# MỤC LỤC

[MỤC LỤC 1](#_bookmark0)

[LỜI NÓI ĐẦU 3](#_bookmark1)

[DANH MỤC HÌNH ẢNH 4](#_bookmark2)

[CHƯƠNG I. PHÁT TRIỂN WEBISTE DỰA TRÊN LARAVEL PHP](#_bookmark3) [FRAMEWORK 5](#_bookmark3)

* 1. [Giới thiệu về Laravel PHP Framework 5](#_bookmark4)
  2. [Ưu điểm của Laravel PHP Framework đối với phát triển website 5](#_bookmark5)
  3. [Cấu trúc thư mục và luồng xử lý trong Laravel 6](#_bookmark6)
  4. [Thành phần cơ bản trong Laravel 6](#_bookmark7)
     1. [Route trong Laravel 6](#_bookmark8)
     2. [Laravel Model 9](#_bookmark9)
     3. [Laravel Controller 10](#_bookmark11)
     4. [Laravel View (Master Template) 12](#_bookmark12)

[CHƯƠNG II. THIẾT KẾ HỆ THỐNG WEBSITE TIN TỨC 17](#_bookmark15)

* 1. [Khảo sát yêu cầu của một website tin tức 17](#_bookmark16)
  2. [Phân tích yêu cầu và thiết kế hệ thống chức năng website 18](#_bookmark17)
     1. [Các đối tượng sử dụng website 18](#_bookmark18)
     2. [Các trường hợp sử dụng (Use Case) 18](#_bookmark19)
  3. [Thiết kế cơ sở dữ liệu 20](#_bookmark23)

[CHƯƠNG III – TRIỂN KHAI XÂY DỰNG WEBSITE TIN TỨC MỨC](#_bookmark25)

[NGƯỜI DÙNG 24](#_bookmark25)

* 1. [Triển khai lõi và cấu hình hệ thống sử dụng Laravel Framework 24](#_bookmark26)
     1. [Cấu hình chung cho hệ thống ứng dụng 24](#_bookmark27)
     2. [Cấu hình cơ sở dữ liệu 25](#_bookmark28)
  2. [Xây dựng trang chủ 27](#_bookmark29)
  3. [Xây dựng trang phân loại theo chủ đề bài viết 29](#_bookmark30)
  4. [Xây dựng trang chi tiết bài đăng tin tức 31](#_bookmark31)
  5. [Xây dựng chức năng tìm kiếm bài viết 32](#_bookmark32)

# [CHƯƠNG 4 – TRIỂN KHAI XÂY DỰNG QUẢN TRỊ CHO WEBSITE TIN](#_bookmark33) [TỨC 33](#_bookmark33)

# [Xây route quản trị và layout dùng chung 33](#_bookmark34)

# [Tạo nhóm route cho admin 33](#_bookmark35)

# [Tạo layout dùng chung cho trang quản trị và tổ chức thư mục 33](#_bookmark36)

# [Xây dựng đăng nhập và đăng xuất 34](#_bookmark38)

# [Xây dựng trang chủ quản trị 38](#_bookmark40)

# [Xây dựng trang cá nhân 39](#_bookmark41)

# [Xây dựng trang quản lý bài viết 40](#_bookmark43)

# [Trang hiển thị danh sách bài viết 41](#_bookmark44)

# [Trang thêm bài viết 43](#_bookmark46)

# [Trang sửa bài viết 44](#_bookmark48)

# [Trang xóa bài viết 45](#_bookmark50)

# [Trang quản lý chuyên mục 45](#_bookmark51)

# [Trang thêm chuyên mục 46](#_bookmark52)

# [Danh sách, Xóa chuyên mục 47](#_bookmark54)

# [Quản lý thẻ và người viết bài (author) 47](#_bookmark56)

# [Lấy danh sách hiển thị 48](#_bookmark57)

# [Thêm các bản ghi 49](#_bookmark60)

# [Sửa các bản ghi 50](#_bookmark63)

# [Xóa các bản ghi 51](#_bookmark66)

# [Quản lý tệp tin 52](#_bookmark68)

# [TÀI LIỆU THAM KHẢO 54](#_bookmark70)

# [PHỤ LỤC 55](#_bookmark71)

# LỜI NÓI ĐẦU

Công nghệ thông tin là ngành đang phát triển mạnh và ngày càng được ứng dụng rộng rãi trên nhiều lĩnh vực. Cùng với xu hướng phát triển của các phương tiện truyền thông như báo, radio … thì việc sử dụng internet ngày càng phổ biến. Internet mang lại một kho thông tin khổng lồ phục vụ mọi nhu cầu và mục đích chỉ bằng một cú nhấp chuột.

Đi kèm theo đó là nhu cầu cập nhật thông tin giải tri,tin tức xã hội,đời sống… Và để đáp ứng nhu cầu đó thì việc website tin tức ra đời là nhu cầu tất yếu.

Để xây dựng một website tin tức có thể dùng các ngôn ngữ lập trình khác nhau như Java, PHP, ASP.Net …

Nhưng phù hợp và dễ tiếp cận nhất là PHP. Framework có mặt ở mọi ngôn ngữ lập trình và nó được coi như là một tiêu chuẩn mà khi sử dụng cần tuân thủ quy định của nó tất cả những qui định đó nhằm giúp tạo ra một bản code sạch. Nhờ tuân thủ qui định, mà khi làm việc nhóm, người này có thể đọc hiểu code của người kia, có thể sử dụng lại hàm được viết bởi một người khác. Framework khá nổi bật trong ngôn ngữ PHP là Laravel Framework đứng top 1 một trong những framework phổ biến nhất, tiếp sau là Phalcon, Symfony2, CodeIgniter và các framework khác. Tháng 8 năm 2014, Laravel Framework được xem như là một dự án PHP phổ biến nhất trên Github.

# DANH MỤC HÌNH ẢNH

[Hình 1. 1 - Mô hình tương tác Model và cơ sở dữ liệu 9](#_bookmark10)

[Hình 1. 2 - Liên kết Route, Controller và View trong Laravel 12](#_bookmark13)

[Hình 1. 3 - Hoạt động của Master Template trong Laravel 14](#_bookmark14)

[Hình 2. 1 - Sơ đồ Use Case của người dùng thông thường 18](#_bookmark20)

[Hình 2. 2 - Sơ đồ Use Case của quản trị viên đăng bài (Author) 19](#_bookmark21)

[Hình 2. 3 - Sơ đồ Use Case của quản trị viên hệ thống 20](#_bookmark22)

[Hình 2. 4 - Sơ đồ quan hệ của cơ sở dữ liệu 22](#_bookmark24)

[Hình 4. 1 - Giao diện trang chủ quản trị 34](#_bookmark37)

[Hình 4. 2 - Giao diện trang đăng nhập cho quản trị 34](#_bookmark39)

[Hình 4. 3 - Giao diện thông tin tài khoản 40](#_bookmark42)

[Hình 4. 4 - Giao diện danh sách bài viết 42](#_bookmark45)

[Hình 4. 5 - Giao diện trang đăng bài viết 43](#_bookmark47)

[Hình 4. 6 - Giao diện trang sửa bài viết 44](#_bookmark49)

[Hình 4. 7 - Giao diện thêm chuyên mục bài viết 46](#_bookmark53)

[Hình 4. 8 - Giao diện danh sách chuyên mục 47](#_bookmark55)

[Hình 4. 9 - Giao diện danh sách thẻ 49](#_bookmark58)

[Hình 4. 10 - Giao diện sach sách quản trị viên đăng bài 49](#_bookmark59)

[Hình 4. 11 - Giao diện thêm thẻ 50](#_bookmark61)

[Hình 4. 12 - Giao diện thêm quản trị viên đăng bài 50](#_bookmark62)

[Hình 4. 13 - Giao diện sửa thẻ 51](#_bookmark64)

[Hình 4. 14 - Giao diện xoá tài khoản quản trị viên đăng bài 51](#_bookmark65)

[Hình 4. 15 - Giao diện xoá thẻ 52](#_bookmark67)

[Hình 4. 16 - Giao diện quản lý tệp tin 52](#_bookmark69)

**CHƯƠNG I. PHÁT TRIỂN WEBISTE DỰA TRÊN LARAVEL PHP**

**FRAMEWORK**

# Giới thiệu về Laravel PHP Framework

Laravel là một PHP framework mã nguồn mở và miễn phí, được phát triển bởi Taylor Otwell và nhắm vào mục tiêu hỗ trợ phát triển các ứng dụng web theo kiếm trúc model-view-controller (MVC). Những tính năng nổi bật của Laravel bao gồm cú pháp dễ hiểu – rõ ràng , một hệ thống đóng gói modular và quản lý gói phụ thuộc, nhiều cách khác nhau để truy cập vào các cơ sở dữ liệu quan hệ, nhiều tiện ích khác nhau hỗ trợ việc triển khai vào bảo trì ứng dụng.

Vào khoảng Tháng 3 năm 2015, các lập trình viên đã có một cuộc bình chọn PHP framework phổ biến nhất, Laravel đã giành vị trí quán quân cho PHP framework phổ biến nhất năm 2015, theo sau lần lượt là Symfony2, Nette, CodeIgniter, Yii2 và một số khác. Trước đó, Tháng 8 2014, Laravel đã trở thành dự án PHP phổ biến nhất và được theo dõi nhiều nhất trên Github.

Laravel được phát hành theo giấy phép MIT, với source code được lưu trữ tại [Github](https://github.com/laravel/laravel).

# Ưu điểm của Laravel PHP Framework đối với phát triển website

* + - Tổ chức tập tin và cấu trúc mã lệnh rõ ràng.
    - Hệ thống thư viện và công cụ phong phú.
    - Xác nhận biểu mẫu (Form Validation).
    - Mã lệnh ít hơn, nhưng làm được nhiều việc hơn.
    - Mô hình MVC rõ ràng, dễ bảo trì.
    - Thích hợp làm việc nhóm và phân chia công việc.
    - Chức năng hỗ trợ bảo mật, xác thực dữ liệu, SQL Injection,…
    - Hệ thống giao diện phong phú, thư viện form helper giúp xây dựng biểu mẫu nhanh chóng và bảo mật.
    - Hỗ trợ đường dẫn thân thiện (Friendly Url).
    - Bảo mật session, cookie,…

# Cấu trúc thư mục và luồng xử lý trong Laravel

* Thư mục chính:
  + App: Chứa mã cốt lõi của ứng dụng, ta dùng nhiều tới thư mục này.
  + Bootstrap: Chứa các tệp tin khởi động và cấu hình tự động chạy.
  + Config: Chứa tất cả các tệp cấu hình của hệ thống.
  + Database: Nơi chúng ta cấu hình các bộ dữ liệu mẫu: migrate, seed.
  + Public: Nơi lưu trữ thư viện CSS, Javascript, các hình ảnh. Chứa tệp index.php
  + Resources: Chứa views, các tệp ngôn ngữ.
  + Routes: Chứa tất cả các route của hệ thống.
* Luồng xử lý:

Đầu tiên người dùng yêu cầu 1 đường dẫn sang phía route, route trả về controller, controller sẽ xử lý các yêu cầu đó, nếu yêu cần dữ liệu thì sẽ phải thông qua Model để truy xuất đến cơ sở dữ liệu, sau khi truy xuất dữ liệu model trả về cho controller, controller đưa dữ liệu về cho view, view xuất dữ liệu ra màn hình cho người dùng xem.

# Thành phần cơ bản trong Laravel

# Route trong Laravel

Mục đích chính của router là định tuyến đến những controller cụ thể nào đó từ phía yêu cầu của người sử dụng. Đối với những Framework khác thì khi muốn thực hiện một công việc nào đó thường sẽ cần xây dựng một controller để xử lý, nhưng trong Laravel thì hoàn toàn có thể xử lý dữ liệu ngay trong phần Route. Đây chính là điểm mạnh cũng như điểm khác biệt lớn của Laravel đối với các Framework khác.

* Cấu trúc Route:
  + Các định tuyến trong Laravel 5.5 đều được viết trong routes/web.php
  + Cú pháp: Route::method(‘URI’, ‘function call back’);

Trong đó:

* + - URI là dạng link trên url.
    - Function call back: hàm sẽ gọi tới link URI phía trên.
    - Method: các dạng phương thức cơ bản post, get, put, delete, any Các method:
    - POST Route: các thao tác lấy từ biểu mẫu (form) như thêm dữ liệu.
    - GET Route: dành cho các thao tác truy cập thông thường tương đương với yêu cầu cơ bản trong PHP. Ví dụ chạy 1 đường dẫn trên URL.
    - PUT Route: dành cho các thao tác lấy từ biểu mẫu nhưng là cập nhật dữ liệu.
    - DELETE Route: dành cho thao tác xóa dữ liệu.
    - ANY Route: là sự tổng hợp các thao tác ở trên. Trong đó POST và GET được sử dụng nhiều nhất.
* Truyền tham số trên Route:

Route::get(‘myroute/{ten}’,function($ten){ Return “Chào bạn”.$ten;

});

Trong đó, *{ten}* là giá trị do người dùng nhập trên liên kết của URL hoặc do nhà phát triển định sẵn.

Đặt điều kiện cho tham số với phương thức where():

Route::get(‘myroute/{ten}’,function($ten){ Return “Chào bạn”.$ten;

})->where([‘ ten ’ => ‘[a-zA-Z]+’]);

* Định danh cho Route:
  + Cách 1: Khai báo ‘as’=> ‘Tên Route’ trong tham số như sau:

Route::get(‘myroute’,[ ‘as’=> ‘newname’, function(){ Return “Đã đổi tên”;}

]);

* + Cách 2: Cách này khá ngắn gọn và dễ dùng: thêm *name(‘tên route’)* ở cuối.

Route::get('myroute',function() { return “Đã đổi tên”;

} )->name(‘tên route’);

* Muốn gọi route bằng tên đã đặt, sử dụng *route(‘tên route’):*

Route::get('myroute',function(){

return redirect()->route(‘tên route’);

} );

* Nhóm Route:

Route::group([ ‘prefix’ => ‘MyGroup’ ] , function(){

//Gọi Route User1: domain/MyGroup/User1 Route::get(‘User1’, function(){ return ‘User1’ });

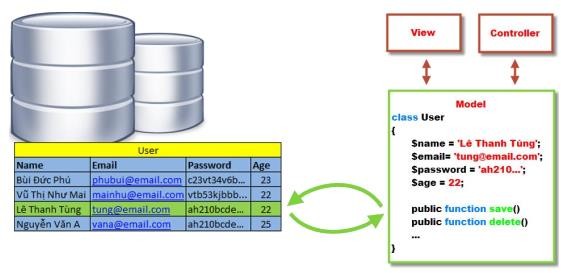
//Gọi Route User2: domain/MyGroup/User2 Route::get(‘User2’, function(){ return ‘User2’ });

//Gọi Route User3: domain/MyGroup/User

Route::get(‘User3’, function(){ return ‘User3’ });

});

# Laravel Model



*Hình 1. 1 - Mô hình tương tác Model và cơ sở dữ liệu*

Model là một lớp dữ liệu có cấu trúc giống với bảng trong cơ sở dữ liệu, dùng để xử lý dữ liệu vào ra trong bảng.

* Tạo model
  + Các Model sẽ được lưu tại mục App/ Thực hiện:
  + Tạo một Model: Khởi động Command Prompt (cmd) và trỏ về thư mục gốc của dự án, gõ lệnh sau để tạo model:

php artisan make:model TenModel

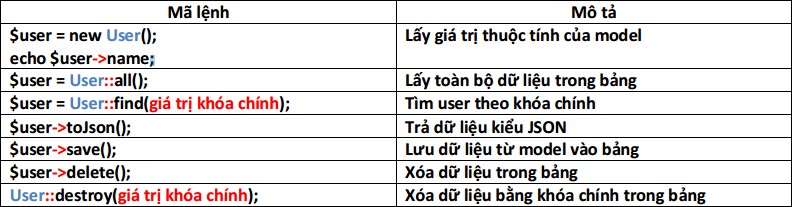
* + Tạo một Model và migrate tương ứng với nó

php artisan make:model TenModel –m

* + Kết nối Model tới bảng trong cơ sở dữ liệu

Protected $table = ‘Tên Bảng’;

* + Một số phương thức hay sử dụng trong Model:



# Laravel Controller

* Cấu trúc Controller
  + Các Controller sẽ được lưu tại app/Http/Controllers, nội dung của một Controller sẽ có dạng:

namespace App\Http\Controllers; use Illuminate\Http\Request;

class MyController extends Controller

{

//Thực hiện các công việc

}

* Tạo Controller
  + Tạo Controller với cmd bằng câu lệnh:

*php artisan make:controller MyController*

* Gọi Controller từ Route
  + Để gọi một Controller, phải thông qua Route bằng cách khai báo như sau:

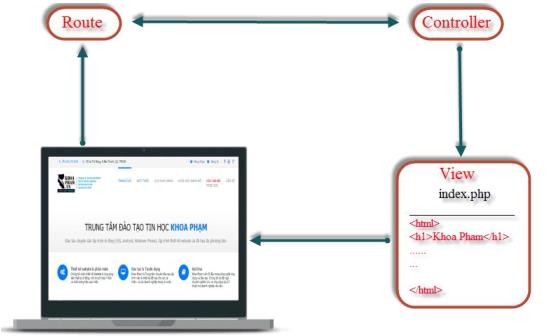
|  |  |
| --- | --- |
| Gọi | Nhận dữ liệu bên Controller |
| Route::get(‘goiController/{id}’, ‘MyController@GetData’); | class MyController extends Controller {  public function GetData($id)  {  Echo “Xin chào!”;  }  } |

* Gửi nhận dữ liệu từ Route sang Controller

|  |  |
| --- | --- |
| Truyền tham số id từ Route | Nhận dữ liệu bên Controller |
| Route::get(‘goiController/{id}’, ‘MyController@GetData’); | class MyController extends Controller  {  public function GetData($id)  {  Echo “Đã nhận  ”.$id;  }  } |

# Laravel View (Master Template)

* Laravel View



*Hình 1. 2 - Liên kết Route, Controller và View trong Laravel*

View là các tệp tin có đuôi .php, chứa mã nguồn html, hiển thị dữ liệu cho người dùng xem và được lưu tại resources/views.

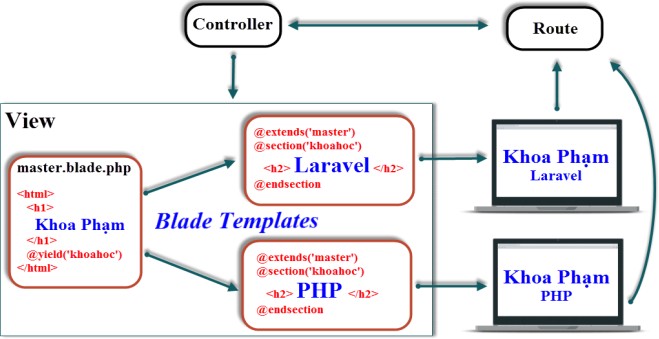
Giả sử truyền lên trình duyệt một đường dẫn, trình duyệt sẽ gửi lên router, Route chuyển sang Controller, sau khi Controller xử lý xong, sẽ gọi tệp tin index.php, đây chính là view. Tập tin này chứa các mã nguồn html, css, hoặc các dữ liệu mà nhà phát triển ứng dụng muốn hiển thị lên màn hình rồi trả về trình duyệt hiển thị cho người dùng xem.

|  |  |  |
| --- | --- | --- |
| Cài đặt Route | Controller | myView.php |
| Route::get('myView', 'MyController@myView'); | Public function myView()  {  Return view(‘myView’);  } | <h1>Đây là view </h1> |

* Truyền tham số sang view

|  |  |  |
| --- | --- | --- |
| Cài đặt Route | Controller | myView.php |
| Route::get('myView/{ts}', 'MyController@myView'); | Public function myView($ts)  {  Return view(‘myView’, [‘thamso’=>&ts]);  } | <h1>Đây là  <?php echo  $thamso;  ?></h1> |

* Master template



*Hình 1. 3 - Hoạt động của Master Template trong Laravel*

Luồng xử lý dữ liệu tương tự như ở View ở trên. Trong View có tệp master.blade.php, tệp này chứa giao diện chung của hệ thống, @section định nghĩa phần nội dung còn @yield sử dụng để hiển thị nội dung mà section đem lại. Các view sẽ kế thừa blade template này bằng cách sử dụng lệnh @extends*.*

Muốn sử dụng Blade Template thì tên tệp phải có chứa .blade đằng trước .php

# Sử dụng các thư viện dành cho thiết kế giao diện website với Laravel

* + 1. **Sử dụng Boostrap CSS Framework**

Bootstrap là 1 framework HTML, CSS, và JavaScript cho phép người dùng dễ dàng thiết kế website theo một chuẩn nhất định, bao gồm typography (kiểu dáng văn bản), forms (biểu mẫu), buttons (nút bấm), tables (bảng biểu), navigation (menu), modals (các phương thức), image carousels (trình chiếu ảnh) và nhiều thứ khác. Trong bootstrap có thêm các trình cắm (plugin) Javascript trong nó, giúp tạo các website thân thiện (web responsive) với các thiết bị cầm tay như mobile, ipad, tablet,... dễ dàng hơn và nhanh chóng hơn.

Trong các phiên bản Laravel 5.x, bootstrap đã được thiết lập sẵn cấu hình, chỉ cần tải về các gói (package) là sử dụng được.

Mỗi trang web, khi muốn sử dụng có 2 cách là tải Bootstrap từ getbootstrap.com hoặc thêm từ CDN. Ví dụ trường hợp sử dụng CDN:

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/boots trap.min.css" integrity="sha384- BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4 u" crossorigin="anonymous">

# Sử dụng Jquery Javascript Framework

Jquery là một thư viện javascript, nó giúp viết javascript nhanh hơn với cú pháp đơn giản hơn. Thay vì phải viết những dòng lệnh dài và lặp lại nhiều lần trong javascript thì jquery đã đóng gói chúng thành những phương thức sẵn có để dễ dàng sử dụng.

Để sử dụng cũng có 2 cách để nhúng thư viện vào trang web như sau:

* Tải thư viện tại <https://jquery.com/download/>sử dụng nó như sau:

Ví dụ ở đây đang sử dụng phiên bản jquery 2.0.2. Khi tải về tìm đến tệp *jquery-2.0.2.min.js* rồi nhúng đường dẫn của file đó vào website là có thể dùng được

<script src="'jquery-2.0.2.min.js'"></script>

* Sử dụng cdn tại <https://code.jquery.com/>chọn phiên bản cần dùng và coppy đoạn nhúng script mà trang web cung cấp. Ví dụ:

<script

src="https://code.jquery.com/jquery-3.2.1.js" integrity="sha256- DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE=" crossorigin="anonymous"></script>

# Sử dụng kỹ thuật Jquery Ajax

AJAX - "Asynchronous JavaScript and XML" - là một bộ công cụ cho phép tải dữ liệu từ máy chủ mà không yêu cầu tải lại trang. Nó sử dụng chức năng sẵn có XMLHttpRequest(XHR) của trình duyệt để thực hiện một yêu cầu đến máy chủ và xử lý dữ liệu trả về.

jQuery cung cấp method *$.ajax* và một số phương thức tiện lợi giúp làm việc với XHRs thông qua trình duyệt một cách dễ dàng hơn.

Hàm Jquery ajax tổng quát :

- *Jquery.ajax( options )*: Tải một trang từ xa sử dụng một HTTP request. Các thông số chính trong *options* như sau:

+ *url****:*** Một chuỗi chứa URL để yêu cầu được gửi tới

+ *data***:** tham số tùy ý này biểu diễn các cặp khoá/giá trị (key/value) mà sẽ được gửi tới máy chủ.

+ type: Là phương thức cần gửi đi (get,post,put … )

+ data*Type***:** tham số tùy ý này biểu diễn kiểu dữ liệu để được trả về tới hàm callback: "xml", "html", "script", "json", "jsonp", hoặc "text".

+ *callback***:** tham số tùy ý này biểu diễn một hàm để được thực thi bất cứ khi nào dữ liệu được tải thành công.

Ví dụ:

$.ajax({

url: 'Demo.php',

type: ‘post’, dataType: 'json', data : {‘value’:var}

})

.done(successFunction)

.fail(errorFunction)

.always(alwaysFunction);

# CHƯƠNG II. THIẾT KẾ HỆ THỐNG WEBSITE TIN TỨC

# Khảo sát yêu cầu của một website tin tức

Bước khảo sát yêu cầu sẽ làm rõ được một số vấn đề sau:

* Mục đích xây dựng ứng dụng: Mục đích sử dụng của dự án này là trình bày bài viết trên website dưới dạng tin tức, cho phép người dùng truy cập và tra cứu tin tức trên trình duyệt web.
* Đối tượng sử dụng: Người dùng bình thường (độc giả), quản trị viên website (gồm quản trị viên đăng bài và quản trị viên toàn bộ hệ thống).
* Yêu cầu về chức năng: Đối với một website tin tức, yêu cầu tối thiểu sẽ phải đạt được là:
  + Hiển thị nội dung theo từng bài viết.
  + Cho phép truy cập và đọc bài viết.
  + Cho phép liệt kê danh sách bài viết theo từng chuyên mục.
  + Cho phép tìm kiếm bài viết.
  + Cho phép bình luận ở cuối bài viết.
  + Các chức năng cao hơn cho tác giả như đăng bài, sửa bài, xoá bài, tải lên các dữ liệu đa phương tiện như ảnh, phim, …
  + Các chức năng cho quản trị viên hệ thống như thêm, sửa, xoá tài khoản, thêm chuyên mục, thêm tệp tin, bài viết nổi bật
* Yêu cầu về thiết kế:
  + Trình bày nội dung bài viết, có tiêu đề, ngày tháng đăng bài.
  + Trang chủ liệt kê danh sách cách bài viết phân chia theo từng danh mục, các bài viết nổi bật.
  + Phần đầu trang web có logo của trang web, phần menu hiển thị các danh mục bài viết cho người dùng lựa chọn.
  + Phần cuối trang web có thông tin chung về trang web, đơn vị vận hành, quản lý, và một số các thông tin khác.

Sau khi đã làm rõ được các yêu cầu đối với ứng dụng, bước tiếp theo sẽ là thiết kế hệ thống theo những yêu cầu đã nêu ở trên.

# Phân tích yêu cầu và thiết kế hệ thống chức năng website

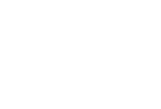
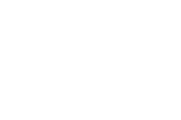
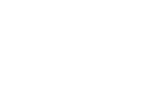
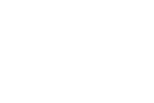
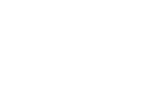
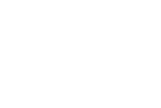
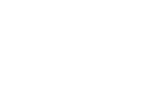
# Các đối tượng sử dụng website

Như đã khảo sát, các đối tượng sử dụng của website bao gồm người dùng (độc giả) và quản trị viên website. Hai đối tượng này, về mặt chung đều là người sử dụng website, nhưng mục đích sử dụng của mỗi đối tượng lại khác nhau. Người dùng bình thường chỉ có quyền đọc bài viết, có thể bình luận về bài viết, quản trị viên sẽ có tất cả các quyền của người dùng bình thường, ngoài ra sẽ có quyền cao hơn là quản lý bài viết (đăng bài, sửa bài, xoá bài viết, tải lên các dữ liệu đa phương tiện).

Vì vậy, việc xác định định danh của người dùng đối với người sử dụng bình thường là không cần thiết, mặt khác, đối với quản trị viên là bắt buộc.

# Các trường hợp sử dụng (Use Case)

Các trường hợp sử dụng của người dùng sẽ được biểu thị thông qua các sơ đồ use case.



**Tìm kiếm bài viết**

**Bình luận**

«extend» **bài viết**

**Ðọc bài viết**

**Người dùng**

«extend»

«extend» **Lọc theo thẻ**

**Truy cập danh mục**

**Lọc theo tên tác giả**

**Truy cập trang web**

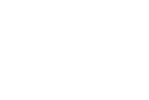
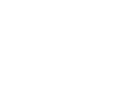
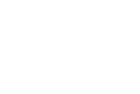
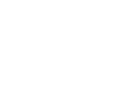
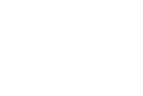
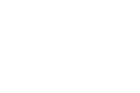
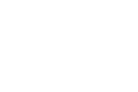
Website tin tức

**uc User Uses Case**

*Hình 2. 1 - Sơ đồ Use Case của người dùng thông thường*

Hình 2.1 biểu thị các trường hợp sử dụng của người dùng thông thường, bao gồm truy cập trang web, truy cập các danh mục bài viết, truy cập và đọc bài viết, tìm

kiếm bài viết theo tiêu đề. Khi người dùng truy cập trang web, hệ thống sẽ đưa người dùng tới trang chủ của website tin tức. Khi người dùng đọc bài viết, người dùng sẽ có thể bình luận bài viết đó.



«extend»

**Đăng xuất**

**Xoá bài viết**

«extend»

**Author**

«extend»

**Đăng nhập**

**Chỉnh sửa bài viết**

«extend» «extend»

«extend»

«extend»

Website tin tức

**Upload**

**Đăng bài**

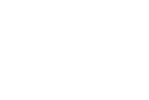
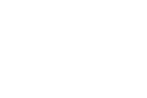
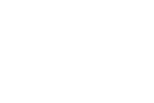
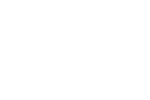
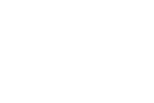
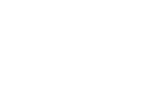
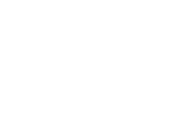
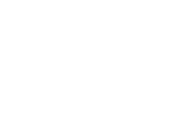
**Chỉnh sửa thông tin**

**uc Author Uses Case**

*Hình 2. 2 - Sơ đồ Use Case của quản trị viên đăng bài (Author)*

Hình 2.2 biểu thị các trường hợp sử dụng của quản trị viên đăng bài. Ngoài các chức năng cơ bản của một người dùng thông thường, quản trị viên đăng bài có thể truy cập vào phần quản trị để sử dụng các chức năng của quản trị viên đăng bài. Để truy cập được vào hệ thống, quản trị viên phải tiến hành đăng nhập. Sau khi đăng nhập thành công, quản trị viên có thể thực hiện được các chức năng như xem, chỉnh sửa thông tin tài khoản, đăng xuất, đăng bài, chỉnh sửa bài viết, xoá bài viết. Đối với các chức năng chỉnh sửa và xoá bài viết, quản trị viên đăng bài sẽ chỉ có thể xem, chỉnh sửa hay xoá bài viết do chính mình đăng lên, các bài viết khác sẽ không có quyền truy cập.

Ngoài người dùng và quản trị viên đăng bài, ứng dụng còn có một đối tượng nữa, đó là quản trị viên hệ thống. Quản trị viên hệ thống là những người có quyền cao nhất của hệ thống, có thể thao tác với toàn bộ dữ liệu của ứng dụng. Như mô tả của sơ đồ dưới đây:



**Quản lý thẻ**

**Đăng xuất**

**Quản lý danh mục**

«extend»

«extend» «extend»

**tin**

«extend»

**Administrator**

«extend»

**Quản lý tập**

**Đăng nhập**

«extend» «extend»

**Quản lý tài khoản**

**Quản lý bài viết**

**Chỉnh sửa thông tin**

Website tin tức

**uc Administrator Uses Case**

*Hình 2. 3 - Sơ đồ Use Case của quản trị viên hệ thống*

Hình 2.3 mô tả các trường hợp sử dụng của quản trị viên hệ thống (Administrator). Cũng giống như quản trị viên đăng bài, quản trị viên hệ thống cũng có các quyền cơ bản của một người dùng bình thường, ngoài ra, quản trị viên hệ thống còn có các quyền cao hơn khi đăng nhập vào phần quản trị của hệ thống. Sau khi đăng nhập thành công, quản trị viên hệ thống sẽ có đầy đủ các chức năng, thao tác toàn quyền với hệ thống của ứng dụng. Bao gồm chỉnh sửa thông tin tài khoản, tất cả các chức năng của quản trị viên đăng bài sẽ nằm trong chức năng quản lý bài viết, quản lý tài khoản của các quản trị viên đăng bài, quản lý các nội dung dữ liệu mà quản trị viên đăng bài tải lên hệ thống, quản lý các danh mục bài viết, quản lý các thẻ (Tags) của các bài viết, và cuối cùng là đăng xuất khỏi hệ thống khi xong việc.

# Thiết kế cơ sở dữ liệu

Khi xây dựng một ứng dụng, việc thiết kế cơ sở dữ liệu đúng chuẩn và đáp ứng được truy cập là rất quan trọng. Một cơ sở dữ liệu tốt là cơ sở dữ liệu đủ tiêu chuẩn đáp ứng được tối thiểu chuẩn 3NF. Trong ứng dụng này, cơ sở dữ liệu sử dụng sẽ là MySQL. Các bác trong cơ sở dữ liệu sẽ bao gồm bảng người dùng (user), bảng bài

viết (post), bảng danh mục (category), bảng thẻ (tags), bảng chứa các thông tin về dữ liệu tải lên (files), và các bảng dùng để làm trung gian kết nối sẽ được tạo sau.

Đầu tiên là dữ liệu dành cho tài khoản (users). Thông tin cần lưu lại của một tài khoản sẽ bao gồm số định danh (ID) của tài khoản, tên tài khoản, địa chỉ email, mật khẩu, một số các thông tin thêm về người sử dụng tài khoản như ảnh đại diện, ngày sinh, … và các thông tin xác thực như token (dạng như chữ ký điện tử được mã hoá), thời gian tạo tài khoản. Dữ liệu này chỉ được sử dụng đối với các quản trị viên, nên có thể đặt tên bảng này là users hoặc admin.

Tiếp theo là dữ liệu dành cho bài viết (posts). Đây là dữ liệu quan trọng nhất của một website tin tức. Cũng giống như với tài khoản, thông tin cần lưu lại trước tiên sẽ là số hiệu định danh (ID) của bài viết dùng làm khoá chính, theo sau đó là các thông tin của bài viết, như tiêu đề bài viết (title), nội dung bài viết (content), mô tả bài viết (description), đường dẫn liên kết (slug), số lượng lượt xem (view). Ngoài ra, để thuận tiện hơn trong việc quản lý và điều chỉnh bài viết, một bài viết cần có thêm các mục như thể loại bài viết (post type), bài viết có là bài nổi bật hay không (hot), bài viết có được phép hiển thị trên trang web hay không (status), người đăng (user id), thông tin này sẽ được kết nối với bảng người dùng (user), danh mục bài viết (category id), thông tin này cũng sẽ được kết nối với bảng danh mục, và cuối cùng là thời gian đăng bài viết.

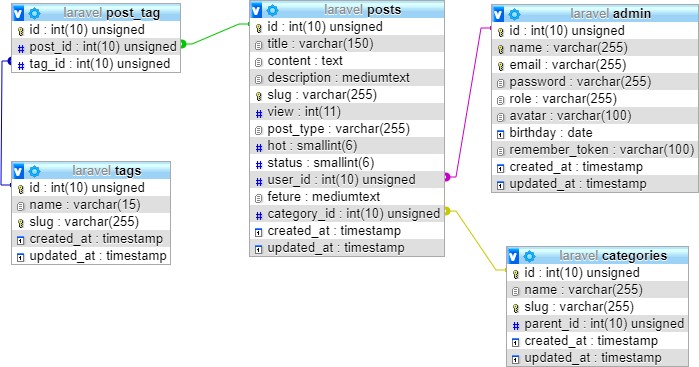
Dữ liệu dành cho danh mục bài viết (categories). Trước hết, trường ID sẽ là sự ưu tiên để làm khoá chính và để làm cầu kết nối giữa hai bảng danh mục bài viết và bảng bài viết (post), sau đó sẽ là các thông tin về danh mục, bao gồm tên danh mục (name), đường dẫn liên kết của danh mục (slug), danh mục cha (parent id) và cuối cùng là thời gian tạo. Bảng danh mục bài viết sẽ được sử dụng để hiển thị menu ở phần đầu của trang web.

Tiếp theo là các bảng dùng để quản lý thẻ (tags) và dữ liệu tải lên (files). Đối với thẻ, thông tin cần lưu lại trong cơ sở dữ liệu sẽ là ID, tên thẻ (name), đường dẫn liên kết của thẻ (slug), và thời gian tạo. Đối với dữ liệu tải lên, thông tin cần lưu lại sẽ là ID, tên (name), đường dẫn liên kết đến tệp tin đó (link), số hiệu bài viết (post

id), dùng để xác định xem tệp tin đó thuộc bài viết nào, và hơn nữa là do ai tải lên, và cuối cùng là thời gian tải lên.

Về cơ bản, cơ sở dữ liệu của ứng dụng đã được hình thành, tiếp theo là kiểm tra lần nữa xem nó đã đủ chuẩn 3NF hay chưa.

* Dạng chuẩn thứ nhất 1NF: không có phần tử/nhóm các phần tử lặp, tức là không thể phân chia một thuộc tính bất kỳ thành các phần nhỏ hơn. Trong khi thiết kế cơ sở dữ liệu, các thuộc tính đã được chia ra thành các phần nhỏ và đưa vào các bảng khác nhau sau đó kết nối với nhau, vậy cơ sở dữ liệu trên đã đáp ứng được chuẩn thứ nhất 1NF.
* Dạng chuẩn thứ hai 2NF: Không có phụ thuộc hàm không đầy đủ vào khoá chính, tức là không có một trường nào đó không thực sự liên quan đến khoá chính hay đến bảng. Dễ dàng nhận thấy, các thông tin trong các bảng đã được phân loại một cách đầy đủ để chỉ mang lại thông tin cho một thực thể nhất định trong ứng dụng, vậy nên sẽ không có thuộc tính không liên quan, vậy nó đã chuẩn 2NF.
* Dạng chuẩn thứ ba 3NF: Không có phụ thuộc hàm vào thuộc tính không khoá. Tất cả các dữ liệu chỉ phụ thuộc vào trường định danh (ID) của các bảng, và trường đó là khoá chính, vậy cơ sở dữ liệu trên đã đạt chuẩn 3NF.



*Hình 2. 4 - Sơ đồ quan hệ của cơ sở dữ liệu*

Hình 2.4 mô tả các quan hệ của các bảng trong cơ sở dữ liệu. Có một bảng trung gian đã được tạo ra để kết nối hai bảng bài viết (posts) và bảng thẻ (tags) là bảng post\_tag, bảng này xác định thẻ thuộc bài viết nào, hay nói cách khác là bài viết có bao nhiêu thẻ, là những thẻ nào.

# CHƯƠNG III – TRIỂN KHAI XÂY DỰNG WEBSITE TIN TỨC MỨC NGƯỜI DÙNG

# Triển khai lõi và cấu hình hệ thống sử dụng Laravel Framework

Để bắt đầu với dự án xây dựng website tin tức với Laravel, cần cài đặt lõi của Framework này (xem phần phụ lục). Sau khi cài đặt xong, bước tiếp theo là cấu hình hệ thống của Framework để phù hợp với môi trường lập trình. Để cụ thể hơn, môi trường lập trình được sử dụng trong tài liệu này là máy chủ web ảo sử dụng XAMPP phiên bản 7.0.8.

# Cấu hình chung cho hệ thống ứng dụng

Các thông tin cấu hình của Laravel đều nằm trong thư mục config, truy cập vào thư mục này và mở tệp tin app.php, đây là tệp tin chứa các cấu hình chung cho ứng dụng. Các thông tin cấu hình cơ bản như sau:

* Tên ứng dụng (name): tên của ứng dụng đang làm việc, mặc định là “Laravel”. Có thể sử dụng tên tuỳ ý.
* Môi trường (env): môi trường mà ứng dụng đang chạy, có 3 môi trường là development (ứng dụng đang được phát triển), testing (ứng dụng đã hoàn thành và đi vào giai đoạn kiểm thử), production (ứng dụng đã được đưa vào sử dụng).
* Phát hiện lỗi (debug): nếu giá trị là true, hệ thống sẽ hiển thị chi tiết lỗi, ngược lại, nếu giá trị là false, hệ thống sẽ chỉ đưa ra một cảnh báo chung mà không hiển thị chi tiết lỗi.
* Liên kết (url): đường dẫn liên kết, hay nói cách khác là địa chỉ trang web, nơi mà ứng dụng đang chạy.
* Múi giờ (time\_zone): Múi giờ của nơi phát triển ứng dụng.

Các thông tin cấu hình trên là các thông tin cơ bản, ngoài ra còn rất nhiều các thông tin cấu hình khác, sử dụng đối với các trường hợp nhất định và đối với những nhà phát triển có nhiều kinh nghiệm làm việc với Laravel.

Ví dụ cấu hình như sau:

'name' => env('APP\_NAME', 'Laravel'), 'env' => env('APP\_ENV', 'production'), 'debug' => env('APP\_DEBUG', false),

'url' => env('APP\_URL', 'http://localhost'), 'timezone' => 'Asia/Ho\_Chi\_Minh',

# Cấu hình cơ sở dữ liệu

Trước khi đi vào cấu hình cơ sở dữ liệu, phải tiến hành tạo cơ sở dữ liệu mới. Truy cập vào phpmyadmin để tạo cơ sở dữ liệu mới, trong dự án này, tên cơ sở dữ liệu được sử dụng là forge.

Sau khi tạo cơ sở dữ liệu mới, tiến hành truy cập và mở tệp tin database.php trong thư mục config để cấu hình cho cơ sở dữ liệu của ứng dụng. Các thông tin cấu hình cơ bản như sau:

* Mặc định (default): tên của kết nối tới cơ sở dữ liệu. Có thể đặt tên tuỳ ý, nhưng thông tin này phải là một trong các phần tử của connections phía dưới.
* Kết nối (connections): Các kết nối đến cơ sở dữ liệu. Đây là một mảng chứa thông tin của một hoặc nhiều kết nối tới các cơ sở dữ liệu, mỗi phần tử của mảng đại diện cho một kết nối, và một trong các phần tử này phải có tên giống với tên của kết nối trong phần default.

Ví dụ, trong dự án này, tên của kết nối là mysql, thông tin cấu hình trong phần defaut sẽ là mysql, và thông tin của kết nối mysql trong phần cấu hình connections như sau:

* Driver: Trình điều khiển kết nối cơ sở dữ liệu đến cơ sở dữ liệu MySQL, trong dự án này là mysql, ngoài ra còn một trình điều khiển khác là mysqli.
* Host: Địa chỉ của máy chủ chứa cơ sở dữ liệu, trong dự án này, do chạy máy chủ web ảo nên địa chỉ của nó sẽ là địa chỉ cục bộ: 127.0.0.1 hoặc localhost.
* Port: Cổng kết nối đến cơ sở dữ liệu, mặc định của MySQL sẽ là cổng số 3306. Thông tin này có thể được thay đổi trong cấu hình của XAMPP.
* Database: Tên cơ sở dữ liệu sẽ sử dụng, dự án này sử dụng tên cơ sở dữ liệu là forge như đã nêu ở trên.
* Username: Tài khoản truy cập vào cơ sở dữ liệu.
* Password: Mật khẩu của tài khoản trên.

Đó là các thông tin cấu hình cơ bản, ngoài ra còn một số thông tin khác ở phía dưới (chi tiết trong tệp tin database.php) dành cho nhà phát triển có nhiều kinh nghiệm làm việc với cơ sở dữ liệu.

Ví dụ cấu hình như sau:

'default' => env('DB\_CONNECTION', 'mysql'),

…

'connections' => [

'mysql' => [

'driver' => 'mysql',

'host' => env('DB\_HOST', '127.0.0.1'),

'port' => env('DB\_PORT', '3306'),

'database' => env('DB\_DATABASE', 'forge'), 'username' => env('DB\_USERNAME', 'forge'), 'password' => env('DB\_PASSWORD', ''), 'unix\_socket' => env('DB\_SOCKET', ''), 'charset' => 'utf8',

'collation' => 'utf8\_unicode\_ci', 'prefix' => '',

'strict' => true, 'engine' => null,

]

]

# Xây dựng trang chủ

Sau khi cấu hình xong, ứng dụng đã có thể khởi chạy hoàn toàn, nhưng nó chỉ có một trang web duy nhất, mặc định của Laravel, cần phải xây dựng các trang web để thay đổi nó. Đầu tiên là trang chủ của website tin tức.

Trước hết là làm việc với Controller của Laravel. Các Controller của Laravel được đặt trong thư mục app\Http\Controllers, tiến hành tạo một tệp tin mới có phần mở rộng là .php dành cho trang chủ, dự án này sử dụng tên tệp tin là PagesController.php. Tệp tin này sẽ chứa mã PHP, tiến hành tạo một lớp, lớp này có tên trùng tên với tên của tệp tin:

<?php

namespace App\Http\Controllers; use Illuminate\Http\Request;

class PagesController extends Controller

{

}

Trong lớp PagesController, tạo một phương thức cho trang chủ, tạm lấy tên là getindex(). Tạm dừng làm việc với Controller ở đây, tiếp theo sẽ làm việc với Route. Để tạo định hướng cho máy chủ, khi người dùng truy cập vào trang chủ của website, thì máy chủ sẽ hướng người dùng đến phương thức getindex() của lớp PagesController vừa tạo, truy cập vào đường dẫn routes của dự án, mở tệp tin web.php, đây là tệp tin chứa các định hướng trên website, xoá nội dung có sẵn của tệp tin này và thêm vào đoạn mã sau để định hướng:

Route::get('', 'PagesController@getindex');

Tiếp đến sẽ làm việc với model. Yêu cầu của trang chủ là hiển thị menu gồm các danh mục bài viết cho người dùng lựa chọn, vậy cần phải có một model lấy thông tin về các danh mục từ cơ sở dữ liệu để đưa lên trang web.

Các model trong Laravel được đặt trong thư mục app của dự án, tạo một tệp tin php trong thư mục này để làm model cho danh mục, dự án này lấy tên là Category.php. Thêm nội dung vào tệp tin này như sau:

namespace App;

use Illuminate\Database\Eloquent\Model; class Category extends Model

{

protected $table = 'categories';

}

đến.

Trong đó, *categories* là tên bảng trong cơ sở dữ liệu mà Controller sẽ truy vấn

Ngoài yêu cầu về danh mục bài viết, trang chủ còn có yêu cầu về các bài viết,

để thoả mãn yêu cầu thứ hai này, tạo thêm một model nữa, dự án này lấy tên là Post.php. Nội dung của tệp tin Post.php cũng tương tự như nội dung tệp tin Category.php:

namespace App;

use Illuminate\Database\Eloquent\Model; class Post extends Model

{

}

Có thể dễ dàng nhận thấy, trong tệp tin Post.php không khai báo biến $table như trong Category.php, do trong Laravel tự quy định bảng cần truy vấn theo tên của model, ở model Post, nó đã tự quy định $table = ‘posts’.

Sau khi đã tạo xong các model cần thiết cho trang chủ, quay lại PagesController, làm việc tiếp trên Controller. Công việc còn lại của phương thức getindex() là lấy ra các danh mục, bài viết, và đưa các dữ liệu đó vào một view.

Nội dung của phương thức getindex() trong PagesController như sau:

public function getindex()

{

$cates = Category::where('name','!=','video')->get();

$videos = Post::where('post\_type','=','video')-

>take(5)->orderBy('created\_at','des')->get();

return view('news.pages.home',['cates'=>$cates,'videos'=>$videos]);

}

Ở đây, phương thức đã đưa đến một view là news.pages.home, tức là đưa các dữ liệu đã lấy được đến tệp tin nằm ở thư mục *resources\views\news\pages\home.blade.php.* Tiến hành tạo tệp tin này và thêm các mã trình bày, chi tiết xem tại home.blade.php.

Đây mới chỉ là phần thân của trang chủ, trên phần thân còn có phần đầu trang web (header) và ở dưới là phần cuối trang web (footer), và đây là những phần mà trang nào cũng có, và trang nào cũng giống nhau, vậy nên, hai phần này sẽ được đưa vào một khuôn khổ riêng, trong Laravel gọi là khuôn mẫu (layout). Trong dự án này, các layout chung được đặt trong thư mục *resources\views\news\layout,* và các phần riêng được tách ra thành các tệp tin trong *resources\views\news\partials.*

# Xây dựng trang phân loại theo chủ đề bài viết

Trang phân loại theo chủ đề bài viết có chức năng hiển thị các bài viết theo danh mục mà người dùng đã chọn, nếu nhiều bài có thể chia thành nhiều phần, mỗi phần từ 10 đến 20 bài viết. Nếu danh mục bài viết không tồn tại hoặc số bài viết trong danh mục đó bằng 0 thì sẽ đưa ra thông báo.

Tương tự như xây dựng trang chủ, đầu tiên sẽ làm việc với Controller, có thể dùng chính Controller đã tạo ở bước dựng trang chủ là PagesController. Thêm một phương thức khác cho lớp PagesController, dự án này lấy tên là getCategory(). Nội dung của phương thức getCategory như sau:

public function getCategory($slug)

{

$cate = Category::where('slug', $slug)->first(); if(count($cate)==0 || count($cate->posts)==0){

return view('news.pages.category',['key'=>$slug]);

} else {

$list = Post::where('category\_id',$cate->id)-

>where('status',1)->orderBy('created\_at','des')->paginate(2);

return view('news.pages.category',['posts'=>$list,'cate'=>$cate-

>name]);

}

}

Trong đó, phương thức getCategory() có một tham số là $slug, tham số này sẽ được lấy trong Route, thêm vào tệp tin web.php của route như sau:

Route::get('category/{slug}','PagesController@getCategory');

Như vậy đã hướng người dùng đến phương thức getCategory() của lớp PagesController khi người dùng xem danh sách các bài viết trong danh mục bất kỳ, được truyền vào theo tham số slug. Thêm một view nữa được sử dụng trong phương thức getCategory() là news.pages.category, tức là tệp tin *resources\views\news\pages\category.blade.php.* Chi tiết tham khảo mã nguồn của tệp tin này.

# Xây dựng trang chi tiết bài đăng tin tức

Trang chi tiết bài đăng tin tức có chức năng hiển thị ra nội dung của bài viết.

Nếu bài viết không tồn tại, sẽ xuất hiện một thông báo lỗi.

Ở phần này, vẫn sử dụng lớp PagesController đã tạo, thêm phương thức cho bài viết để xác định bài viết mà người dùng đã chọn. Trong dự án này, tên phương thức đó là getPost(). Nội dung của phương thức getPost() như sau:

public function getPost($slug)

{

$post = Post::where('status',1)->where('slug', $slug)-

>first();

if(count($post)==0){ return

view('news.pages.singlepost',['key'=>$slug]);

} else

{

$post->view = $post->view + 1;

$post\_lq = Post::where('status',1)-

>where('slug','!=', $slug)->where('category\_id','=',$post-

>category\_id)->take(5)->get();

$post->save(); return

view('news.pages.singlepost',['post'=>$post,'lq'=>$post\_lq]);

Cũng tương tự như trang danh mục bài viết đã xây dựng, ở phương thức getPost() có tham số là $slug, là tham số truyền vào bởi route, sử dụng model là Post để lấy các dữ liệu từ cơ sở dữ liệu và đưa các dữ liệu đó ra view là singlepost.blade.php.

# Xây dựng chức năng tìm kiếm bài viết

Trong một website bất kỳ, chức năng tìm kiếm nội dung là không thể thiếu, website càng chứa nhiều thông tin, chức năng tìm kiếm càng cần thiết. Trong dự án này, chức năng tìm kiếm được xây dựng để tìm kiếm theo tiêu đề bài viết.

Trước hết, tạo một biểu mẫu (form) tìm kiếm trên thanh menu của website và trỏ hành động (action) của biểu mẫu về route search (xem mã nguồn tệp tin

public function getSearch(Request $request)

{

$key= $request->input('key');

$posts = Post::where('status',1)->where('title', 'like', '%'.$key.'%')->get();

return view('news.pages.search',['posts'=>$posts,'key'=>$key]);

}

resources\views\news\partials\menu.blade.php để biết thêm chi tiết). Sau đó, tạo phương thức để thực thi tìm kiếm trong Controller PagesController, dự án này lấy tên của phương thức là getSearch():

Cũng giống như các phương thức trước, phương thức getSearch() sử dụng model Post để lấy các dữ liệu trong bảng posts của cơ sở dữ liệu, và đưa các dữ liệu lấy được ra view search.blace.php.

Sau đó, chỉnh sửa route để đưa người dùng đến phương thức getSearch() đã tạo:

Route::get('search','PagesController@getSearch')-

>name('search');

Về cơ bản, các trang chính của website mức người dùng đã hoàn thành, các trang khác như lọc danh mục theo thẻ (tags) và theo tác giả tương tự như các trang vừa tạo. Chi tiết xem thêm mã nguồn. Việc tiếp theo là xây dựng phần quản trị dành cho quản trị viên.

# CHƯƠNG 4 – TRIỂN KHAI XÂY DỰNG QUẢN TRỊ CHO WEBSITE TIN TỨC

* 1. **Xây route quản trị và layout dùng chung**
     1. **Tạo nhóm route cho admin**

Để triển khai trang quản trị vào dự án trước hết cần tạo một nhóm route riêng biệt cho việc quản trị. Truy cập tệp web.php theo đường dẫn routes/web.php thêm vào một route như sau:

}

/\* Route con \*/

'admin', 'middleware' => 'auth'],

=>

Route::group(['prefix'

function(){

Tạo route theo nhóm tức bên trong hàm của route bao gồm các route con mà đường dẫn của nó sẽ là /admin/route-con dựa vào thông số prefix (tiền tố), tiếp theo để xác thực người dùng và ở đây là admin và người đăng bài viết sẽ dùng một middlewave là auth (sẽ nói rõ ở phần 4.2).

# Tạo layout dùng chung cho trang quản trị và tổ chức thư mục

Truy cập theo đường dẫn resource/view tạo một thứ mục mới admin. Tại đây sẽ lưu tất cả tệp view của trang quản trị. Tạo tiếp các thư mục con tương ứng với các chức năng trong trang quản trị. Lúc này cấu trúc thư mục sẽ như sau:

- Views

/----- admin

/----- author

/----- category

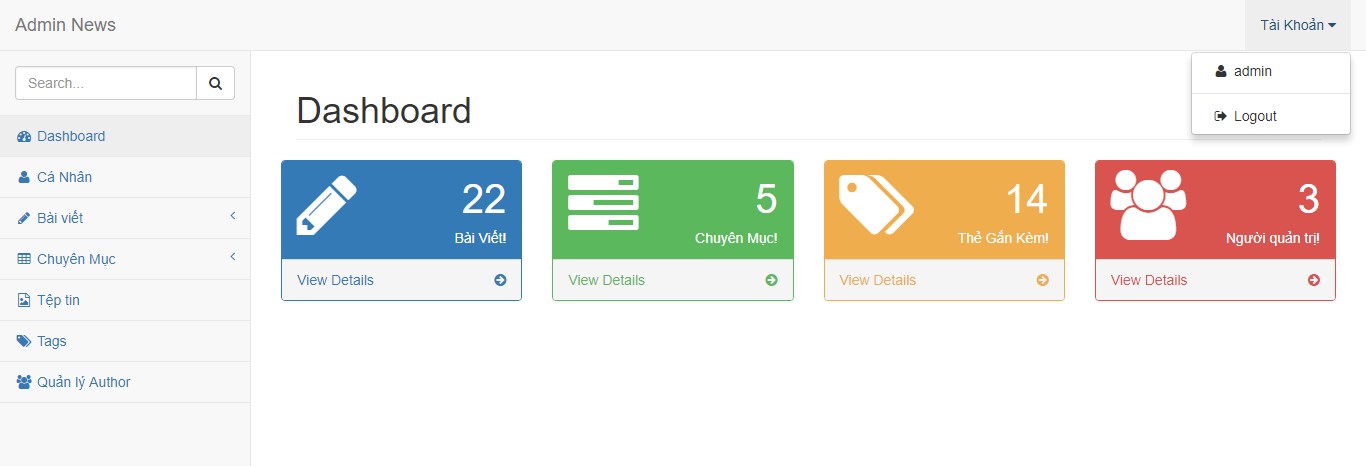
/----- layout

/----- patials

/----- post

/----- tag

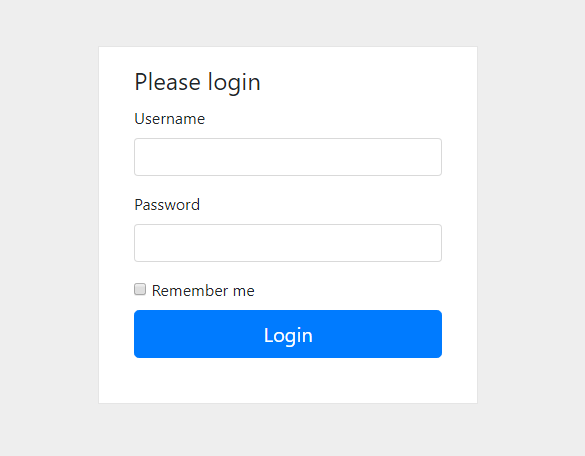
Tại thư mục layout tạo một tệp có tên layout.blade.php để có làm layout dùng chung cho các trang khác trong quản trị. Các trang con chỉ việc kế thừa lại layout này và tùy biến với nội dung trang đó. Giao diện trang quản trị cơ bản như sau:



*Hình 4. 1 - Giao diện trang chủ quản trị*

# Xây dựng đăng nhập và đăng xuất

Để xác thực người dùng truy cập vào trang quản trị với đường dẫn /admin thì bắt buộc phải xây dựng một trang đăng nhập và nếu qua được bước xác thực này thì mới cho vào trong trang quản trị. Giao diện trang đăng nhập như sau:



*Hình 4. 2 - Giao diện trang đăng nhập cho quản trị*

thực.

Có 2 trường là username và password là thông tin người dùng nhập vào để xác

Tiếp theo tạo các route trong web.php như sau:

Route::get('login', 'LoginController@getLogin');

Route::post('login', 'LoginController@postLogin')-

>name('login');

Route::get('logout', 'LoginController@getLogout');

Các route trên được xử lý tương ứng với các hàm trong lớp LoginController, lớp này sẽ dụng lớp Auth được thiết kế sẵn của laravel để xác thực người dùng.

Với route thứ nhất sẽ quy định đường dẫn /login và phương thức get do hàm getLogin trong lớp LoginController và thực hiện các hành động mà hàm này quy định.

public function getLogin()

{

if (Auth::check())

{

return redirect()->route('dashbroad');

}

return view('admin.login');

}

Hàm này cơ bản để kiểm tra tình trạng người dùng đã đăng nhập hay chưa, nếu người dùng đăng nhập thì hàm Auth::check() sẽ trả về true và sẽ chuyến hướng người dùng này vào ngay trong trang quản trị. Ngược lại nếu người dùng chưa đăng nhập thì sẽ trả về trang login trong thư mục views/admin/login.

Các thông tin người dùng nhập sẽ được gửi lên đường dẫn /login với phương thức post và do hàm postLogin trong lớp LoginController xử lý.

public function postLogin(Request $request)

{

$rules = [

'username' => 'required | min : 4 | max : 25 ', 'password' => 'required | min: 8'

];

$msg = [

'required' => ':attribute không được bỏ trống.', 'password.min' => 'Password phải lớn hơn :min ký tự.', 'username.min' => 'Username phải lớn hơn :min ký tự.', 'username.max' => 'Username phải nhỏ hơn :max ký tự.',

];

$validator = Validator::make($request->all(), $rules,

$msg);

if ($validator->fails()) { return redirect()->back()

->withErrors($validator)

->withInput();

} else {

$username = $request->input('username');

$password = $request->input('password');

if( Auth::attempt(['name' => $username, 'password' =>

$password], $request->input('remember') ) ){

return redirect()->route('dashbroad');

} else {

$msg = new MessageBag(['errlogin'=> 'Sai thông tin đăng nhập.']);

return redirect()->back()-

>withErrors($msg);

}

}

Hàm này cơ bản sẽ kiểm tra dữ liệu người dùng nhập vào để tăng độ an toàn cho hệ thống, tiếp sẽ sẽ lấy thông tin từ hai trường người dùng nhập vào là username và password rồi dùng hàm Auth::attemp() để xác thực, hàm này sẽ lấy thông tin người dùng nhập vào làm 2 đối số rồi dùng mã hoá aes-256 bit để mã hóa password mà khi tạo đã quy ước trước đó. So sánh thông tin người dùng được lưu trong bảng Admin, nếu thông tin chính xác sẽ chuyển hướng người dùng sang trang quản trị đồng thời cung cấp session cho người dùng ngược lại nếu sai thì sẽ chuyển hướng sang trang đăng nhập và hiển thị thông tin lỗi.

Để đăng xuất người dùng dùng hàm Auth::logout() thì sẽ xóa session người dùng và chuyển hướng sang trang đăng nhập. Đường dẫn /logout trong route được gán vào hàm getLogout trong class LoginController:

function getLogout()

{

if( Auth::check() ) Auth::logout(); return redirect()->route('login');

}

# Xây dựng trang chủ quản trị

Vì là một thành phần trong trang quản trị nên route sẽ nằm trong nhóm route:

Route::group(['prefix' => 'admin', 'middleware' => 'auth'], function(){

Route::get('', 'HomeController@getdashbroad')-

>name('dashbroad');

}

Vì vậy đường dẫn tương ứng của route này để đến trang chủ quản trị sẽ là

/admin. Vì không có gì trong tham số thứ nhất nên mặc định sẽ được coi là đại diện trong nhóm này.

Đường dẫn trong route là /admin được giao quyền xử lý của hàm getdashbroad trong lớp HomeController.

public function getdashbroad() {

$post\_count = Post::count();

$admin\_count = Admin::count();

$tag\_count = Tag::count();

$category\_count = Category::count(); return view('admin.index',['num\_post'=>

$post\_count,'num\_admin'=>$admin\_count,'num\_tag'=>$tag\_coun t,'num\_cate'=>$category\_count]);

}

Hàm này sẽ dùng các model Post,Admin,Tag,Category rồi đếm các bản ghi tương ứng có trong database. Trả về trang hiển thị là admin.index (tương ứng với file admin/index.blade trong views) kèm theo là các biến đếm số bản ghi. Giao diện quản trị như hình 4.1.

# Xây dựng trang cá nhân

Đầu tiên tạo route cho trang cá nhân trong file routes/web.php như sau

Route::get('profile', 'ProfileController@getProfile');

Route::post('profile/update', 'ProfileController@profileUpdate');

Hai route này thuộc trang quản trị nên cũng sẽ được nằm trong nhóm route admin. Đường dẫn để vào trang là /admin/profile.

* Route thứ nhất Khi truy cập với phương thức get /admin/profle hàm getProfile sẽ xử lý như sau:

public function getProfile()

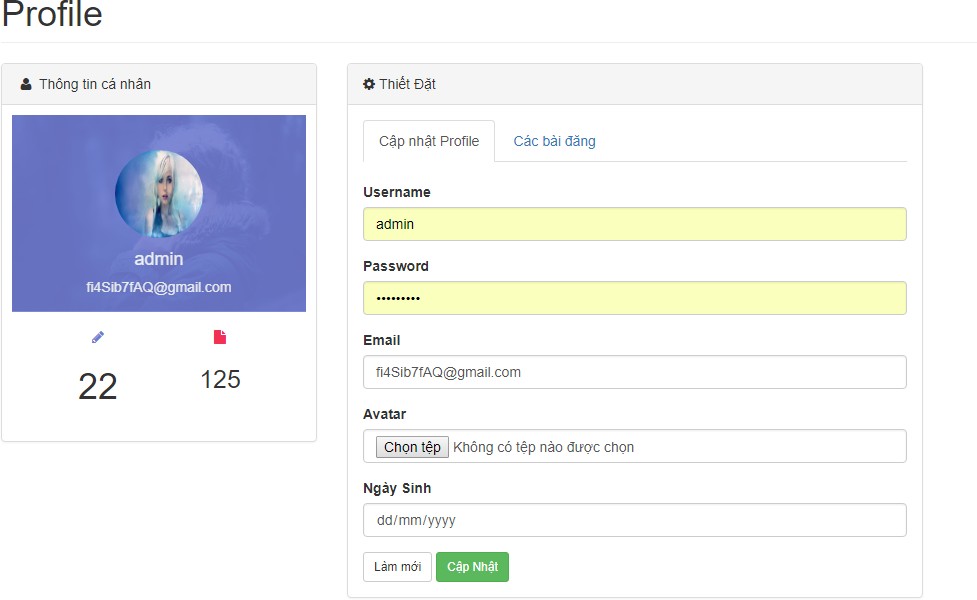
{

$profile = Admin::find(Auth::user()->id);

return view('admin.author.profile',['profile'=>$profile]);

}

Lấy thông tin ID người dùng được lưu trong session với Auth::user()->id và tìm thông tin người dùng này trong model Admin rồi trả về trang hiển thị admin.author.profile (tương ứng file views/admin/author/profle.balde.php) kèm theo biến profile chứa thông tin người dùng. Trang thông tin người dùng như sau:



*Hình 4. 3 - Giao diện thông tin tài khoản*

* Route thứ 2 khi người dùng gửi thông tin cập nhật lên với đường dẫn

/admin/profile/update với method post. Xử lý cập nhật trong hàm profileUpdate như sau: Kiểm tra thông tin người dùng nhập vào để đảm bảo độ tin cậy và an toàn, tiếp theo sẽ lấy thông tin người dùng nhập vào rồi cập nhật bản ghi tương ứng với ID người dùng trong session qua Auth::user()->id, nếu cập nhật thành công trả lại trang admin.author.profile (tương ứng file views/admin/author/profle.balde.php) và kèm thông báo thành công ngược lại nếu lỗi cũng trả về trang này và kèm theo thông báo lỗi.

# Xây dựng trang quản lý bài viết

Đăng ký các route cho trang quản lý bài viết nằm trong nhóm route admin như sau:

Route::prefix('post')->group(function () {

Route::get('/', 'PostController@getList')->name('list-post'); Route::get('add', 'PostController@getAdd'); Route::put('updateStatus', 'PostController@updateStatus');

Route::put('updateHot', 'PostController@updateHot'); Route::post('add', 'PostController@postAdd'); Route::get('update/{id}', 'PostController@getUpdate'); Route::post('update/{id}', 'PostController@postUpdate'); Route::get('delete/{id}', 'PostController@getDelete');

});

Gom các route của quản lý bài viết tại vào một nhóm (prefix(‘post’)).

# Trang hiển thị danh sách bài viết

Trang danh sách được xử lý trong hàm getList thuộc lớp PostController với nội dung:

public function getList()

{

$posts = Post::all(); if(Auth::user()->role=='author'){

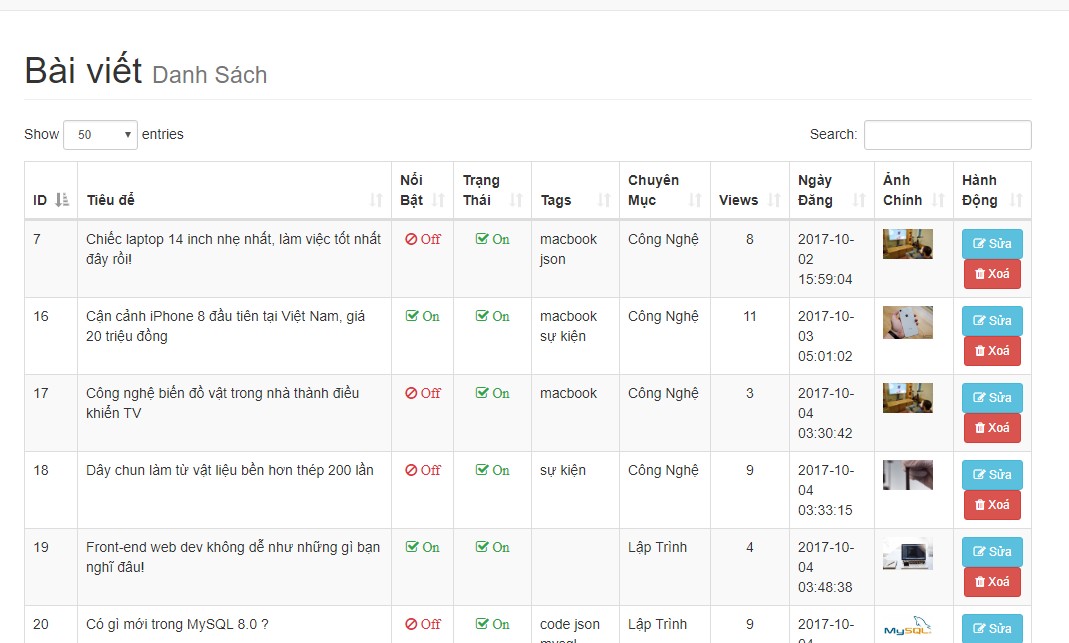
$posts = $posts->where('user\_id',Auth::user()->id);

}

return view('admin.post.list',['posts'=>$posts]);

}

Hàm này có chức năng sẽ kiểm tra người dùng đăng nhập có quyền như thế nào, nếu người dùng là ‘author’ thì sẽ chỉ lấy các bài vết của chính người này, và ngược lại nếu có quyền là ‘admin’ thì sẽ lấy tất cả các bài viết trong bảng posts thông qua model Posts. Để lấy thông tin về quyền của người dùng sẽ truy cập thông qua lớp Auth cuối cùng sẽ trả về trang hiển thị trong view admin.post.list kèm với biến post có lưu danh sách bài viết. Trang hiển thị danh sách sẽ như sau:



*Hình 4. 4 - Giao diện danh sách bài viết*

Bên phía giao diện để có thể xem các bản ghi cách thuận tiện và tiện ích sẽ sử dụng plugin DataTables Table kết hợp với thư viện jQuery. Nó làm tăng sự tiện lợi và tương tác ví dụ như tìm kiếm bản ghi, xem các bản ghi theo sắp xếp trường,… Tất cả thực hiện nhanh chóng và không cần tải lại trên server.

Đối với trường hợp người dùng là ‘author’ sẽ được hiển thị danh sách bài viết của chính mình kèm chức năng sửa hoặc xóa các bài viết đó ngoài ra không hiển thị toàn bộ danh sách bài viết cũng như không có quyền thay đổi hay xóa bỏ bài viết của người khác.

Đối với trường hợp người dùng là ‘admin’ sẽ được hiển thị toàn bộ danh sách bài viết ,

Có thể sửa xóa bài viết của chính mình, xóa bài viết của người khác nhưng không có quyền thay đổi nó. Ngoài ra còn có thêm chức năng thay đổi trạng thái bài viết hoặc chọn bài viết là bài viết nổi bật người dùng là author sẽ bị vô hiệu hóa chức năng này. Hai chức năng này đều sử dụng AJAX đễ xử lý. Bắt sự kiện nhấn vào biển tượng với bản ghi tương ứng sẽ gửi lên server thông tin thay đổi với bản ghi và được cập nhật.

# Trang thêm bài viết

Route của trang này Route::get('add', 'PostController@getAdd') hàm getAdd sẽ xử lý như sau:

public function getAdd()

{

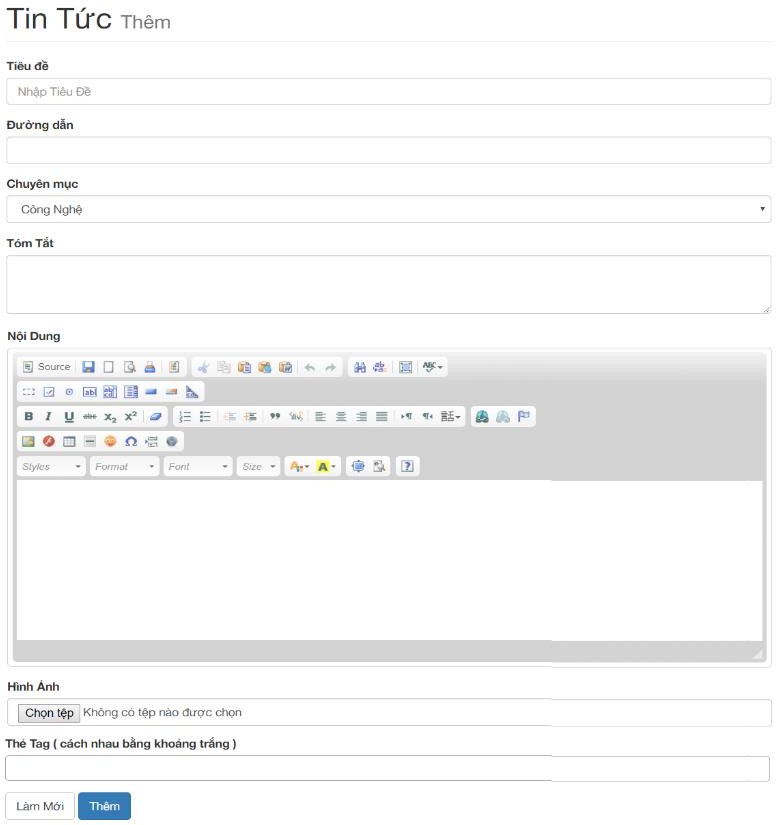
$cates = Category::all();

$tags = Tag::all(); return

view('admin.post.add',['cates'=>$cates,'tags'=>$tags]);

}

Lấy thông tin tất cả các bản ghi của chuyên mục ($cates) và thẻ ($tags) sau đó trả về trang hiển thị bên view là trang admin.post.add kèm thoe đó là 2 biến $cates và $tag.



*Hình 4. 5 - Giao diện trang đăng bài viết*

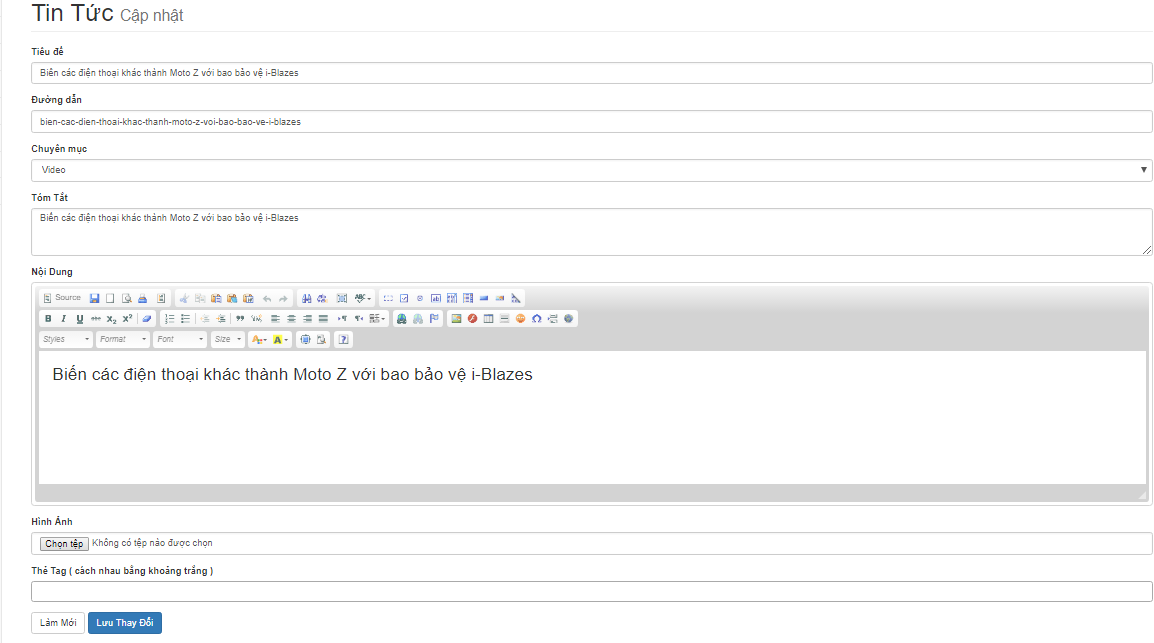
Trang thêm bài viết bao gồm các trường : Tiêu đề, đường dẫn, chuyên mục, tóm tắt nội dung, nội dung, hình ảnh (nếu có) ,thẻ. Form này sẽ gửi thông tin tới admin/post/add theo phương thức post tức route :

Route::post('add', 'PostController@postAdd')

Tại hàm postAdd sẽ thực hiện kiểm tra dữ liệu nhập vào và thực hiện thêm mới bản ghi. Nếu thêm thành công sẽ điều hướng về trang danh sách bài viết kèm thông báo thành công ngược lại nếu có lỗi xảy ra sẽ điều hướng về trang thêm bài viết kèm thông báo lỗi và thông tin cũ người dùng nhập vào để tiện lợi hơn.

# Trang sửa bài viết

Route thực hiện việc sửa bài viết *Route::get('update/{id}', 'PostController@getUpdate')*. *{id}* ở đây chính là một biên tương ứng với id bài viết sẽ sửa. Chi tiết hàm getUpdate() xem thêm trong mã nguồn.



*Hình 4. 6 - Giao diện trang sửa bài viết*

Form này sẽ gửi thông tin tới admin/post/update/id theo phương thức post tức route:

Route::post('update/{id}', 'PostController@postUpdate');

Hàm postUpdate sẽ thực hiện việc kiểm tra dữ liệu mà form này gửi lên và lưu thay đổi vào database.

# Trang xóa bài viết

Route của trang xoá bài viết:

Route::get('delete/{id}', 'PostController@getDelete')

Hàm thực hiện lấy biến id từ đường dẫn và kiểm tra xem bài viết đó cò tồn tại không đồng thời kiểm tra người dùng có quyền là admin hay là người đăng bài viết thì thực hiện xóa nếu không sẽ gửi lại thông báo lỗi tương ứng.

# Trang quản lý chuyên mục

Tất cả các route chuyên mục,thẻ,quản lý người dùng,quản lý file chỉ có admin mới có quyền truy cập vì thế gom nhóm các route này trong một nhóm route từ đó thiết lập middlewave để kiểm tra nhóm này.

Route::middleware(['role'])->group(function () {

}

Dùng lớp Auth để lấy thông tin về quyền người dùng nếu có quyền là admin sẽ tiếp tục cho phép thực hiện các route còn nếu không sẽ chuyển hướng về trang chính của quản trị.

Route của trang quản lý chuyên mục bao gốm

Route::prefix('category')->group(function () { Route::get('/', 'CategoryController@getList'); Route::get('add', 'CategoryController@getAdd'); Route::post('add', 'CategoryController@postAdd');

Route::get('data', 'CategoryController@dataTable')-

>name('data');

Route::post('update', 'CategoryController@postUpdate'); Route::delete('delete', 'CategoryController@delete');

});

# Trang thêm chuyên mục

Route thực hiện là hàm getAdd, tương ứng với đường dẫn /category/add. Hàm getAdd trong lớp CategoryController như sau:

public function getAdd()

{

$cates = Category::all();

return view('admin.category.add',["cates"=>$cates]);

}

Lấy thông tin các chuyên mục đang có và lưu vào biến $ cates. Trả về trang hiển thị tại view là admin.category.add kèm theo danh sách chuyên mục là biến

$cates. Giao diện trang thêm chuyên mục như sau:



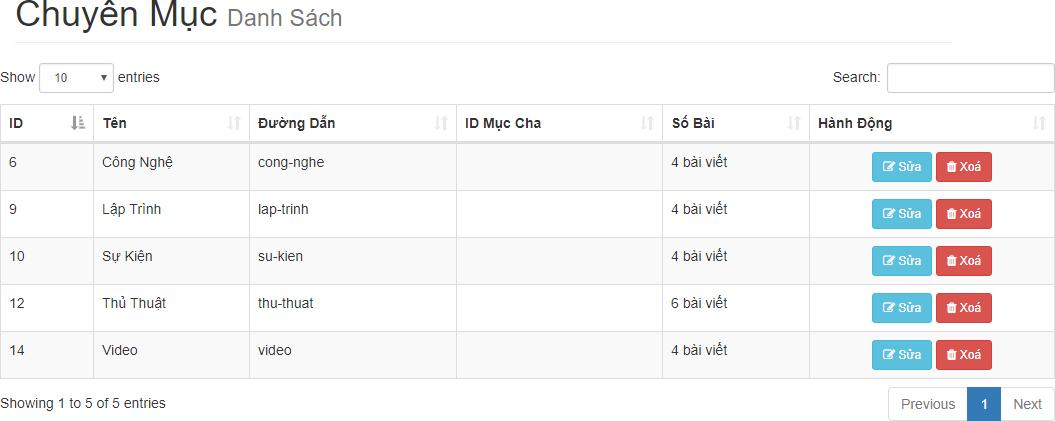
*Hình 4. 7 - Giao diện thêm chuyên mục bài viết*

Hàm postAdd sẽ thực hiện kiểm tra dữ liệu gửi lên nếu thông tin có định dạng đúng sẽ lấy thông tin gửi lên và lưu chuyên mục mới vào bảng Categories sau đó chuyển hướng về trang thêm chuyên mục kèm thông báo thành công.

# Danh sách, Xóa chuyên mục

Trang danh sách chuyên mục có route trỏ tới hàm getList(), hàm này sẽ thực hiện trả về trang hiển thị admin.category.list tại views kèm thoe danh sách chuyên mục.

Trang danh sách như sau:



*Hình 4. 8 - Giao diện danh sách chuyên mục*

Dữ liệu được đổ ra qua biến cates mà controler trả về, ở đây cũng sử dụng plugin DataTables Table kết hợp với thư viện jQuery để đữ liệu có sự tương tác tốt. Bên cạnh đó nút xóa cũng được tích hợp luôn trong trang này. Khi nhấn vào nút xóa sử thực hiện gửi thông tin bản ghi cần xóa lên server qua đường dẫn

/admin/cateogry/delete thuộc route

Route::delete('delete', 'CategoryController@delete')

# Quản lý thẻ và người viết bài (author)

Các chức năng thêm sửa xóa với thẻ và người viết bài hoàn toàn dùng AJAX giống với cách dùng để xử lý trong chuyên mục.

Danh sách route của thẻ :

Route::prefix('tag')->group(function () {

Route::get('/', 'TagController@getList')->name('list-tag'); Route::get('data', 'TagController@dataTable')->name('data-

tag');

Route::post('add', 'TagController@postAdd'); Route::put('update', 'TagController@putUpdate'); Route::delete('delete', 'TagController@delete');

});

Danh sách route của quản lý người viết bài :

Route::prefix('author')->group(function () {

Route::get('/', 'AdminController@getList')->name('list- author');

Route::get('data', 'AdminController@dataTable')-

>name('data-author');

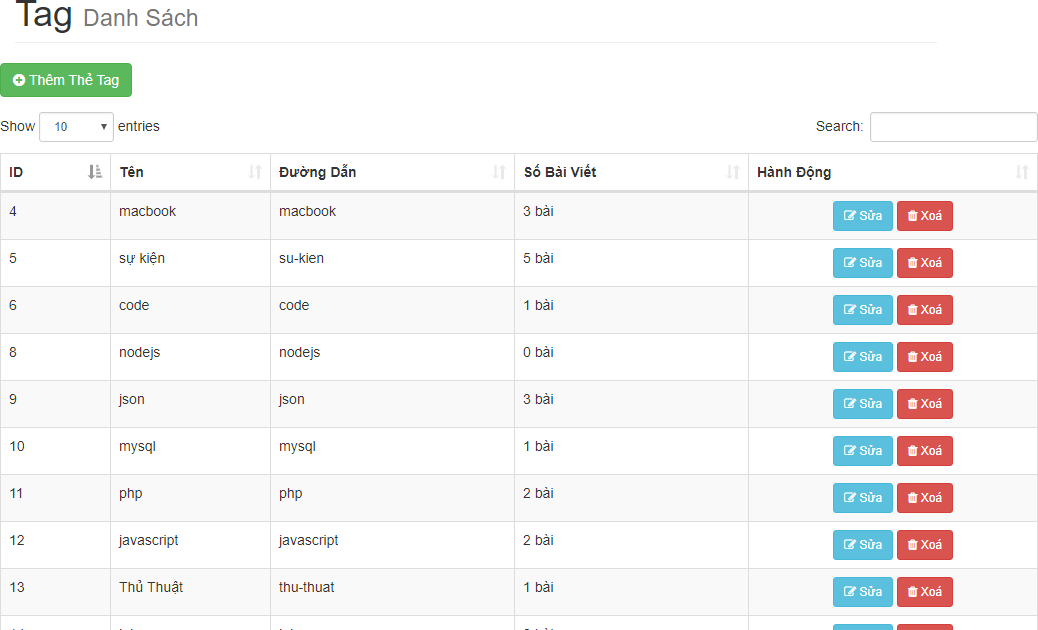
Route::post('add', 'AdminController@postAdd'); Route::delete('delete', 'AdminController@delete');

});

# Lấy danh sách hiển thị

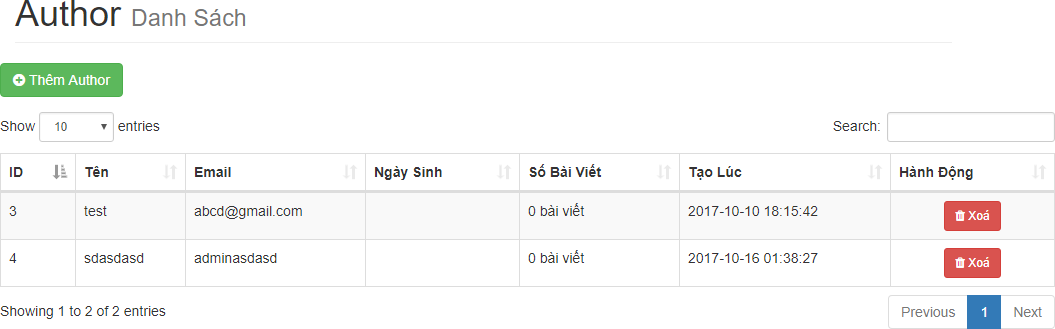
Để lấy danh sách hiển thị hai chức năng đều dùng AJAX để lấy các bản ghi về qua phương thức get với đường dẫn admin/tag/getdata, admin/author/getdata. Tại các hàm dataTable trong Controller sẽ trả về danh sách các bản ghi để khi có yêu cầu gửi AJAX lên có thể lấy được danh sách các bản ghi và đổ ra.

Trang danh sách của thẻ như sau:



*Hình 4. 9 - Giao diện danh sách thẻ*

Trang danh sách người viết bài:

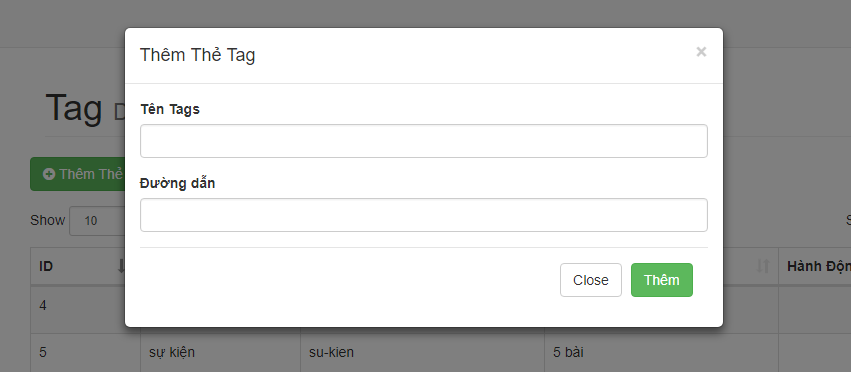


*Hình 4. 10 - Giao diện sach sách quản trị viên đăng bài*

# Thêm các bản ghi

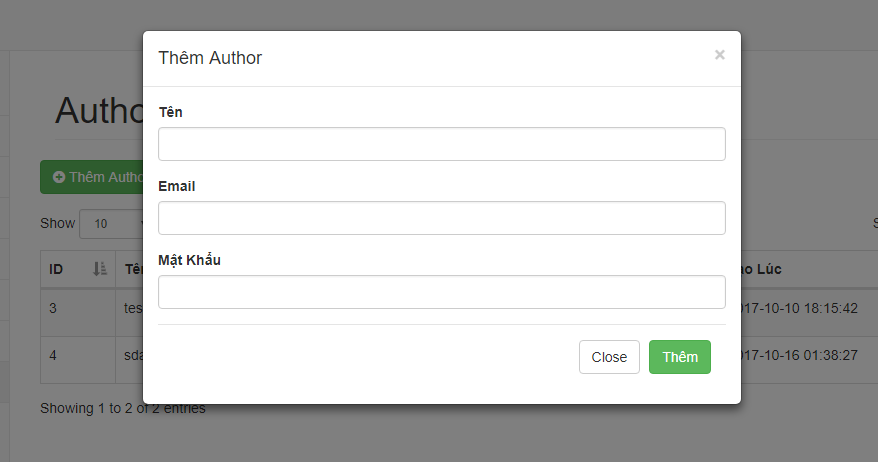
Việc thêm các bản ghi dùng AJAX để thực hiện, các bước thực hiện như sau khi ngườ dùng nhấn vào nút thêm sẽ hiển thị ra một popup với nội dung là một form và các trường để gửi lên server qua AJAX. Khi người dùng nhấn nút thêm thì sẽ gửi thông tin từ form lên server xử lý nếu kết quả trả về thành công thì sẽ chạy đoạn script lấy danh sách để cập nhật danh sách bản ghi mới nhất.

Popup form thêm như sau:



*Hình 4. 11 - Giao diện thêm thẻ*

Thêm bản ghi trong quản lý quản trị viên đăng bài:

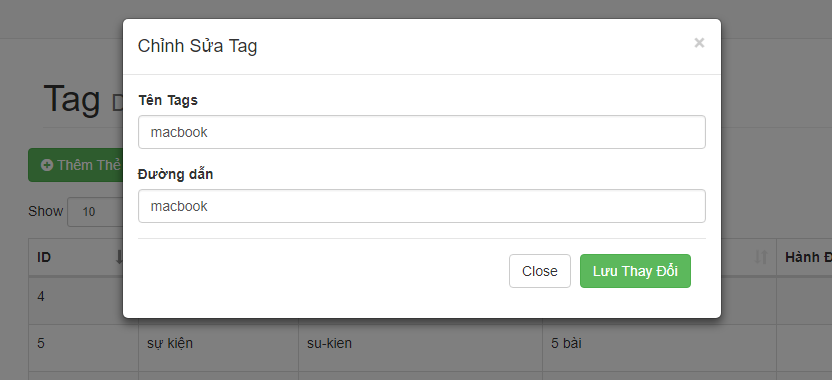


*Hình 4. 12 - Giao diện thêm quản trị viên đăng bài*

# Sửa các bản ghi

Việc sửa đổi bản ghi cũn dùng AJAX để thực hiện, quy trình như sau: khi người dùng ấn vào nút sửa thi sẽ hiển thị lên popup với nội dung là form đã điền thông tin của bản ghi tương ứng trong các trường. Khi người dùng nhấn vào nút thay đổi sẽ gửi thông tin từ form lên thông qua ajax, server sẽ kiểm tra thông tin và trả lại thông báo.

PopUp sửa như sau:

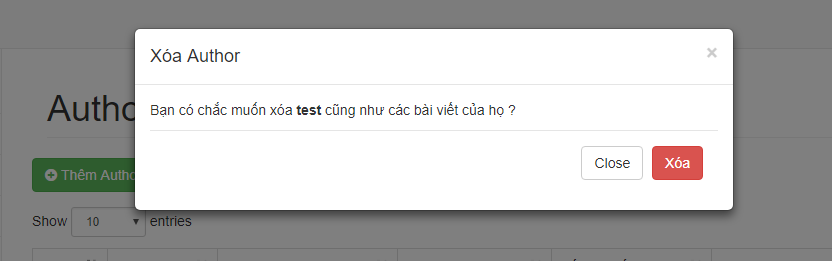


*Hình 4. 13 - Giao diện sửa thẻ*

Chức năng này sẽ thực hiện việc kiểm tra bản ghi với tham số id đưa lên nếu tồn tại sẽ thực hiện xóa và trả lại thông báo ‘ok’ nếu lỗi sẽ trả về thông báo lỗi tương ướng.

Việc xóa bản ghi trong người đăng bài hoàn toàn tương tự. Giao diện xóa như

sau:

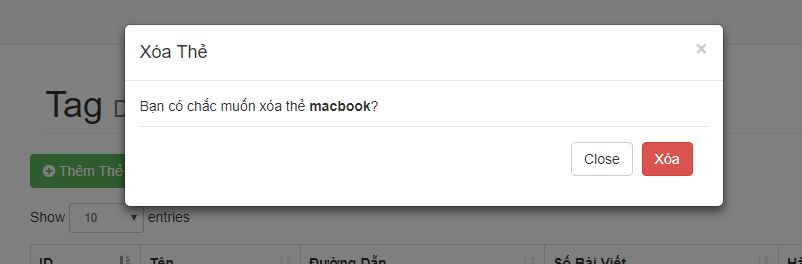


*Hình 4. 14 - Giao diện xoá tài khoản quản trị viên đăng bài*

# Xóa các bản ghi

Việc xóa các bản ghi cũng dùng AJAX để thực hiện, quy trình như sau : Khi ngườ dùng ấn vào xóa bản ghi sẽ lấy ra ID của bản ghi tương ứng sau đó sẽ gửi thông tin ID cần xóa lên server thông qua AJAX trên server xử lý xong sẽ trả lại thông báo thành công hoặc lỗi, nếu thành công thì sẽ làm mới danh sách các bản ghi để cập nhật thông tin qua hàm AJAX lấy danh sách đã nêu ở trên.

Giao diện xóa như sau:



*Hình 4. 15 - Giao diện xoá thẻ*

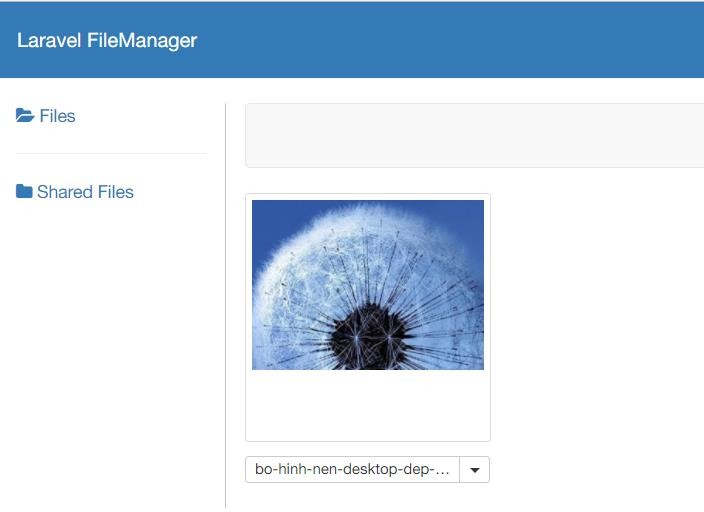
Chức năng này sẽ thực hiện việc kiểm tra bản ghi với tham số id đưa lên nếu tồn tại sẽ thực hiện cập nhật và trả lại thông báo ‘ok’ nếu lỗi sẽ trả về thông báo lỗi tương ướng.

# Quản lý tệp tin

Trang quản lý tệp tin được sử dụng một plugin tên Laravel FileManager cho phép quản lý các tệp tin cách rât tiện lợi như : cắt ảnh, đổi kích cỡ, đổi tên…

Khi tích hợp plugin này vào trong phần cấu hình sẽ thêm middlewave để yêu cầu người dùng đăng nhập và có quyền là admin mới có thể truy cập vào plugin này. Vào file config/lfm.php và thay đổi thông số như sau:

'middlewares' => ['web', 'auth','role']



*Hình 4. 16 - Giao diện quản lý tệp tin*

# KẾT LUẬN

Trải qua một tháng xây dựng website tin tức với Laravel Framework thì nhóm em đã phần nào hoàn thiện tương đối các chức năng cơ bản của một website tin tức thực tế. Người dùng có thể dễ dàng xem các tin tức trên trang web theo các tiêu chí lọc tin tức theo chuyên mục, theo thẻ, theo người đăng bài để tiếp cận với các tin tức cách phù hợp với mình nhất, bên cạnh đó các bài viết có thể được sự đóng góp từ nhóm người đăng bài riêng, với công cụ soạn bài viết hỗ trợ tốt được tính hợp từ CKEditor. Nhóm người đăng bài giúp mở rộng quy mô về dữ liệu cho website. Tất cả đều dưới sự quản lý từ quản trị viên, quản trị viên có thể dễ dàng quản lý thêm, sửa, xóa với chuyên mục, thẻ, người đăng bài, tệp tin… một cách nhanh chóng.

Tuy vậy để hoạt động tốt và tối ưu hơn còn một số mặt hạn chế cần khắc phục như sau:

* Các tệp tin hình ảnh, video … đính kèm với bài viết đều lưu trực tiếp trên máy chủ của website, với số lượng nhỏ thì không là vấn đề nhưng khi quy mô được mở rộng thì sẽ tốn rất nhiều dung lượng cho máy chủ. Vì thế để khắc phục vấn đề này cần có cơ chế xử lý các tệp tin được đính kèm cần chuyển sang một nơi lưu trữ khác uy tín, an toàn, chi phí thấp… VD: google drive, flick pro, youtube

…

* Cần nhiều thiết đặt tùy chỉnh hơn trong trang quản trị đối với người đăng bài và quản trị viên. Vd: Các thiết đặt tùy chỉnh số bài được hiện trong một trang đối với những nơi dùng phân trang.
* Thêm chức năng nhận thông báo qua mail hoặc thông báo trên trình duyệt khi có bài mới nếu người dùng có đăng ký nhận tin giúp người đăng ký cập nhật tin tức mới nhất từ webstie cũng như giữ số lượng độc giả cho website. Để làm được chức năng này cần lưu các thông tin liên lạc người dùng sau đó cấu hình một email để gửi.

# TÀI LIỆU THAM KHẢO

[1]. Bootstrap • The most popular HTML, CSS, and JS library in the world. [http://getbootstrap.com](https://l.facebook.com/l.php?u=http%3A%2F%2Fgetbootstrap.com%2F&amp;h=ATO7Q3SgQKCON4XzJJKM46TbVKJToQqECcDuZllSCWLQJh6kSXLfNQGnWjYIgspFB6DP2YCfH8pbKlRhoMR7ukS8BaC27nFcT1iAC_4TZ2apRQzS43jheAWiqeGJT24rV7M6dZ-mAAsU)

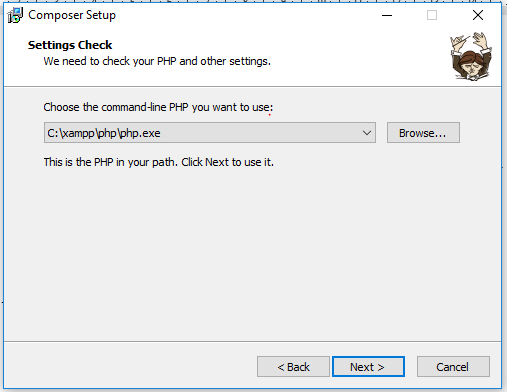
[2]. jQuery AJAX Methods, [https://www.w3schools.com/jquery/jquery\_ref\_ajax.asp](https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.w3schools.com%2Fjquery%2Fjquery_ref_ajax.asp&amp;h=ATO7Q3SgQKCON4XzJJKM46TbVKJToQqECcDuZllSCWLQJh6kSXLfNQGnWjYIgspFB6DP2YCfH8pbKlRhoMR7ukS8BaC27nFcT1iAC_4TZ2apRQzS43jheAWiqeGJT24rV7M6dZ-mAAsU) [3]. jQuery API Document, [https://api.jquery.com](https://l.facebook.com/l.php?u=https%3A%2F%2Fapi.jquery.com%2F&amp;h=ATO7Q3SgQKCON4XzJJKM46TbVKJToQqECcDuZllSCWLQJh6kSXLfNQGnWjYIgspFB6DP2YCfH8pbKlRhoMR7ukS8BaC27nFcT1iAC_4TZ2apRQzS43jheAWiqeGJT24rV7M6dZ-mAAsU)

[4]. Laravel Document, [https://laravel.com/docs/5.4](https://l.facebook.com/l.php?u=https%3A%2F%2Flaravel.com%2Fdocs%2F5.4&amp;h=ATO7Q3SgQKCON4XzJJKM46TbVKJToQqECcDuZllSCWLQJh6kSXLfNQGnWjYIgspFB6DP2YCfH8pbKlRhoMR7ukS8BaC27nFcT1iAC_4TZ2apRQzS43jheAWiqeGJT24rV7M6dZ-mAAsU)

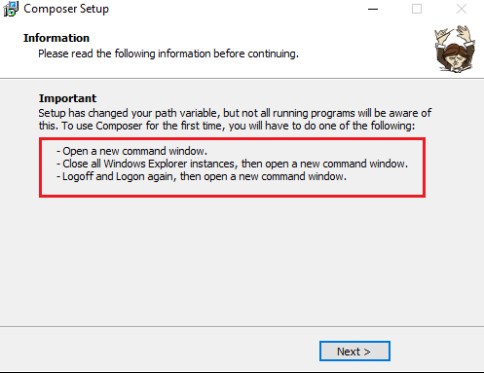
**PHỤ LỤC**

1. Cài đặt Composer
   * Tải composer: <https://getcomposer.org/Composer-Setup.exe>

Sau khi nhấn vào đường dẫn, tập tin cài đặt sẽ được tải về. Trong quá trình cài đặt sẽ được composer thông báo chọn đến tập tin php.exe trong XAMPP.



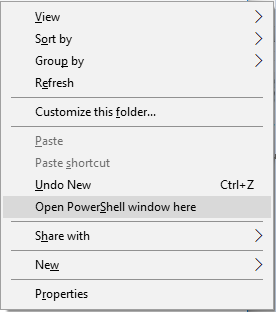
Sau khi cài đặt xong composer yêu cầu làm theo các bước sau:



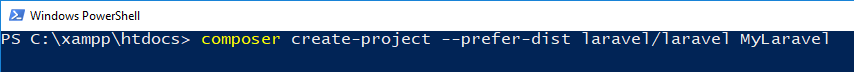
1. Mở cmd.exe
2. Đóng tất cả các cửa sổ windows lại ( bao gồm cả cmd.exe )
3. Mở lại cmd.exe
4. Đóng lại rồi logout ra khỏi windows , sau đó login lại.
5. Cuối cùng ta bật cmd.exe lên là xong.

2. Cài đặt Laravel

Truy cập thư mục C:\xampp\htdocs. Thực hiện tổ hợp phím Shift + chuột phải chọn “Open PowerShell window here” hoặc “Open Command window here” tùy theo máy.



Thực hiện câu lệnh composer create-project --prefer-dist laravel/laravel {Tên project Laravel}



Nhấn Enter đợi một khoảng thời gian để composer cài đặt Laravel phiên bản mới nhất.

Cấu trúc thư mục của Laravel sau khi cài đặt:

