TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN



PRJ – REPORT  
Laravel Report

Giảng Viên: Phan Thanh Nhuần

**Nhóm D**:

1.Trần Phong Nhã - 18211TT1000

2.Nguyễn Phương Tân - 18211TT2680

3.Trần Triều Thuận - 18211TT0530

Ngày 14/12/2020

MỤC LỤC

[I. Hướng Dẫn Cài Laravel 8.x 5](#_Toc59308472)

[1. Giới thiệu 5](#_Toc59308473)

[2. Hướng dẫn cài đặt và tạo project laravel 8.x 5](#_Toc59308474)

[**2.1 Cài đặt Wamp Server** 5](#_Toc59308475)

[**2.2 Cài đặt Composer** 7](#_Toc59308476)

[**2.3 Cách cài project laravel 8.x đơn giản** 15](#_Toc59308477)

[II. Tạo một gói Composer/Packagist cơ bản 19](#_Toc59308478)

[1. Tạo một kho lưu trữ với github 19](#_Toc59308479)

[2. Tạo Package 25](#_Toc59308480)

[III. Tạo Chức năng login thông qua thư viện jetstream 34](#_Toc59308481)

[**1 Tạo chức năng login thông qua email** 34](#_Toc59308482)

[**2. Tạo chức năng login thông qua email hoặc username** 42](#_Toc59308483)

**Mục Lục Hình**

[Hình 1: Biểu tượng Wamp Server 5](#_Toc58832146)

[Hình 2: Biểu tượng Wampserver khi khởi động 6](#_Toc58832147)

[Hình 3: Phiên bản PHP 7.3.12 7](#_Toc58832148)

[Hình 4: Giao diện trang web của composer 8](#_Toc58832149)

[Hình 5: Nhấn vào nút download composer 8](#_Toc58832150)

[Hình 6: Giao diện download composer 9](#_Toc58832151)

[Hình 7: Download file setup.exe 10](#_Toc58832152)

[Hình 8: Cửa sổ giao diện install 10](#_Toc58832153)

[Hình 9: Giao diện Installation Option 11](#_Toc58832154)

[Hình 10: Select Location composer setup 11](#_Toc58832155)

[Hình 11: Setting check your php 12](#_Toc58832156)

[Hình 12: Proxy setting composer 13](#_Toc58832157)

[Hình 13: Ready to install composer 13](#_Toc58832158)

[Hình 14: Hoàn tất cài composer 14](#_Toc58832159)

[Hình 15: Command Prompt và gõ composer 15](#_Toc58832160)

[Hình 16: Kết quả cài composer thành công 15](#_Toc58832161)

[Hình 17: Cửa sổ khi nhấn shift và chuột phải 16](#_Toc58832162)

[Hình 18: Giao diện Windows PowerShell 16](#_Toc58832163)

[Hình 19: Gõ câu lệnh tạo laravel 17](#_Toc58832164)

[Hình 20: Cài thành công laravel 17](#_Toc58832165)

[Hình 21: Folder demolaravel được tạo 18](#_Toc58832166)

[Hình 22: Giao diện laravel trên trình duyệt 18](#_Toc58832167)

[Hình 23: Nút tạo repo mới 19](#_Toc58832168)

[Hình 24: Nhập đầy đủ thông tin cần thiết và nhấn nút tạo 20](#_Toc58832169)

[Hình 25: Kho lưu trữ minh họa sau khi vừa tạo xong 21](#_Toc58832170)

[Hình 26: Lệnh clone một kho lưu trữ 22](#_Toc58832171)

[Hình 27: Copy project laravel có tên "blog" vào kho lưu trữ 22](#_Toc58832172)

[Hình 28: Đến đường dẫn của repo và gõ lệnh git add . 22](#_Toc58832173)

[Hình 29: Câu lệnh commit cam kết bắt buộc 23](#_Toc58832174)

[Hình 30: Câu lệnh push lên kho lưu trữ qua nhánh chính 24](#_Toc58832175)

[Hình 31: Kho lưu trữ đã được thêm thành công project laravel 24](#_Toc58832176)

[Hình 32: Form đăng ký tài khoản packagist 25](#_Toc58832177)

[Hình 33: Cách lấy API tooken để submit repo 26](#_Toc58832178)

[Hình 34: Tạo webhook 26](#_Toc58832179)

[Hình 35: Yêu cầu nhập lại mật khẩu 27](#_Toc58832180)

[Hình 36: Điều chỉnh thông tin cho webhook 28](#_Toc58832181)

[Hình 37: Webhook được tạo thành công 29](#_Toc58832182)

[Hình 38: Submit package 29](#_Toc58832183)

[Hình 39: Ckeck repo 30](#_Toc58832184)

[Hình 40: Lỗi chưa có file composer.json 30](#_Toc58832185)

[Hình 41: đặt file ở cùng cấp thư mục cha 31](#_Toc58832186)

[Hình 42: Ba câu lệnh để push lên git 32](#_Toc58832187)

[Hình 43: Check thành công 32](#_Toc58832188)

[Hình 44: Result submit 33](#_Toc58832189)

[Hình 45: Cài đặt thư viện jetstream 34](#_Toc58832190)

[Hình 46: Cài đặt gói hỗ trợ thư viện jetsteam 35](#_Toc58832191)

[Hình 47: Cài đặt npm 36](#_Toc58832192)

[Hình 48: Quá trình khởi chạy npm dev 36](#_Toc58832193)

[Hình 49: Hoàn tất khởi chạy npm dev 37](#_Toc58832194)

[Hình 50: Lỗi update table lên database 37](#_Toc58832195)

[Hình 51: Cập nhật số lượng ký tự cho email trong table create\_user 38](#_Toc58832196)

[Hình 52: Cập nhật số lượng ký tự email trong table create\_password\_resets 38](#_Toc58832197)

[Hình 53: Cập nhật số lượng ký tự cho table failed\_jobs 39](#_Toc58832198)

[Hình 54: Cập nhật số lượng ký tự cho table create\_session 39](#_Toc58832199)

[Hình 55: Cập nhật table create\_personal tránh xung đột 40](#_Toc58832200)

[Hình 56: Chọn tất cả các table và drop 40](#_Toc58832201)

[Hình 57: Chấp nhận drop tất cả các table 40](#_Toc58832202)

[Hình 58: Cập nhật tất cả table lên database 41](#_Toc58832203)

[Hình 59: Kích hoạt máy ảo laravel 41](#_Toc58832204)

[Hình 60: Chức năng login và register 41](#_Toc58832205)

[Hình 61: Thay đổi giá trị type và value 42](#_Toc58832206)

[Hình 62: Thông báo người dùng login thông qua username hoặc email 42](#_Toc58832207)

[Hình 63: Cập nhật giá trị đầu vào username hoặc email 43](#_Toc58832208)

[Hình 64: Kiểm điều kiện đầu vào 44](#_Toc58832209)

## I. Hướng Dẫn Cài Laravel 8.x

### 1. Giới thiệu

Laravel hiện nay là một trong những PHP framework mã nguồn mở và hoàn toàn miễn phí, tuy vậy cách cài đặt Laravel có phần hơi khó khăn với người dùng cơ bản, đặc biệt với những người mới vào học về lập trình PHP.

### 2. Hướng dẫn cài đặt và tạo project laravel 8.x

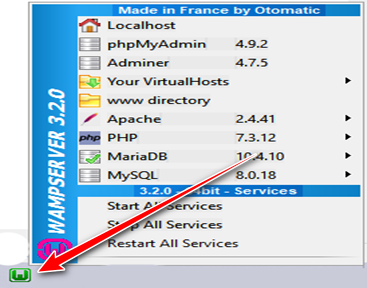
#### **2.1 Cài đặt Wamp Server**

Nếu các bạn chưa có Wampserver sẵn trong máy thì có thể vào link: <https://www.wampserver.com/en/download-wampserver/> để tải và cài đặt. Sau khi tải và cài đặt thành công vào click vào biểu tượng W (wampserver) để khởi động Wampserver.



Hình 1: Biểu tượng Wamp Server

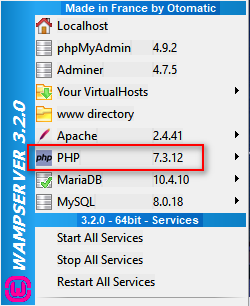
Sau khi click vào khởi động, bạn phải chờ ít phút. Trong lúc Wampserver chạy sẽ xuất hiện biểu tượng **W** ở phía dưới bên góc phải màn hình.



Hình 2: Biểu tượng Wampserver khi khởi động

Khi khởi động biểu tượng sẽ thay đổi màu từ Đỏ -> Cam ->Xanh, khi biểu tượng màu xanh như hình 8 là wamp đã khởi động thành công.

Ở đây các bạn chú ý phiên bản PHP các bạn phải sử dụng phiên bản >= 7.3.



Hình 3: Phiên bản PHP 7.3.12

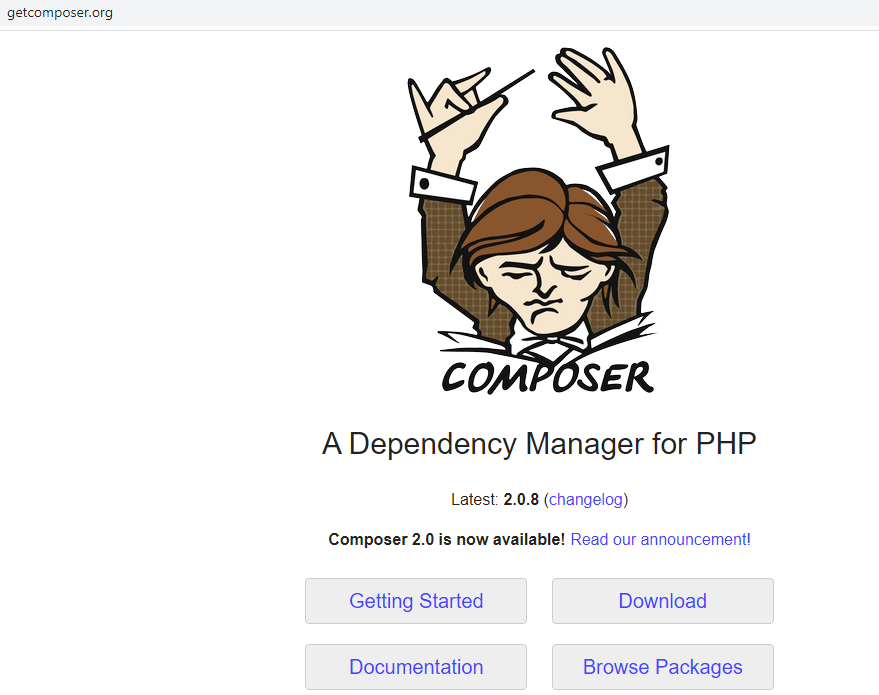
#### **2.2 Cài đặt Composer**

Composer là một trình quản lý cấp gói ứng dụng, thư viện cho ngôn ngữ lập trình PHP.

Trước tiên chúng ta phải tải về cài đặt composer trước rồi mới tiến hành cài đặt và tạo một project laravel.

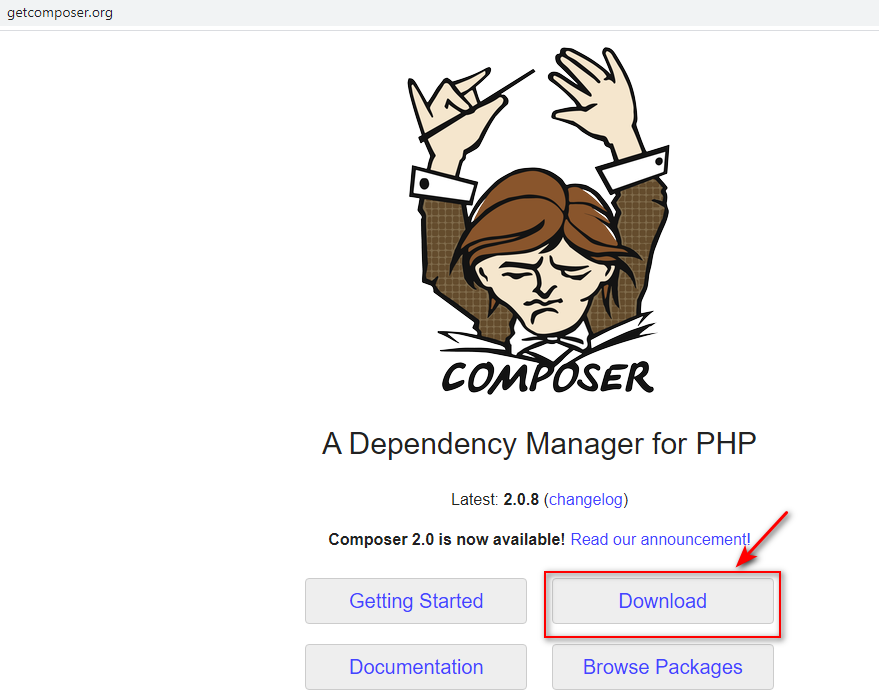
Bạn có thể tải composer qua link sau: <https://getcomposer.org/>

Khi các bạn truy cập vào đường link trên các bạn sẽ thấy giao diện như hình sau:



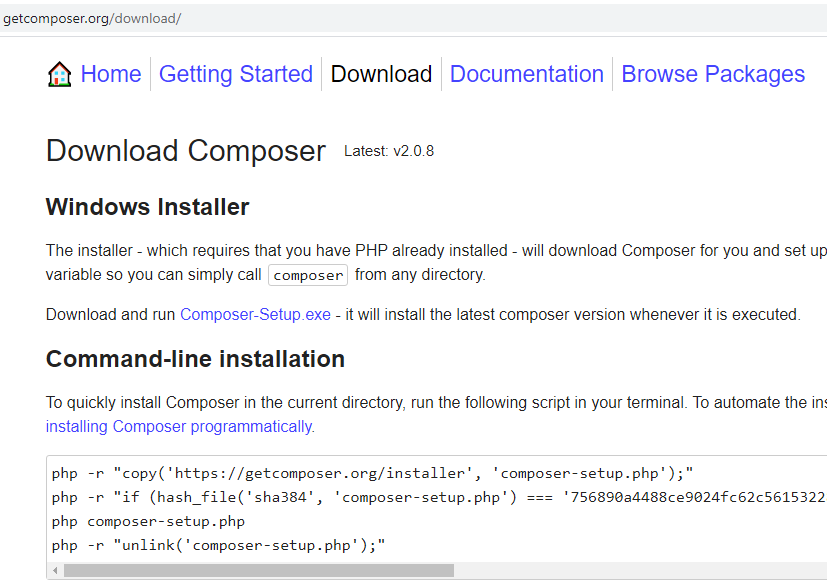
Hình 4: Giao diện trang web của composer

Sau đó nhấn vào nút **Download**



Hình 5: Nhấn vào nút download composer

Khi nhấn vào sẽ dẫn bạn đến trang web download, ở đây bạn có thể tải file **setup.exe** hoặc gõ lệnh.



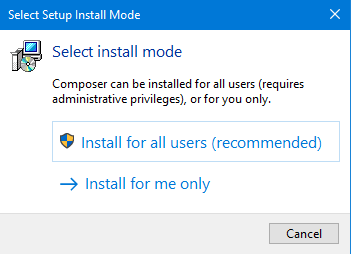
Hình 6: Giao diện download composer

Để tải file cài đặt bạn nhấn vào **Composer-Setup.exe**. Sau khi nhấn thì file cài đặt sẽ được tải xuống.



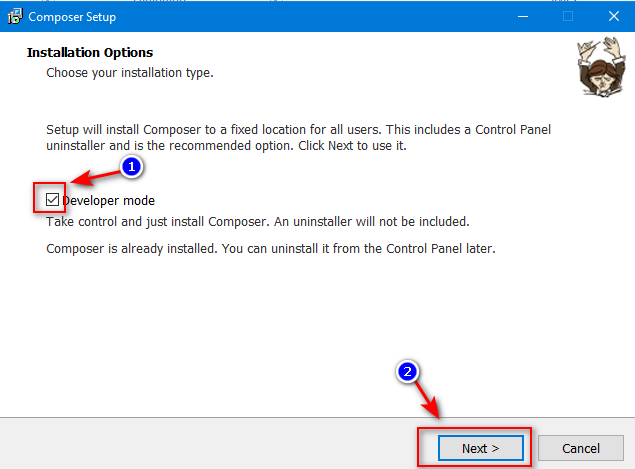
Hình 7: Download file setup.exe

Sau đó nhấn vào file đã download sẽ hiện ra cửa sổ cài đặt của composer:



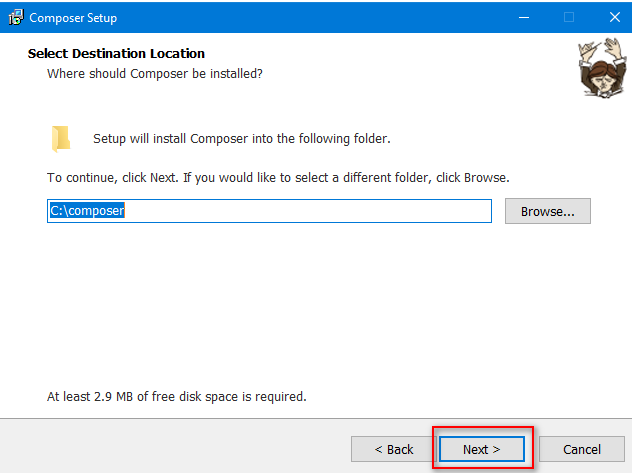
Hình 8: Cửa sổ giao diện install

Ở đây các bạn nhấn vào “**Install for all users”** để bắt đầu cài đặt.



Hình 9: Giao diện Installation Option

Các bạn có thể click vào ô trống **Developer mode** hoặc để trống điều được. Sau đó nhấn **Next**.

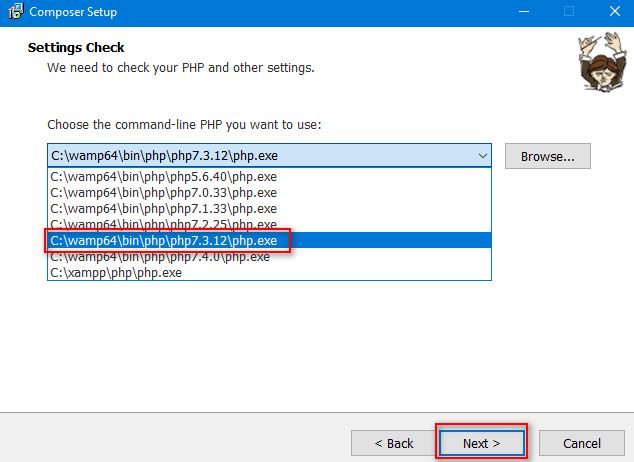


Hình 10: Select Location composer setup

Tùy chọn **Select Destination Location**, composer sẽ được cài theo đường dẫn.

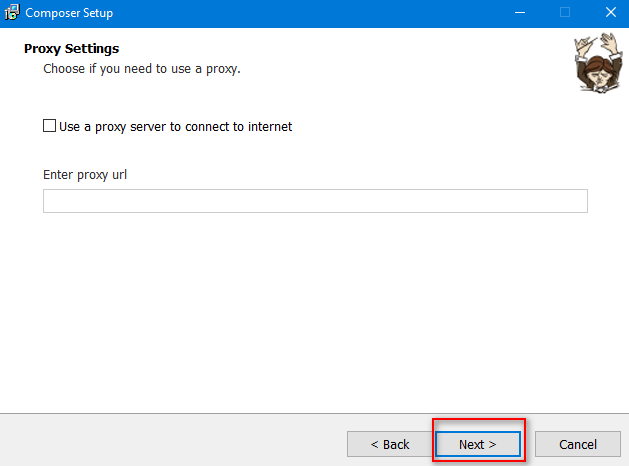
**Ví dụ:** theo hình thì composer sẽ cài theo đường dẫn: **C:\composer**.

Tiếp tục nhấn next để tiếp tục cài đặt.



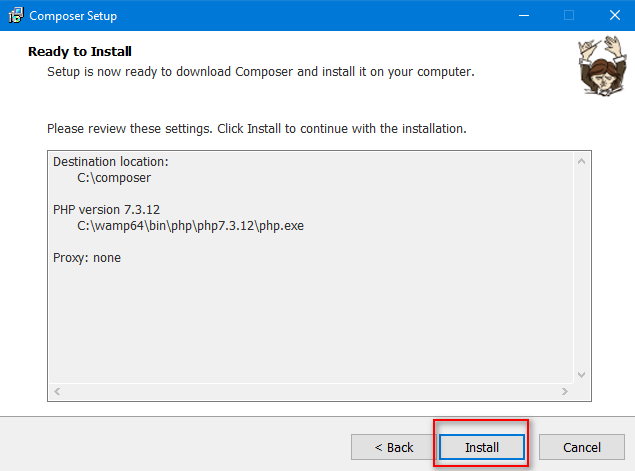
Hình 11: Setting check your php

Các bạn sẽ chọn phiên bản **php7.3.12** trở lên. Vì laravel 8.x trở lên **chỉ có thể** chạy trên nền **php >= 7.3**. Và tiếp tục nhấn next.



Hình 12: Proxy setting composer

Ở giao diện proxy setting các bạn để trống và tiếp tục nhấn next.

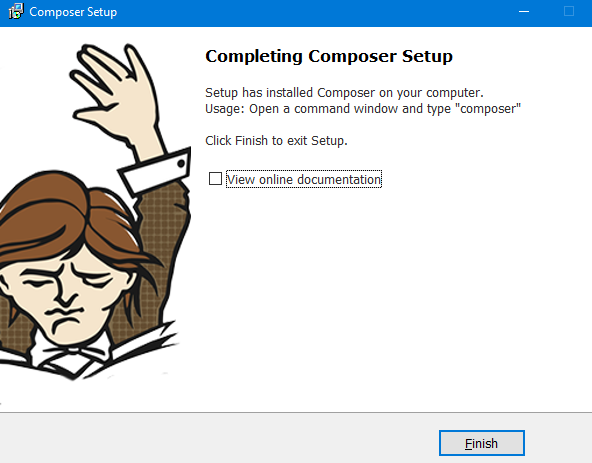


Hình 13: Ready to install composer

Ở giao diện **Ready to install** cho người cài đặt check lại các phần đã cài trước đó. Nếu không có thay đổi gì thì bạn nhấn **Install**.

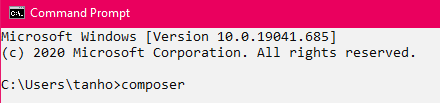
Hệ thống sẽ tự download composer cho bạn và yêu cầu bạn phải kết nối internet để cài đặt.

Sau khi cài xong như hình dưới và bạn nhấn **Finish** là hoàn tất việc cài đặt composer.



Hình 14: Hoàn tất cài composer

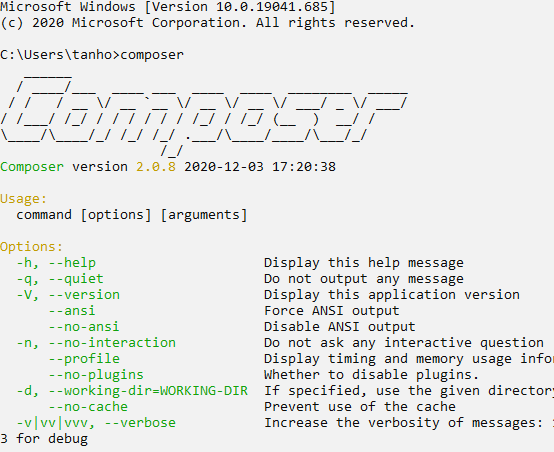
Để check xem composer đã được cài chưa bạn có thể gọi **Command Prompt** lên và gõ lệnh composer.



Hình 15: Command Prompt và gõ composer

Sau đó nhấn Enter.

Nếu cài thành công bạn sẽ thấy như hình 13 bên dưới.



Hình 16: Kết quả cài composer thành công

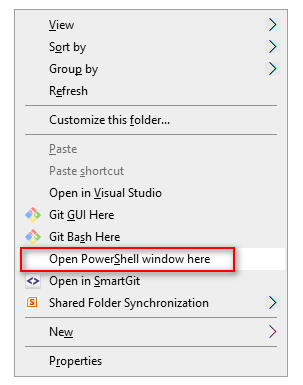
#### **2.3 Cách cài project laravel 8.x đơn giản**

Để cài một project laravel 8.x bạn có thể tham khảo qua trang web : <https://laravel.com/docs/8.x>

Ỡ đây chúng ta sẽ cài project laravel bằng cách gõ lệnh:

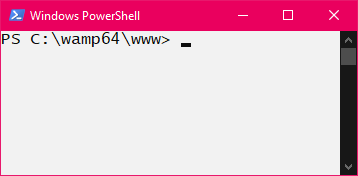
**Bước 1:**

Các bạn vào thư mục theo đường dẫn **C:\wamp64\www**. Nhấn **Shift + click** **chuột phải**. Sau đó sẽ hiện ra một cửa sổ các bạn chọn **Open PowerShell window here**.



Hình 17: Cửa sổ khi nhấn shift và chuột phải

Sau khi nhấn thì bạn sẽ nhận được một giao diện như sau:



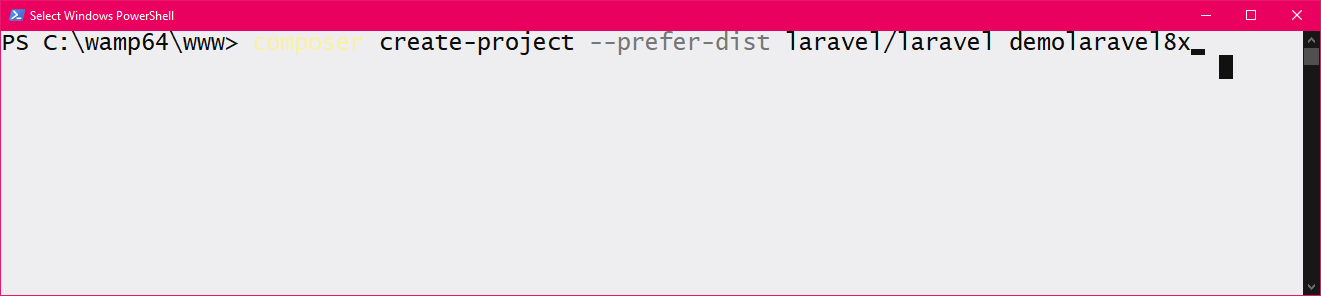
Hình 18: Giao diện Windows PowerShell

Để cài tự động laravel về thì các bạn gõ câu lệnh như sau vào **Window PowerShell**:

composer create-project --prefer-dist laravel/laravel blog

Trong đó bạn để ý chữ “**blog**” ở cuối dòng. Các bạn có thể đặt tên cho project theo ý mình (vd: demolaravel8x)

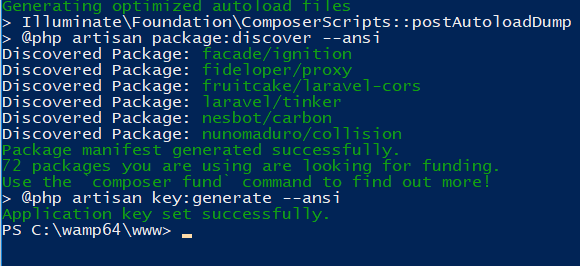
composer create-project --prefer-dist laravel/laravel demolaravel8x



Hình 19: Gõ câu lệnh tạo laravel

Sau đó nhấn Enter. Thì hệ thống sẽ tự động download các file, package cần thiết cho laravel.

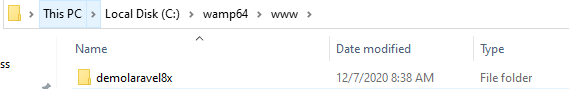
**Yêu cầu:** Các bạn phải kết nối internet trước khi tải laravel.



Hình 20: Cài thành công laravel

Khi nào các bạn thấy cửa sổ **Window PowerShell** như hình trên thì bạn đã thành công cài đặt một project laravel rồi.

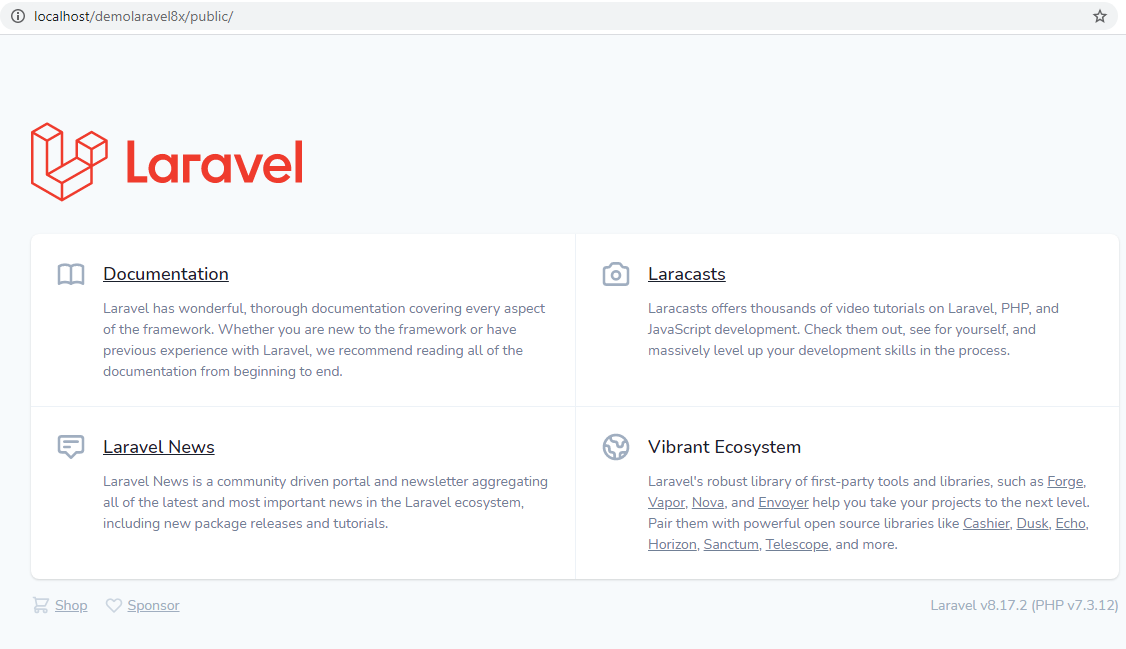
Bạn vào đường dẫn **C:\wamp64\www** sẽ thấy 1 folder có tên **demolaravel8x** đã được tạo và bên trong là các file cần thiết cho laravel 8.x.



Hình 21: Folder demolaravel được tạo

Để kiểm tra các bạn vào trình duyệt internet gõ đường dẫn như sau: **localhost/demolaravel8x/public**

Nếu bạn đến được giao diện như hình sau:



Hình 22: Giao diện laravel trên trình duyệt

Vậy là bạn đã cài thành công một project đơn giản với laravel 8.x.

## II. Tạo một gói Composer/Packagist cơ bản

Sau khi tạo một package, bạn có thể tạo một dự án mới hoặc cập nhật một dự án đã tồn tại để sử dụng Composer. Với phương pháp này, bạn có thể tạo các dự án phức tạp và chia sẻ chúng với các lập trình viên khác trên toàn thế giới.

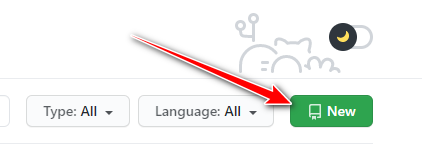
### 1. Tạo một kho lưu trữ với github

Trước tiên, chúng ta sẽ tạo một **kho lưu trữ rỗng** trên **github** và sau đó ta sẽ đưa source code của **project laravel 8.x** mà ta đã tạo ở phần trên vào.

Ở bài hướng dẫn này sẽ chỉ cách tạo một **kho lưu trữ** trực tiếp trên github website.

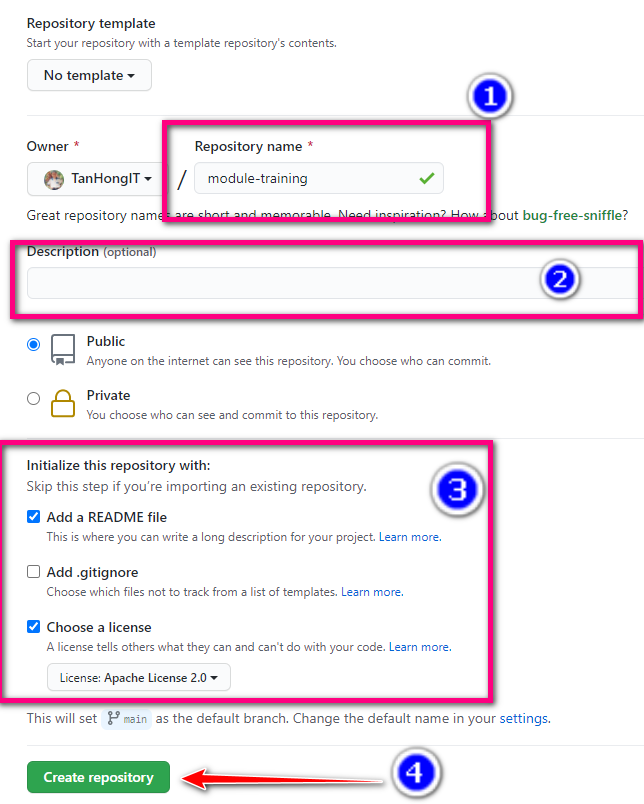
Các bạn hãy đăng nhập tài khoản git của mình và truy cập vào đường dẫn sau: <https://github.com/new>

Hoặc bạn có thể nhấn vào nút **New** có background màu xanh tại trang danh sách các **repositories** để tạo một project mới.



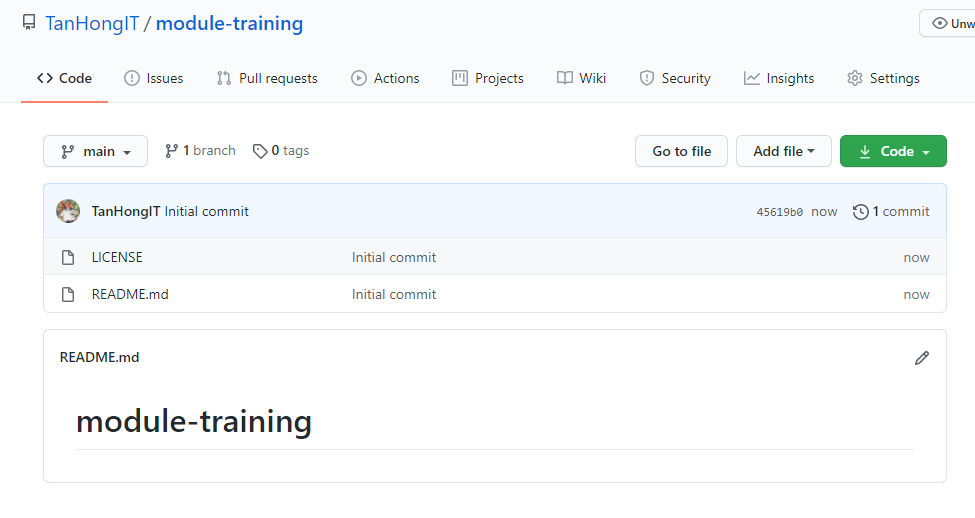
Hình 23: Nút tạo repo mới

Tiếp theo bạn hãy điền thông tin tương ứng và nhấn vào button Create repository.



Hình 24: Nhập đầy đủ thông tin cần thiết và nhấn nút tạo

Sau khi chờ đợi bạn sẽ có một kho lưu trữ rông như hình sau:



Hình 25: Kho lưu trữ minh họa sau khi vừa tạo xong

Hình trên là ví dụ về một kho lưu trữ mà mình đã tạo với nhánh chính có tên là “**main**”.

Bước tiếp theo chúng ta sẽ Clone kho lưu trữ trên về máy tính.

**Lưu ý:**

Trước khi thực hiện clone về máy bạn cần phải cài Git vào máy tính. Các bạn có thể tải về và cài đặt theo đường link sau: <https://git-scm.com/downloads>

Các bạn vào thư mục theo đường dẫn **C:\wamp64\www**. Nhấn **Shift + click** **chuột phải**. Sau đó sẽ hiện ra một cửa sổ các bạn chọn **Open Command Prompt here**.

Sau khi mở lên ban hãy thực hiện theo lệnh:

**git clone link\_project**

Trong đó **link\_project** là đường dẫn đến kho lưu trữ mà bạn vừa mới tạo xong.

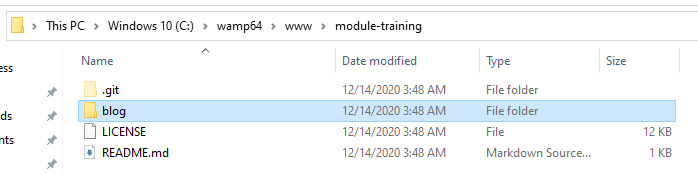
**Ví dụ:** git clone https://github.com/TanHongIT/module-training



Hình 26: Lệnh clone một kho lưu trữ

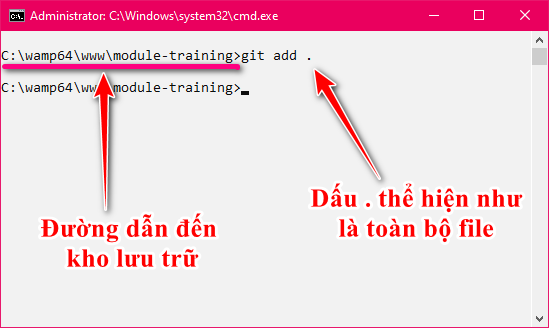
Sau khi đã clone về bạn mở hãy copy source code **project laravel 8.x** mà ta đã tạo vào.

Ở hình dưới mình đã copy project có tên là blog vào kho lưu trữ.



Hình 27: Copy project laravel có tên "blog" vào kho lưu trữ

Sau khi copy xong, quay lại màn hình của sổ **Command Prompt**, bạn hãy thực hiện câu lệnh **git add .** để thêm toàn bộ thay đổi vào bộ đệm của git dưới máy tính.

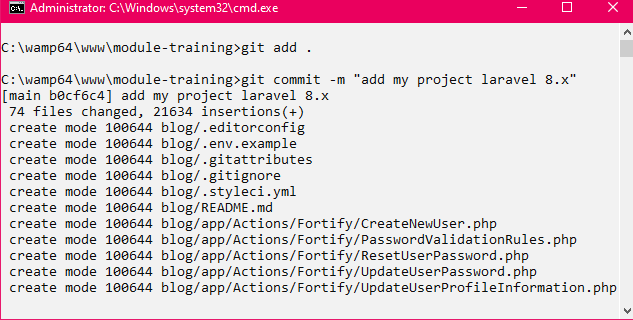


Hình 28: Đến đường dẫn của repo và gõ lệnh git add .

Tiếp theo thực hiện câu lệnh **git commit -m “your add message”**

Phần ký tự nằm trong dấu ngoặc kép là phần tin nhắn cam kết mà bạn muốn upload lên kho lưu trữ. Bạn thay đổi lại theo ý muốn và sau đó thì nhấn **Enter**.

**Ví dụ:** git commit -m “add my project laravel 8.x”



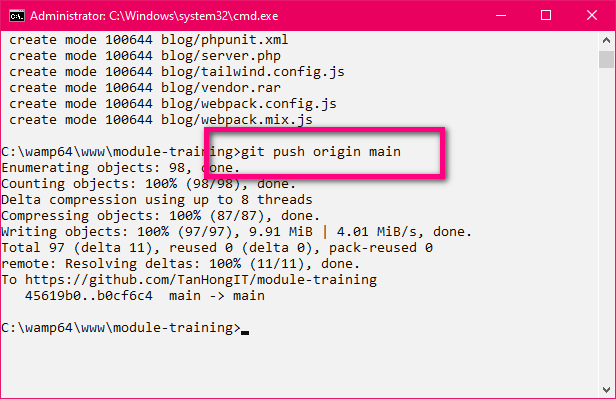
Hình 29: Câu lệnh commit cam kết bắt buộc

Tiếp theo chúng ta sẽ thực hiện câu lệnh push lên github bằng câu lệnh:

**git push origin main**

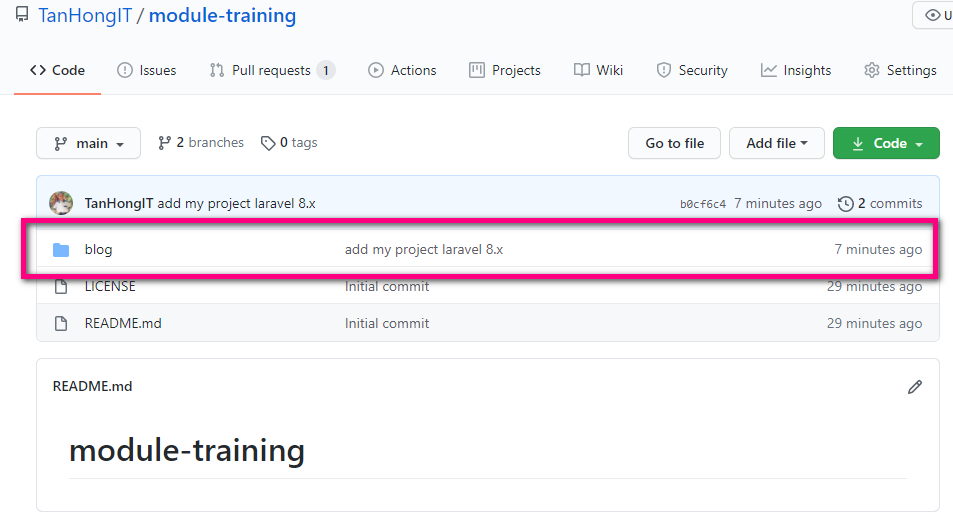
**Trong đó:** từ khóa “**main**” là tên nhánh chính của kho lưu trữ.

Bạn phải kiểm tra tên nhánh chính của kho lưu trữ và thay đổi lại cho phù hợp nhé. Khi đã thay đổi đúng tên thì nhấn Enter để tiến hành push. Điều quan trọng là máy tính của bạn phải có kết nối internet nhé.



Hình 30: Câu lệnh push lên kho lưu trữ qua nhánh chính

Sau đó chúng ta hãy lên đường dẫn của kho lưu trữ trên trình duyệt để kiểm tra lại.



Hình 31: Kho lưu trữ đã được thêm thành công project laravel

Chúng ta thấy rằng source code đã được upload lên github thành công với dòng commit là “add my project laravel 8.x”

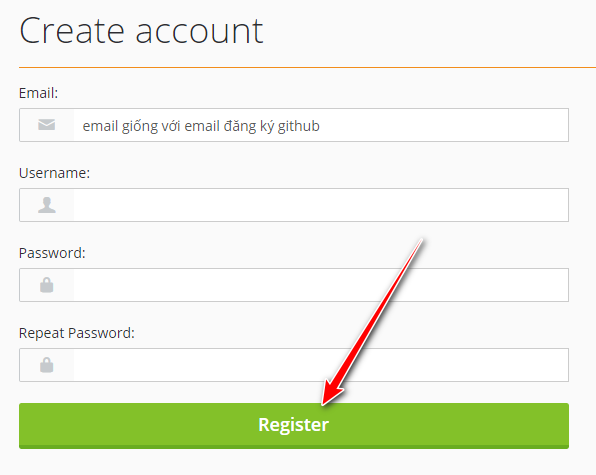
### 2. Tạo Package

Chúng ta sẽ tạo package với **github** kết nối với **packagist**

Chúng ta cần có một tài khoản trên trang **packagist.org**. Nếu bạn chưa có hãy bắt đầu tạo một tài khoản mới với email tạo phải trùng với email đã đăng ký tài khoản github.

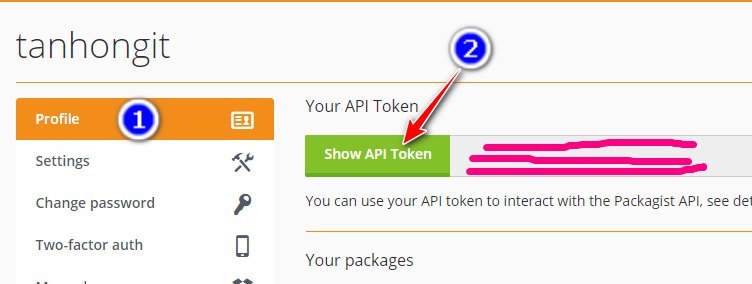
Vài link <https://packagist.org/register/> để tiến hành đăng ký tài khoản.

Nhập đầy đủ thông tin và sau đó nhấn vào button **Register**.



Hình 32: Form đăng ký tài khoản packagist

Sau khi tạo xong tài khoản, bạn hãy vào phần **Profile** và nhấn vào **Show API Token**.

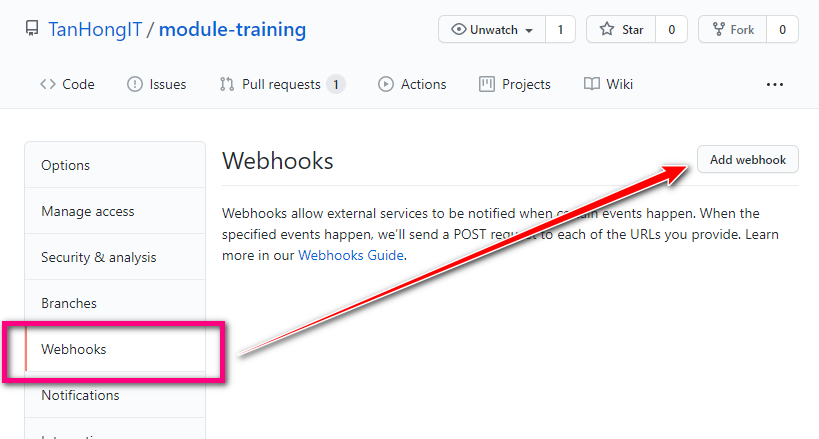


Hình 33: Cách lấy API tooken để submit repo

Khi này API sẽ được hiển thị ở thẻ input kế bên cạnh, bạn hãy bảo mật cẩn thận API này đừng cho bất ý ai biết.

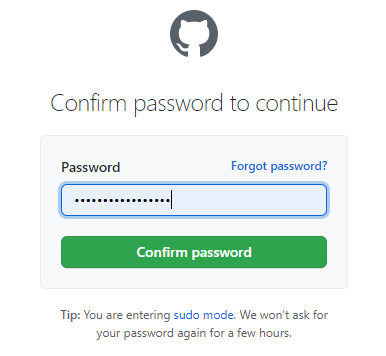
Quay lại trang chứa kho lưu trữ của bạn và đi qua tab **Setting**s.

Sau đó vào mục **Webhooks** và nhấn vào button “**Add webhook”**.



Hình 34: Tạo webhook

Bước tiếp theo github sẽ yêu cầu bạn nhập lại mật khẩu để xác nhận bảo mật hành động.



Hình 35: Yêu cầu nhập lại mật khẩu

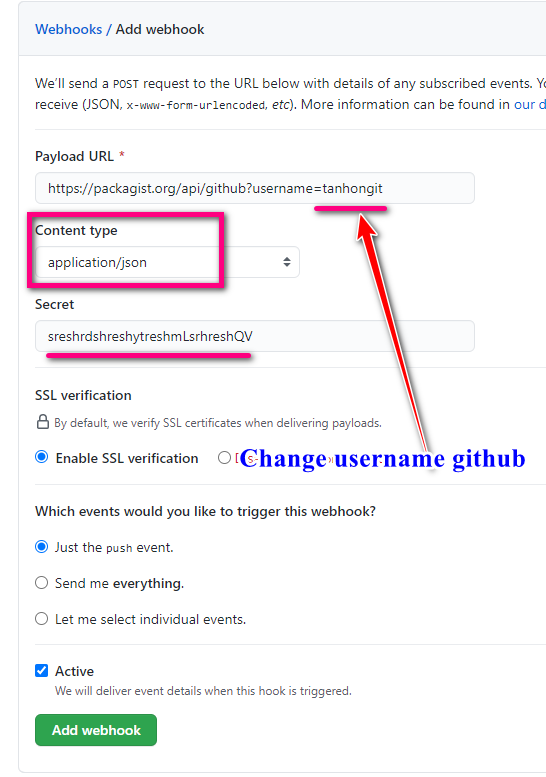
Tại đây bạn hãy thay đổi lại **Payload URL** là https://packagist.org/api/github?username=tanhongit

Bạn nhớ đổi usernane lại thành username tài khoản github của các bạn nhé.

Về **Content type**, bạn chuyển nó về: application/json

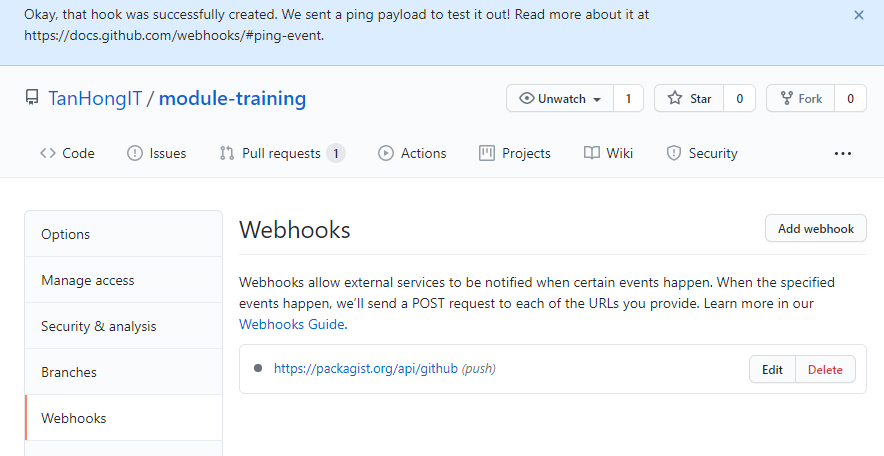
Ở trường nhập **Secret**, bạn hãy dán đoạn API token mà nãy chúng ta vừa lấy bên **packagist** vào.

Sau khi đã hoàn tất các yêu cầu trên thì ta nhấn vào nút **Add webhook**.



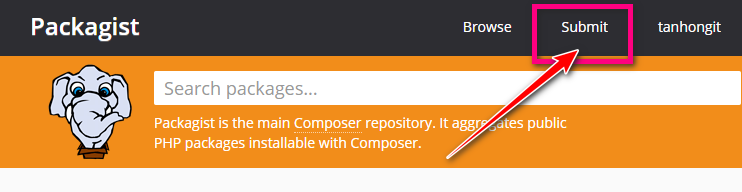
Hình 36: Điều chỉnh thông tin cho webhook

Chúng ta sẽ được một webhook trong kho lưu trữ như sau:



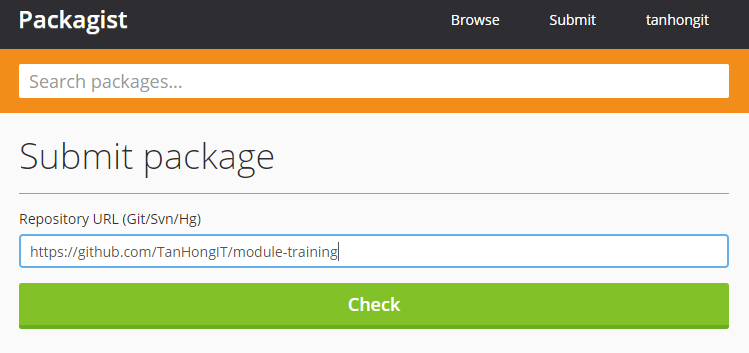
Hình 37: Webhook được tạo thành công

Giờ chúng ta lại quay sang **packagist.org** và nhấn vào tab **Submit**.



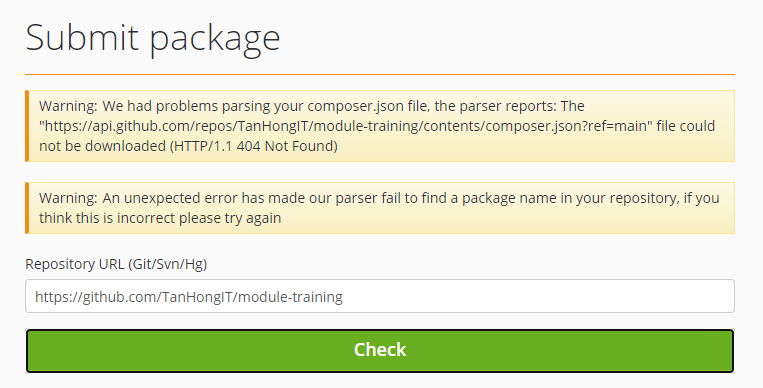
Hình 38: Submit package

Tiến hành paste đường dẫn của kho lưu trữ vào trường nhập liệu và nhấn nút Check.



Hình 39: Ckeck repo

Chắc chắn khi này chúng ta sẽ thấy thông báo lỗi bởi bì chúng ta chưa tạo file **composer.json** để cho **packagist.org** hiểu được.



Hình 40: Lỗi chưa có file composer.json

Tiến hành tạo một file **composer.json** có nọi dung như sau:

{

    "require": {

        "vendor/package": "1.3.2",

        "vendor/package2": "1.\*",

        "vendor/package3": "^2.0.3",

        "php": ">=7.3",

        "another-vendor/package": "1.\*"

    },

    "name": "tanhongit/laravel-training-module",

    "description": "Training Project Laravel Module",

    "keywords": [

        "tanhongit"

    ],

    "authors": [

        {

            "name": "tanhongit",

            "email": "phuongtan12357nguyen@gmail.com"

        }

    ],

    "license": "MIT",

    "autoload": {

        "psr-4": {

            "tanhongit\\module\\": "/"

        }

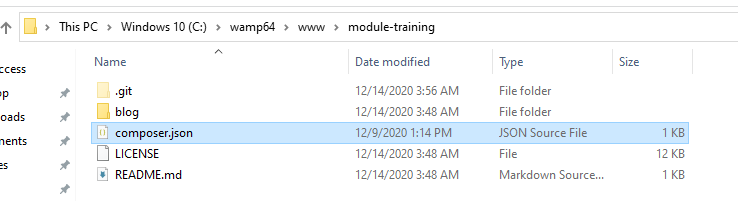
    },

    "minimum-stability": "dev"

}

Bạn hãy sửa lại thông tin trong đoạn code cho phù hợp với kho lưu trữ của bạn.

Bạn lưu file này nằm ở cấp thư mục cha trong kho lưu trữ.



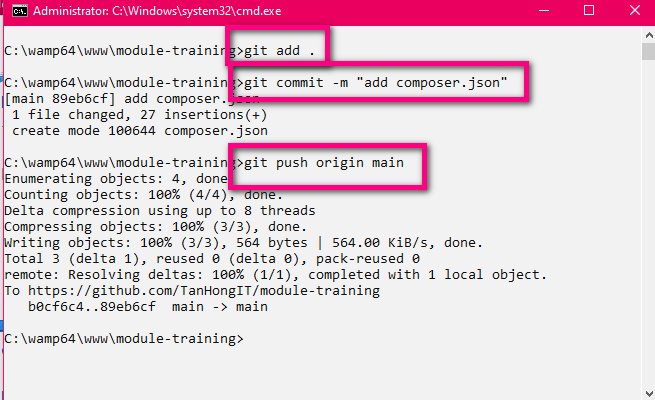
Hình 41: đặt file ở cùng cấp thư mục cha

Sau đó ta thực hiện 3 câu lệnh huyền thoại lần lượt là:

git add .

git commit -m “tin nhắn cam kết”

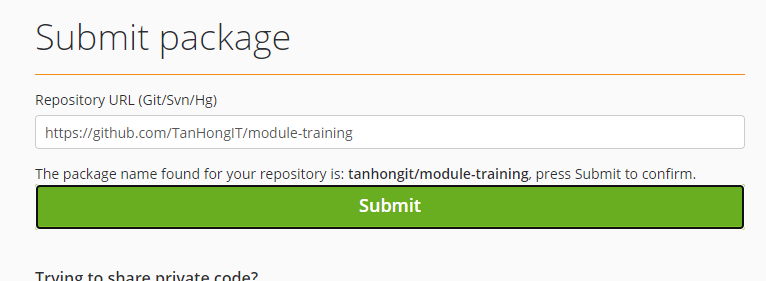
git push origin nhánh\_chính



Hình 42: Ba câu lệnh để push lên git

Cuối cùng, sau khi đã push lên github xong, ta tiến hành check lại một lần nữa.

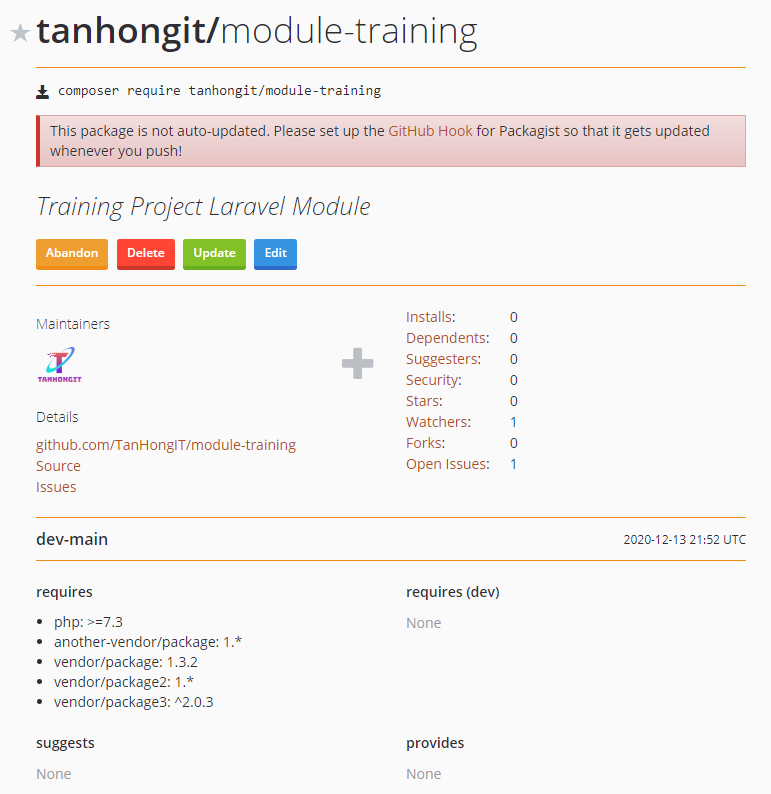
Nếu bạn thấy có thông báo như hình dưới thì bạn đã kiểm tra thành công.



Hình 43: Check thành công

Bây giờ bạn hãy nhấn vào nút **Submit** để kiến hành xác nhận.

Sau khi đã nhấn **Submit** bạn sẽ có một trang hiển thị thông tin package giống như hình bên dưới.



Hình 44: Result submit

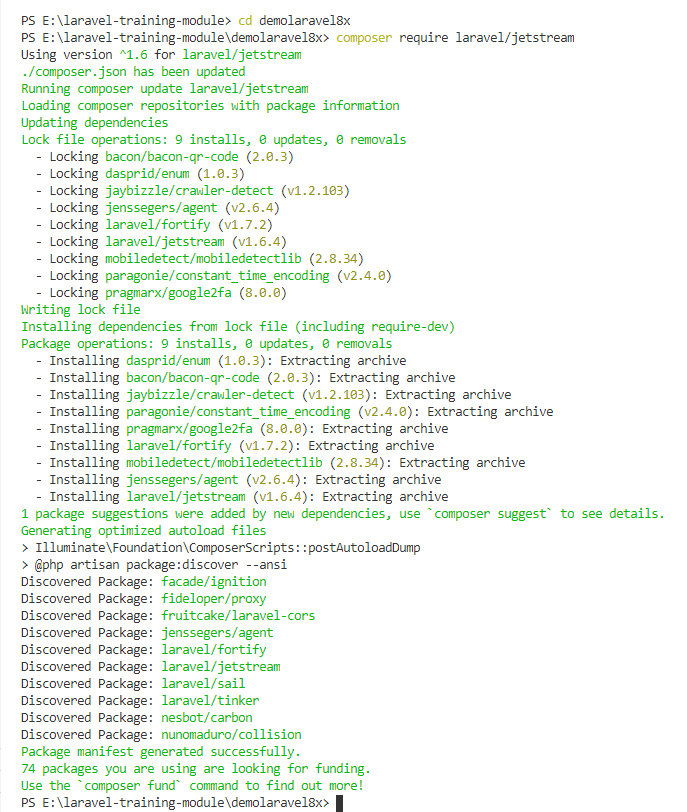
## III. Tạo Chức năng login thông qua thư viện jetstream

#### **1 Tạo chức năng login thông qua email**

Sau khi đã tạo thành công project cơ bản với laravel và bây giờ sẽ tiếp tục thêm tính năng login, register thông qua thư viện hỗ trợ laravel mang tên **jetstream** với câu lệnh như sau:

**cd demolaravel8x**

**composer require laravel/jetstream**

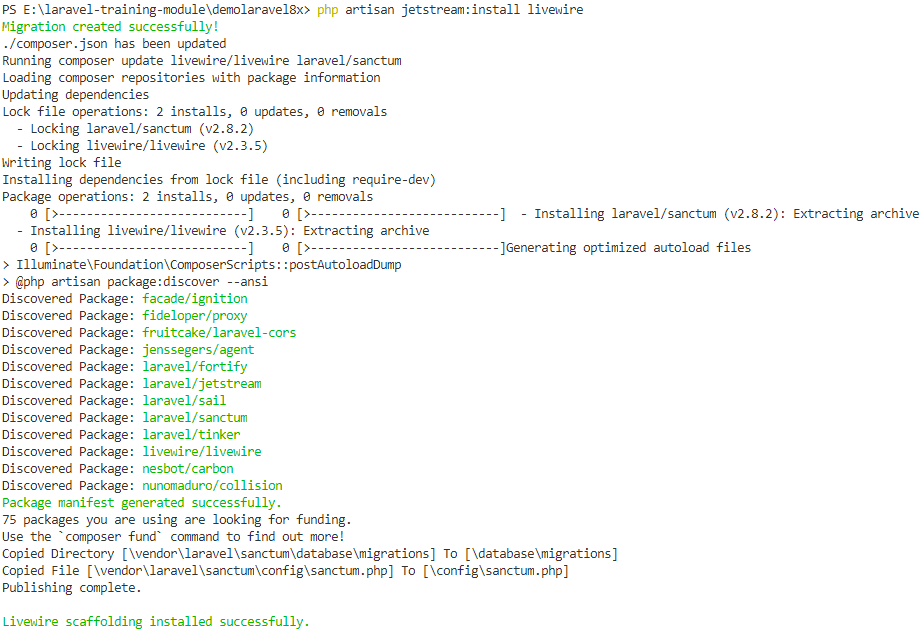


Hình 45: Cài đặt thư viện jetstream

Trong đó, **demolaravel8x** là tên đường dẫn đến tên project laravel mà bạn đã tạo.

Sau khi đã cài đặt thư viện **jeatstream** và được kết quả như hình trên tiếp tục cài đặt gói hỗ trợ của thư viện jetstream thông qua câu lệnh:

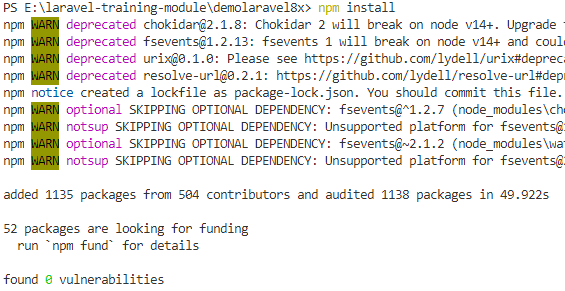
**php artisan jetstream:install livewire**



Hình 46: Cài đặt gói hỗ trợ thư viện jetsteam

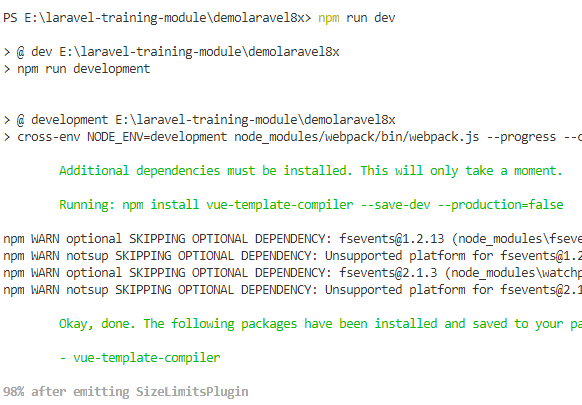
Sau khi cài đặt thành công lúc này bạn sẽ cần phải cài đặt thêm **npm** với câu lệnh như sau:

**npm install** và sẽ được kết quả như hình dưới đây:

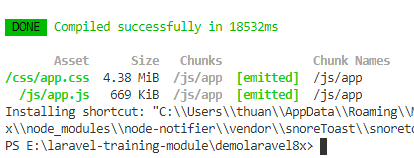


Hình 47: Cài đặt npm

Sau khi hoàn thành cài đặt **npm** và đạt được kết quả như hình (số lượng packages có thể thay đổi tùy vào số lượng package cần được cài đặt và cập nhất trong máy) tiến hành chạy câu lệnh: **npm run dev**

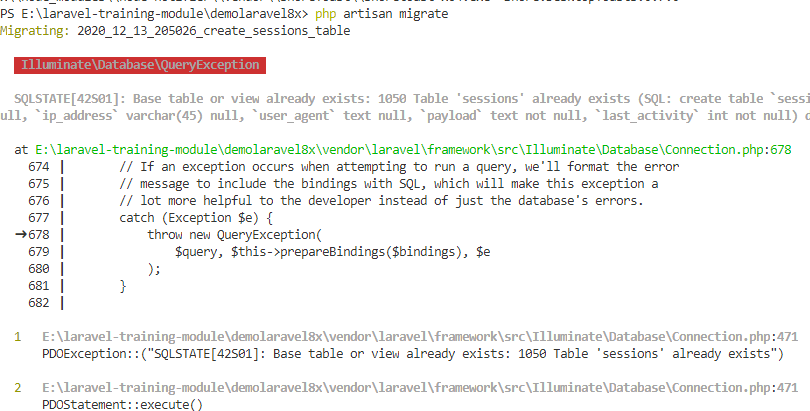


Hình 48: Quá trình khởi chạy npm dev



Hình 49: Hoàn tất khởi chạy npm dev

Sau khi cài đặt và thực thi câu lệnh **npm run dev** sẽ tiếp tục chạy lệnh: **php artisan migrate** hoàn tất việc cập nhật các table lên database. Tuy nhiên khi chạy bạn sẽ phát hiện lỗi như hình dưới đây:

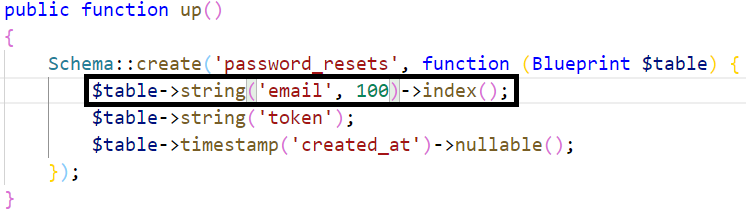
****

Hình 50: Lỗi update table lên database

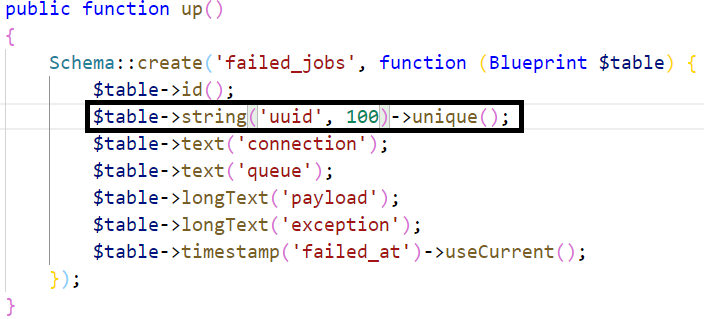
Vào file migration theo đường dẫn: **demolaravel8x/database/migrations/** tìm đến file **create\_user\_table.php** tiến hành cập nhật số lượng ký tự cho email trong các table theo các hình dưới đây:



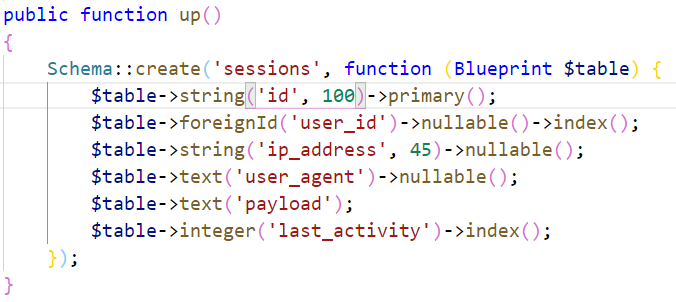
Hình 51: Cập nhật số lượng ký tự cho email trong table create\_user



Hình 52: Cập nhật số lượng ký tự email trong table create\_password\_resets

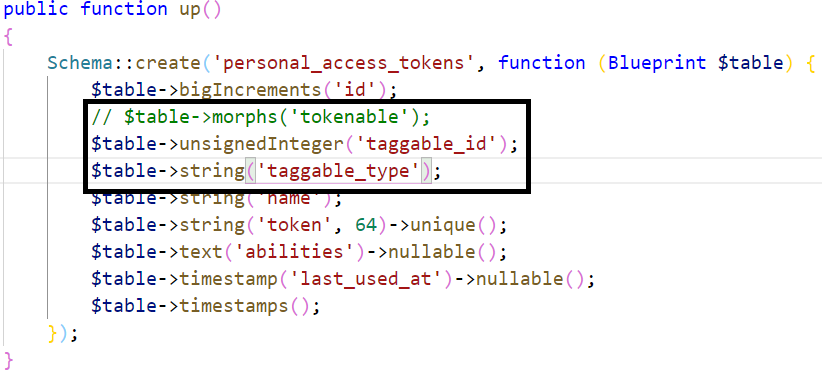


Hình 53: Cập nhật số lượng ký tự cho table failed\_jobs



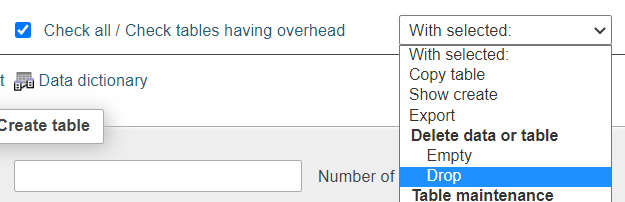
Hình 54: Cập nhật số lượng ký tự cho table create\_session

Sau khi đã chỉnh sửa các file liên quan đến ký tự ta sẽ tiếp tục chỉnh sửa trong file **create\_personal\_access\_tokens\_table.php** để tránh tình trạng xung đột giữa các table với nhau nên sẽ thêm một cột mới với tên là **taggable**\_**id** vó kiểu dữ liệu là số nguyên không dấu (**unsignedInteger**) theo hình dưới đây:

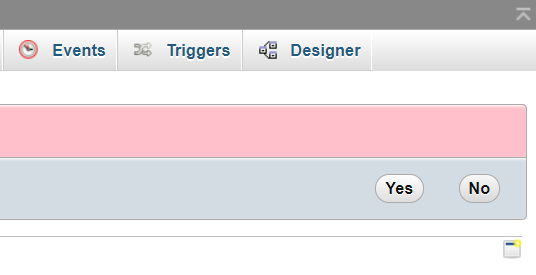


Hình 55: Cập nhật table create\_personal tránh xung đột

Sau khi cập nhật các ký tự cho các table cần thiết, ta tiến hành truy cập phpmyadmin và chọn drop các table cũ trong database và sau đó chọn **yes**:

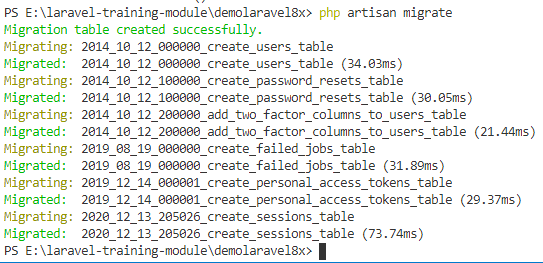


Hình 56: Chọn tất cả các table và drop



Hình 57: Chấp nhận drop tất cả các table

Tiến hành cập nhật lại file database thông qua câu lệnh: **php artisan migrate** sẽ được kết quả như hình dưới đây:



Hình 58: Cập nhật tất cả table lên database

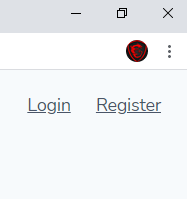
Sau khi đã cập nhật thành công file database cho project và giờ ta sẽ tiến hành chạy câu lệnh dùng để kích hoạt máy ảo của laravel như sau:

**php artisan serve**



Hình 59: Kích hoạt máy ảo laravel

Sau khi truy cập vào máy ảo của laravel bạn sẽ thấy **login** và **register** vừa được cập nhật tại góc trên cùng bên phải và tiến hành đăng ký, đăng nhập bình thường:

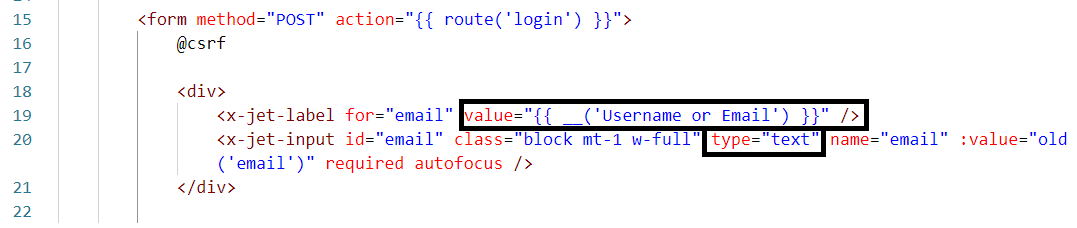


Hình 60: Chức năng login và register

#### **2. Tạo chức năng login thông qua email hoặc username**

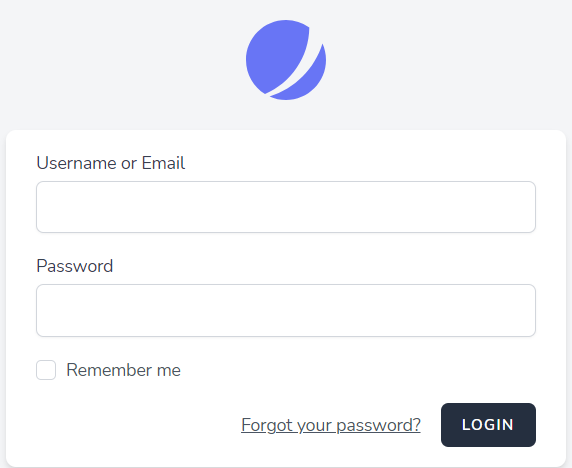
Để có thể đăng nhập thông qua email hoặc username ta sẽ phải đưa **type** của field nhập liệu trong form từ dạng **email** về dạng **text** (nằm trong file **login.blade.php**) theo đường dẫn như sau:

**\resources\views\auth\login.blade.php**

****

Hình 61: Thay đổi giá trị type và value

Tại đoạn value chỉ đơn thuần là xuất ra câu thông báo Username or Email cho người dùng nhận biết như hình dưới đây:



Hình 62: Thông báo người dùng login thông qua username hoặc email

Sau khi đã chỉnh sửa **type input** thì ta sẽ tiếp tục chỉnh sửa trong **CORE(vendor)** nhận giá trị username hoặc là email.

**LƯU Ý:**

Vì đây là **core** của cả framework laravel nên sẽ cực kỳ nguy hiểm nếu bạn không biết bạn đang làm gì vì điều này có thể làm framework **hoạt động không bình thường** hoặc **không thể hoạt động.**

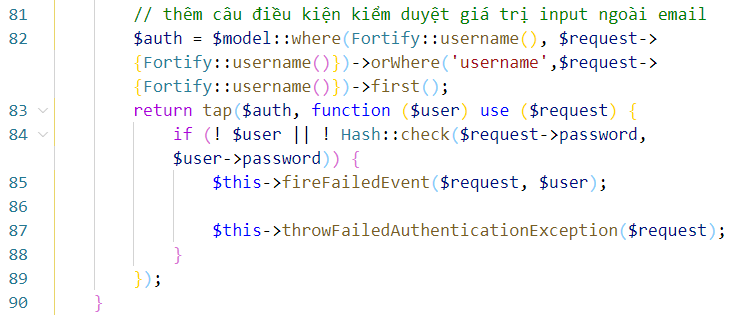
Để có thể gọi là tạm thời biết được bạn đang chỉnh sửa gì trong core thì hãy tìm đến file của git có tên là **.gitignore** và xóa dòng **/vendor** để git nhận và cho biết đã và đang chỉnh sửa gì, file nào trong **core.**

Sau khi đã xóa dòng **/vendor** ta sẽ tiến hành chỉnh sửa như sau:

Tìm đến file theo đường dẫn: **\vendor\laravel\fortify\src\Actions\RedirectIfTwoFactorAuthenticatable.php**

Khóa câu lệnh cũ bằng cách comment code thông qua tổ hợp phím **Ctrl + Shift.**

Và thực hiện cập nhật chức năng như hình dưới đây:



Hình 63: Cập nhật giá trị đầu vào username hoặc email

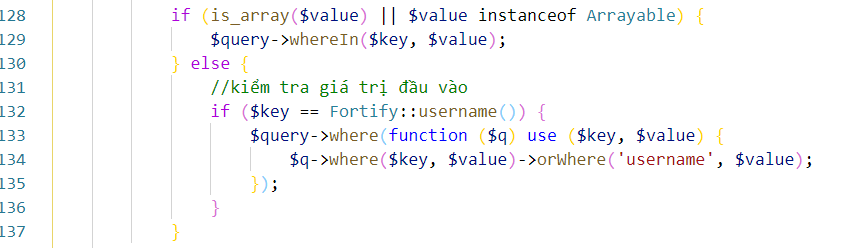
Ta sẽ tạo một biến mới có tên là **$auth** và cho nhận giá trị nhập vào là username hoặc email thông qua: **orWhere**. Và truyền vào hàm **tap(),** hiện tại chỉ dừng ở mức tinh chỉnh cho việc login bằng username hoặc email nên chúng ta sẽ không đi sâu vào cách vận hành của thư viện **jetstream**.

Tiếp đến ta sẽ truy cập vào file theo đường dẫn:

**\vendor\laravel\framework\src\Illuminate\Auth\EloquentUserProvider.php**

Đừng quên comment lại code mặc định của core trước khi chỉnh sửa code. (Dễ dàng quay lại code cũ nếu có trở ngại)

Tìm đến hàm **retrieveByCredentials()** Tại hàm **else** ta sẽ tiến hành cập nhật lại giá trị vào là username hoặc email như hình dưới đây:



Hình 64: Kiểm điều kiện đầu vào

Nếu trong **else** chỉ có **$q->where($key, $value)** thì giá trị nhận vào mặc định sẽ là **email** do **thư viện định nghĩa từ trước**, vậy nên lúc này ta sẽ cần điều kiện **if** nhận giá trị đầu vào và tiến hành xây dựng hàm lấy ra giá trị **username** hoặc **email** do người dùng nhập vào.