và cuối cùng là sử dụng Mysql dành cho việc quản lý cơ sở dữ liệu.

## ABSTRACT

## MỤC LỤC

<b>DANH MỤC TỪ VIẾT TẮT .....</b>	<b>7</b>
<b>DANH MỤC HÌNH VẼ.....</b>	<b>8</b>
<b>MỞ ĐẦU.....</b>	<b>11</b>
<b>Chương 1. TỔNG QUAN ĐỀ TÀI.....</b>	<b>12</b>
1.1. Giới thiệu đề tài .....	12
1.2. Lý do đề tài.....	12
1.3. Bố cục đề tài.....	13
1.4. Mục tiêu nghiên cứu.....	13
<b>Chương 2. CƠ SỞ LÝ THUYẾT.....</b>	<b>14</b>
2.1. Giới thiệu về server side .....	14
2.1.1. Khái niệm.....	14
2.1.2. Ưu điểm.....	14
2.1.3. Nhược điểm.....	14
2.1.4. Giới thiệu về framework Laravel.....	15
2.2. Giới thiệu về client side .....	26
2.2.1. Khái niệm.....	26
2.2.2. Ưu điểm.....	26
2.2.3. Nhược điểm.....	27
2.2.4. Giới thiệu về framework VueJS.....	27
<b>Chương 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....</b>	<b>36</b>
3.1. Mô tả đề tài.....	36
3.2. Thiết kế cơ sở dữ liệu.....	37
3.2.1. Mô hình cơ sở dữ liệu và mối quan hệ giữa các bảng.....	37
3.2.2. Các bảng trong cơ sở dữ liệu.....	37
3.3. Giải thích mối quan hệ.....	43
3.4. Đặc tả usecase.....	44
3.4.1. Quản trị.....	44

3.4.2. Khách hàng .....	45
<b>Chương 4. THỰC NGHIỆM.....</b>	<b>46</b>
4.1. Cấu hình cài đặt .....	46
4.1.1. Mysql workbench 8.0 .....	46
4.1.2. PHP storm: .....	46
4.2. Giao diện chức năng .....	46
4.2.1. Quản trị .....	46
4.2.2. Khách hàng .....	52
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>55</b>
<b>PHỤ LỤC .....</b>	<b>56</b>



## **DANH MỤC TỪ VIẾT TẮT**

## DANH MỤC HÌNH VẼ

Hình 2.1.4.2.1: Cấu trúc viết routing .....	15
Hình 2.1.4.2.2: Cấu trúc của url .....	16
Hình 2.1.4.3.1: Mô hình MVC trong laravel .....	17
Hình 2.1.4.4.1: Dependency trong laravel .....	18
Hình 2.1.4.4.2: Danh sách các service provider .....	20
Hình 2.1.4.4.3: Repository trong laravel .....	21
Hình 2.1.4.4.4: Facades trong laravel .....	21
Hình 2.1.4.4.5: Observe trong laravel .....	22
Hình 2.1.4.5.1: Phần mềm Wampp .....	23
Hình 2.1.4.5.2: Bảng Environment Variables .....	24
Hình 2.1.4.5.3: Edit Environment Variables .....	25
Hình 2.1.4.5.4: Composer PHP - Laravel .....	25
Hình 2.1.4.5.5: Khởi tạo dự án laravel đầu tiên. ....	26
Hình 2.2.4.2.1: Lifecycle vuejs .....	29
Hình 2.2.4.3.1: Component trong Vuejs .....	30
Hình 2.2.4.5.1: Cài đặt Vue-loader .....	34
Hình 2.2.4.6.1: Cài đặt Axios .....	34
Hình 2.2.4.7.1: Cài đặt vue-router .....	34
Hình 2.2.4.9.1: Cài đặt PrmieVue .....	35
Hình 3.2.1.1.1: Mô hình cơ sở dữ liệu .....	37
Hình 4.2.1.1.1: Chức năng đăng nhập .....	46
Hình 4.2.1.1.2: Chức năng hiển thị danh sách khách hàng .....	47
Hình 4.2.1.1.3: Chức năng chỉnh sửa danh sách khách hàng .....	47
Hình 4.2.1.1.4: Chức năng xoá thông tin khách hàng .....	48
Hình 4.2.1.1.5: Chức năng hiển thị danh sách loại sản phẩm .....	48
Hình 4.2.1.1.6: Chức năng chỉnh sửa danh sách loại sản phẩm .....	49
Hình 4.2.1.1.7: Chức năng xoá danh sách loại sản phẩm .....	49
Hình 4.2.1.1.8: Chức năng hiển thị danh sách sản phẩm .....	50
Hình 4.2.1.1.9: Chức năng tạo sản phẩm .....	50
Hình 4.2.1.1.10: Chức năng hiển thị phiên .....	51

Hình 4.2.1.11 : Chức năng tạo phiên .....	51
Hình 4.2.2.1 : Chức năng đăng ký .....	52
Hình 4.2.2.2 : Chức năng đăng nhập .....	52
Hình 4.2.2.3 : Chức năng xem danh sách phiên đấu giá .....	53
Hình 4.2.2.4 : Chức năng đấu giá .....	53
Hình 4.2.2.5 : Chức năng hiển thị người đấu giá chiến thắng .....	54

## DANH MỤC BẢNG

Bảng 3.2.2.1: Bảng admins .....	38
Bảng 3.2.2.2: Bảng users .....	38
Bảng 3.2.2.3: Bảng categories .....	38
Bảng 3.2.2.4: Bảng products .....	39
Bảng 3.2.2.5: Bảng bids .....	39
Bảng 3.2.2.6: Bảng auctions .....	39
Bảng 3.2.2.7: Bảng product_auctions .....	39
Bảng 3.2.2.8: Bảng win_biddings .....	40
Bảng 3.2.2.9: Bảng histories .....	40
Bảng 3.2.2.10: Bảng transports .....	40
Bảng 3.2.2.11: Bảng transport_admins .....	41
Bảng 3.2.2.12: Bảng payments .....	41
Bảng 3.2.2.13: Bảng comments .....	41
Bảng 3.2.2.14: Bảng personal_access_tokens .....	42
Bảng 3.2.2.15: Bảng migration .....	42
Bảng 3.2.2.16: Bảng failed_jobs .....	42
Bảng 3.2.2.17: Bảng password_resets .....	42

## MỞ ĐẦU

## **Chương 1. TỔNG QUAN ĐỀ TÀI**

*Chương 1 khái quát về đề tài nghiên cứu, lý do chọn đề tài cũng như sự nhận xét sự phát triển của công nghệ đối với con người trong thời kì số hoá hiện nay. Thời kì 4.0.*

### **1.1. Giới thiệu đề tài**

Trước 2019 tình hình mua sắm online bắt đầu phát triển và dần trở nên phổ biến hơn ở nước ta cũng như trên khắp thế giới. Nhưng sau 2019, do tình hình dịch COVID nên bắt buộc thói quen mọi người phải thực sự thay đổi để thích nghi với tình hình dịch bệnh cũng như hướng đến một cuộc cải cách mới mang tính bước ngoặt trong thị trường mua sắm trực tuyến. Do đó việc mua sắm online trở nên rầm rộ hơn bao giờ hết. Bắt đầu sinh ra các sàn thương mại điện tử, các trang mua sắm như taobao, tiki, shopee và trong đó web đấu giá realtime là một nền tảng trực tuyến đầy hứa hẹn. Website này cho phép người dùng tham gia vào các phiên đấu giá trực tiếp và theo dõi sự thay đổi giá cả một cách chính xác và ngay lập tức. Không giống như các hình thức đấu giá truyền thống, ở đây, mọi sự tham gia và đặt giá diễn ra trực tuyến thông qua mạng internet, tạo điều kiện thuận lợi cho các người chơi từ khắp nơi trên thế giới. Đặc điểm nổi bật của web đấu giá realtime bao gồm tính minh bạch, tốc độ nhanh chóng và cơ hội tham gia vào các phiên đấu giá độc đáo. Điều này đã tạo ra một cộng đồng đam mê với những trải nghiệm độc đáo trong việc mua sắm và đấu giá trực tuyến.

### **1.2. Lý do đề tài**

Hiện nay tính tới năm 2023, nhu cầu mua sắm online đã trở nên quá phổ biến không chỉ riêng Việt Nam mà trên toàn thế giới. Nhưng chỉ mua sắm thôi thì không thể cạnh tranh được với những ông lớn đã đi trước thế nên tạo ra một môi trường cạnh tranh giữa các người khách hàng mua sắm là một hướng đi hợp lý trong thời điểm hiện tại. Nhưng hiện nay vẫn chưa có một công cụ hay một ứng dụng nào tại Việt Nam thực hiện được điều đó thực sự hiệu quả vậy nên em đã quyết định chọn đề tài này để xây dựng một hệ thống có các chức năng cơ bản cho việc đấu giá trực tuyến một cách hiệu quả.

Với sự phát triển của công nghệ một cách vượt bậc hiện nay việc chọn ngôn ngữ để viết ra một trang web thực sự không khó. Có rất nhiều ngôn ngữ phổ biến hiện nay có thể xây dựng được trang web như PHP, Python, C#, Java, Javascript, Ruby. Với mỗi ngôn ngữ đều có các framework tương ứng nhất định để cho việc xây dựng phía back-

end một cách dễ dàng hơn như Laravel, CakePHP, ExpressJS, NodeJS, EntityFramework... Còn với phía front-end cũng có các framework như ReactJS, VueJS, Angular, Goolang giúp tăng hiệu suất render dữ liệu nhận được từ phía back-end. Trong đề tài này em đã quyết định chọn PHP-Laravel để xây dựng phía server-side và VueJS để xây dựng phía client-side cho trang web single page (SPA) của mình.

### **1.3. Bố cục đề tài**

Chương 1:Giới thiệu và tổng quan đề tài.

Chương 2: Cơ sở lý thuyết của công nghệ.

Chương 3: Hệ thống website đấu giá thời gian thực.

Chương 4: Kết luận và đánh giá kết quả của hệ thống.

### **1.4. Mục tiêu nghiên cứu**

Khảo sát thị trường cũng như những yếu tố ảnh hưởng đến sự thành công cũng như thất bại của hệ thống. Xây dựng hệ thống với giao diện thân thiện với người sử dụng, hiệu suất thời gian đạt được khi đấu giá có tỉ lệ độ trễ là thấp nhất.

Sử dụng ngôn ngữ lập trình phổ biến hiện nay là PHP và Javascript cho dự án này. Thay vì sử dụng ngôn ngữ thuần thì trong đề tài website thời gian thực này sử dụng framework là Laravel, VueJS và xây dựng mô hình quan hệ cơ sở dữ liệu là Mysql, một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay. Ngoài ra còn sử dụng các thẻ HTML và CSS. Những công cụ này là nền tảng cốt lõi cho việc xây dựng hệ thống một cách nhanh chóng và hiệu quả.

## **Chương 2. CƠ SỞ LÝ THUYẾT**

### **2.1. Giới thiệu về server side**

#### **2.1.1. Khái niệm**

Server-side rendering (SSR) là một phương pháp cho việc hiển thị trang web hoặc ứng dụng web, trong đó dữ liệu và giao diện người dùng được tạo ra trên máy chủ và sau đó được gửi đến trình duyệt của người dùng. Điều này khác với phương pháp client-side rendering (CSR), trong đó trình duyệt tải về một tệp HTML trống và sau đó sử dụng JavaScript để đổ dữ liệu và tạo giao diện người dùng.

#### **2.1.2. Ưu điểm**

- SEO tốt hơn: Do trang web được tạo ra hoàn chỉnh trên máy chủ trước khi nó được gửi đến trình duyệt, các công cụ tìm kiếm có thể dễ dàng hiểu nội dung của trang web, làm cho SEO dễ dàng hơn.
- Hiệu suất tốt hơn cho người dùng đầu tiên: Người dùng không phải chờ đợi mã JavaScript tải về và thực thi trước khi trang web hiển thị, giúp cải thiện trải nghiệm người dùng đầu tiên.
- Hỗ trợ tốt cho các trình duyệt và thiết bị cũ: Vì SSR không phụ thuộc nhiều vào mã JavaScript phía máy khách, nó hoạt động tốt trên nhiều trình duyệt và thiết bị khác nhau, bao gồm cả các thiết bị có cấu hình yếu.
- Bảo mật tốt hơn: SSR có thể giúp ẩn giấu thông tin quan trọng khỏi mã JavaScript và tránh các vấn đề bảo mật như thiết lập và bảo mật đúng cách của ứng dụng trên phía máy khách.

#### **2.1.3. Nhược điểm**

- Tải máy chủ cao hơn: SSR đòi hỏi máy chủ phải thực hiện nhiều công việc để tạo giao diện người dùng và đổ dữ liệu vào trang trước khi nó được gửi đến trình duyệt của người dùng. Điều này có thể tạo áp lực lớn cho máy chủ, đặc biệt là khi có nhiều người dùng truy cập cùng lúc.
- Thời gian phản hồi ban đầu chậm hơn: Do quá trình xử lý trên máy chủ, thời gian phản hồi ban đầu có thể chậm hơn so với phương pháp Client-side Rendering (CSR). Người dùng có thể phải chờ đợi lâu hơn để trang web xuất hiện.



- Phức tạp trong việc triển khai: SSR có thể đòi hỏi việc cấu hình máy chủ và quản lý tài nguyên phức tạp hơn so với CSR. Điều này có thể làm tăng độ phức tạp của quá trình triển khai và bảo trì.
- Khả năng tương thích trình duyệt hạn chế: Mặc dù SSR hoạt động trên hầu hết các trình duyệt hiện đại, nhưng có thể gây ra vấn đề cho người dùng sử dụng các trình duyệt cũ hoặc không hỗ trợ các tính năng cụ thể.

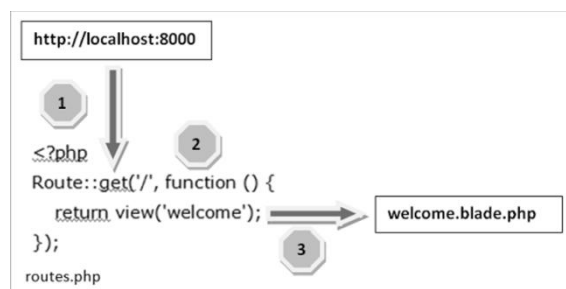
#### 2.1.4. Giới thiệu về framework Laravel

##### 2.1.4.1. Tổng quan

Laravel là một framework ra mắt lần đầu tiên vào năm 2011 được tạo ra bởi Tylor Otwell. Được xây dựng dựa trên ngôn ngữ lập trình PHP đã nổi tiếng từ lâu và rất phổ biến trong giới lập trình trên toàn thế giới. Được xây dựng theo cấu trúc mô hình MVC phổ biến trong việc phát triển ứng dụng. Mặc dù lúc mới ra mắt vào năm 2011 không nhận được sự hưởng ứng từ cộng đồng lập trình viên nhưng sau nhiều năm phát triển Laravel đã trở thành một trong những framework nổi tiếng nhất của PHP nhờ sự rõ ràng và tiện lợi mà nó đem lại.

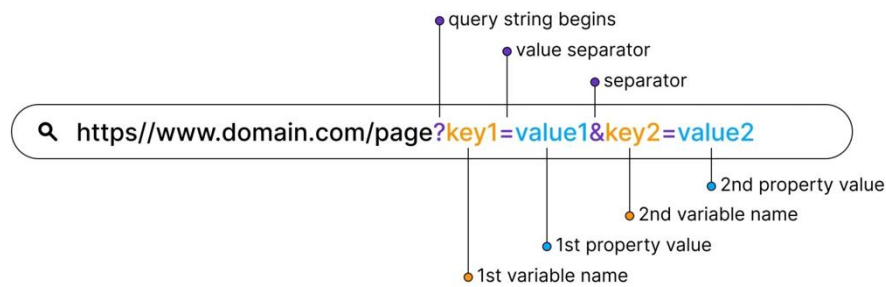
##### 2.1.4.2. Kiến trúc của Laravel

- Như đã đề cập trong phần giới thiệu tổng quan, Laravel được xây dựng theo mô hình MVC (Model – Controller – View) và routing. Trong đó routing chính đường dẫn URL.
- Routing là một thành phần quan trọng trong việc xác định các yêu cầu http của người dùng. Một router gồm có các thành phần chính đó là Http Method (Phương thức HTTP), Route definition, controller action, route params. Có 2 loại routing là API và Web, tùy vào mục đích sử dụng để viết cho hợp lý. Trong đó http method là các phương thức: Get, Post, put, delete.



Hình 2.1.4.2.1: Cấu trúc viết routing

Sau khi đã viết routing thì cấu trúc của url hiển thị trên browser cũng tuân thủ theo quy tắc đã quy ước.



Hình 2.1.4.2.2: Cấu trúc của url

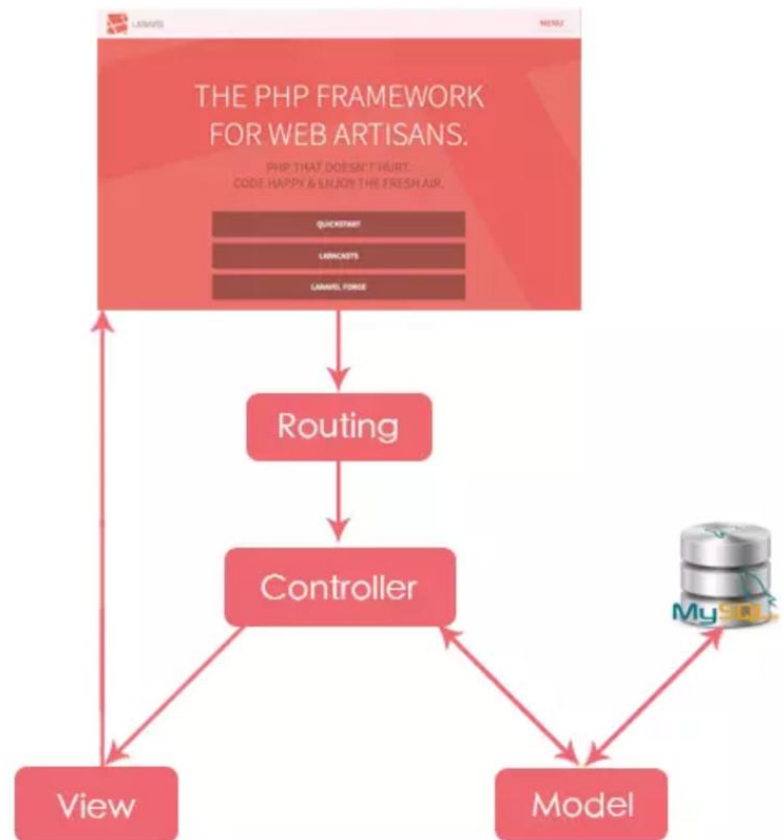
### 2.1.4.3. Mô hình MVC

Mô hình MVC là mô hình với 3 thành phần chính là Model, View và Controller.

Mỗi thành phần có chức năng như sau

- **Model**: Model là đại diện cho thành phần logic dữ liệu trong ứng dụng của bạn. Nó thường được sử dụng để truy vấn dữ liệu từ các cơ sở dữ liệu như thêm, sửa, xóa và tìm kiếm. Model cũng thực chức năng logic khác như tính toán dữ liệu trước khi gửi đến trên giao diện người sử dụng. Hiện nay các framework đã tích hợp ORM (Object Realtion Mapping) để đơn giản hoá việc truy vấn này.
- **View**: Như với chính tên của nó, view chính giao diện mà người tương tác thực tế. View không chứa logic xử mà chỉ là nơi hiển thị dữ liệu cũng như nhận thao tác của người dùng đưa đến Controller trước khi controller gửi đến Model để xử lý yêu cầu.
- **Controller**: Controller là một thành cực kì quan trọng trong mô hình MVC. Vì nó là nơi trung gian giao tiếp giữa Model và View. Controller là nơi nhận các yêu cầu từ phía người sử dụng (View) để gửi đến Model và là nơi trả về các phản hồi từ Model đến View. Nó còn là nơi điều hướng ứng dụng, quyết định xem yêu cầu của người sử dụng sẽ là View nào và Model nào.

⇒ Mô hình MVC thực sự hiệu quả đem lại lợi ích vì nó tách biệt logic dữ liệu, phân chia công việc cụ thể, tái sử dụng mã nguồn cũng như đơn giản hoá việc bảo trì.

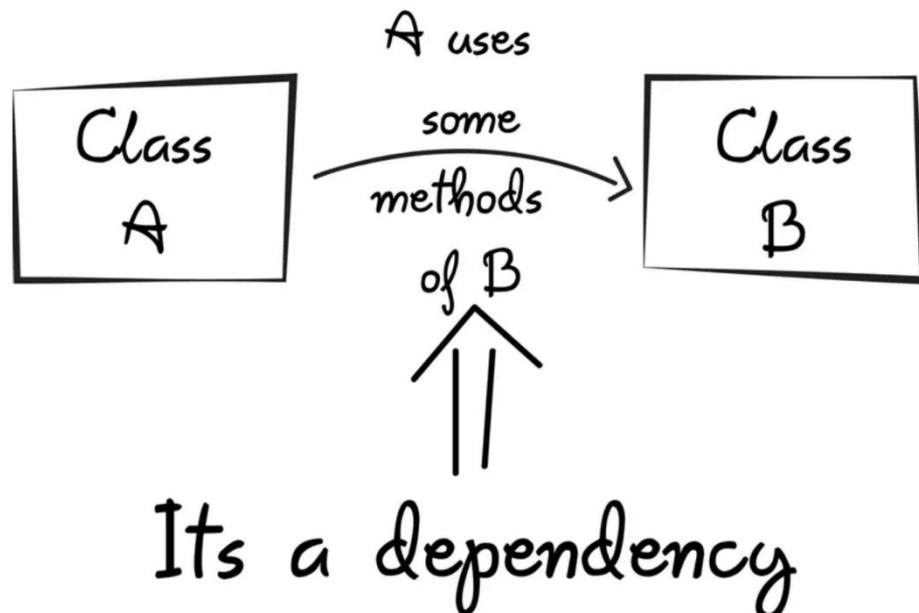


*Hình 2.1.4.3.1: Mô hình MVC trong laravel*

#### **2.1.4.4. Design pattern trong laravel**

Design pattern là một quy tắc thiết kế chung mà ở đó các lập trình viên có thể giải quyết được các vấn đề phức tạp một cách hiệu quả dựa trên quy tắc đã được thử nghiệm. Một số loại design pattern trong laravel:

- Dependency injection: Là một loại design pattern để đạt được mục đích là tạo nên sự không phụ thuộc giữa các object với nhau khi có mối quan hệ giữa object này với một object khác.



Hình 2.1.4.4.1: Dependency trong laravel

Trong trường hợp dưới đây, dependency có thể hiểu là thay vì việc phải khai báo nhiều lần các instance của object Comment và User thì chúng ta chỉ cần inject thông qua các phương thức \_\_setter và \_\_constructor.

- Service Container: Là một thành phần quan trọng của laravel, là nơi quản lý các dependency injection của ứng dụng. Có thể đăng ký các dịch vụ của ứng dụng thông qua 3 phương thức cơ bản là bind, singleton, instance. Sau khi đã đăng kí xong thì có thể sử dụng chúng bằng resolving.

```

class User {}
class Comment {}
class Post
{
    protected $user;
    protected $comment;
    public function __construct($user, $comment)
    {
        $this->user = $user;
        $this->comment = $comment;
    }
}

$post = new Post(new User(), new Comment());
  
```

Ví dụ về simple bidding:

```
$this->app->bind('HelpSpot\API', function ($app) {  
    return new \HelpSpot\API($app->make('HttpClient'));  
});
```

Ví dụ về bidding a singleton:

```
$this->app->singleton('HelpSpot\API', function ($app) {  
    return new \HelpSpot\API($app->make('HttpClient'));  
});
```

Lấy các instance trong container bằng make hoặc resolve

```
$api = $this->app->make('HelpSpot\API');
```

```
$api = resolve('HelpSpot\API');
```

- Service Provider: Service container là nơi chứa các dependency injection thì service provider chính là nơi chứa các service provider. Service gồm có 2 phương pháp chính đó là register() và boot(). Mỗi phương pháp đều có công dụng riêng của nó. Các service provider được khai báo tại config/App.php

```

'providers' => [
    /*
     * Laravel Framework Service Providers...
     */
    Illuminate\Auth\AuthServiceProvider::class,
    Illuminate\Broadcasting\BroadcastServiceProvider::class,
    Illuminate\Bus\BusServiceProvider::class,
    Illuminate\Cache\CacheServiceProvider::class,
    Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,
    Illuminate\Cookie\CookieServiceProvider::class,
    Illuminate\Database\DatabaseServiceProvider::class,
    Illuminate\Encryption\EncryptionServiceProvider::class,
    Illuminate\Filesystem\FilesystemServiceProvider::class,
    Illuminate\Foundation\Providers\FoundationServiceProvider::class,
    Illuminate\Hashing\HashServiceProvider::class,
    Illuminate\Mail\MailServiceProvider::class,
    Illuminate\Notifications\NotificationServiceProvider::class,
    Illuminate\Pagination\PaginationServiceProvider::class,
    Illuminate\Pipeline\PipelineServiceProvider::class,
    Illuminate\Queue\QueueServiceProvider::class,
    Illuminate\Redis\RedisServiceProvider::class,
    Illuminate\Auth\Passwords>PasswordResetServiceProvider::class,
    Illuminate\Session\SessionServiceProvider::class,
    Illuminate\Translation\TranslationServiceProvider::class,
    Illuminate\Validation\ValidationServiceProvider::class,
    Illuminate\View\ViewServiceProvider::class,

    /*
     * Package Service Providers...
     */

    /*
     * Application Service Providers...

```

*Hình 2.1.4.4.2: Danh sách các service provider*

Nơi đăng ký các service container:

```

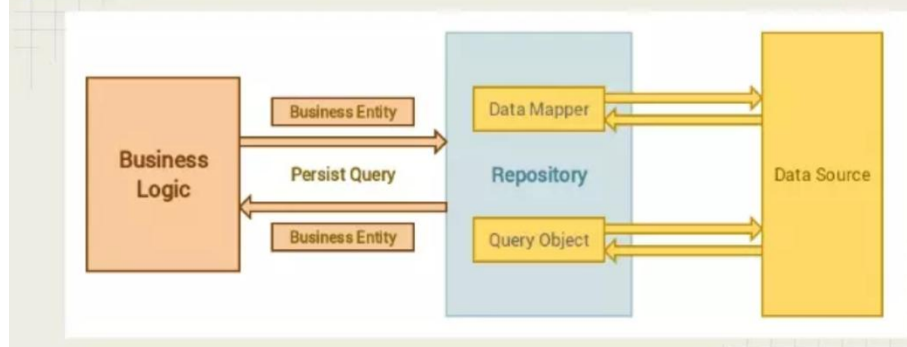
protected function register()
{
    $this->app->singleton('upload', function ($app) {
        return tap(new UploadManager($app), function ($manager) {
            $this->registerHandlers($manager);
        });
    });
}

```

Nơi cấp phép truy cập cho service đã được đăng ký trong register

- Repository: Repository có thể hiểu đơn giản là lớp trung gian giữa 2 tầng của cấu trúc là Business logic và data access. Giúp cho việc truy cập và truy vấn dữ liệu trở nên rõ ràng, chặt chẽ và bảo mật hơn. Thay vì phải truy vấn trực tiếp trong controller sẽ khiến controller của route bị lớn lên gây ra việc mất kiểm soát logic code và khó bảo trì trong tương lai.

## Repository Design Pattern



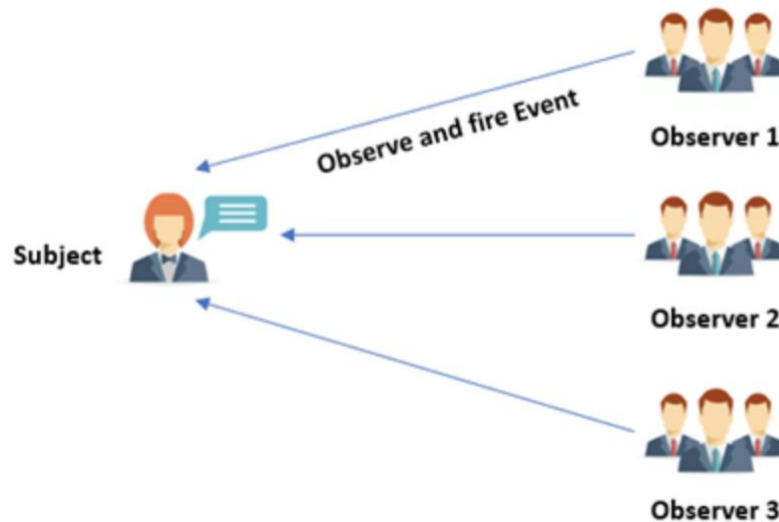
Hình 2.1.4.4.3: Repository trong laravel

- Facades: Đây là nơi giúp sử dụng dưới những tên ngắn gọn theo cấu trúc mà người có thể đặt ra thay vì viết các namespace đầy đủ của chúng. Dù công dụng không lớn bằng các design parttern khác nhưng vẫn là một trong những thành phần quan trọng để tối ưu hoá việc viết mã.

```
'App' => Illuminate\Support\Facades\App::class,
'Arr' => Illuminate\Support\Arr::class,
'Artisan' => Illuminate\Support\Facades\Artisan::class,
'Auth' => Illuminate\Support\Facades\Auth::class,
'Blade' => Illuminate\Support\Facades\Blade::class,
'Broadcast' => Illuminate\Support\Facades\Broadcast::class,
'Bus' => Illuminate\Support\Facades\Bus::class,
'Cache' => Illuminate\Support\Facades\Cache::class,
'Config' => Illuminate\Support\Facades\Config::class,
'Cookie' => Illuminate\Support\Facades\Cookie::class,
'Crypt' => Illuminate\Support\Facades\Crypt::class,
'DB' => Illuminate\Support\Facades\DB::class,
'Eloquent' => Illuminate\Database\Eloquent\Model::class,
'Event' => Illuminate\Support\Facades\Event::class,
'File' => Illuminate\Support\Facades\File::class,
'Gate' => Illuminate\Support\Facades\Gate::class,
'Hash' => Illuminate\Support\Facades\Hash::class,
'Lang' => Illuminate\Support\Facades\Lang::class,
'Log' => Illuminate\Support\Facades\Log::class,
```

Hình 2.1.4.4.4: Facades trong laravel

- Observer: Observer là một sự kiện xảy ra tự động khi có một sự kiện khác xảy ra. Nó hoạt động giống như cơ chế trigger của sql. Đây chính là nơi theo dõi các sự kiện xảy ra trong model. Thường được sử dụng để xử lý các sự kiện như tạo, cập nhật, xoá trong cơ sở dữ liệu.



Hình 2.1.4.4.5: Observe trong laravel

#### 2.1.4.5. Query buider và ORM

- Query buider là công cụ cho việc xây dựng câu truy vấn sql bằng cách sử dụng mã PHP. Query buider là cú pháp gần gũi, dễ dàng sử dụng nhưng cũng có một vài nhược điểm như tỉ lệ bảo mật kém hơn so với eloquent ORM như sql injection (một loại tấn công mã độc sql).

```
$users = DB::table('users')->get();
```

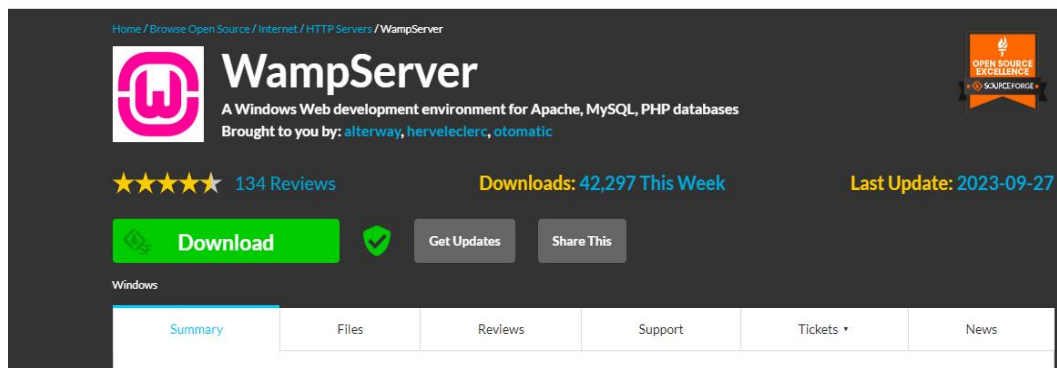
- Eloquent ORM (Object-Relational Mapping) là phương pháp truy vấn bằng đối tượng. Phương pháp này hầu hết đều có trong các freamework hiện đại vì nó sử dụng truy vấn thông qua các đối tượng được khai báo model. Một điểm cực kỳ mạnh của phương pháp này là độ bảo mật cao hơn nhiều so với query builder. Mặc dù có nhiều điểm mạnh nhưng eloquent được cho là có hiệu suất kém hơn query buider vì phải qua một lớp trung gian trước khi thực hiện truy vấn.



```
$users = User::all();
```

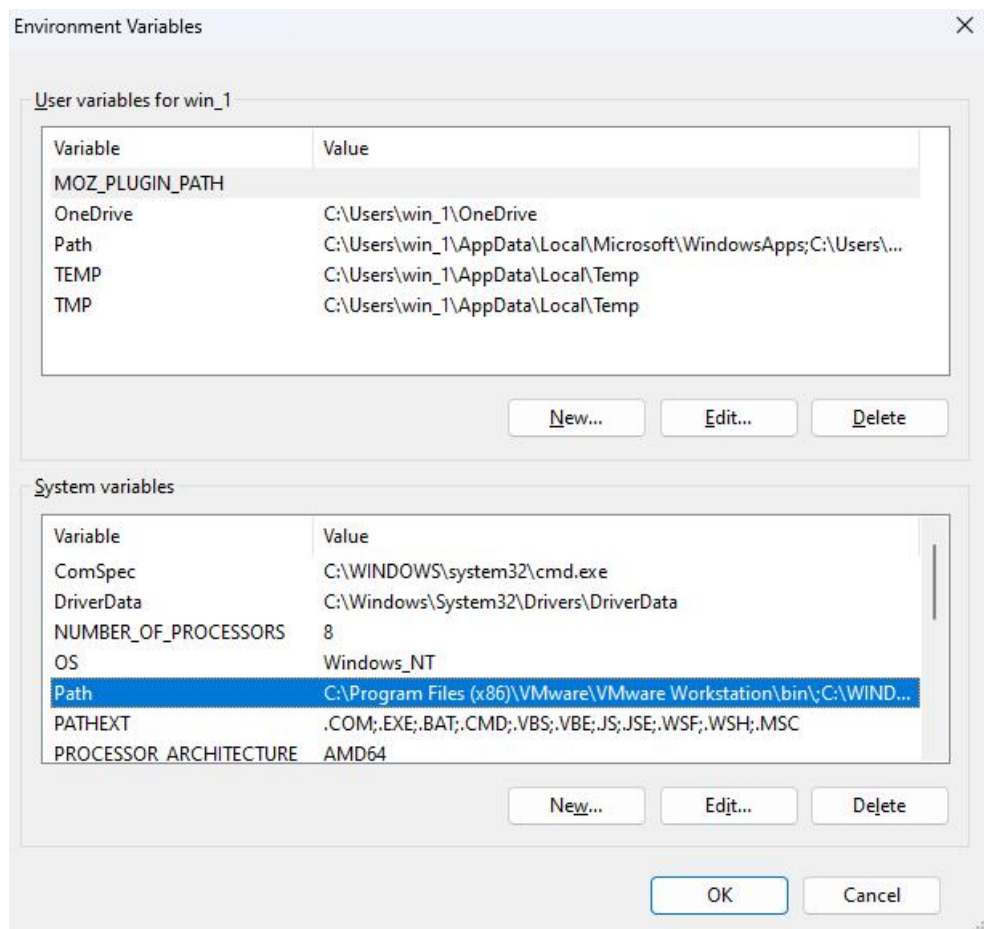
#### 2.1.4.6. Khởi tạo chương trình đầu tiên

- Cài đặt PHP: Để chạy được ngôn ngữ PHP thì tất nhiên phải cài PHP trên máy. Có thể cài thông qua một số phần mềm như laragon, xampp, wampp tùy vào mục đích sử dụng. Ở đây trong dự án này sử dụng wampp vì wampp có một lợi thế là có thể tùy chỉnh được phiên bản PHP thích hợp. Hiện nay 2023, đã có rất nhiều phiên bản PHP được phát hành. Để đạt được hiệu suất tốt nhất cũng như tránh gặp những lỗi không đáng có thì chúng ta nên sử dụng phiên bản PHP có version mới nhất hiện tại là 8.2.x.

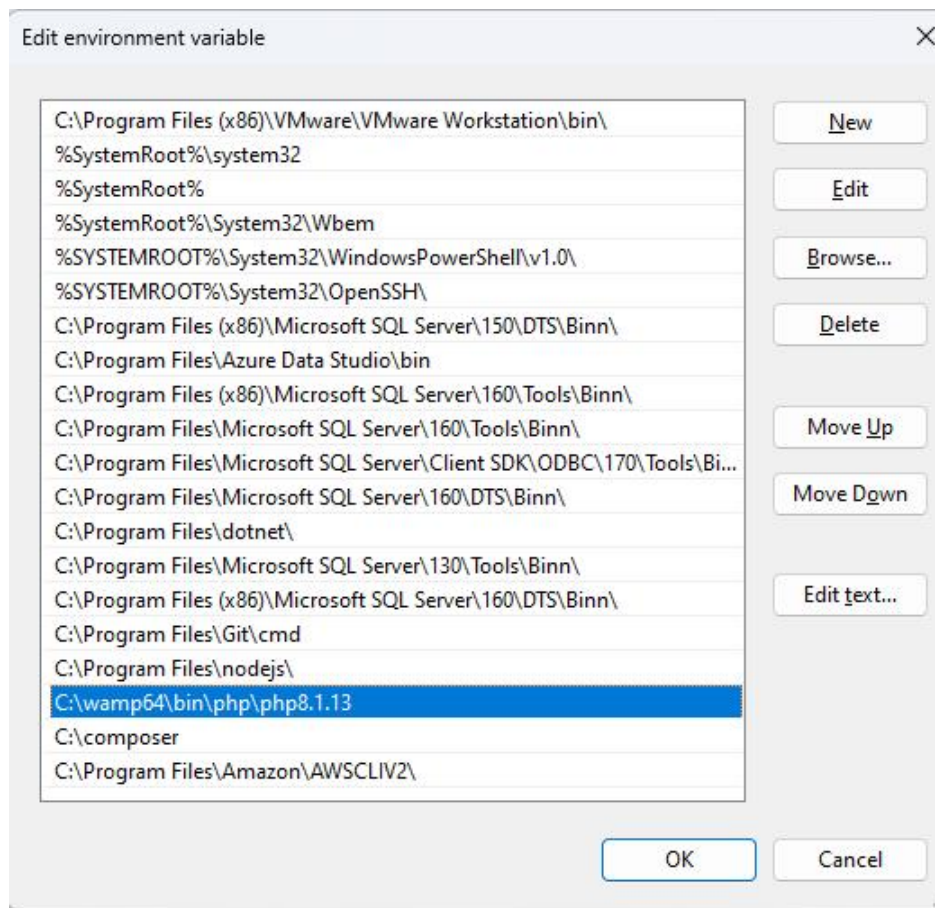


*Hình 2.1.4.5.1: Phần mềm Wampp*

Sau khi đã cài đặt xong wampp việc tiếp theo là gán phiên bản php vào biến môi trường theo cách sau: vào Advanced setting -> Environment Variables. Ở mục System variables chọn vào path. Thêm vào đường dẫn đến thư mục chứa phiên bản php thích hợp với dự án.

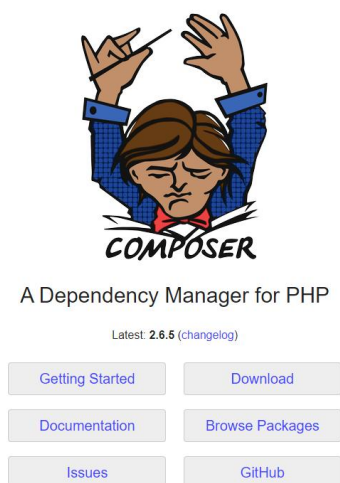


*Hình 2.1.4.5.2: Bảng Environment Variables*



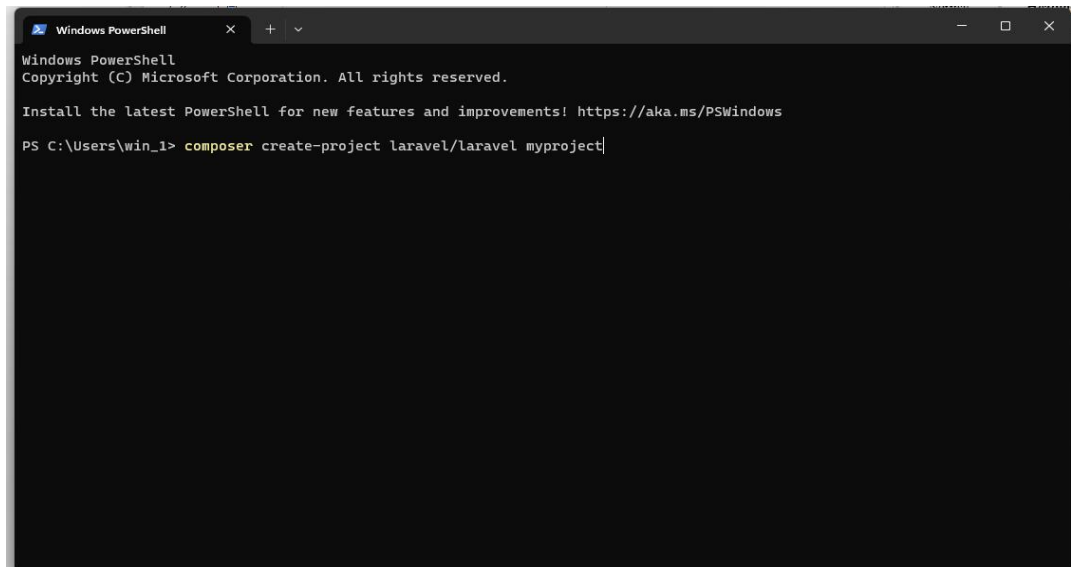
*Hình 2.1.4.5.3: Edit Environment Variables*

- Cài đặt composer: Composer là một công cụ cực kì quan trọng trong việc khởi tạo một dự án Laravel. Nó là nơi quản lý cái gói phụ thuộc và cài đặt các gói mở rộng trong Laravel.



*Hình 2.1.4.5.4: Composer PHP - Laravel*

Sau khi đã tạo cài đặt cũng như thiết lập môi trường cần thiết, bây giờ chúng ta có thể tạo một dự án laravel thông qua terminal với lệnh *composer create-project laravel/laravel myproject*. Trong đó myproject là tên của project mà bạn muốn đặt tên.



*Hình 2.1.4.5.5: Khởi tạo dự án laravel đầu tiên.*

## 2.2. Giới thiệu về client side

### 2.2.1. Khái niệm

Client-side rendering (CSR) là một phương pháp cho việc hiển thị trang web hoặc ứng dụng web, trong đó công việc tạo giao diện người dùng và đồ dữ liệu vào trang được thực hiện trên máy khách (trình duyệt của người dùng). Điều này khác với Server-side Rendering (SSR), trong đó trang web hoặc ứng dụng web được tạo ra trước trên máy chủ và sau đó gửi đến trình duyệt.

### 2.2.2. Ưu điểm

- Tải trang nhanh hơn ban đầu: Với CSR, trình duyệt chỉ cần tải một trang HTML tĩnh ban đầu, sau đó sử dụng JavaScript để tải dữ liệu và hiển thị nội dung. Điều này có thể làm cho trang web trở nên nhanh hơn vì trình duyệt không cần đợi máy chủ tạo ra một trang hoàn chỉnh.
- Tích hợp dữ liệu dễ dàng: CSR cho phép bạn tải dữ liệu từ nhiều nguồn khác nhau và tích hợp chúng dễ dàng vào trang web của mình. Điều này giúp bạn xây dựng ứng dụng web động và phong phú hơn.

- Trải nghiệm người dùng tốt hơn: Thay vì phải tải lại toàn bộ trang web khi người dùng thực hiện một hành động nhỏ, CSR cho phép cập nhật chỉ phần của trang cần thiết. Điều này có thể làm cho trải nghiệm người dùng trở nên mượt mà hơn và giảm tải cho máy chủ.
- Khả năng tạo ứng dụng đơn trang (SPA): CSR thường được sử dụng để xây dựng các ứng dụng đơn trang (SPA), trong đó chỉ có một trang HTML duy nhất được tải lần đầu, và sau đó tất cả các trang con và dữ liệu được tải và hiển thị bằng cách sử dụng JavaScript. Điều này có thể cải thiện trải nghiệm người dùng và giảm tải cho máy chủ.

### **2.2.3. Nhược điểm**

- SEO khó khăn: Trong mô hình CSR, trang web thường được tạo ra bằng JavaScript, và nội dung trên trang có thể không thể hiện trong mã nguồn HTML ban đầu. Điều này làm cho việc tối ưu hóa cho các công cụ tìm kiếm (SEO) trở nên khó khăn hơn, vì các công cụ tìm kiếm không thể thấy nội dung của bạn nếu nó không xuất hiện trong mã nguồn HTML.
- Tốn nhiều tài nguyên máy khách: CSR yêu cầu trình duyệt của người dùng tải và thực thi JavaScript để hiển thị trang web. Điều này có thể tốn nhiều tài nguyên máy khách, đặc biệt là trên các thiết bị có cấu hình thấp hoặc kết nối internet chậm.
- Tải chậm ban đầu: Vì trình duyệt phải tải và thực thi JavaScript trước khi hiển thị nội dung, trang web có thể tải chậm ban đầu, đặc biệt là trên các kết nối internet chậm hoặc thiết bị di động.
- Khả năng xử lý trên máy khách: Với CSR, phần lớn xử lý phải diễn ra trên máy khách, điều này có thể tạo ra một số vấn đề bảo mật, nhất là khi xử lý dữ liệu quan trọng hoặc nhạy cảm. Cần phải có biện pháp bảo mật cẩn thận để ngăn chặn các tấn công như XSS (Cross-Site Scripting).

### **2.2.4. Giới thiệu về framework VueJS**

#### **2.2.4.1. Giới thiệu**

Vuejs là một thư viện javascript mã nguồn mở được xây dựng rộng rãi trên toàn thế giới. Được phát hành phiên bản đầu tiên vào năm 2014, mặc dù ra đời khá

muộn so với các thư viện khác như angular nhưng đã đón được sự hưởng ứng đáng kể từ cộng đồng lập trình viên. Nhờ có hướng đi rõ ràng cùng sự cập nhật và phát triển liên tục nên vuejs luôn được ưa chuộng không kém cạnh gì so với ReactJs của Facebook và Angular của Google. Cũng giống như các thư viện khác VueJS được tạo ra nhằm đáp ứng nhu cầu về mặt UI cho người sử dụng cụ thể là SPA (single page application).

#### **2.2.4.2. Kiến trúc của VueJS**

Một vòng đời (lifecycle) của VueJS được chia làm 3 phần chính.

**Phần 1 Khởi tạo (Creation):** Phần này bao gồm các lifecycle hook được gọi khi component đang được khởi tạo. Các hook trong phần này giúp bạn thiết lập dữ liệu ban đầu, khởi tạo các thành phần, và chuẩn bị cho việc hiển thị component lên giao diện người dùng. Các hook trong phần này bao gồm:

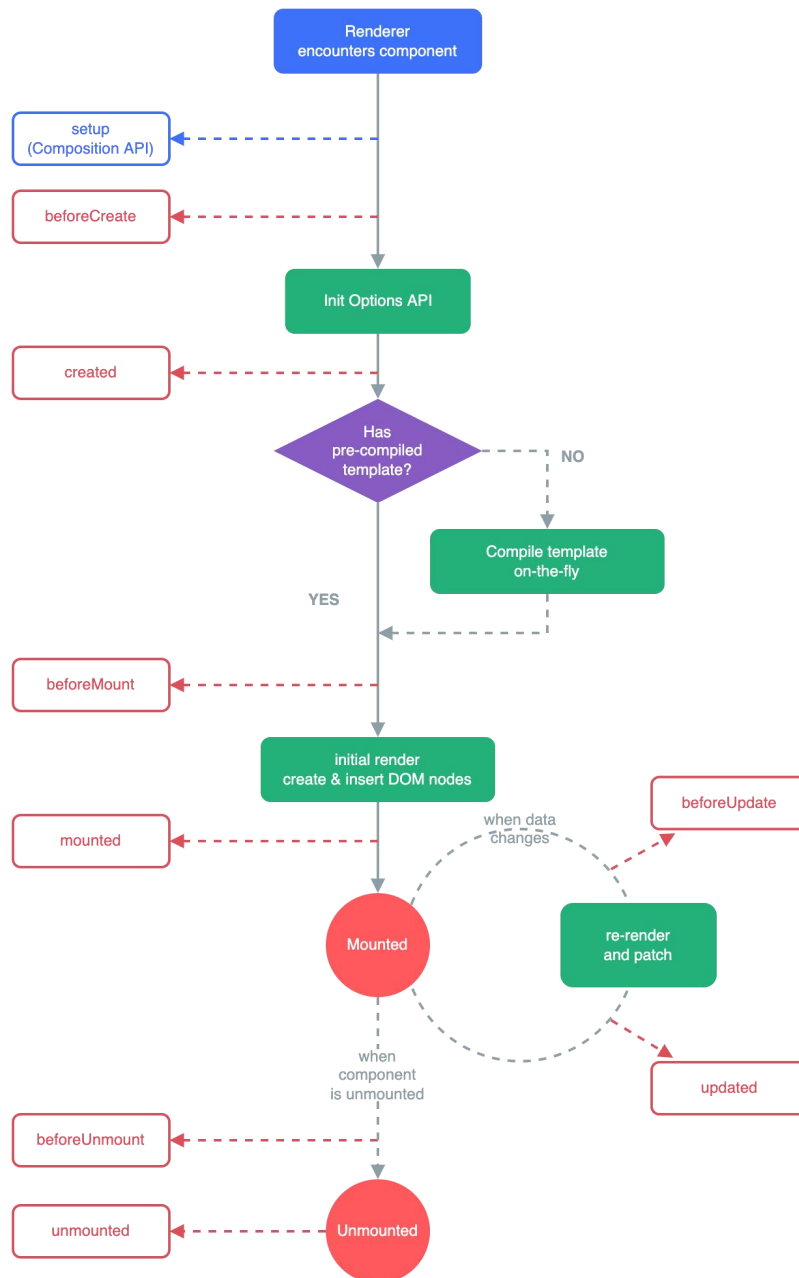
- beforeCreate: Hook này được gọi trước khi component được tạo ra và trước khi dữ liệu và sự kiện được kết nối.
- created: Hook này được gọi sau khi component được tạo ra và đã có dữ liệu và sự kiện được kết nối, nhưng trước khi component được thêm vào giao diện.

**Phần 2 Lắp đặt (Mounting):** Phần này bao gồm các lifecycle hook được gọi khi component đã được tạo và sẵn sàng để lắp đặt vào giao diện. Các hook trong phần này giúp bạn thực hiện các tác vụ liên quan đến DOM như lắp đặt component lên trang web và tương tác với các thư viện bên ngoài. Các hook trong phần này bao gồm:

- beforeMount: Hook này được gọi trước khi component được lắp đặt vào giao diện.
- mounted: Hook này được gọi sau khi component đã được lắp đặt vào giao diện.

**Phần 3 Cập nhật (Updating):** Phần này bao gồm các lifecycle hook được gọi khi dữ liệu của component thay đổi và gây ra cập nhật giao diện. Các hook trong phần này giúp bạn quản lý việc cập nhật dữ liệu và giao diện. Các hook trong phần này bao gồm:

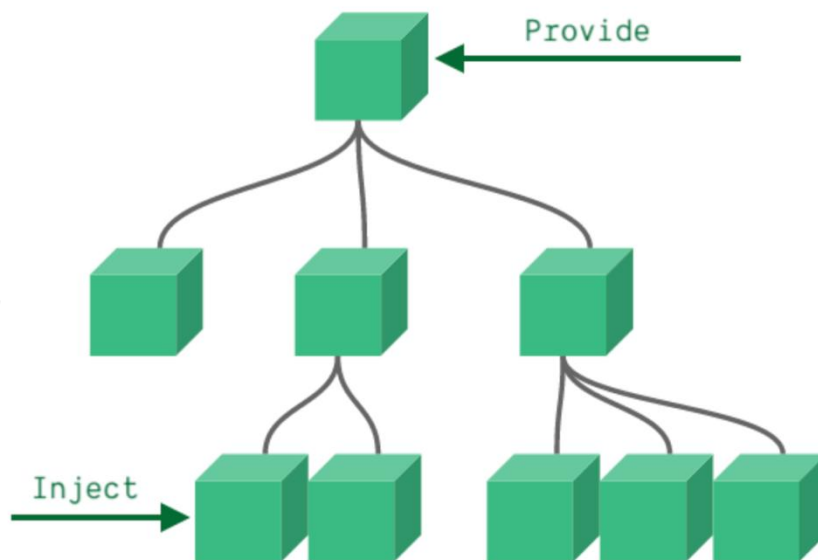
- `beforeUpdate`: Hook này được gọi trước khi component được cập nhật, sau khi dữ liệu thay đổi, nhưng trước khi giao diện được cập nhật.
- `updated`: Hook này được gọi sau khi component đã được cập nhật và giao diện đã được cập nhật tương ứng.



Hình 2.2.4.2.1: Lifecycle vuejs

### 2.2.4.3. Khái niệm component trong VueJS

Trong VueJS, component chính là khái niệm quan trọng nhất. Để thực sự có thể làm được một dự án sử dụng thư viện này thì việc nắm rõ cách hoạt động của nó là điều kiện cần. Component cho phép bạn chia sẽ giao diện người dùng thành các thành phần nhỏ và tái sử dụng chúng ở nhiều nơi khác nhau trong ứng dụng của bạn. Thay vì bạn phải viết một đoạn mã có cấu trúc tương tự nhau nhiều lần thì component đã giải quyết vấn đề đó cho bạn. Component trong vuejs là một khối mã độc lập bao gồm HTML, Javascript và CSS, chứa tất cả các logic xử lý và giao diện cần thiết để thực hiện một phần cụ thể của một trang web hay một ứng dụng nhất định.



Hình 2.2.4.3.1: Component trong Vuejs

Trong 1 trang web có thể chứa nhiều component khác nhau, trong mỗi component đó có thể chứa nhiều component con. Việc tách ra nhiều component như này ban đầu có vẻ hơi rắc rối nhưng khi bảo trì và phát triển bạn sẽ thấy tác dụng tuyệt vời của chúng.



#### 2.2.4.4. Phương thức trong Vue

- Data: Như cái tên đã nói ra chức năng của phương thức này. Đây là nơi chứa các dữ liệu instance và được truy cập qua

```
var vm = new Vue({  
  data: {  
    message: 'Xin chào, Vue.js!'  
  }  
});
```

- Computed: computed là một đối tượng chứa các thuộc tính toán. Các thuộc tính tính toán này được tính dựa trên dữ liệu hiện có và được lưu trữ như các thuộc tính thông thường. Thuộc tính hiện có có thể lấy trong phương thức data.

```
var vm = new Vue({  
  data: {  
    radius: 5  
  },  
  computed: {  
    area: function () {  
      return Math.PI * this.radius * this.radius;  
    }  
  }  
});
```

- Methods: method là một đối tượng chứa các phương thức của javascript. Thường các hàm, function sẽ được khai báo và viết ở trong đối tượng này. Các hàm, function này có thể được gọi từ template hoặc bất cứ nơi nào trong javascript.

```
var vm = new Vue({
  data: {
    count: 0
  },
  methods: {
    increment: function () {
      this.count++;
    }
  }
});

vm.increment(); // Gọi phương thức increment
```

- Watch: watch là đối tượng chứa các hàm theo dõi. Khá giống với observer trong laravel và trigger trong sql. Nó thực hiện nhiệm vụ là theo dõi sự thay đổi của các thuộc tính khác.

```
var vm = new Vue({
  data: {
    message: 'Hello'
  },
  watch: {
    message: function (newVal, oldVal) {
      console.log('Giá trị mới: ' + newVal);
    }
  }
});
```

- Created và mounted: Đây là các hook trong lifecycle của VueJS. Trong đó, created được gọi sau khi instance được tạo và dữ liệu được khởi tạo. Phương thức mounted được gọi khi component đã kết nối thành công và gắn vào DOM.

```
var vm = new Vue({
  data: {
    message: 'Xin chào, Vue!'
  },
  created: function () {
    console.log('Instance đã được tạo.');
```

- Props: Props là thành phần không thể thiếu trong việc truyền dữ liệu giữa các component với nhau. Sự tương tác dữ liệu giữa component cha và component con đều thông qua props. Việc chia sẻ thông tin dữ liệu này giúp cho các component có tính tương tác và tính kết nối cao hơn bao giờ hết.

```
<script setup>
const props = defineProps(['foo'])

console.log(props.foo)
</script>
```

#### 2.2.4.5. Thư viện Vue Loader

Vue loader là một trình tải cho webpack cho phép tạo các thành phần Vue theo định dạng là Single File Components (SFCs) để quản lý giao diện người dùng và logic trong ứng dụng VueJS của bạn. Một số chức năng chính trong vue loader. Biên dịch SFCs chính là điểm quan trọng nhất ở đây vì trình duyệt không thể hiểu được các tập tin có phần mở rộng của tệp là .vue nên cần phải biên dịch thành file Javascript, HTML, CSS lúc đó trình duyệt mới có thể hiển thị được.

Cài đặt Vue loader có thể thông qua nhiều cách khác nhau nhưng các đơn giản nhất là sử dụng `npm -i vue-loader`.

```
> npm i vue-loader
```



*Hình 2.2.4.5.1: Cài đặt Vue-loader*

#### **2.2.4.6. Thư viện Vue Axios**

Vue Axios là thư viện giúp cho việc gọi API một cách dễ dàng hơn. Là một thư viện Javascript phổ biến để thực hiện gọi các yêu cầu HTTP trong ứng dụng web, có thể là GET, POST, PUT, DELETE. Không phải riêng VueJS mà cách thư viện khác như ReactJS hay bất cứ dạng client side rendering nào đều cũng có thể sử dụng axios cho dự án đó.

```
> npm i axios
```

*Hình 2.2.4.6.1: Cài đặt Axios*

#### **2.2.4.7. Thư viện Vue Router**

Vue-router được sử dụng để quản lý định tuyến routing trong ứng dụng web vue.js. Nó cho phép tạo các định tuyến routes, điều hướng giữa các trang và cung cấp tích hợp hiệu quả giữa các component để xây dựng ứng dụng đơn trang hoặc ứng dụng đa trang với định tuyến (multi-page application with routing).

```
npm install vue-router@4
```

*Hình 2.2.4.7.1: Cài đặt vue-router*

#### **2.2.4.8. Thư viện Vuex**

Vuex là một trong những thư viện quản lý trạng thái (state management). Cũng có một số thư viện phổ biến khác như pinia cũng thực hiện nhiệm vụ này nhưng tùy vào quy mô dự án để sử dụng cho phù hợp. Vuex được sử dụng để tạo ra một nơi lưu trữ chung ở đó nó quản lý và chia sẻ dữ liệu trạng thái của ứng dụng giữa các component một cách hiệu quả. Vuex thường được dùng cho các ứng dụng lớn hoặc có chứa nhiều các component phức tạp. Nếu việc truyền dữ liệu

sử dụng props trở nên khó khăn do có quá nhiều cấp componet thì Vuex chính là sự lựa chọn cho giải pháp này.

```
> npm i vuex
```

#### 2.2.4.9. Thư viện Vue Prime

Hiện nay có nhiều template viết sẵn của thư viện vuejs, mục đích là giúp người sử dụng có thể thao tác một cách nhanh hơn, dễ dàng hơn về cả mặt xử lý và giao diện. Đã có nhiều template được tạo ra như vue material admin, vuetify, primevue. Mỗi cái đều có điểm mạnh và điểm yếu riêng. Trong đó primevue là một thư viện người dùng UI đang được sử dụng rộng rãi. Primevue cung cấp hàng loạt các như button, menu, thanh tiêu đề, input và nhiều thành phần khác sử dụng trong ứng dụng vuejs.

```
// with npm  
npm install primevue
```

*Hình 2.2.4.9.1: Cài đặt PrimeVue*

## **Chương 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

### **3.1. Mô tả đề tài**

Website đấu giá trực tuyến được mô tả như sau:

Được chia ra làm hai phần chính đó là trang quản lý và trang khách hàng.

Trang quản lý:

Thực hiện các chức năng cơ bản như quản lý danh mục sản phẩm, quản lý sản phẩm, quản lý người dùng và quản lý phiên đấu giá. Trong đó chức năng tạo phiên được xem là một trong những chức năng quan trọng nhất ảnh hưởng đến sự thành công của hệ thống. Trong mỗi chức năng sẽ có các hạng mục như thêm, sửa, xóa phù hợp với việc vận hành.

Chức năng thêm hạng mục sản phẩm: Chức năng này là nơi phân loại các sản phẩm trước khi tạo ra một sản phẩm chi tiết. Ví dụ như trang sức, công nghệ, thời trang... Có các chức năng như thêm danh mục sản phẩm, sửa danh mục sản phẩm, tìm kiếm danh mục sản phẩm và xóa danh mục sản phẩm.

Chức năng thêm sản phẩm: Dựa vào danh mục sản phẩm để có thể tạo sản phẩm sao cho phù hợp. Ví dụ như trong đồ công nghệ thì có điện thoại, có máy tính... Có các chức năng có bản như thêm sản phẩm, sửa sản phẩm, tìm kiếm sản phẩm và xóa sản phẩm.

Chức năng quản lý khách hàng: Quản lý các tài khoản mà khách hàng đã đăng ký. Có các chức năng như hiển thị danh sách khách hàng, chỉnh sửa thông tin khách hàng.

Chức năng tạo phiên đấu giá: Chọn sản phẩm để tạo phiên đấu giá. Chọn thời gian đấu giá có thể trong thời gian ngắn hoặc có thể trong thời gian dài. Có 3 trạng thái của một phiên đấu giá là hết thời gian, đang diễn ra và sắp diễn ra.

Trang đấu giá: Đây là trang mà người dùng thực hiện các phiên đấu giá. Mỗi sản phẩm sẽ có một phiên đấu giá với một mốc thời gian nhất định. Người dùng sẽ chọn một trong nhiều sản phẩm để đấu giá. Yêu cầu khi đặt giá của một sản phẩm là giá của sản phẩm phải lớn hơn giá của người đặt trước. Sau khi đã thực hiện một phiên đấu giá thì sẽ ghi lại lịch sử đấu giá của người dùng giúp cho khách hàng dễ dàng quản lý được phiên của mình.

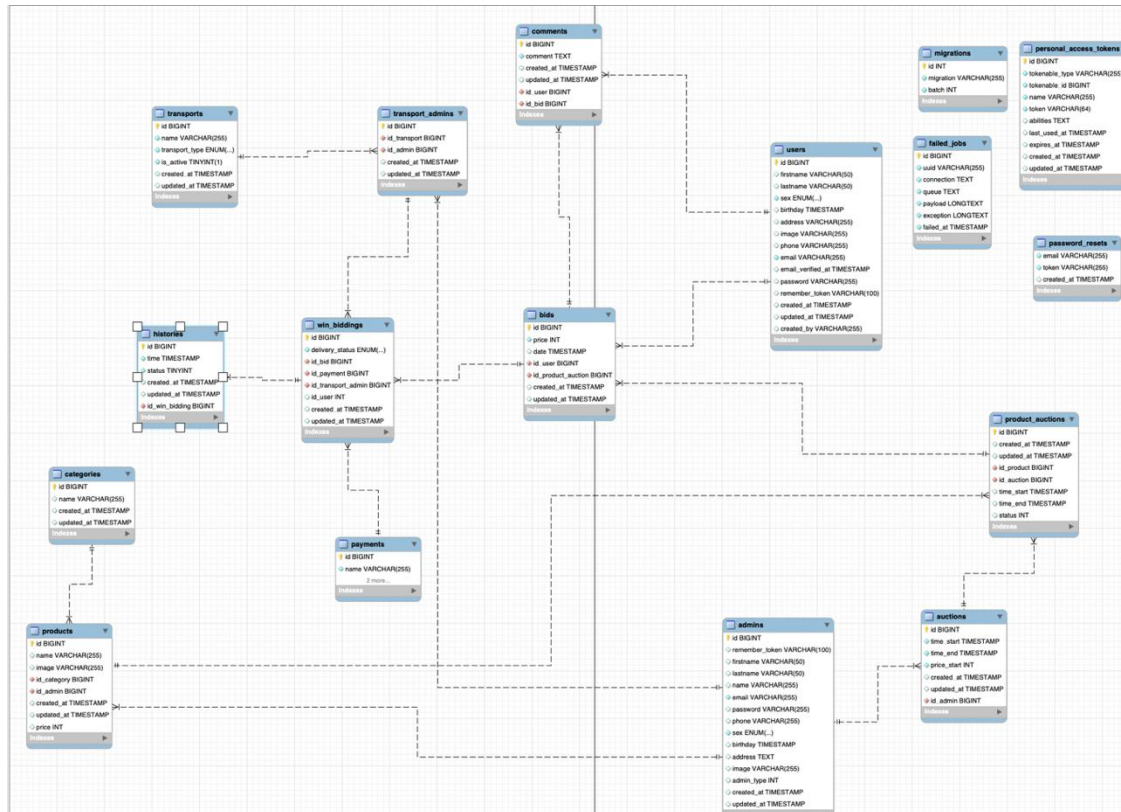
Chức năng đăng ký tài khoản: Người dùng muốn tham gia đấu giá phải thực hiện đăng ký tài khoản, nhập các thông tin cơ bản như email, họ tên...

Chức năng đăng nhập: Sau khi đã đăng ký thành công tài khoản thì thực hiện đăng nhập để vào đấu giá.

Chức năng giỏ hàng: Sau khi chiến thắng một sản phẩm trong phiên đấu giá, sản phẩm sẽ được ghi nhận vào trong giỏ hàng của khách hàng.

## 3.2. Thiết kế cơ sở dữ liệu

### 3.2.1. Mô hình cơ sở dữ liệu và mối quan hệ giữa các bảng



Hình 3.2.1.1: Mô hình cơ sở dữ liệu

### 3.2.2. Các bảng trong cơ sở dữ liệu

#### 3.2.2.1. Bảng admins

Trong bảng admins có các trường như sau:

admins (id, remember\_token, firstname, lastname, name, email, password, phone, sex, birthday, address, image, admin\_type, created\_at, updated\_at)

Id	Auto increments
remember_token	Varchar(100)
firstname	Varchar(50)
lastname	Varchar(50)
name	Varchar(255)
email	Varchar(255) - Unique
password	Varchar(255)
phone	Varchar(255)
sex	Enum – default (0)
birthday	Timestamp
address	Text

image	Varchar(255)
admin_type	Interger
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.1: Bảng admins*

### 3.2.2.2. Bảng users

Trong bảng users có các trường như sau:

users (id, firstname, lastname, sex, birthday, address, image, phone, email, email\_verified\_at, password, remember\_token, created\_at, updated\_at, created\_by)

Id	Auto increments
remember_token	Varchar(100)
firstname	Varchar(50)
lastname	Varchar(50)
name	Varchar(255)
email	Varchar(255) - Unique
password	Varchar(255)
phone	Varchar(255)
sex	Enum – default (0)
birthday	Timestamp
address	Text
image	Varchar(255)
admin_type	Interger
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.2: Bảng users*

### 3.2.2.3. Bảng categories

Trong bảng categories có các trường như sau:

categories (id, name, created\_at, updated\_at)

Id	Auto increments
name	Varchar(255)
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.3: Bảng categories*

### 3.2.2.4. Bảng products

Trong bảng products có các trường như sau:

products (id, name, image, price, created\_at, updated\_at, #id\_category, #id\_admin)

Id	Auto increments
name	Varchar(255)
image	Varchar(255)
price	Interger



created_at	Timestamp
updated_at	Timestamp
#id_category	BigInt
#id_admin	BigInt

*Bảng 3.2.2.4: Bảng products*

### 3.2.2.5. Bảng bids

Trong bảng bids có các trường như sau:

bids (id, price, date, created\_at, updated\_at, #id\_user, #id\_product\_auction)

Id	Auto increments
price	Integer
date	Timestamp
created_at	Timestamp
updated_at	Timestamp
#id_user	BigInt
#id_product_auction	BigInt

*Bảng 3.2.2.5: Bảng bids*

### 3.2.2.6. Bảng auctions

Trong bảng auctions có các trường như sau:

auctions (id, time\_start, time\_end, price\_start, created\_at, updated\_at, #id\_admin)

Id	Auto increments
time_start	Timestamp
time_end	Timestamp
price_start	Integer
created_at	Timestamp
updated_at	Timestamp
#id_admin	BigInt

*Bảng 3.2.2.6: Bảng auctions*

### 3.2.2.7. Bảng product\_auctions

Trong bảng product\_auctions có các trường như sau:

product\_auctions (id, time\_start, time\_end, status, created\_at, updated\_at, #id\_product, #id\_auction)

id	Auto increments
time_start	Timestamp
time_end	Timestamp
status	Int
created_at	Timestamp
updated_at	Timestamp
#id_product	BigInt
#id_auction	BigInt

*Bảng 3.2.2.7: Bảng product\_auctions*

### 3.2.2.8. Bảng win\_biddings

Trong bảng win\_biddings có các trường như sau:

win\_biddings (id, delivery\_status, created\_at, updated\_at, #id\_bid, #id\_payment, #id\_transport\_admin)

id	Auto increments
delivery_status	Enum
created_at	Timestamp
updated_at	Timestamp
#id_bid	BigInt
#id_payment	BigInt
#id_transport_admin	BigInt

*Bảng 3.2.2.8: Bảng win\_biddings*

### 3.2.2.9. Bảng histories

Trong bảng win\_biddings có các trường như sau:

histories (id, time, status, created\_at, updated\_at, #id\_win\_bidding)

id	Auto increments
time	Timestamp
status	TinyInt
created_at	Timestamp
updated_at	Timestamp
#id_win_bidding	BigInt

*Bảng 3.2.2.9: Bảng histories*

### 3.2.2.10. Bảng transports

Trong bảng win\_biddings có các trường như sau:

transports (id, name, transport\_type, is\_active, created\_at, updated\_at)

id	Auto increments
name	Varchar(255)
transport_type	Enum
is_active	Tinyint
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.10: Bảng transports*

### 3.2.2.11. Bảng transport\_admins

Trong bảng win\_biddings có các trường như sau:

transport\_admin (id, created\_at, updated\_at, #id\_transport, #id\_admin)

id	Auto increments
created_at	Timestamp
updated_at	Timestamp
#id_transport	BigInt
#id_admin	BigInt

*Bảng 3.2.2.11: Bảng transport\_admins*

### 3.2.2.12. Bảng payments

Trong bảng payments có các trường như sau:

payments (id, name, created\_at, updated\_at)

id	Auto increments
name	Varchar(255)
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.12: Bảng payments*

### 3.2.2.13. Bảng comments

Trong bảng comments có các trường như sau:

comments (id, comment, created\_at, updated\_at, #id\_user, #id\_bid)

id	Auto increments
comment	Text
created_at	Timestamp
updated_at	Timestamp
#id_user	BigInt
#id_bid	BigInt

*Bảng 3.2.2.13: Bảng comments*

### 3.2.2.14. Bảng personal\_access\_tokens

Trong bảng personal\_access\_tokens có các trường như sau:

personal\_access\_tokens (id, tokenable\_type, tokenable\_id, name, token, abilities, last\_used\_at, expires\_at, created\_at, updated\_at)

id	Auto increments
tokenable_type	Varchar(255)
tokenable_id	BigInt
name	Varchar(255)
token	Varchar(64)
abilites	Text
last_used_at	Timestamp
expires_at	Timestamp
created_at	Timestamp
updated_at	Timestamp

*Bảng 3.2.2.14: Bảng personal\_access\_tokens*

### 3.2.2.15. Bảng migration

Trong bảng migration có các trường như sau:

migrations (id, migration, batch)

id	Auto increments
migration	Varchar(255)
batch	Int

*Bảng 3.2.2.15: Bảng migration*

### 3.2.2.16. Bảng failed\_jobs

Trong bảng failed\_jobs có các trường như sau:

failed\_jobs (id, uuid, connection, queue, payload, exception, failed\_at)

id	Auto increments
uuid	Varchar(255)
connection	Text
queue	Text
payload	LongText
exception	LongText
failed_at	Timestamp

*Bảng 3.2.2.16: Bảng failed\_jobs*

### 3.2.2.17. Bảng password\_resets

Trong bảng password\_resets có các trường như sau:

password\_resets (email, token, created\_at)

email	Varchar(255) - Unique
token	Varchar(255)
created_at	Timestamp

*Bảng 3.2.2.17: Bảng password\_resets*

### 3.3. Giải thích mối quan hệ

- Bảng admin gồm có 2 đối tượng, gồm có admin (người quản trị) và shop (người bán). Đối tượng shop sẽ thuộc phần mở rộng vì không đủ thời gian để phát triển. admin sẽ tạo ra loại sản phẩm, sau khi tạo loại sản phẩm, admin tiếp tục tạo sản phẩm dựa trên loại sản phẩm đã có. Một loại sản phẩm có thể có nhiều sản phẩm và một sản phẩm chỉ thuộc một loại sản phẩm (1 – n).
- Admin tiếp tục tạo phiên đấu giá bảng auctions cho cụm sản phẩm muốn tạo phiên. Một admin có thể tạo nhiều phiên lớn và một phiên lớn chỉ thuộc 1 admin khởi tạo (1 - n).
- Một sản phẩm có thể thuộc nhiều phiên lớn. Một phiên lớn có thể có nhiều sản phẩm. (n – n).
- Tạo ra bảng phụ là bảng product\_auction với thời gian bắt đầu, thời gian kết thúc của từng sản phẩm.
- Người dùng (user) đấu giá các sản phẩm thông qua bảng bids. 1 người dùng có thể đấu giá nhiều lần. 1 lần đấu giá chỉ thuộc 1 user (1 - n).
- Bảng win\_bidding sẽ lưu lại những người có đấu giá cao nhất cho sản phẩm theo từng phiên nhỏ. Giống như một dạng giỏ hàng bảng này cũng sẽ lưu thông tin lựa chọn thanh toán và lựa chọn vận chuyển.
- Bảng payments là thông tin và phương thức thanh toán.
- Bảng transport là thông tin của các bên vận chuyển.
- Bảng transport\_admins chính là bảng kết giữa bảng admins và bảng transport (n – n).
- Bảng comments sẽ lưu lại comment của từng user theo từng phiên đấu giá. 1 user có nhiều bình luận, 1 bình luận chỉ thuộc một user (1 – n).
- Bảng migration để tự động sinh bảng trong mysql theo cấu trúc đã được viết sẵn có trong code.
- Bảng personal\_access\_tokens để lưu lại token đăng nhập của người sử dụng. Phân quyền cho phép truy cập API theo từng đối tượng một cách hợp lý.
- Một số bảng khác như password\_resets, failed\_job được sinh ra mặc định khi migrate trong Laravel Framework.

### 3.4. Đặc tả usecase

#### 3.4.1. Quản trị

Use case đăng nhập:

Tóm tắt: Admin sẽ sử dụng use case này để tiến hành đăng nhập tài khoản đã cấp trước để sử dụng website

Actor: Admin

Bước 1: Trên trang đăng nhập người dùng nhập các thông tin là email và mật khẩu.

Bước 2: Nhấn đăng nhập.

Bước 3: Nếu không khớp yêu cầu nhập lại, nếu khớp thì kết thúc use case.

Use case tạo khách hàng:

Tóm tắt: Admin sẽ sử dụng case này để tiến hành đăng ký tài khoản cho khách hàng, ngoài việc khách hàng có thể tự tạo tài khoản thì admin cũng có thể tạo.

Bước 1: Vào trang quản lý danh sách khách hàng.

Bước 2: Nhập thông tin cần thiết.

Bước 3: Nhấn tạo người dùng.

Use case tạo loại sản phẩm

Actor: Admin

Tóm tắt: Admin sẽ sử dụng case này để tiến hành tạo loại sản phẩm trước khi tạo sản phẩm.

Bước 1: Vào trang quản lý danh sách loại sản phẩm

Bước 2: Nhập thông tin cần thiết.

Bước 3: Nhấn tạo loại sản phẩm

Use case tạo loại sản phẩm

Actor: Admin

Tóm tắt: Admin sẽ sử dụng case này để tiến hành tạo phiên đấu giá

Bước 1: Vào trang quản lý danh sách phiên

Bước 2: Nhập thông tin cần thiết

Bước 3: Nhấn tạo phiên

### 3.4.2. Khách hàng

Use case đăng kí:

Tóm tắt: Người dùng sẽ sử dụng use case này để tiến hành đăng kí tài khoản để sử dụng website

Actor: Người dùng

Bước 1: Trên trang giao diện đăng nhập sẽ có chữ đăng ký hoặc người dùng cũng có thể truy cập bằng link trực tiếp

Bước 2: Điền các thông tin cần thiết vào các mục trong trang đăng ký.

Bước 3: Nhấn submit để tạo tài khoản thành công.

Bước 4: Nếu như hệ thống tự quay trở lại trang login thì việc đăng ký đã hoàn tất

Các yêu cầu của use case: Email phải là duy nhất

Trạng thái sau khi kết thúc use case: Người dùng sẽ được di chuyển về lại trang đăng nhập và sẽ nhập tài khoản vừa đăng ký để sử dụng các dịch vụ.

Use case đăng nhập:

Tóm tắt: Người dùng sẽ thực hiện use case này để tiến hành đăng nhập tài khoản để sử dụng website.

Actor: Người dùng.

Bước 1: Trên trang đăng nhập người dùng nhập các thông tin là email và mật khẩu.

Bước 2: Nhấn đăng nhập.

Bước 3: Nếu không khớp yêu cầu nhập lại, nếu khớp thì kết thúc use case.

Các yêu cầu của use case: Email và mật khẩu phải khớp trong database.

Trạng thái sau khi kết thúc use case: Người dùng sẽ được điều hướng vào trang chính. để sử dụng.

Use case đấu giá sản phẩm:

Tóm tắt: Người dùng chọn sản phẩm được hiển thị trên màn hình. Nếu muốn đấu giá sản phẩm nào người dùng có thể chọn vào sản phẩm đó. Mỗi sản phẩm đều có một mức giá nhất định.

Actor: Người dùng

Bước 1: Chọn sản phẩm cần đấu giá,

Bước 2: Nhập giá đấu giá (Giá phải cao hơn giá trước).

Các yêu cầu của use case: Giá nhập vào phải là số và phải cao hơn giá trước đó.

Trạng thái sau khi kết thúc use case: Người dùng đã đấu giá sản phẩm thành công.

## Chương 4. THỰC NGHIỆM

### 4.1. Cấu hình cài đặt

#### 4.1.1. Mysql workbench 8.0

- Yêu cầu phần cứng tối thiểu:
  - ✓ CPU: Intel Core hoặc Xeon 3GHz (hoặc Dual Core 2GHz) hoặc CPU AMD tương đương.
  - ✓ RAM: 6GB.
  - ✓ Độ phân giải màn hình: 1024 x 768 (khuyến nghị 1280 x 1024).

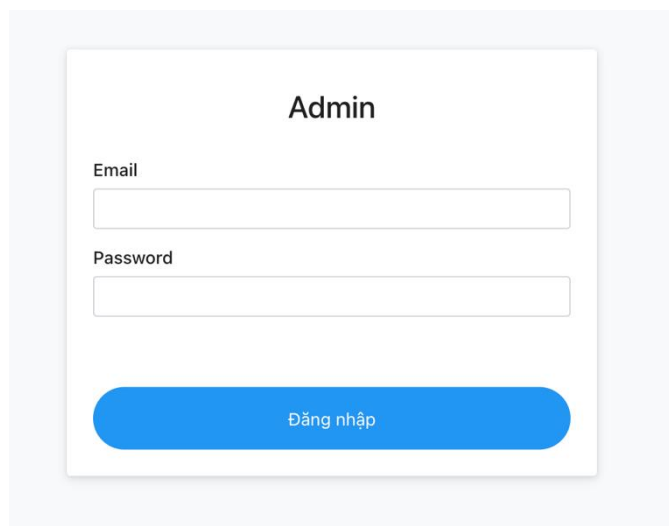
#### 4.1.2. PHP storm:

- Yêu cầu phần cứng tối thiểu:
  - ✓ CPU: CPU x86 64-bit (Intel / ADM)
  - ✓ RAM: 4GB.
  - ✓ Dung lượng đĩa trống: 4GB

### 4.2. Giao diện chức năng

#### 4.2.1. Quản trị

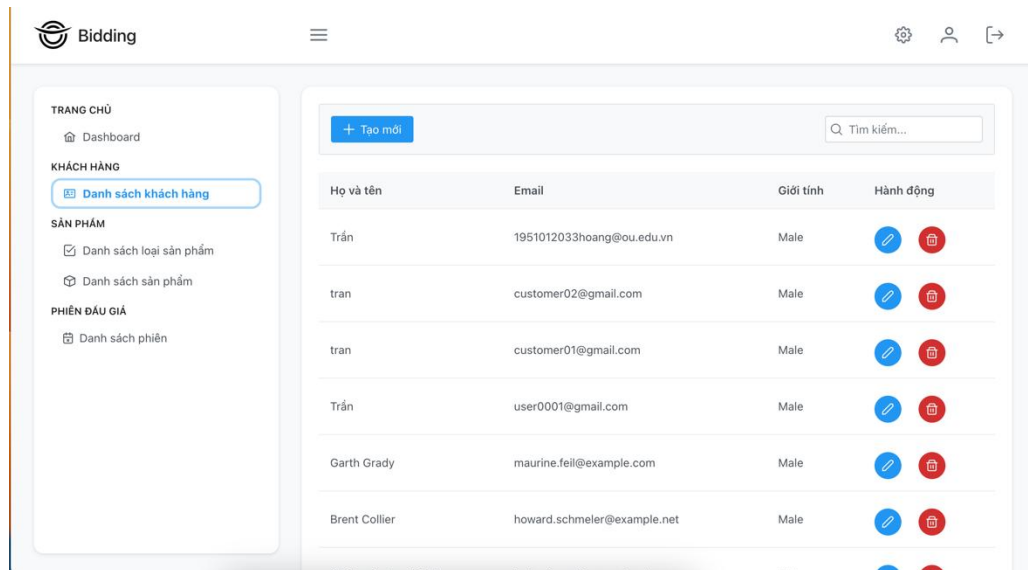
- Chức năng đăng nhập

A screenshot of an admin login interface. It features a white rectangular box with a light gray border on a light gray background. At the top center of the box is the word "Admin" in bold black text. Below it are two input fields: the first is labeled "Email" and the second is labeled "Password", both in a small gray font. At the bottom center of the box is a blue rounded rectangular button with the text "Đăng nhập" in white.

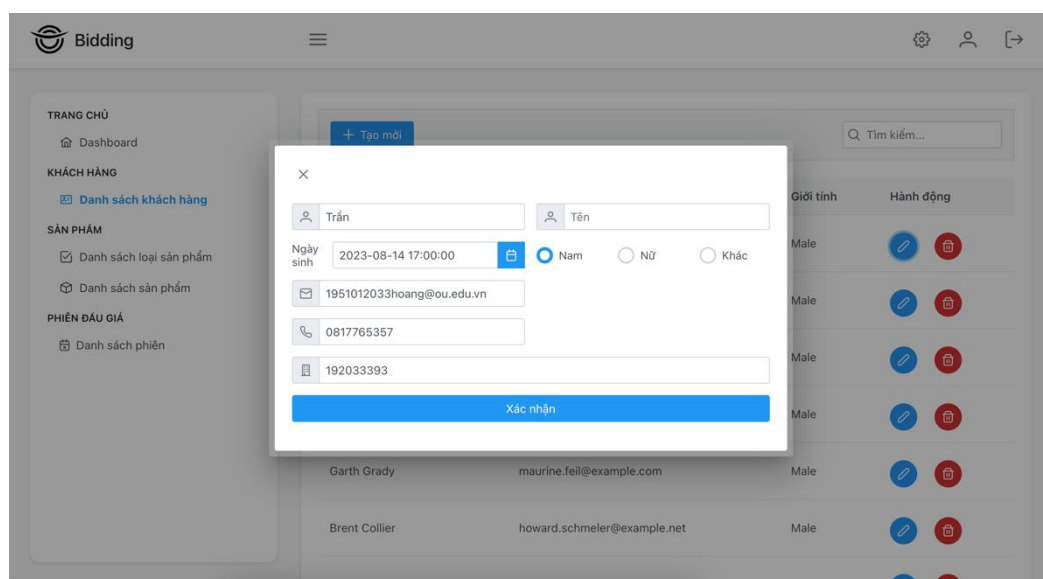
*Hình 4.2.1.1: Chức năng đăng nhập*



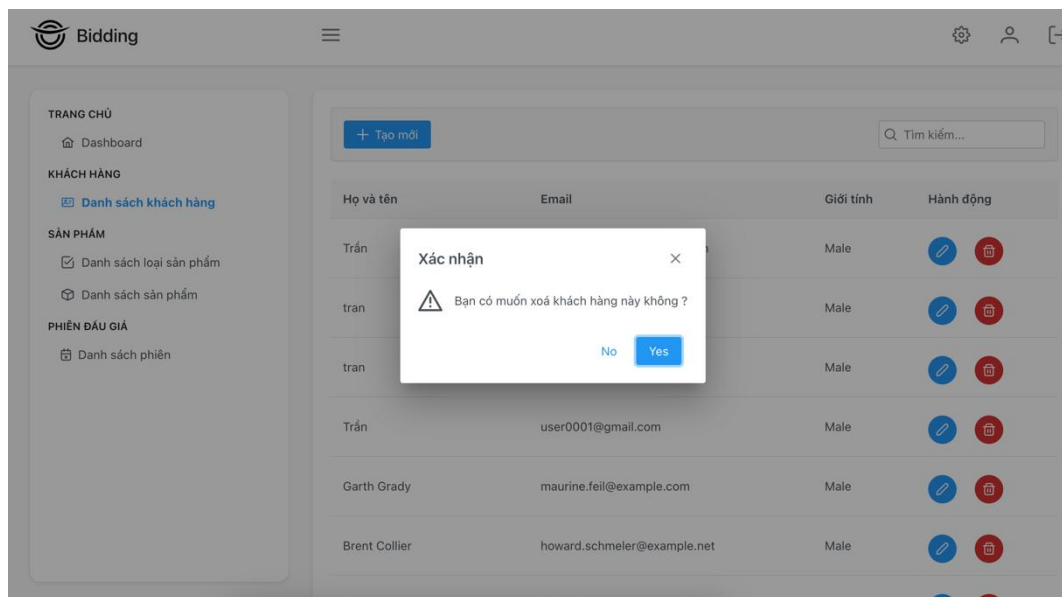
- Chức năng danh sách khách hàng



Hình 4.2.1.2: Chức năng hiển thị danh sách khách hàng

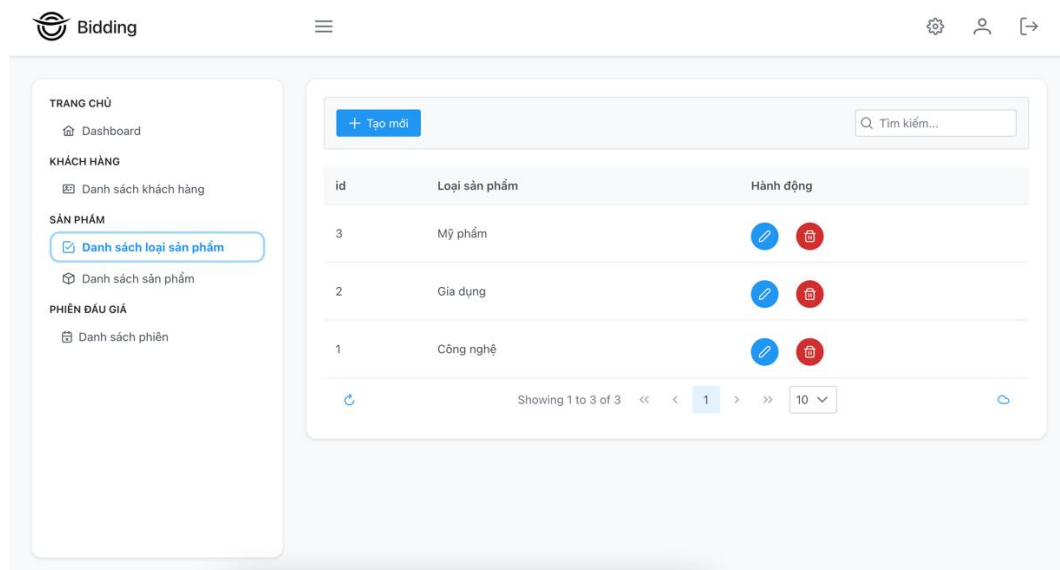


Hình 4.2.1.3: Chức năng chỉnh sửa danh sách khách hàng

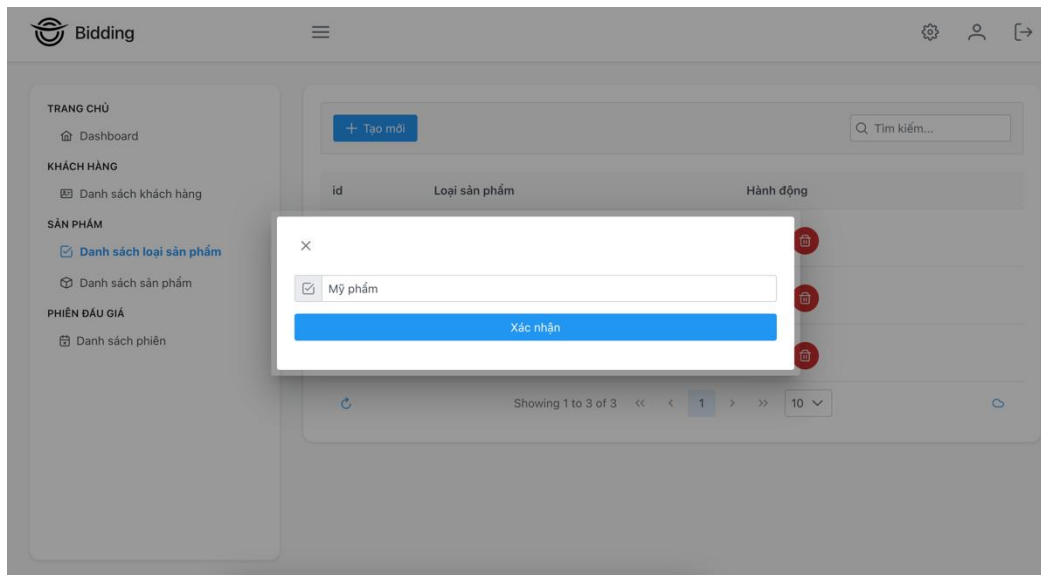


Hình 4.2.1.4: Chức năng xóa thông tin khách hàng

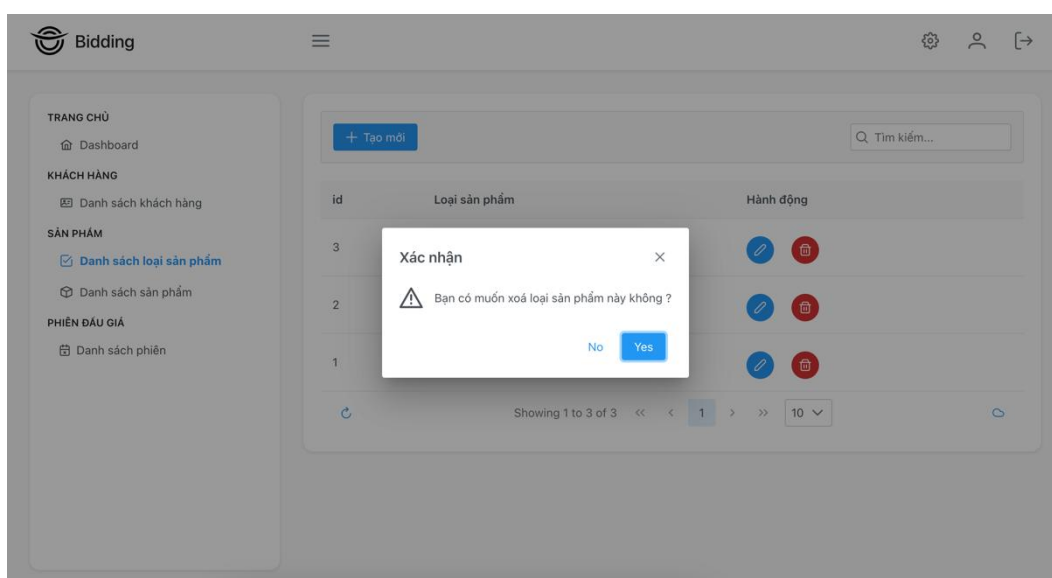
- Chức năng danh sách loại sản phẩm



Hình 4.2.1.5: Chức năng hiển thị danh sách loại sản phẩm

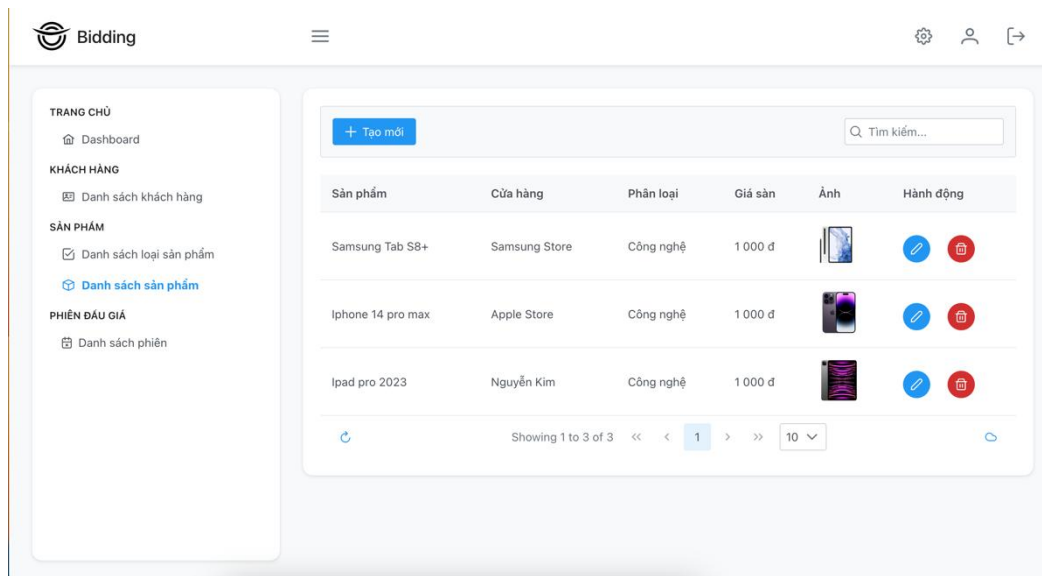


Hình 4.2.1.6: Chức năng chỉnh sửa danh sách loại sản phẩm

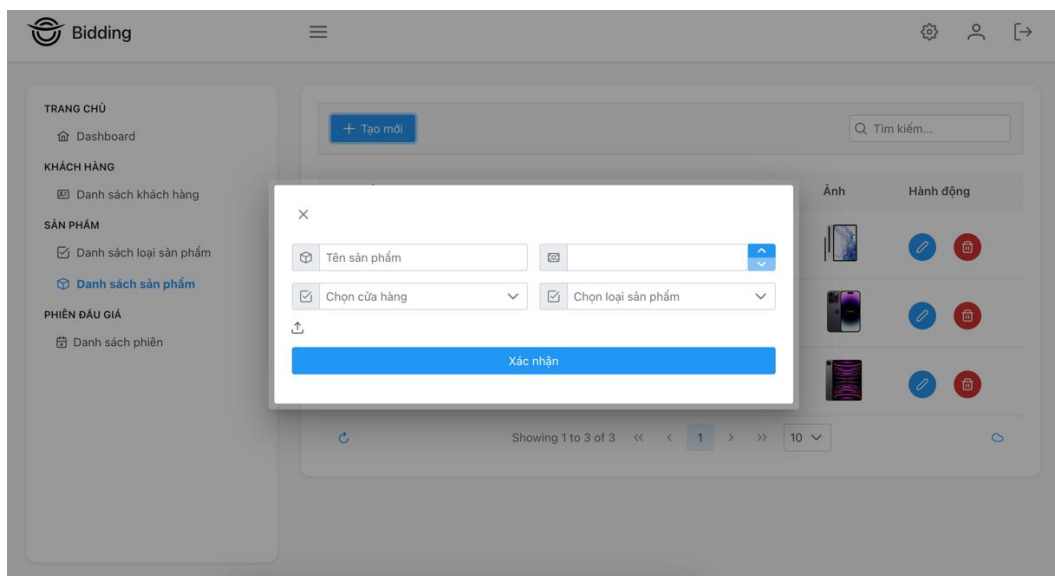


Hình 4.2.1.7: Chức năng xoá danh sách loại sản phẩm

- Chức năng danh sách sản phẩm

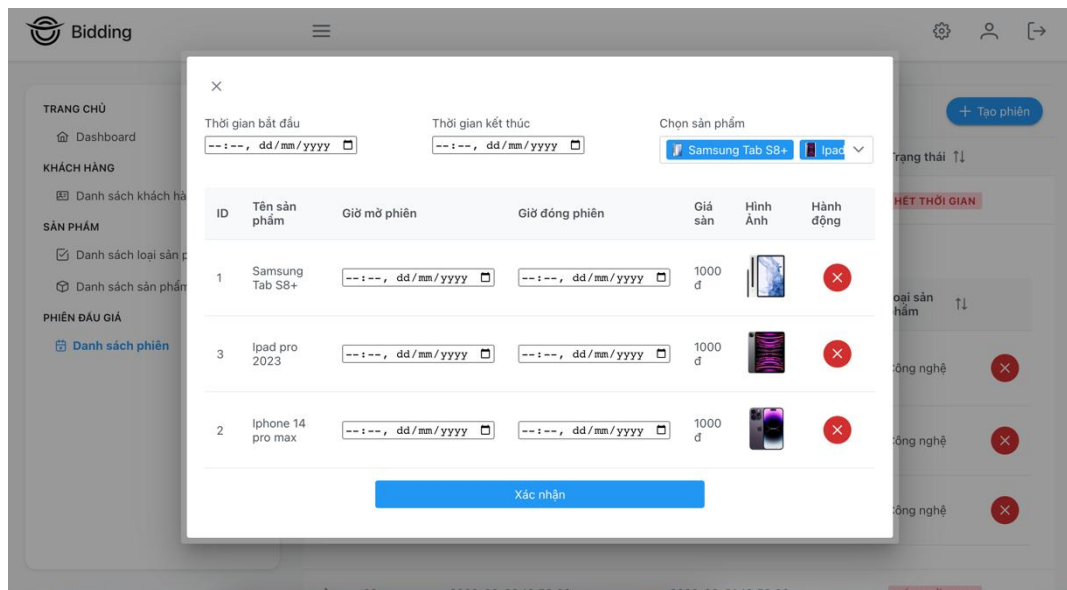


Hình 4.2.1.8: Chức năng hiển thị danh sách sản phẩm

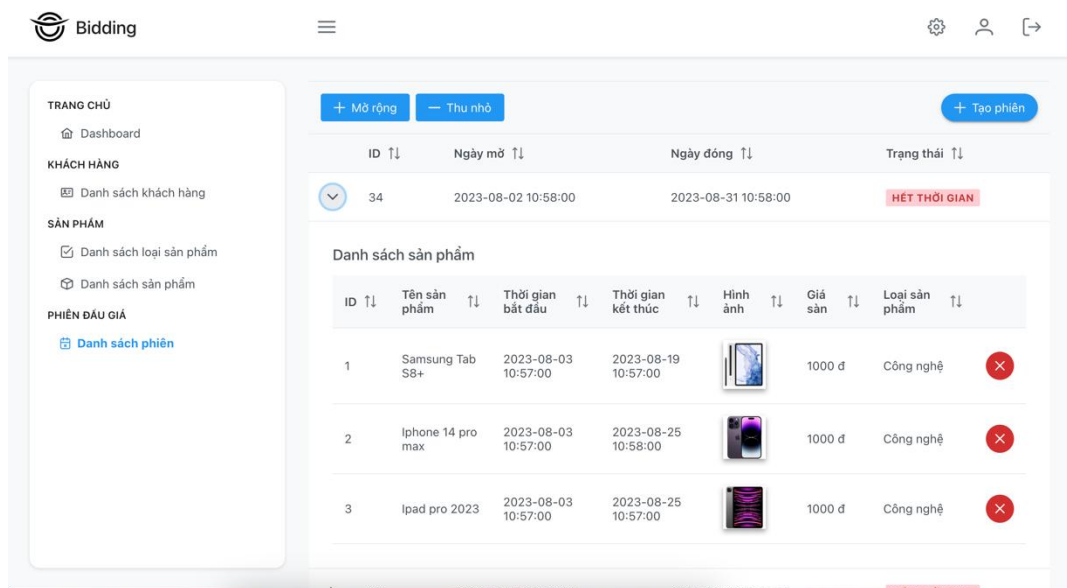


Hình 4.2.1.9: Chức năng tạo sản phẩm

- Chức năng phiên



Hình 4.2.1.10: Chức năng hiển thị phiên




Hình 4.2.1.11 : Chức năng tạo phiên

#### 4.2.2. Khách hàng

- Chức năng đăng ký

**Đăng ký**








**Tạo tài khoản**

*Hình 4.2.2.1 : Chức năng đăng ký*


- Chức năng đăng nhập



**Đăng nhập**

Email

Mật khẩu

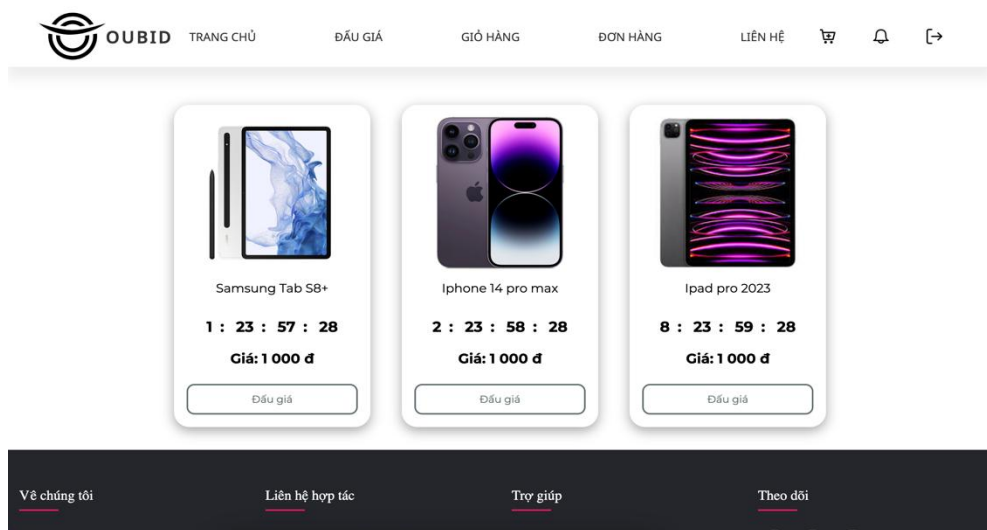


[Chưa có tài khoản ?](#)

**Đăng nhập**

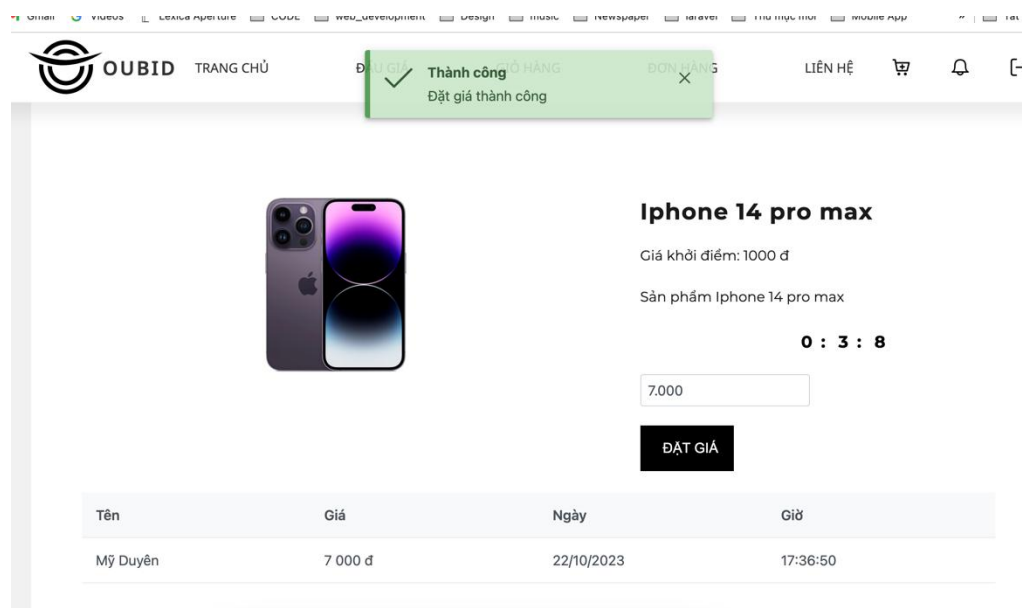
*Hình 4.2.2.2 : Chức năng đăng nhập*

- Chức năng xem danh sách phiên đấu

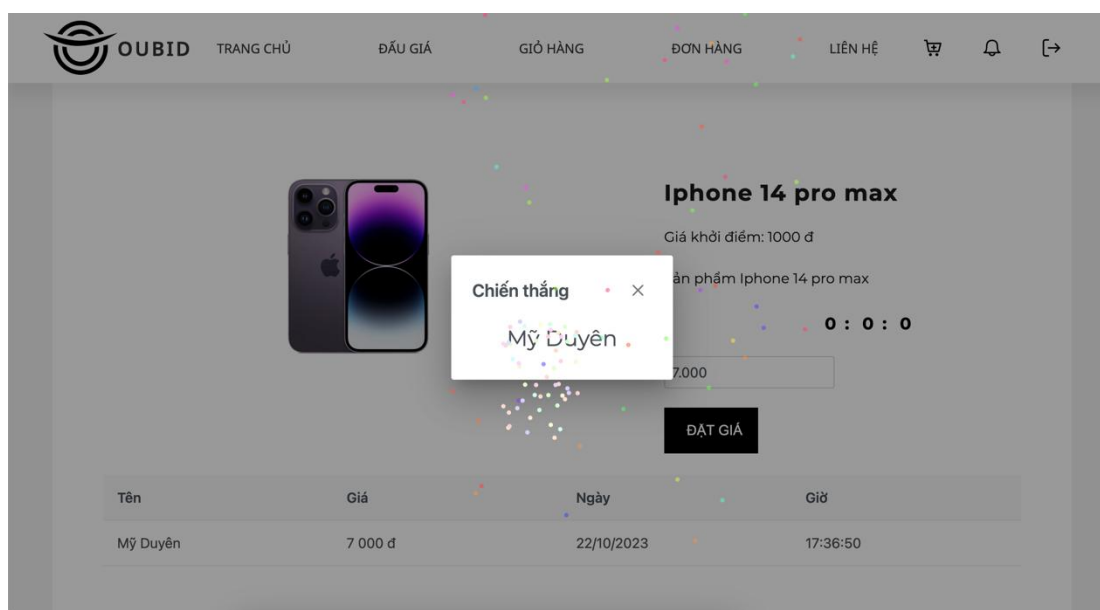


Hình 4.2.2.3 : Chức năng xem danh sách phiên đấu giá

- Chức năng đấu giá



Hình 4.2.2.4 : Chức năng đấu giá



*Hình 4.2.2.5 : Chức năng hiển thị người đấu giá chiến thắng*



## TÀI LIỆU THAM KHẢO

### Các trang web:

- [1]. <https://laravel.com/>
- [2]. <https://vuejs.org/>
- [3]. <https://vuex.vuejs.org/>
- [4]. <https://dashboard.pusher.com/>
- [5]. <https://www.w3schools.com/>

## PHỤ LỤC