

**TRƯỜNG ĐẠI HỌC HẠ LONG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO KẾT THÚC HỌC PHẦN**

**MÔN: PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ**

Đề tài: Xây dựng hệ thống phần mềm mã nguồn mở quản lý

kí túc xá của sinh viên ở trường Đại học Hạ Long

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** TS. Nguyễn Văn Thẩm | |
| **Lớp:** KHMTK6B | |
|  |  |
| **Sinh viên thực hiện** | **Điểm** |
| 1. Max THONGMEEXAY |  |
| 2. Hong SYHANON |  |
| 3. Oudomphet LAOLYETEXA |  |
|  | |
| **Giảng viên chấm 1 Giảng viên chấm 2** | |

*Quảng Ninh, tháng 06 năm 2024*

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến tất cả quý thầy cô và các bạn đã đồng hành cùng em trong quá trình thực hiện báo cáo học phần với đề tài "Xây dựng hệ thống phần mềm mã nguồn mở quản lý kí túc xá của sinh viên ở trường Đại học Hạ Long". Sự hướng dẫn và hỗ trợ từ quý thầy cô đã giúp em hoàn thành bài báo cáo này một cách thành công và đạt được những kết quả đáng kể.

Em xin dành một phần cảm ơn đặc biệt đến giáo viên Nguyễn Văn Thẩm, người đã dành thời gian và nỗ lực tận tình để hướng dẫn và giúp đỡ em trong quá trình thực hiện báo cáo này. Với sự am hiểu và kiến thức sâu rộng, thầy đã không ngừng truyền đạt những kiến thức quý báu cũng như tạo điều kiện thuận lợi nhất để em có thể tiến bộ trong học tập và nghiên cứu. Sự hướng dẫn và hỗ trợ của thầy không chỉ giúp em hoàn thành tốt báo cáo mà còn là nguồn động viên lớn lao trong quá trình nâng cao năng lực và kỹ năng của bản thân. Em cảm kích sâu sắc những lời khuyên, sự chỉ dẫn và sự đồng hành của thầy trong suốt thời gian qua.

Sự tận tâm và lòng nhiệt thành của thầy đã góp phần quan trọng vào sự thành công của dự án này. Một lần nữa, em xin bày tỏ lòng biết ơn sâu sắc và lòng kính trọng đến giáo viên Nguyễn Văn Thẩm. Mong rằng, những đóng góp và sự hỗ trợ của thầy sẽ tiếp tục là nguồn động viên quý báu cho em trong những công việc tương lai.

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc167456063)

[CHƯƠNG 1.TỔNG QUAN BÀI TOÁN QUẢN LÝ KÍ TÚC XÁ CỦA SINH VIÊN Ở TRƯỜNG ĐẠI HỌC HẠ LONG 4](#_Toc167456064)

[1.1. Trình bày tổng quan bài toán quản lý kí túc xá của sinh viên ở trường đại học hạ long 4](#_Toc167456065)

[1.2. Trình bày các giải pháp thực hiện 4](#_Toc167456066)

[CHƯƠNG 2. PHÁT TRIỂN MÃ NGUỒN WEBSITE QUẢN LÝ KÍ TÚC XÁ CỦA SINH VIÊN Ở TRƯỜNG ĐẠI HỌC HẠ LONG 6](#_Toc167456067)

[2.1. Mô hình cơ sở dữ liệu của website quản lý kí túc xá của sinh viên ở trường đại học hạ long 6](#_Toc167456068)

[2.1.1.Sơ đồ phân rã chức năng 6](#_Toc167456069)

[2.1.2. Các bảng dữ liệu 6](#_Toc167456070)

[2.1.3. Mô hình quan hệ 8](#_Toc167456071)

[2.2. Cấu trúc mô hình MVC trên mã nguồn mở Laravel 8](#_Toc167456072)

[2.2.1. Cấu trúc chung 8](#_Toc167456073)

[2.2.2. Cấu hình các route 9](#_Toc167456074)

[2.2.3. Cấu hình cơ sở dữ liệu 10](#_Toc167456075)

[2.2.4. Tầng Controller 11](#_Toc167456076)

[2.2.5. Tầng Model 12](#_Toc167456077)

[2.2.6. Tầng View 13](#_Toc167456078)

[2.2.7. Tầng Controller 14](#_Toc167456079)

[CHƯƠNG 3.CÀI ĐẶT VÀ CÁC GIAO DIỆN WEBSITE 16](#_Toc167456080)

[3.1. Giao diện trang chủ 16](#_Toc167456081)

[3.2. Giao diện các trang cho nhà quản trị 16](#_Toc167456082)

[3.3. Giao diện các trang cho người dùng 21](#_Toc167456083)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 25](#_Toc167456084)

[1. Kết luận 25](#_Toc167456085)

[2. Hướng phát triển 25](#_Toc167456086)

[TÀI LIỆU THAM KHẢO 26](#_Toc167456087)

CHƯƠNG 1

TỔNG QUAN BÀI TOÁN **QUẢN LÝ KÍ TÚC XÁ CỦA SINH VIÊN Ở TRƯỜNG ĐẠI HỌC HẠ LONG**

1.1. Trình bày tổng quan bài toán **quản lý kí túc xá của sinh viên ở trường đại học hạ long**

Bài toán quản lý ký túc xá của sinh viên tại Trường Đại học Hạ Long đòi hỏi một hệ thống quản lý toàn diện để điều phối và giám sát các khía cạnh khác nhau của cuộc sống sinh viên trong ký túc xá, bao gồm quản lý sinh viên, phòng ở, tài khoản, hình ảnh ký túc xá, và các kênh liên hệ.

Quản lý sinh viên là phần quan trọng của hệ thống, bao gồm việc theo dõi và cập nhật thông tin cá nhân, học tập, và tình trạng lưu trú của sinh viên. Hệ thống cần có khả năng đăng ký và lưu trữ thông tin cá nhân, quản lý hồ sơ sinh viên, và theo dõi tình trạng lưu trú. Việc quản lý phòng đòi hỏi hệ thống phải đăng ký và cập nhật thông tin phòng, quản lý tình trạng phòng, và phân bổ phòng cho sinh viên dựa trên nhu cầu và tình trạng thực tế.

Quản lý tài khoản giúp đảm bảo an ninh và sự truy cập hợp lý vào hệ thống, bao gồm quản lý người dùng, bảo mật thông tin, và hỗ trợ khôi phục tài khoản. Quản lý hình ảnh ký túc xá minh họa các tiện nghi và điều kiện sống trong ký túc xá, bao gồm lưu trữ và quản lý hình ảnh, cập nhật và xóa hình ảnh, và hiển thị hình ảnh trên website. Cuối cùng, quản lý liên hệ tạo kênh giao tiếp hiệu quả giữa sinh viên, phụ huynh, và ban quản lý ký túc xá, bao gồm cung cấp thông tin liên lạc, biểu mẫu liên hệ trực tuyến, và hệ thống hỗ trợ và phản hồi. Một hệ thống quản lý ký túc xá tích hợp các chức năng này sẽ đảm bảo hoạt động hiệu quả và thuận tiện, cải thiện trải nghiệm của sinh viên và nâng cao hiệu quả quản lý của nhà trường.

Bên cạnh đó, việc tích hợp các tính năng hỗ trợ quản lý tài chính, bảo trì và sửa chữa cơ sở vật chất cũng như theo dõi tình trạng vệ sinh, an ninh sẽ giúp nâng cao chất lượng dịch vụ và tạo ra môi trường sống và học tập tốt nhất cho sinh viên. Sự phát triển của một hệ thống quản lý toàn diện không chỉ đáp ứng nhu cầu hiện tại mà còn tạo điều kiện thuận lợi cho việc mở rộng và nâng cấp trong tương lai, góp phần xây dựng một cộng đồng ký túc xá an toàn, hiện đại và thân thiện.

1.2. Trình bày các giải pháp thực hiện

Để thực hiện bài toán quản lý kí túc xá của sinh viên tại Trường Đại học Hạ Long một cách hiệu quả, việc áp dụng các giải pháp trên là cần thiết. Từ việc xây dựng hệ thống quản lý thông tin sinh viên đến việc quản lý thu chi minh bạch và đào tạo nhân viên, mỗi bước đều đóng vai trò quan trọng trong việc tạo ra một môi trường sống và học tập tốt nhất cho sinh viên.

**Xây dựng hệ thống quản lý thông tin sinh viên**: Để quản lý một cách hiệu quả, cần có một hệ thống quản lý thông tin sinh viên hoàn chỉnh. Hệ thống này bao gồm các thông tin cá nhân, thông tin về phòng ở, học phí, lịch sử thanh toán, và các dịch vụ đã sử dụng. Sử dụng công nghệ thông tin để tự động hóa việc cập nhật và theo dõi thông tin sinh viên.

**Quản lý phòng ở hiệu quả**: Việc phân chia và quản lý phòng ở một cách tổ chức và công bằng là rất quan trọng. Đảm bảo rằng mỗi sinh viên đều có điều kiện sống và học tập tốt nhất. Sử dụng phần mềm quản lý phòng ở để theo dõi tình trạng phòng, sửa chữa và bảo dưỡng.

**Quản lý thu chi minh bạch**: Xây dựng hệ thống quản lý thu chi minh bạch để ghi nhận và theo dõi các khoản thu học phí, chi tiêu cho vận hành và bảo dưỡng kí túc xá, cũng như chi tiêu cho các dịch vụ hỗ trợ sinh viên. Sử dụng phần mềm quản lý tài chính để tự động hóa quy trình này.

**Quản lý tài khoản chính xác**: Xây dựng hệ thống quản lý tài khoản đầy đủ và chính xác, ghi nhận và kiểm soát các giao dịch thu chi, báo cáo tài chính định kỳ. Sử dụng phần mềm quản lý tài khoản để tăng tính chính xác và minh bạch.

**Quản lý thông tin và giao tiếp**: Xây dựng hệ thống quản lý thông tin và giao tiếp để truyền đạt thông tin liên quan đến kí túc xá và sinh viên một cách đồng nhất và kịp thời. Sử dụng các kênh truyền thông hiện đại như email, tin nhắn, và mạng xã hội để tối ưu hóa việc giao tiếp.

**Đào tạo và hỗ trợ nhân viên**: Đào tạo và hỗ trợ nhân viên quản lý kí túc xá về các quy trình và công nghệ mới, giúp họ hiểu rõ hơn và làm việc hiệu quả hơn. Điều này cũng đảm bảo rằng dịch vụ được cung cấp cho sinh viên là tốt nhất có thể.

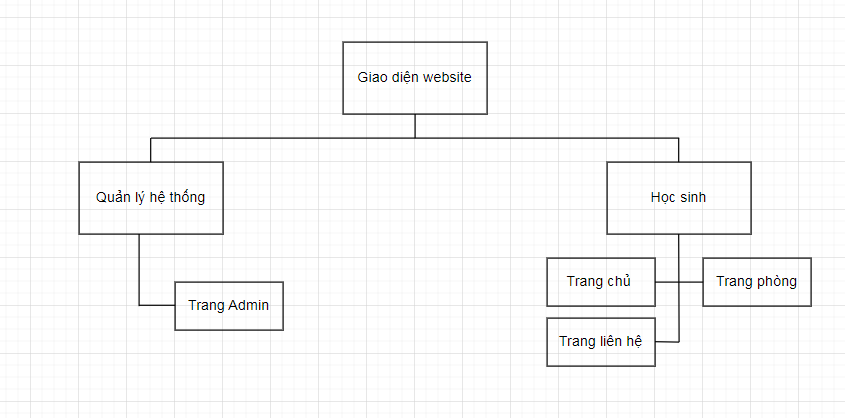
Bằng cách áp dụng các giải pháp trên một cách toàn diện và liên tục, Trường Đại học Hạ Long có thể thực hiện bài toán quản lý kí túc xá của sinh viên một cách hiệu quả và minh bạch, giúp tạo ra một môi trường sống và học tập lý tưởng cho sinh viên.

CHƯƠNG 2

PHÁT TRIỂN MÃ NGUỒN WEBSITE **QUẢN LÝ KÍ TÚC XÁ CỦA SINH VIÊN Ở TRƯỜNG ĐẠI HỌC HẠ LONG**

2.1. Mô hình cơ sở dữ liệu của website **quản lý kí túc xá của sinh viên ở trường đại học hạ long**

2.1.1.Sơ đồ phân rã chức năng



2.1.2. Các bảng dữ liệu

**Bảng hình ảnh**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | id | Int | Khóa chính |
| 2 | hinhanh | Varchar(255) | Hình ảnh |

**Bảng học sinh**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | id | Int | Khóa chính |
| 2 | hoten | Varchar(255) | Họ tên |
| 3 | sdt | Varchar(255) | Số điện thoại |
| 4 | diachi | Varchar(255) | Địa chỉ |
| 5 | lop | Varchar(255) | Lớp |
| 6 | trangthai | Tinyint | Trạng thải |
| 7 | phong\_id | Int | Phòng |

**Bảng liên hệ**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | id | Int | Khóa chính |
| 2 | hoten | Varchar(255) | Họ tên |
| 3 | gmail | Varchar(255) | Gmail |
| 4 | tinnhan | Longtext | Tinh nhắn |

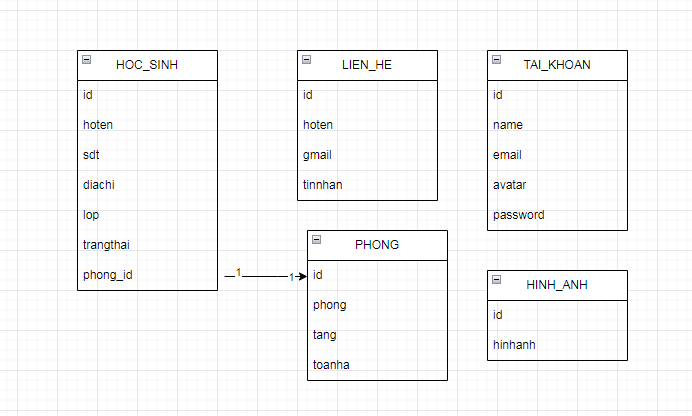
**Bảng phòng**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | id | int | Khóa chính |
| 2 | phong | Varchar(255) | Phòng |
| 3 | tang | Varchar(255) | Tầng |
| 4 | toanha | Varchar(255) | Toàn nhà |

**Bảng tài khoản**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu dữ liệu** | **Mô tả** |
| 1 | id | Bigint | Khóa chính |
| 2 | name | Varchar(255) | Tên |
| 3 | email | Varchar(255) | Email |
| 4 | avatar | Varchar(255) | Ảnh đại diện |
| 5 | password | Varchar(255) | Mật khẩu |

2.1.3. Mô hình quan hệ



2.2. Cấu trúc mô hình MVC trên mã nguồn mở Laravel

2.2.1. Cấu trúc chung

Cấu trúc mô hình MVC trong Laravel, một framework PHP mã nguồn mở, được tổ chức một cách rõ ràng và hiệu quả để giúp các nhà phát triển dễ dàng quản lý và mở rộng ứng dụng.

Thư mục app/ chứa mã nguồn chính của ứng dụng và bao gồm các thư mục con như Console/ (chứa các lệnh Artisan tùy chỉnh), Exceptions/ (xử lý ngoại lệ của ứng dụng), và Http/ (chứa các controller, middleware và các thành phần liên quan đến HTTP). Các controller nằm trong app/Http/Controllers/, chịu trách nhiệm xử lý yêu cầu và trả về phản hồi, trong khi các middleware được đặt trong app/Http/Middleware/ để lọc các yêu cầu HTTP. Các model, đại diện cho dữ liệu và logic nghiệp vụ, được đặt trong app/Models/.

Thư mục bootstrap/ chứa các tệp khởi động và cấu hình ban đầu của ứng dụng. Thư mục config/ lưu trữ các tệp cấu hình quan trọng như cấu hình cơ sở dữ liệu, mail, session, và cache.

Cơ sở dữ liệu được quản lý thông qua thư mục database/, chứa các tệp migration để tạo và chỉnh sửa cấu trúc bảng, các factory để tạo dữ liệu mẫu cho testing và các seeder để tạo dữ liệu mặc định.

Thư mục public/ chứa các tệp công khai như index.php (điểm vào chính của ứng dụng) và các tệp tĩnh như CSS, JavaScript, và hình ảnh. Các tài nguyên của ứng dụng được đặt trong thư mục resources/, bao gồm js/, lang/, sass/, và views/.

Định tuyến của ứng dụng được quản lý trong thư mục routes/ với các tệp như web.php, api.php, console.php, và channels.php.

Thư mục storage/ chứa các tệp được tạo bởi ứng dụng trong quá trình chạy, bao gồm các tệp của ứng dụng, các tệp framework như session, cache và views đã biên dịch, và các tệp log.

Thư mục tests/ chứa các tệp kiểm thử, phân chia thành các kiểm thử tính năng và kiểm thử đơn vị.

Cuối cùng, thư mục vendor/ chứa các thư viện của bên thứ ba được quản lý bởi Composer. Tổng thể, cấu trúc MVC trong Laravel giúp tách biệt các phần khác nhau của ứng dụng, làm cho mã nguồn dễ hiểu, dễ bảo trì và mở rộng, đồng thời cung cấp một framework mạnh mẽ để phát triển ứng dụng web một cách nhanh chóng và hiệu quả.

2.2.2. Cấu hình các route

Trong cấu trúc mô hình MVC của Laravel, việc cấu hình các route là một phần quan trọng giúp định tuyến các yêu cầu HTTP đến các controller phù hợp, qua đó điều khiển luồng xử lý và phản hồi của ứng dụng. Các route được định nghĩa trong thư mục routes/, nơi chứa các tệp như web.php, api.php, console.php, và channels.php, mỗi tệp có một vai trò cụ thể trong định tuyến. Tệp web.php chứa các route dành cho ứng dụng web và thường sử dụng các phiên và CSRF protection, phù hợp với các ứng dụng có giao diện người dùng.

Ví dụ, một route trong web.php có thể định nghĩa một đường dẫn đến một phương thức trong controller như sau: Route::get('/posts', [PostController::class, 'index'])->name('posts.index');, trong đó get chỉ định phương thức HTTP GET, /posts là URL, và PostController@index là phương thức trong controller để xử lý yêu cầu này. Tương tự, api.php chứa các route dành cho API, thường không sử dụng session hay CSRF protection, cho phép tạo các ứng dụng RESTful. Một ví dụ về route trong api.php có thể là: Route::get('/posts', [PostController::class, 'index']);, tương tự như trong web.php nhưng với cấu hình phù hợp cho API.

Tệp console.php định nghĩa các lệnh Artisan tùy chỉnh, cho phép thực hiện các tác vụ từ dòng lệnh, ví dụ như tạo model, chạy migration, hay seed dữ liệu. Cấu trúc của một lệnh trong console.php có thể như sau: Artisan::command('inspire', function () { $this->comment(Inspiring::quote()); });, định nghĩa một lệnh tùy chỉnh mang tên inspire.

Cuối cùng, tệp channels.php được sử dụng để đăng ký các kênh broadcast, giúp ứng dụng có thể phát và nghe các sự kiện trong thời gian thực. Ví dụ, một định nghĩa kênh trong channels.php có thể là: Broadcast::channel('order.{orderId}', function ($user, $orderId) { return $user->id === Order::find($orderId)->user\_id; });, trong đó xác thực người dùng có quyền nghe kênh cụ thể dựa trên điều kiện nhất định.

Tóm lại, cấu hình các route trong Laravel là một quá trình xác định rõ ràng các điểm vào của ứng dụng, phân tách logic xử lý theo các loại yêu cầu khác nhau, và kết nối các yêu cầu này với các phương thức thích hợp trong các controller, từ đó giúp tổ chức và điều hướng luồng xử lý của ứng dụng một cách hiệu quả và linh hoạt.

2.2.3. Cấu hình cơ sở dữ liệu

Trong cấu trúc mô hình MVC (Model-View-Controller) của mã nguồn mở Laravel, việc cấu hình cơ sở dữ liệu đóng vai trò quan trọng trong việc kết nối và tương tác với các nguồn dữ liệu. Laravel cung cấp một cách tiếp cận đơn giản và linh hoạt để thiết lập và quản lý cơ sở dữ liệu thông qua tập tin cấu hình .env. Tập tin này chứa các thông tin thiết yếu như loại cơ sở dữ liệu (MySQL, PostgreSQL, SQLite, SQL Server), tên cơ sở dữ liệu, tên người dùng, mật khẩu, và địa chỉ máy chủ. Ví dụ, để cấu hình một cơ sở dữ liệu MySQL, bạn sẽ cần chỉ định các giá trị tương ứng cho DB\_CONNECTION, DB\_HOST, DB\_PORT, DB\_DATABASE, DB\_USERNAME, và DB\_PASSWORD. Những giá trị này sẽ được đọc bởi Laravel khi ứng dụng khởi động và sẽ thiết lập kết nối với cơ sở dữ liệu dựa trên các thông số đã định nghĩa.

Hơn nữa, Laravel hỗ trợ nhiều kết nối cơ sở dữ liệu, cho phép bạn dễ dàng tương tác với nhiều cơ sở dữ liệu khác nhau trong cùng một ứng dụng bằng cách định nghĩa các kết nối bổ sung trong tập tin cấu hình config/database.php. Tập tin này không chỉ cho phép bạn thiết lập nhiều kết nối mà còn cung cấp các tùy chọn cấu hình nâng cao như cấu hình kết nối lại tự động, sử dụng bộ đệm query, và điều chỉnh các thông số kết nối. Laravel cũng cung cấp một ORM (Object-Relational Mapping) mạnh mẽ tên là Eloquent, giúp việc làm việc với cơ sở dữ liệu trở nên dễ dàng hơn bằng cách cho phép bạn làm việc với các mô hình và quan hệ giữa chúng một cách tự nhiên như làm việc với các đối tượng trong PHP.

Thêm vào đó, Laravel tích hợp sẵn các công cụ di trú cơ sở dữ liệu (database migrations), giúp bạn dễ dàng tạo, thay đổi và quản lý cấu trúc bảng cơ sở dữ liệu theo thời gian một cách nhất quán và có thể kiểm soát được. Các di trú được viết dưới dạng các lớp PHP và được lưu trữ trong thư mục database/migrations, cho phép bạn dễ dàng chia sẻ và kiểm soát phiên bản của cấu trúc cơ sở dữ liệu trong quá trình phát triển nhóm. Với sự hỗ trợ mạnh mẽ từ các công cụ này, cấu hình cơ sở dữ liệu trong Laravel không chỉ đơn giản và trực quan mà còn cực kỳ linh hoạt, phù hợp với các nhu cầu phát triển ứng dụng từ nhỏ đến lớn, từ đơn giản đến phức tạp. Hơn nữa, Laravel hỗ trợ nhiều kết nối cơ sở dữ liệu, cho phép bạn dễ dàng tương tác với nhiều cơ sở dữ liệu khác nhau trong cùng một ứng dụng bằng cách định nghĩa các kết nối bổ sung trong tập tin cấu hình config/database.php. Tập tin này không chỉ cho phép bạn thiết lập nhiều kết nối mà còn cung cấp các tùy chọn cấu hình nâng cao như cấu hình kết nối lại tự động, sử dụng bộ đệm query, và điều chỉnh các thông số kết nối. Laravel cũng cung cấp một ORM (Object-Relational Mapping) mạnh mẽ tên là Eloquent, giúp việc làm việc với cơ sở dữ liệu trở nên dễ dàng hơn bằng cách cho phép bạn làm việc với các mô hình và quan hệ giữa chúng một cách tự nhiên như làm việc với các đối tượng trong PHP.

Thêm vào đó, Laravel tích hợp sẵn các công cụ di trú cơ sở dữ liệu (database migrations), giúp bạn dễ dàng tạo, thay đổi và quản lý cấu trúc bảng cơ sở dữ liệu theo thời gian một cách nhất quán và có thể kiểm soát được. Các di trú được viết dưới dạng các lớp PHP và được lưu trữ trong thư mục database/migrations, cho phép bạn dễ dàng chia sẻ và kiểm soát phiên bản của cấu trúc cơ sở dữ liệu trong quá trình phát triển nhóm. Với sự hỗ trợ mạnh mẽ từ các công cụ này, cấu hình cơ sở dữ liệu trong Laravel không chỉ đơn giản và trực quan mà còn cực kỳ linh hoạt, phù hợp với các nhu cầu phát triển ứng dụng từ nhỏ đến lớn, từ đơn giản đến phức tạp.

2.2.4. Tầng Controller

Tầng Controller trong cấu trúc mô hình MVC của Laravel đóng vai trò quan trọng trong việc xử lý các yêu cầu từ người dùng và điều phối các tác vụ của ứng dụng. Mỗi Controller trong Laravel là một lớp PHP, được đặt trong thư mục app/Http/Controllers, và thường được chia thành các phương thức (hoặc hành động) tương ứng với các chức năng cụ thể của ứng dụng.

Công việc chính của Controller là nhận yêu cầu từ phía người dùng thông qua các route, xử lý các yêu cầu này bằng cách gọi các phương thức từ tầng Model để truy vấn hoặc cập nhật dữ liệu, sau đó chuyển kết quả này đến tầng View để hiển thị cho người dùng. Điều này tạo ra một luồng làm việc rõ ràng và dễ hiểu, giúp cho việc phát triển và bảo trì ứng dụng trở nên dễ dàng hơn.

Bên cạnh việc xử lý yêu cầu từ người dùng, các Controller trong Laravel cũng thường thực hiện các công việc phụ trợ như kiểm tra quyền truy cập, xác thực người dùng, gửi các thông báo hoặc thông báo lỗi, và xử lý các yêu cầu AJAX. Điều này giúp cho ứng dụng trở nên an toàn, linh hoạt và dễ sử dụng.

Laravel cung cấp một cách tiếp cận linh hoạt và mạnh mẽ để viết Controller, với sự hỗ trợ của nhiều tính năng như Dependency Injection, Route Model Binding, Middleware, và Eloquent ORM. Điều này giúp cho việc viết code trở nên đơn giản, hiệu quả và dễ bảo trì.

Trong tổ chức của một ứng dụng Laravel, Controller thường được tổ chức theo các module hoặc chức năng cụ thể của ứng dụng, giúp cho code trở nên dễ đọc và dễ bảo trì. Việc này cũng giúp cho các nhóm phát triển có thể làm việc độc lập trên các phần khác nhau của ứng dụng mà không gây ra xung đột hoặc trùng lặp code.

2.2.5. Tầng Model

Tầng Model trong cấu trúc mô hình MVC của Laravel đóng vai trò vô cùng quan trọng trong việc quản lý và tương tác với dữ liệu của ứng dụng. Bằng cách tổ chức các model trong thư mục app/Models/, Laravel giúp cho việc quản lý mã nguồn trở nên dễ dàng và có tổ chức hơn. Mỗi model trong Laravel thường tương ứng với một bảng trong cơ sở dữ liệu và chứa các phương thức để thực hiện các hoạt động như truy vấn dữ liệu, tạo, cập nhật hoặc xóa bản ghi, cũng như xử lý các logic nghiệp vụ liên quan.

Điểm đáng chú ý nhất của Model trong Laravel là khả năng kết nối với cơ sở dữ liệu thông qua Eloquent ORM. Mỗi model được Laravel biểu diễn là một bảng trong cơ sở dữ liệu, với các thuộc tính của model tương ứng với các cột trong bảng. Điều này giúp cho việc thực hiện các thao tác CRUD (Create, Read, Update, Delete) trở nên dễ dàng và linh hoạt hơn bao giờ hết.

Ngoài ra, các model cũng chứa các phương thức để xử lý logic nghiệp vụ của ứng dụng. Điều này bao gồm các quy tắc xác thực, tính toán phức tạp và bất kỳ logic nào khác liên quan đến dữ liệu của model. Sự kết hợp giữa việc quản lý dữ liệu và logic nghiệp vụ trong cùng một địa điểm giúp mã nguồn trở nên dễ bảo trì và tái sử dụng hơn.

Một trong những ưu điểm lớn của Model trong Laravel là tính kế thừa. Bằng cách tạo các model con kế thừa từ các model cha, chúng ta có thể tái sử dụng code và tổ chức logic một cách hiệu quả. Điều này giúp giảm sự lặp lại và tăng tính module hóa của mã nguồn.

Cuối cùng, việc kiểm thử các model trong Laravel cũng trở nên dễ dàng nhờ vào các bộ kiểm thử tích hợp sẵn trong framework. Chúng ta có thể tạo các kiểm thử để đảm bảo rằng các phương thức của model hoạt động như mong đợi và xử lý các trường hợp biên và lỗi một cách hiệu quả.

Tóm lại, tầng Model trong cấu trúc MVC của Laravel đóng vai trò quan trọng trong việc quản lý và xử lý dữ liệu của ứng dụng, cung cấp một cơ chế linh hoạt và mạnh mẽ để tương tác với cơ sở dữ liệu và xử lý logic nghiệp vụ. Điều này giúp cho việc phát triển ứng dụng trở nên dễ dàng, linh hoạt và dễ bảo trì hơn.

2.2.6. Tầng View

Tầng View trong cấu trúc MVC của Laravel đóng vai trò quan trọng trong việc hiển thị dữ liệu và tạo ra giao diện tương tác cho người dùng. Với việc sử dụng Blade template engine, các view được tổ chức và quản lý một cách dễ dàng trong thư mục resources/views/, mang lại sự linh hoạt và tiện lợi trong quá trình phát triển ứng dụng.

Một trong những đặc điểm nổi bật của Tầng View là khả năng mô-đun hóa và tái sử dụng. Blade template engine của Laravel cho phép chúng ta tạo ra các thành phần giao diện như header, footer, menu và sử dụng chúng lại trong nhiều trang khác nhau mà không cần phải viết lại mã nguồn. Điều này giúp cho việc quản lý mã nguồn view trở nên dễ dàng và hiệu quả.

Ngoài ra, Tầng View cũng cho phép tích hợp dữ liệu từ các controller và model một cách dễ dàng thông qua các biến được truyền vào từ controller hoặc thông qua các lệnh Blade để trực tiếp lấy dữ liệu từ model. Điều này giúp cho việc hiển thị dữ liệu động và tương tác với cơ sở dữ liệu trở nên linh hoạt và thuận tiện.

Tính năng tích hợp logic của Blade cũng là một điểm mạnh của Tầng View. Blade template engine cung cấp các cú pháp điều kiện và vòng lặp để tích hợp logic trực tiếp vào các view, giúp chúng ta có thể điều chỉnh cách hiển thị dữ liệu dựa trên các điều kiện hoặc thực hiện các vòng lặp để hiển thị danh sách dữ liệu một cách linh hoạt.

Việc định dạng và thiết kế giao diện cũng trở nên dễ dàng với Tầng View trong Laravel. Chúng ta có thể sử dụng các class CSS, framework CSS như Bootstrap hoặc tự custom để tạo ra giao diện đẹp mắt và chuyên nghiệp. Laravel cung cấp công cụ mix để quản lý các asset như CSS, JavaScript và hình ảnh, giúp tạo ra các trang web hiệu quả và tương thích với nhiều thiết bị.

Cuối cùng, việc kiểm thử cũng trở nên dễ dàng với Tầng View trong Laravel. Chúng ta có thể tạo các kiểm thử để đảm bảo rằng các view hiển thị dữ liệu đúng cách và đảm bảo rằng trải nghiệm người dùng cuối là như mong đợi.

Tóm lại, Tầng View trong cấu trúc MVC của Laravel đóng vai trò quan trọng trong việc tạo ra trải nghiệm người dùng cuối một cách dễ dàng, linh hoạt và hấp dẫn. Với sự kết hợp của Blade template engine và các tính năng mạnh mẽ của Laravel, chúng ta có thể xây dựng các trang web chuyên nghiệp và hiệu quả, từ việc quản lý mã nguồn đến thiết kế giao diện và tích hợp dữ liệu.

2.2.7. Tầng Controller

Tầng Controller trong cấu trúc MVC của Laravel đóng vai trò cực kỳ quan trọng trong quá trình phát triển ứng dụng web. Nó không chỉ đảm nhận vai trò điều phối và xử lý các yêu cầu từ người dùng mà còn là trung tâm của toàn bộ logic ứng dụng. Controllers thường được lưu trữ trong thư mục app/Http/Controllers/, và chúng thực hiện các phương thức để xử lý các yêu cầu HTTP như GET, POST, PUT, DELETE.

Một trong những đặc điểm nổi bật nhất của Tầng Controller là khả năng xử lý yêu cầu HTTP. Mỗi phương thức trong controller thường tương ứng với một loại yêu cầu HTTP như GET, POST, PUT, DELETE. Điều này giúp cho việc quản lý và tổ chức mã nguồn trở nên dễ dàng hơn, đồng thời giúp tách biệt logic xử lý cho từng loại yêu cầu, tạo điều kiện cho việc mở rộng và bảo trì dễ dàng hơn.

Controllers cũng đóng vai trò chính trong việc tương tác với model. Chúng là người điều khiển việc lấy, cập nhật, xóa dữ liệu từ cơ sở dữ liệu thông qua các phương thức được gọi từ các model. Thông qua controllers, chúng ta có thể thực hiện các hoạt động như lấy dữ liệu từ cơ sở dữ liệu, thêm mới bản ghi, cập nhật thông tin, hoặc xóa bản ghi một cách linh hoạt và hiệu quả.

Ngoài ra, Controllers còn chứa logic ứng dụng để xử lý các yêu cầu từ phía người dùng. Điều này bao gồm kiểm tra xác thực, xác định các điều kiện và thực hiện các hành động tương ứng dựa trên các điều kiện đó. Controllers giúp chúng ta đưa ra quyết định thông minh và thực hiện các hành động phù hợp với nhu cầu của ứng dụng.

Quản lý dữ liệu đầu vào và đầu ra cũng là một nhiệm vụ quan trọng của Controllers. Chúng đảm bảo rằng dữ liệu được chuyển đi và từ người dùng được xử lý một cách an toàn và đúng đắn. Controllers kiểm tra và xử lý dữ liệu đầu vào từ người dùng trước khi chuyển cho model xử lý, đồng thời đảm bảo rằng dữ liệu đầu ra được chuẩn bị một cách đúng đắn trước khi được hiển thị lên view.

Bên cạnh đó, Controllers có khả năng tích hợp middleware, một cơ chế mạnh mẽ trong Laravel cho việc thực hiện các công việc trung gian trước hoặc sau khi xử lý các yêu cầu HTTP. Middleware có thể được sử dụng trong controllers để thực hiện các tác vụ như xác thực người dùng, kiểm tra quyền truy cập, và thực hiện các tác vụ khác trước khi xử lý yêu cầu.

Cuối cùng, việc kiểm thử Controllers trong Laravel cũng trở nên dễ dàng hơn bao giờ hết. Chúng ta có thể tạo các bộ kiểm thử để đảm bảo rằng các phương thức trong controllers hoạt động như mong đợi, giúp đảm bảo tính ổn định và độ tin cậy của ứng dụng.

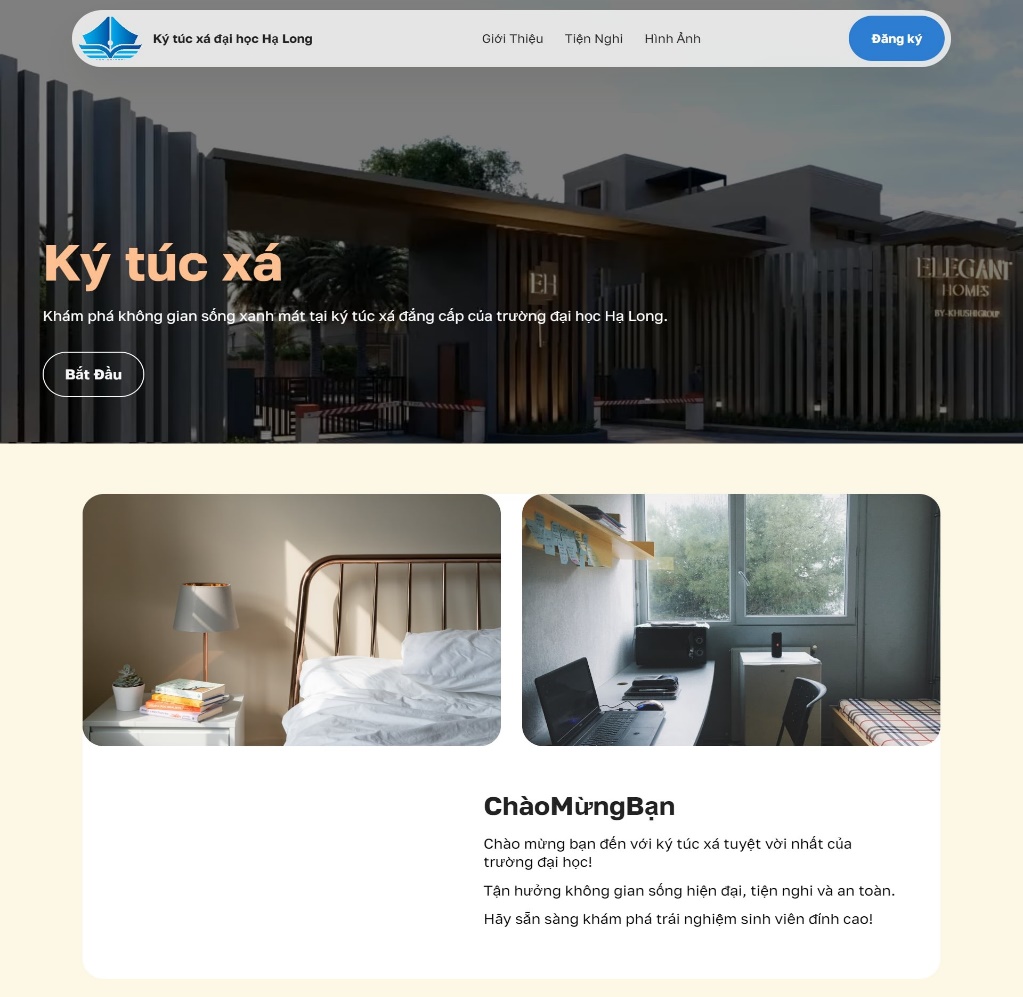
Tóm lại, Tầng Controller trong cấu trúc MVC của Laravel không chỉ đóng vai trò quan trọng trong việc điều phối và xử lý yêu cầu từ người dùng mà còn chịu trách nhiệm quản lý logic ứng dụng và tương tác với model và view. Với tính linh hoạt, hiệu quả và khả năng kiểm thử dễ dàng, Controllers là một thành phần không thể thiếu trong việc phát triển ứng dụng Laravel.

CHƯƠNG 3

CÀI ĐẶT VÀ CÁC GIAO DIỆN WEBSITE

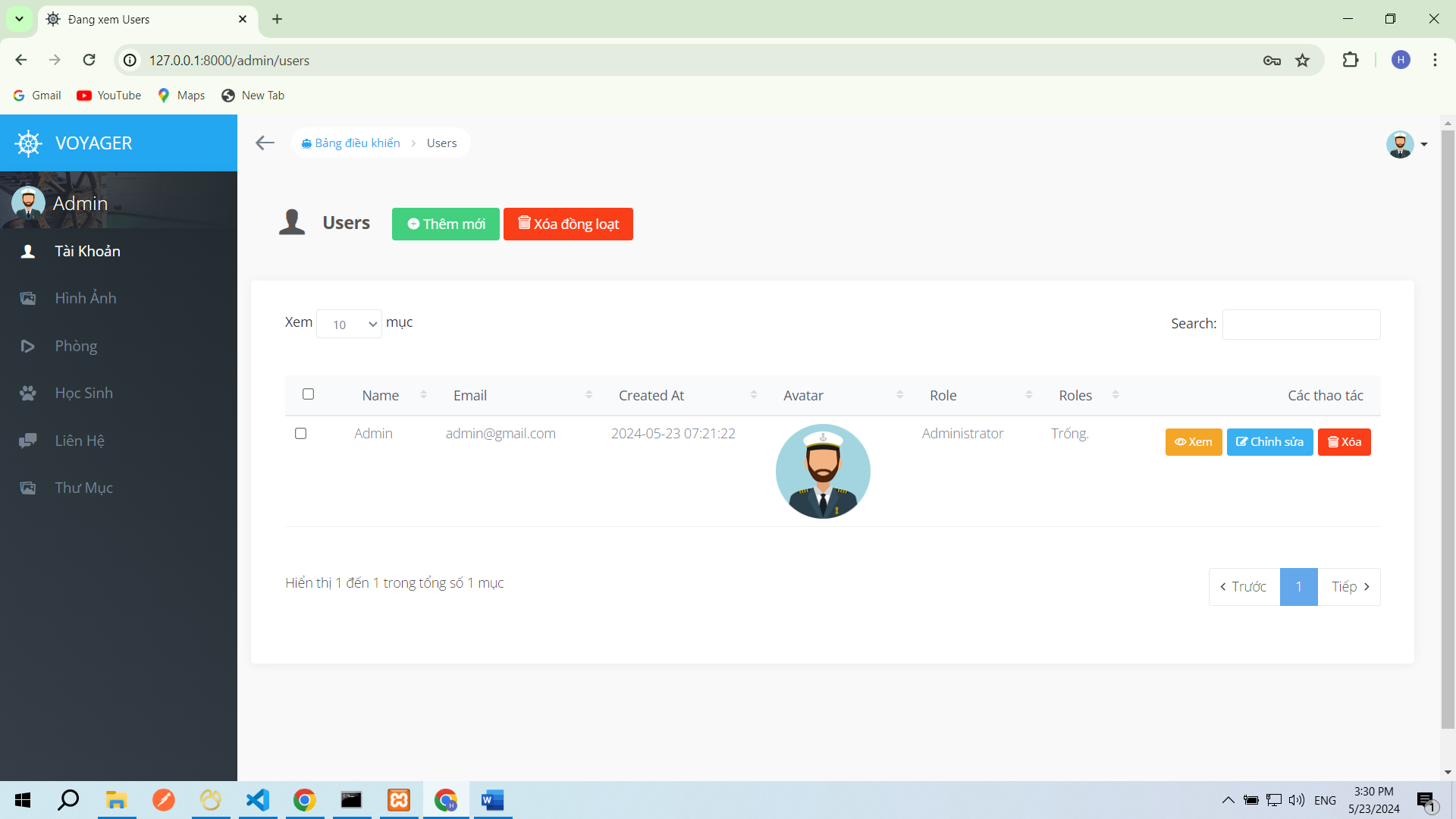
3.1. Giao diện trang chủ

**Trang chủ**

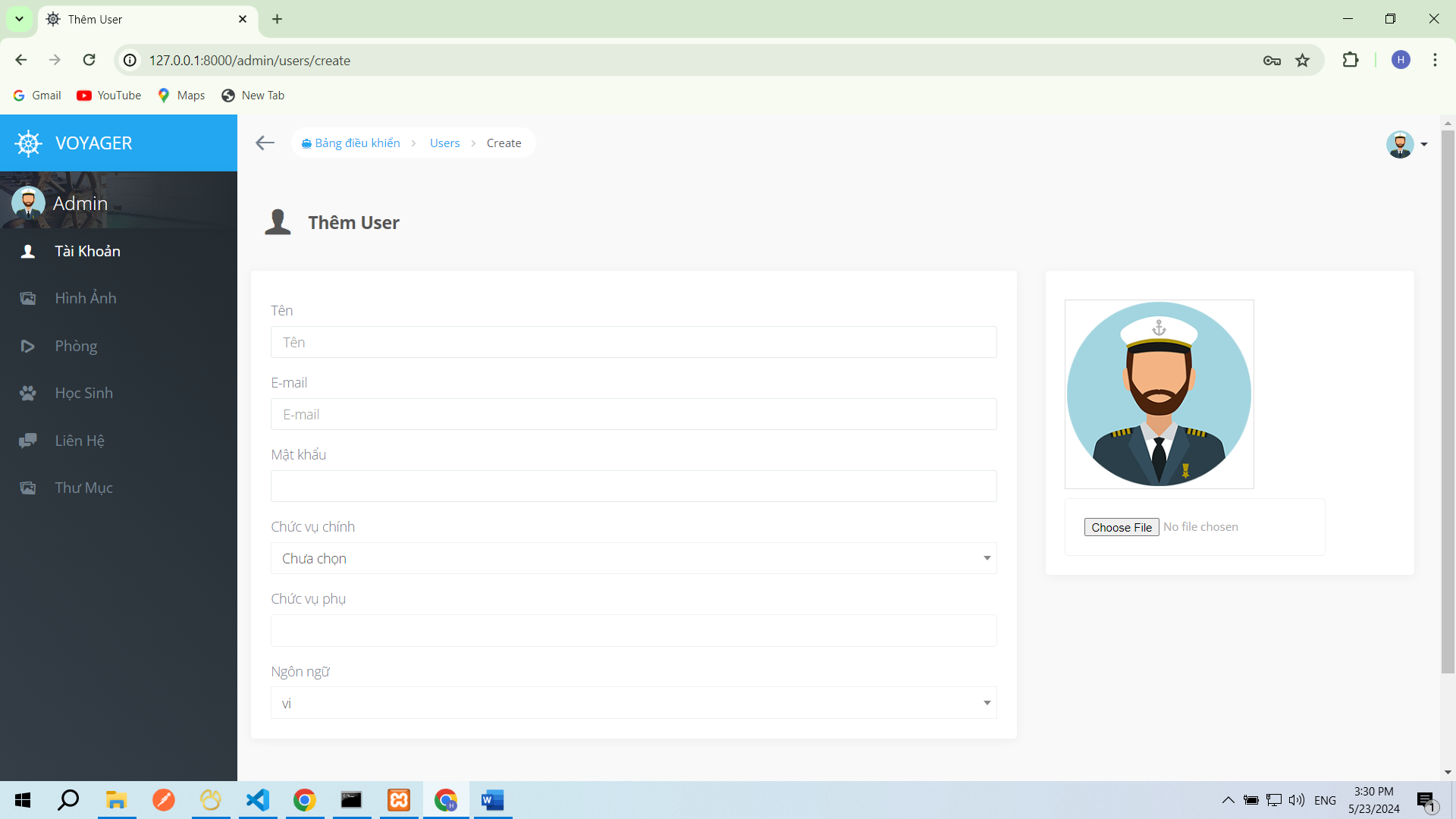


3.2. Giao diện các trang cho nhà quản trị

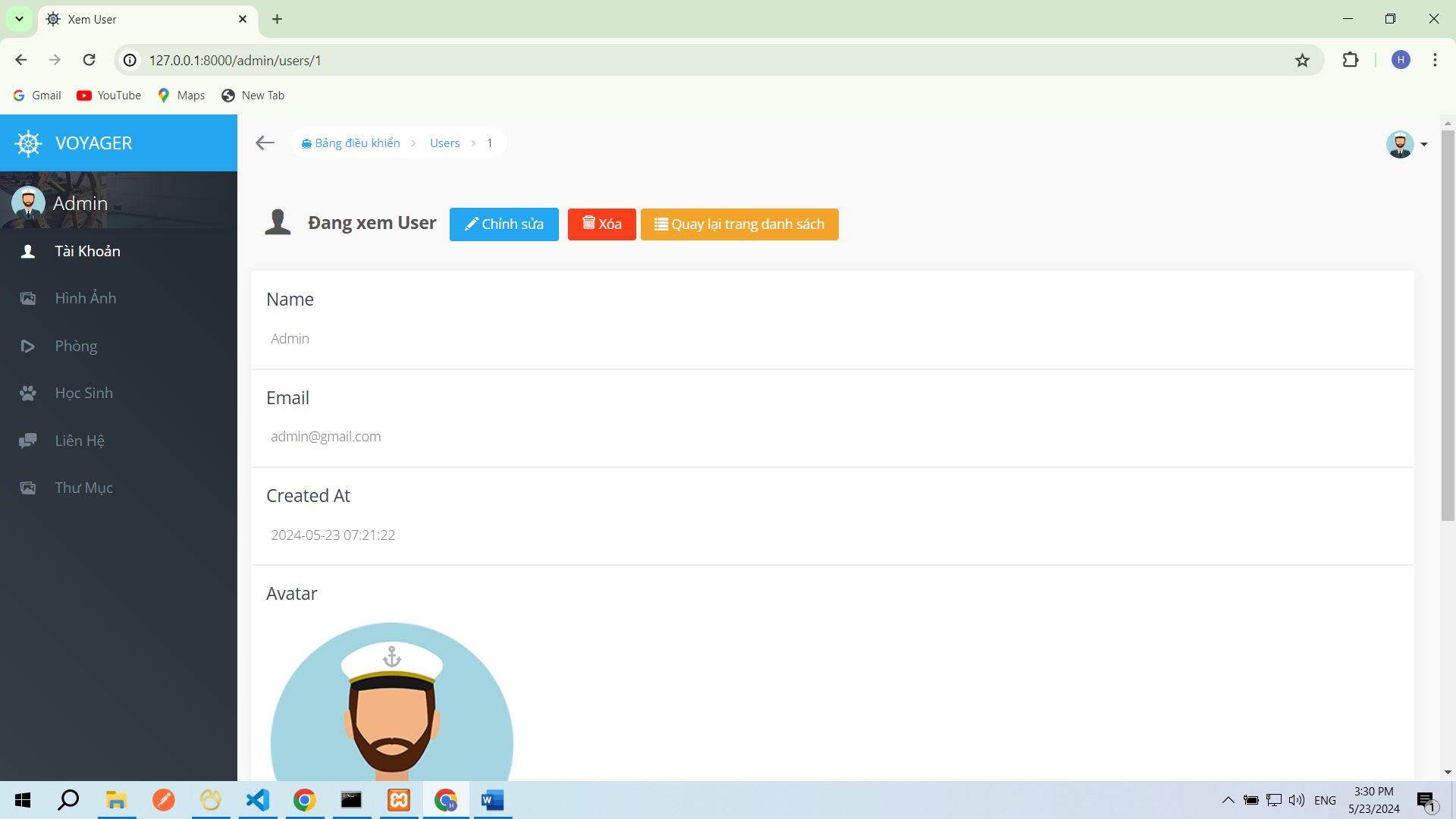
**Trang tài khoản** : có các chức năng thêm, sửa, xóa tài khoản



hình 1.1: xem danh sách tài khoản

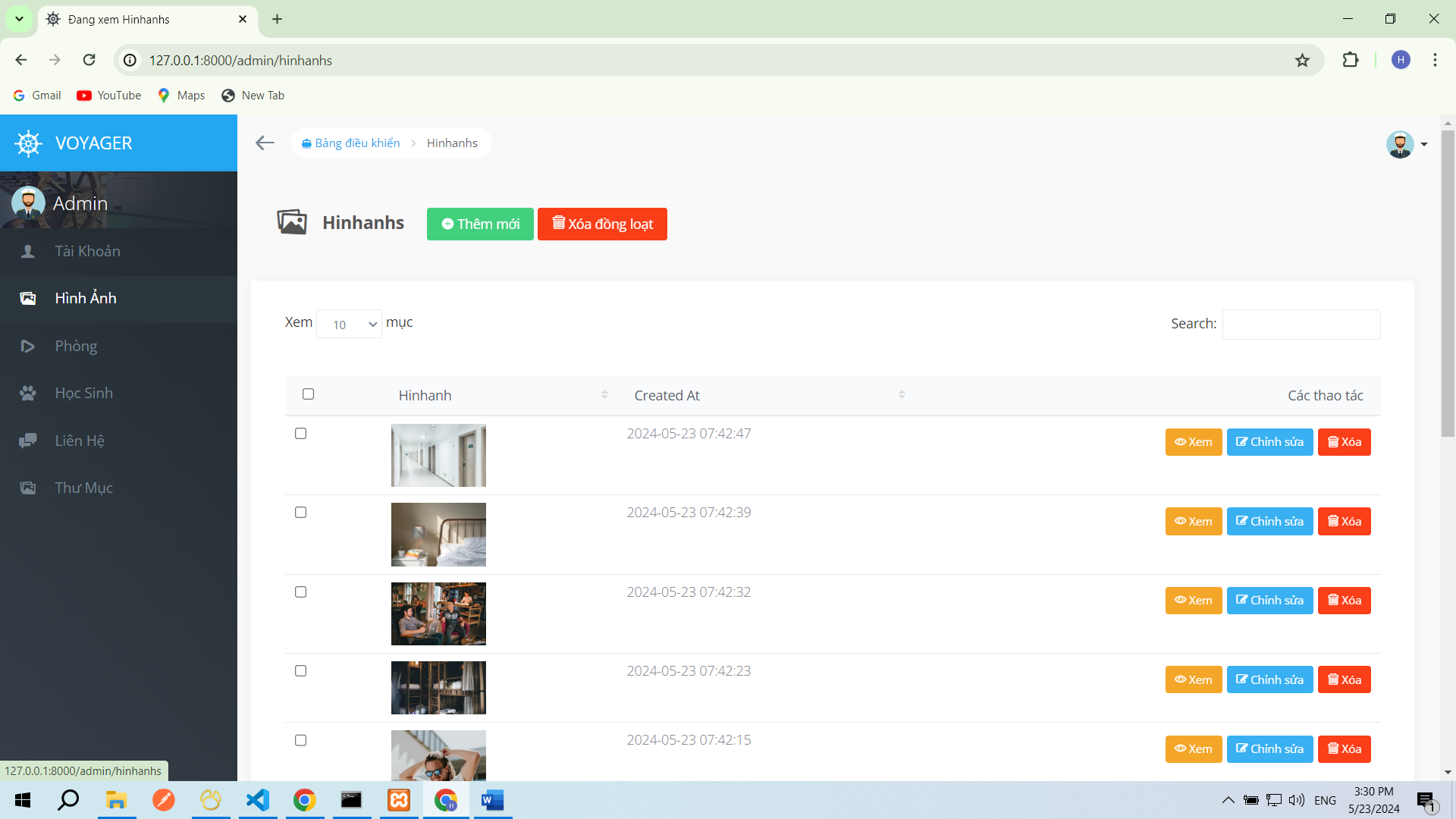


hình 1.2: thêm mới tài khoản

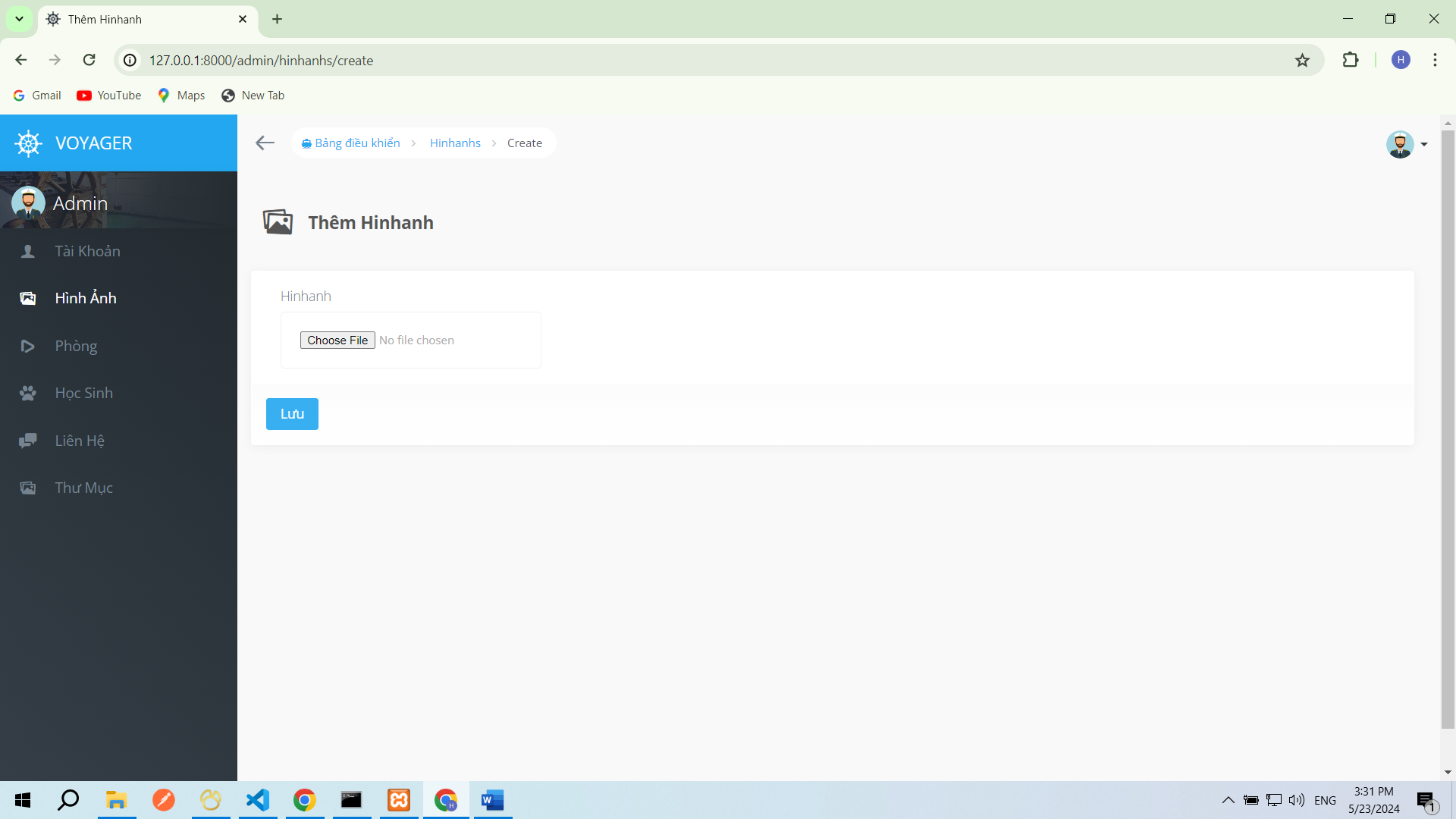


hình 1.3: xem chi tiết tài khoản

**Trang hình ảnh** : có các chức năng thêm, sửa, xóa hình ảnh



hình 2.1: xem danh sách hình ảnh

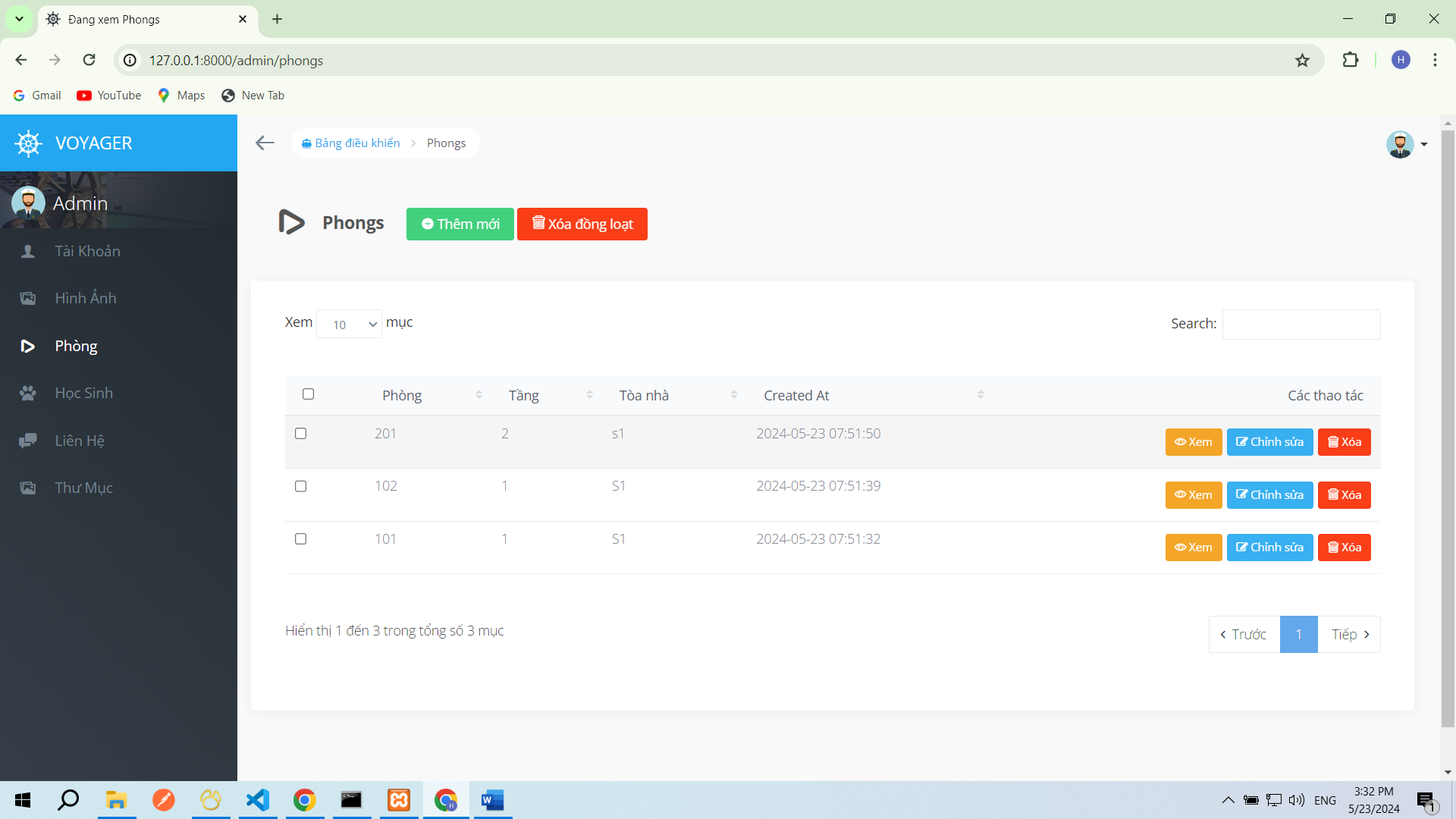


hình 2.2: thêm mới hình ảnh

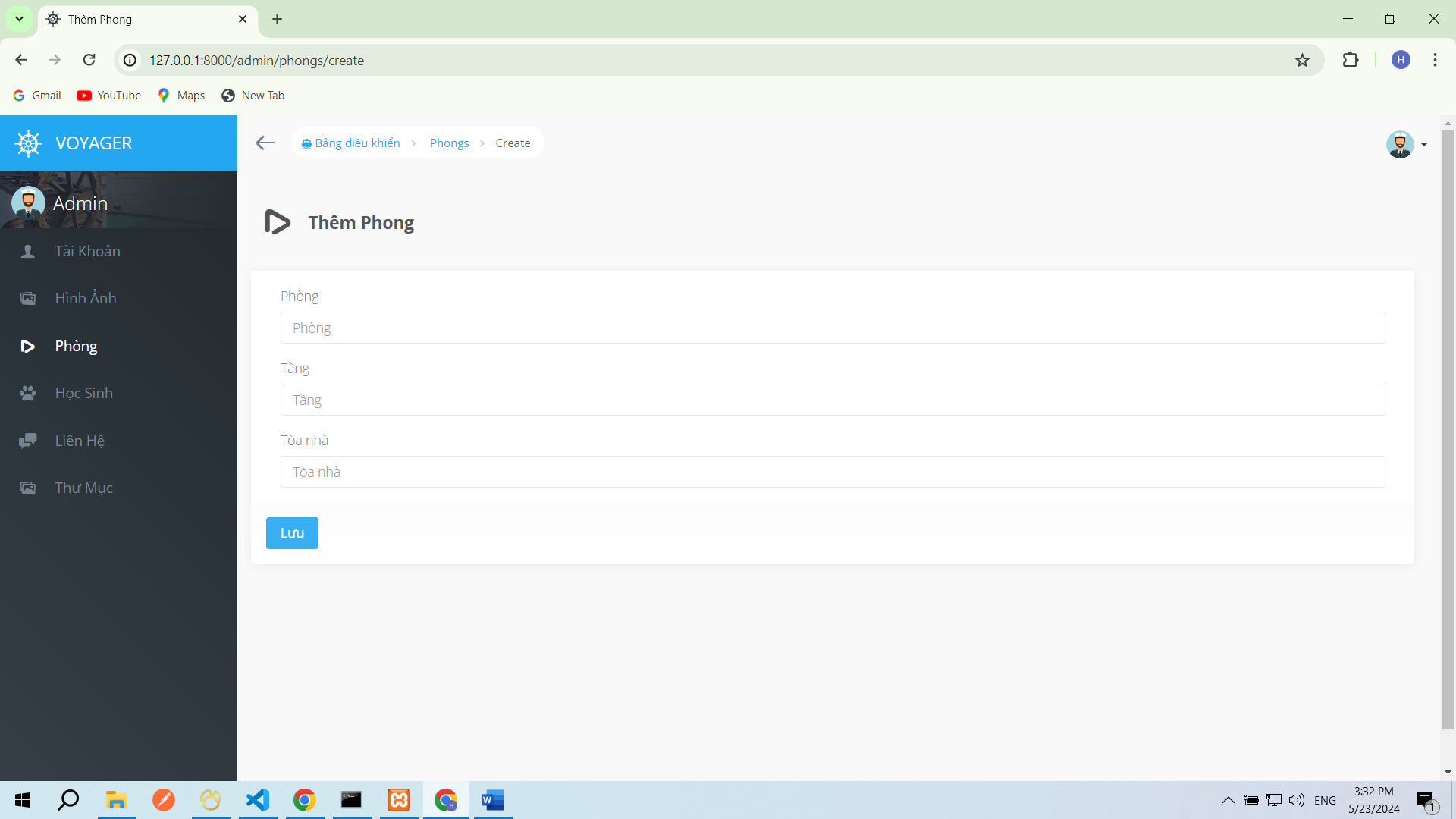


hình 2.3: xem chi tiết hình ảnh

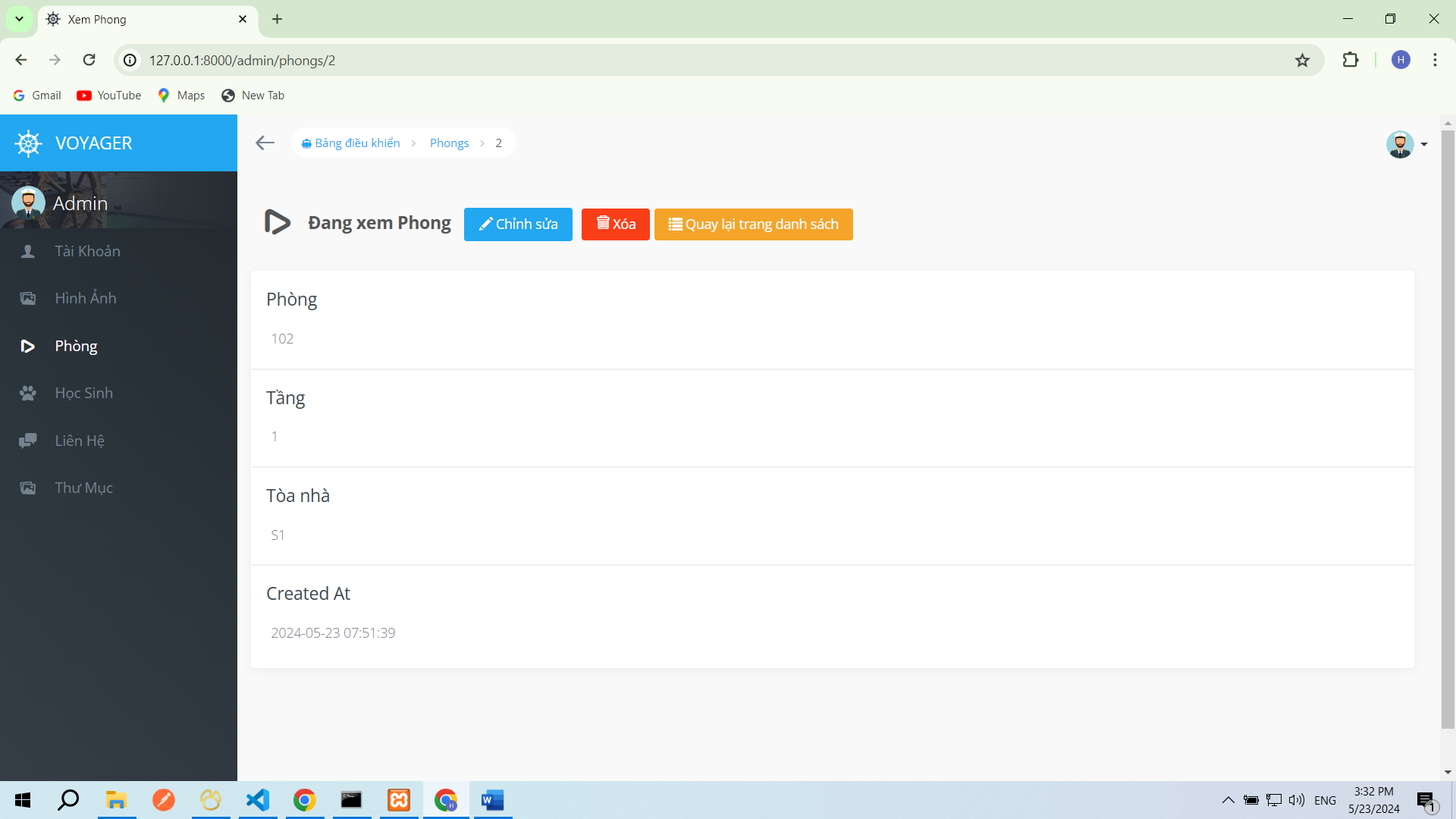
**Trang phòng** : có các chức năng thêm, sửa, xóa phòng



hình 3.1: xem danh sách phòng

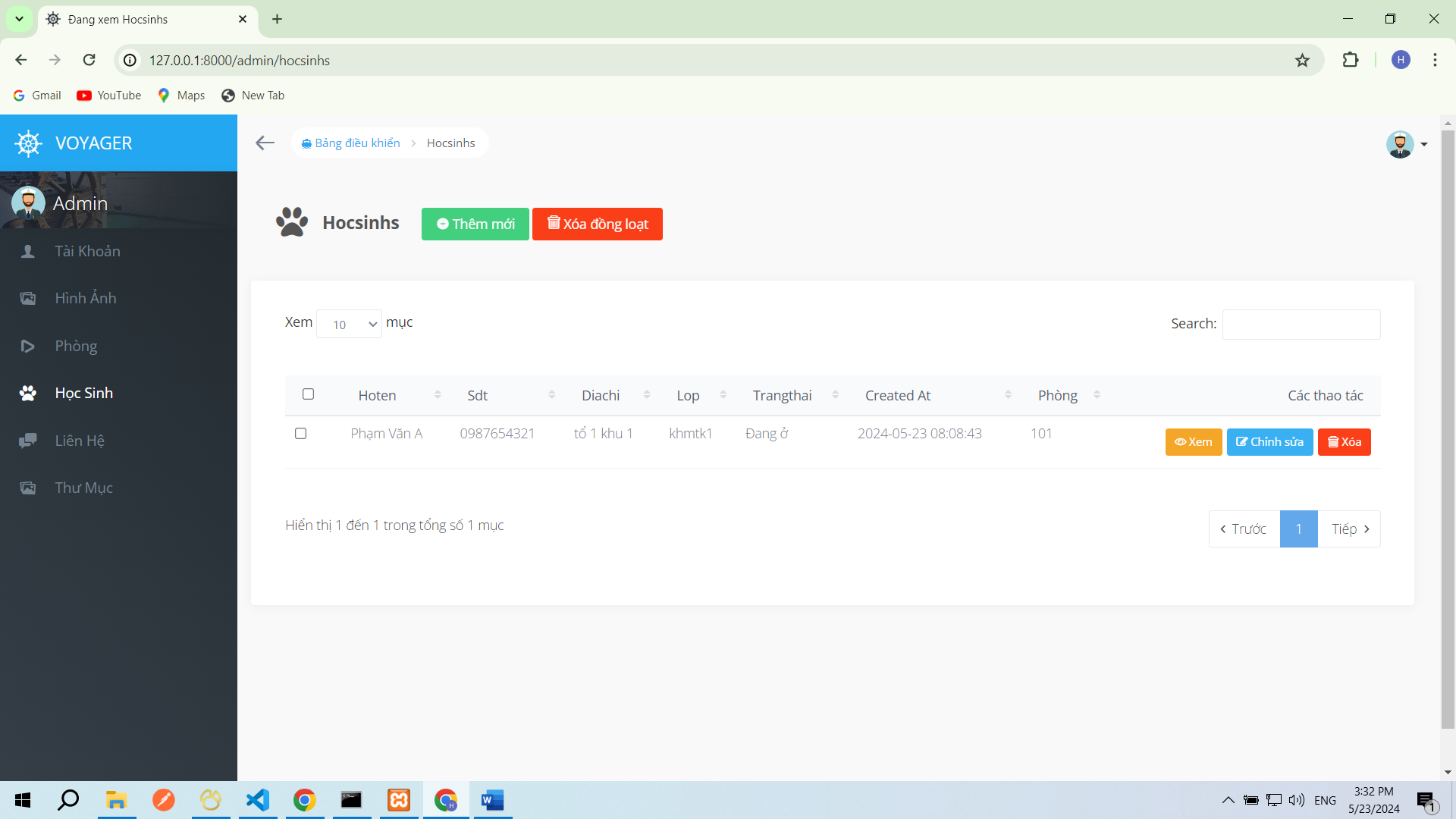


hình 3.2: thêm mới phòng

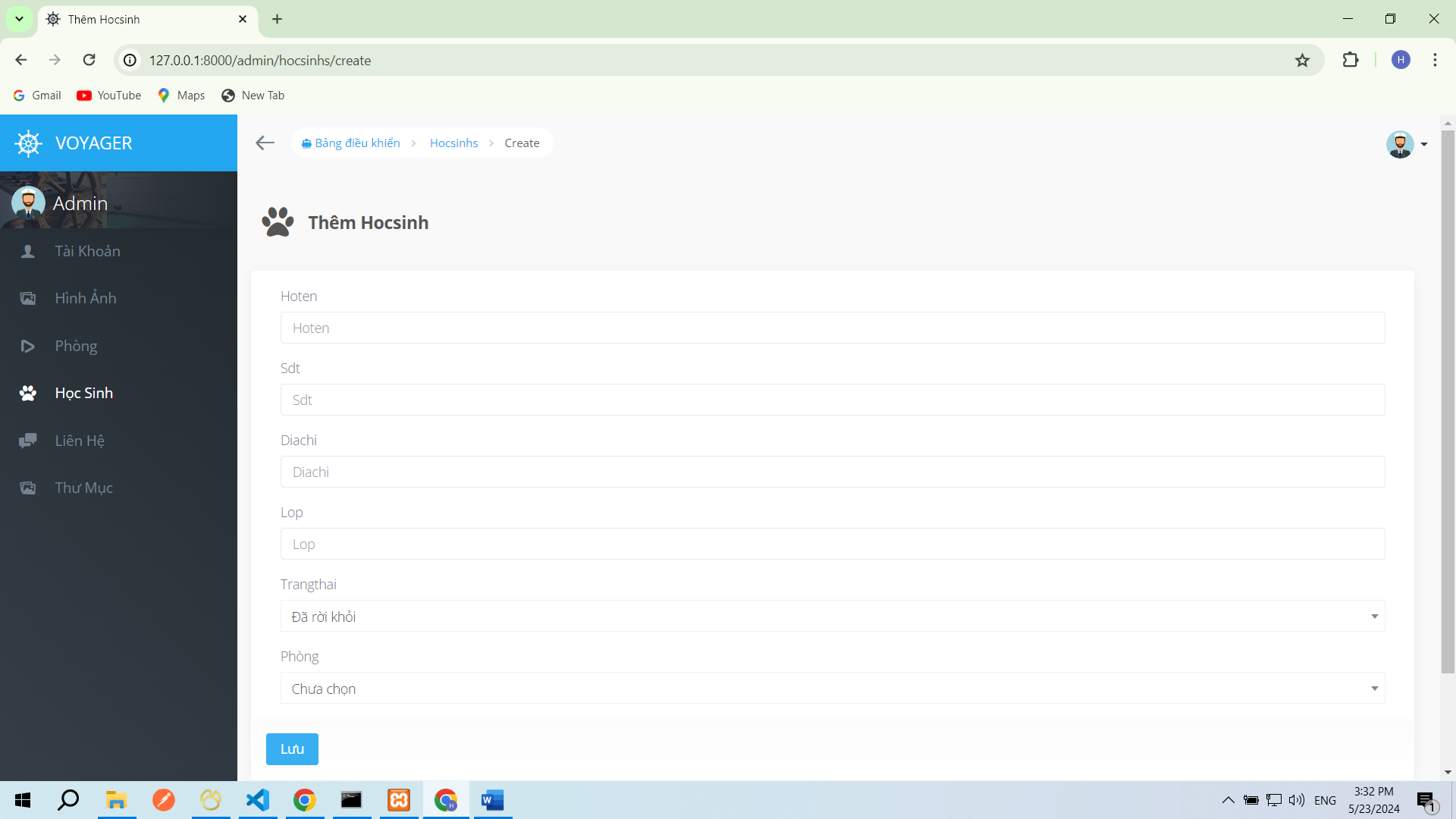


hình 3.3: xem chi tiết phòng

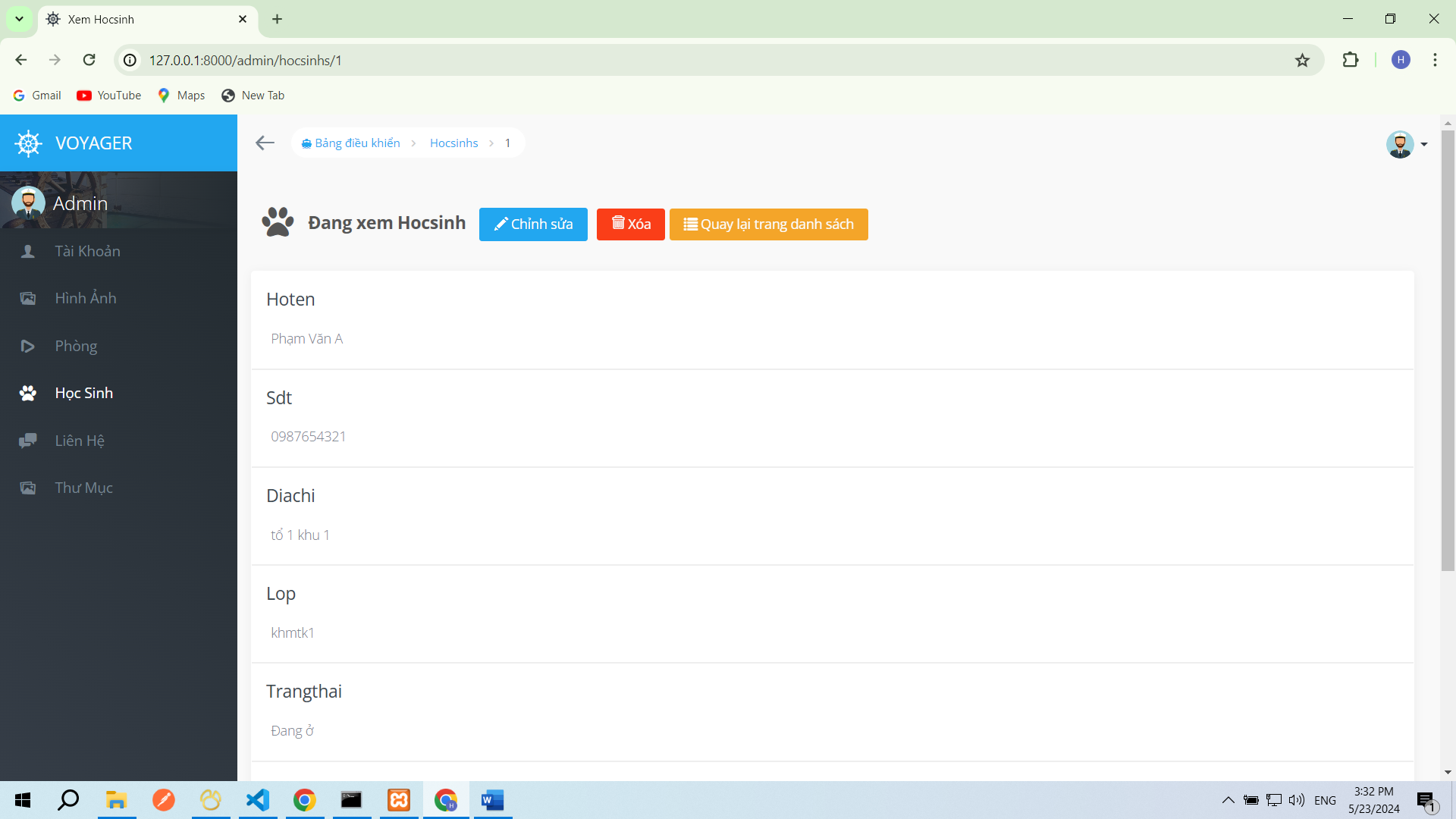
**Trang học sinh** : có các chức năng thêm, sửa, xóa học sinh



hình 4.1: xem danh sách học sinh

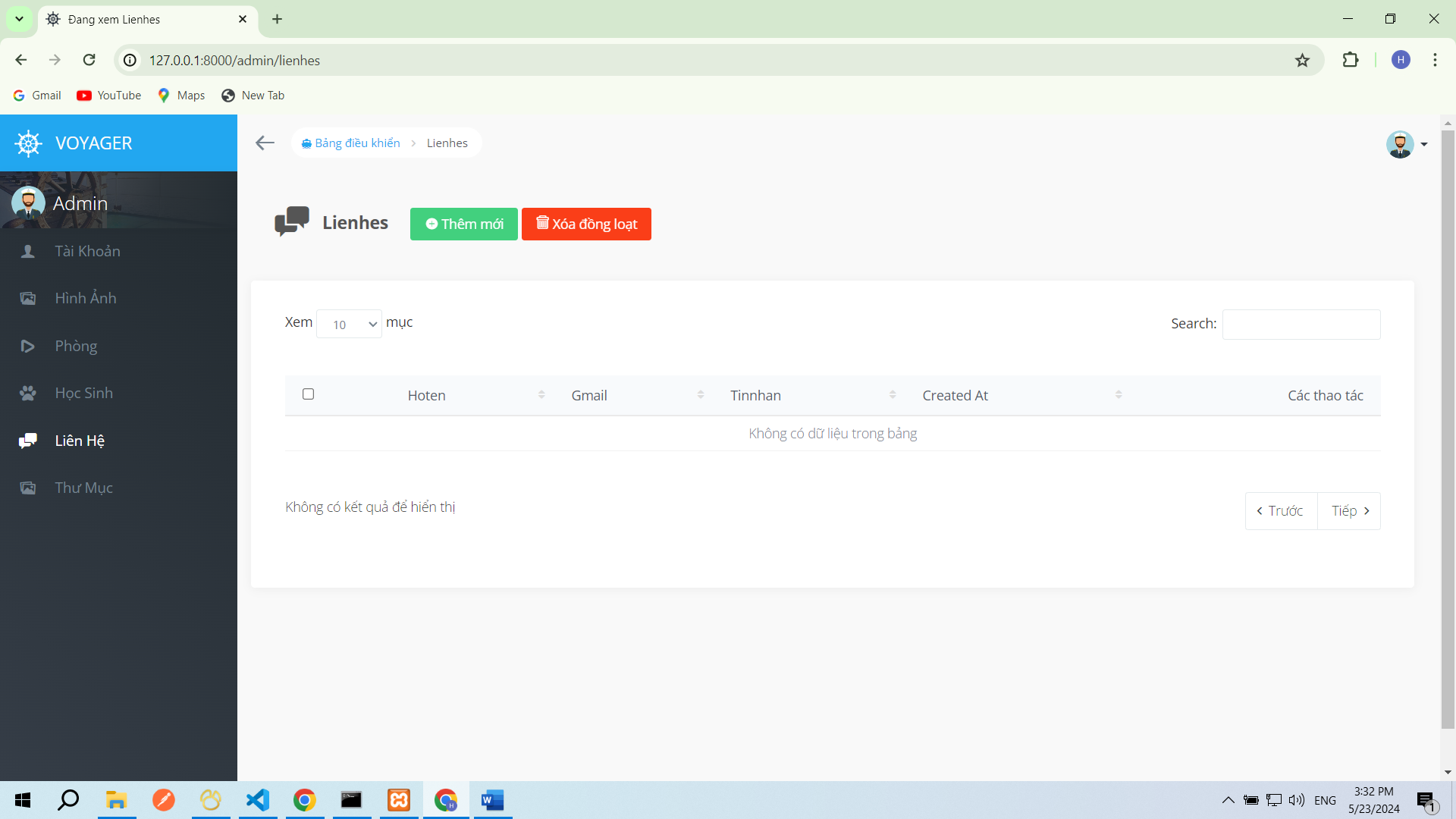


hình 4.2: thêm mới học sinh



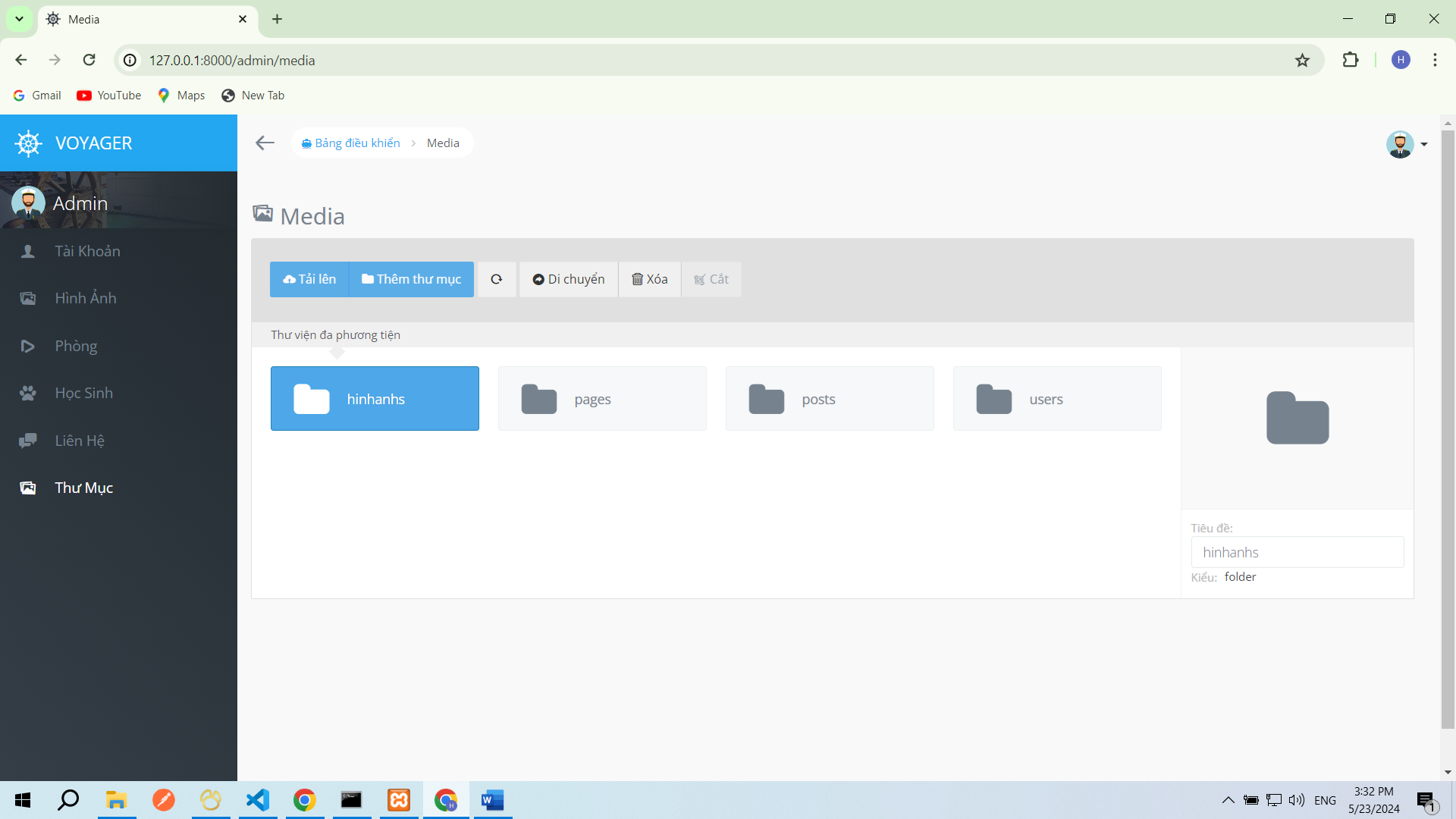
hình 4.3: xem chi tiết học sinh

**Trang liên hệ**



hình 5: trang liên hệ

**Trang quản lý thư mục của web**

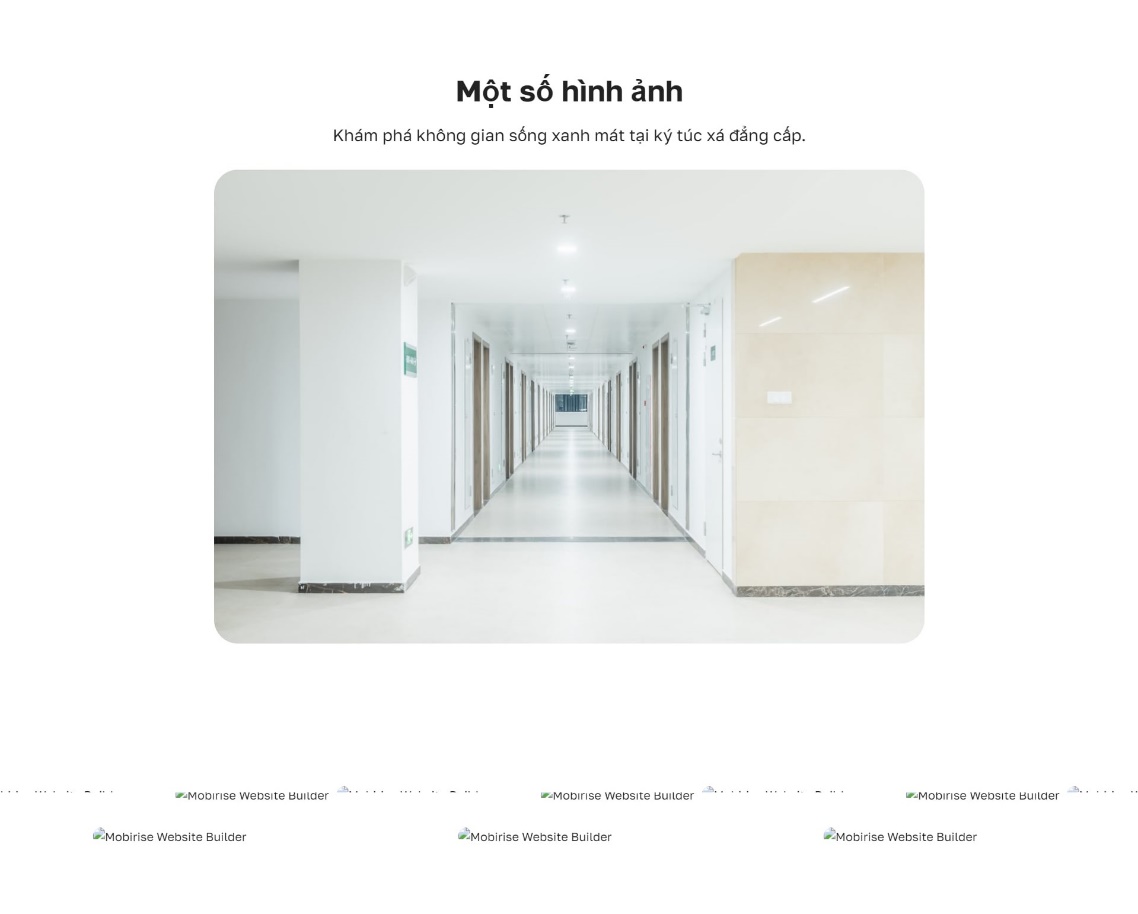


hình 6: trang quản lý thư mục

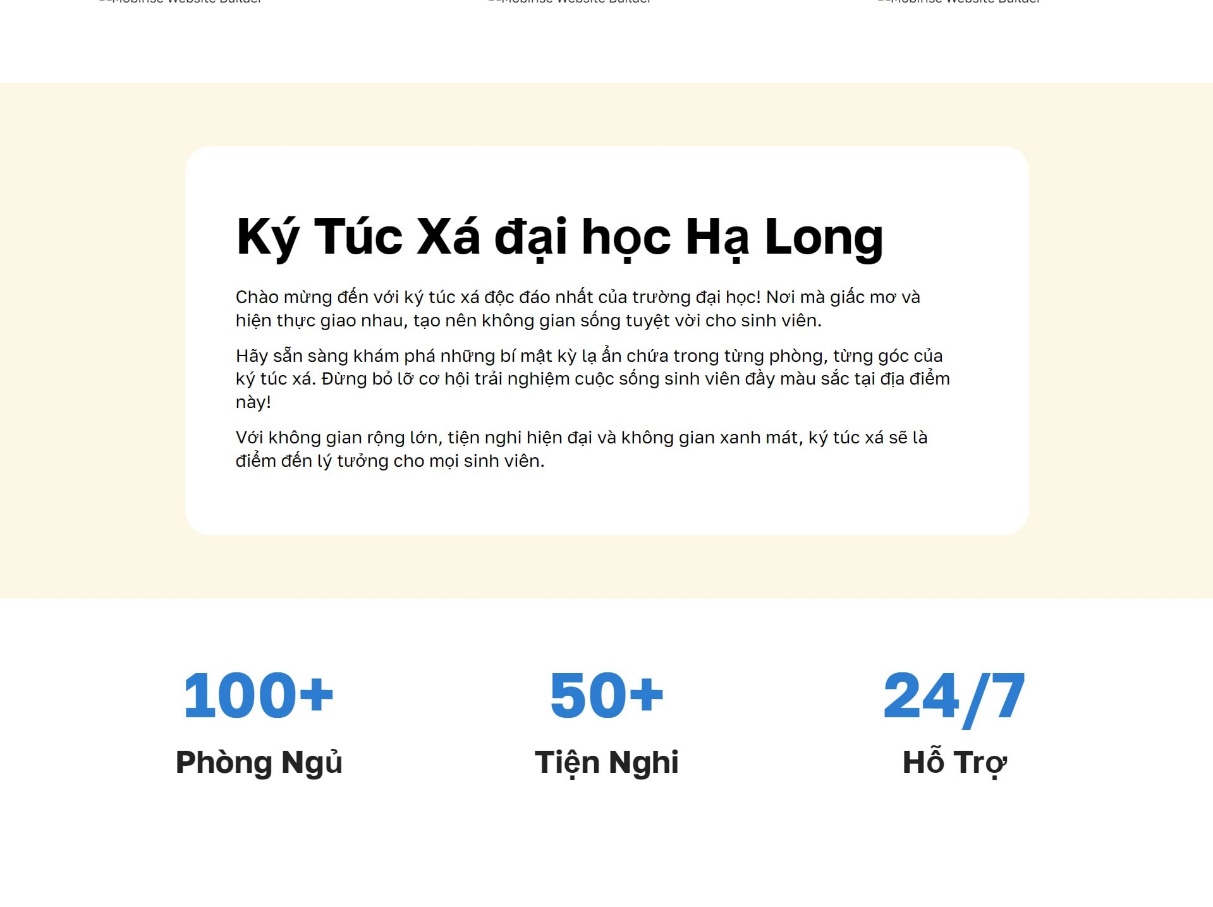
3.3. Giao diện các trang cho người dùng



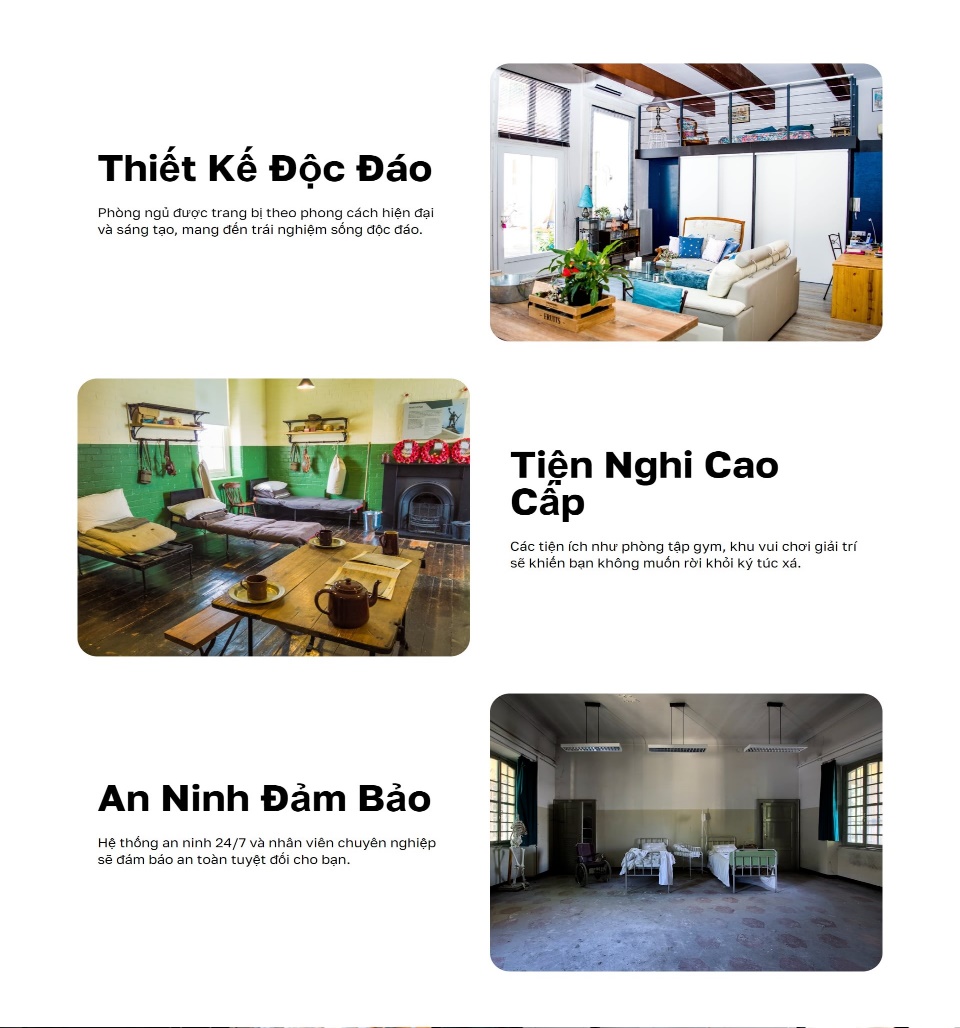
hình 1



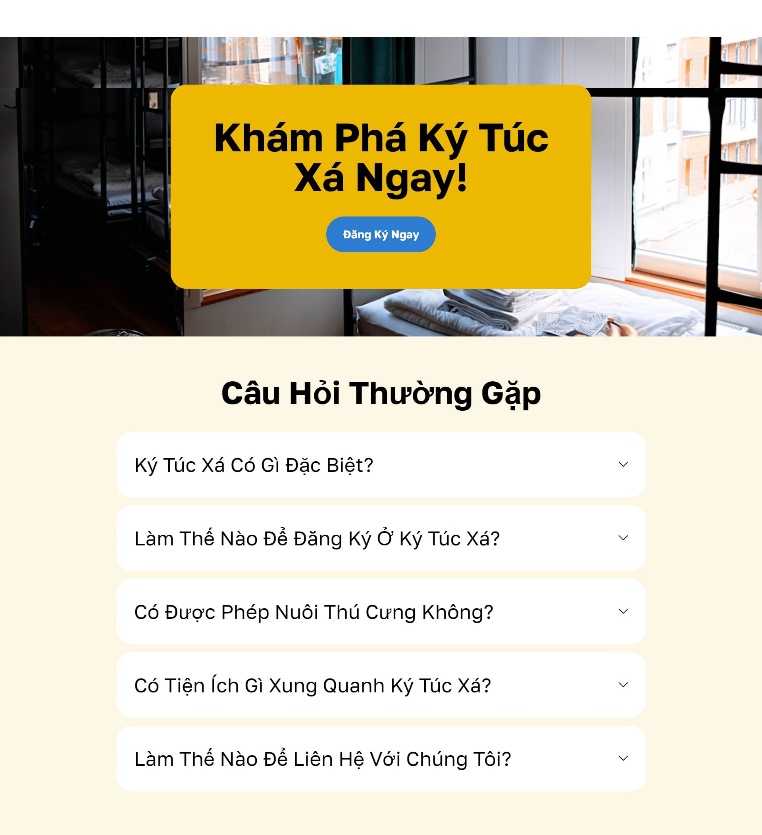
hình 2



hình 3



hình 4



hình 5



hình 6

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Em đã dành rất nhiều công sức để phát triển đề tài "Xây dựng hệ thống phần mềm mã nguồn mở quản lý kí túc xá của sinh viên tại Trường Đại học Hạ Long". Các chức năng đã được triển khai và đang tiếp tục được hoàn thiện để tạo ra một trang web ngày càng tiện lợi và thân thiện hơn với người dùng. Hệ thống quản trị cũng đã được cải thiện, giúp cho việc điều hành các hoạt động trên trang web trở nên dễ dàng hơn đối với quản trị viên.

2. Hướng phát triển

Hướng phát triển của dự án sẽ tập trung vào việc cập nhật và bổ sung các tính năng còn thiếu sót, cải thiện hiệu suất của trang web, tối ưu hóa các chức năng quản lý, và tối ưu hóa thuật toán để nâng cao trải nghiệm người dùng và hiệu quả của hệ thống. Em cam kết đặt mục tiêu nâng cao chất lượng sản phẩm và mang lại trải nghiệm tốt nhất cho người dùng.

Để đạt được điều này, tôi rất mong muốn nhận được sự hỗ trợ, đề xuất và phản hồi từ các thầy cô để có thể hoàn thiện sản phẩm một cách toàn diện và chuyên nghiệp hơn. Sự đóng góp của quý thầy cô là một phần không thể thiếu trong quá trình phát triển dự án, và tôi rất trân trọng mọi ý kiến đóng góp từ phía quý thầy cô.

Mặc dù đã nỗ lực hoàn thành các yêu cầu trong quá trình nghiên cứu và thiết kế, nhưng do kiến thức và kinh nghiệm còn hạn chế, dự án vẫn còn một số thiếu sót. Em nhận thức rằng sự hỗ trợ và chỉ bảo của thầy Nguyễn Văn Thẩm và các bạn đã đóng vai trò quan trọng trong việc giúp tôi vượt qua những thách thức này. Vì vậy, tôi xin bày tỏ lòng biết ơn sâu sắc và lòng kính trọng đến thầy Nguyễn Văn Thẩm và tất cả các bạn đã đồng hành cùng tôi trong dự án này.

TÀI LIỆU THAM KHẢO

[1]. Modern PHP: Building a PHP Application bởi Josh Lockhart

[2]. PHP: The Right Way bởi Josh Lockhart

[3]. Laravel: Code Happy bởi Dayle Rees