

# CÔNG NGHỆ WEB AN TOÀN

## Bài 5.2 Khung phát triển ứng dụng web

1

Mô hình MVC

2

Khung phát triển ứng dụng web

3

Giới thiệu Laravel

4

Ứng dụng CRUD với Laravel

# Mục tiêu bài học

---

1. Hiểu được bản chất, đặc điểm của mẫu phát triển ứng dụng web MVC
2. Có khả năng đánh giá, lựa chọn, sử dụng khung phát triển ứng dụng web

1

Mô hình MVC

2

Khung phát triển ứng dụng web

3

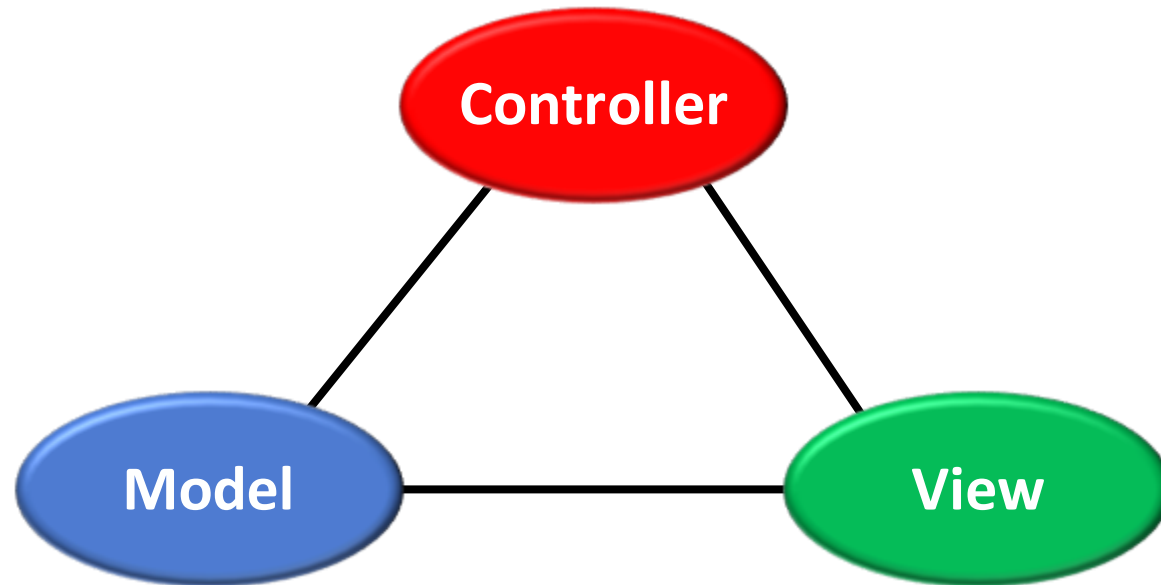
Giới thiệu Laravel

4

Ứng dụng CRUD với Laravel

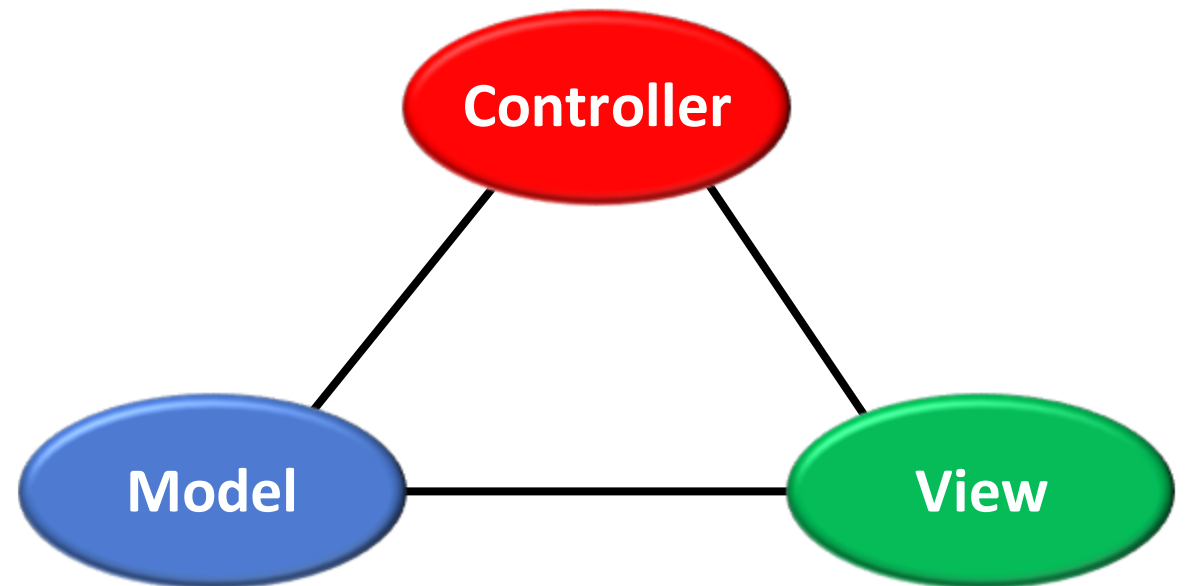
# Model-View-Controller

- MVC là một mẫu thiết kế trong phát triển ứng dụng web
- MVC giúp tách biệt phần thể hiện thông tin với cơ chế xử lý thông tin.



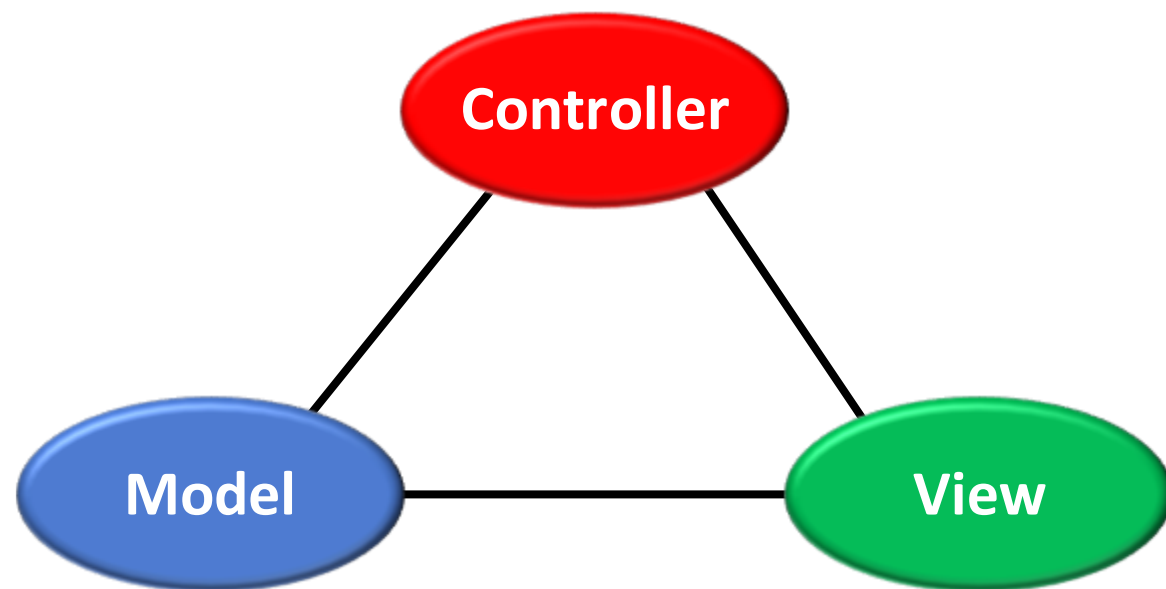
# Model

- **Model** trực tiếp thực hiện các tác vụ trên dữ liệu, thường là thao tác với database.
- Nói cách khác, **Model** chịu trách nhiệm bảo trì dữ liệu.
- Nhìn chung, **Model** có tính tự trị cao, nó không cần biết về sự tồn tại của **Controller** và **View**



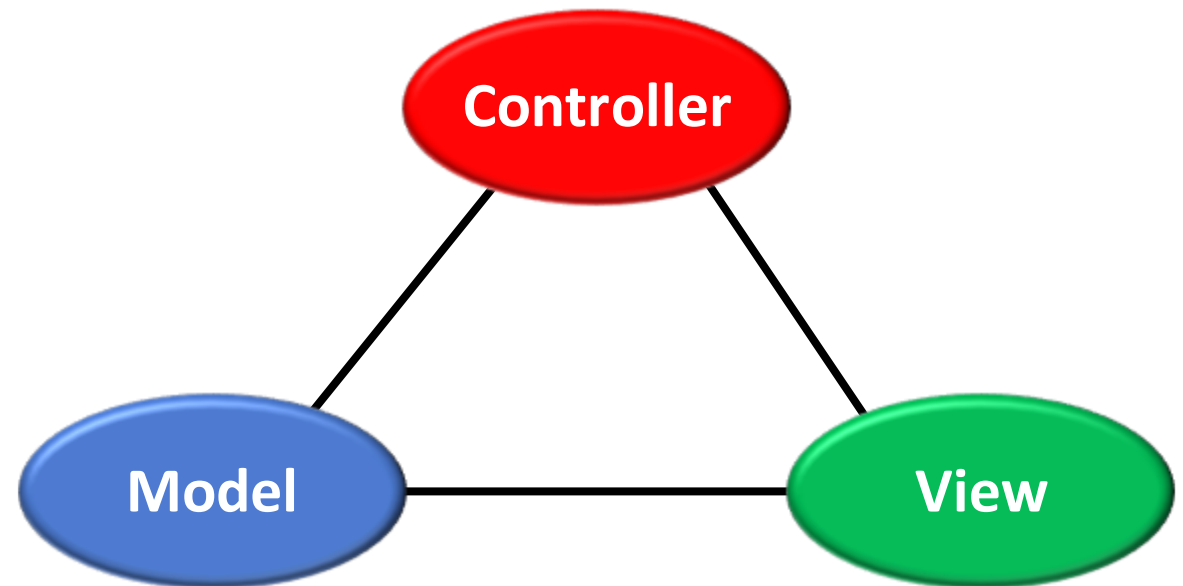
# View

- **View** chịu trách nhiệm hiển thị thông tin cho người dùng; tức là sinh ra mã (HTML) của trang web.
- **View** cần sử dụng dữ liệu do **Model** quản lý.



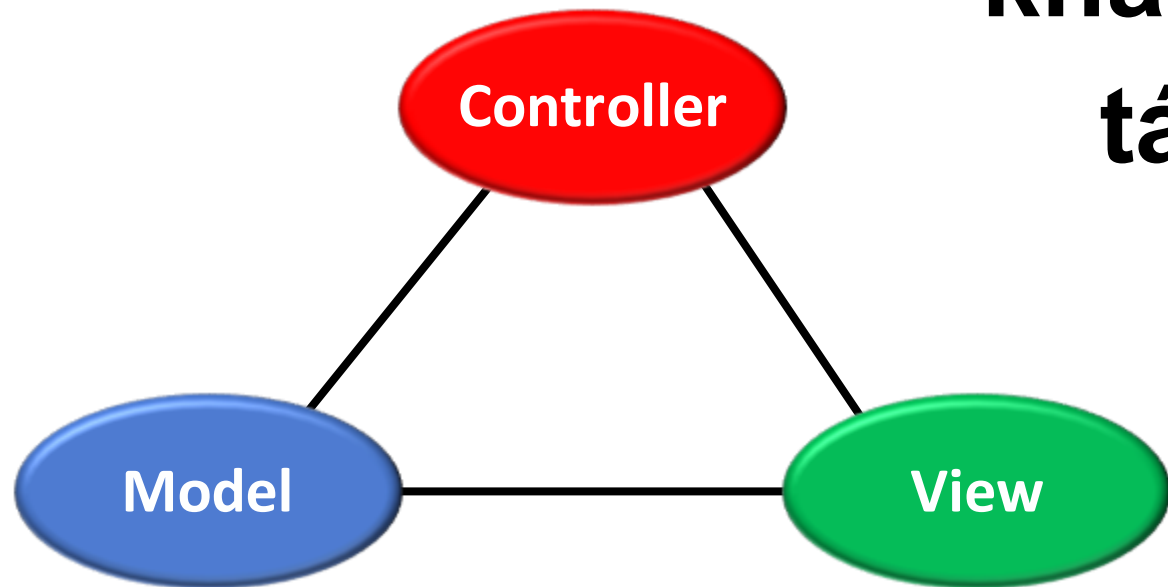
# Controller

- **Controller** là thành phần chịu trách nhiệm tương tác với người dùng.
- **Controller** có vai trò chính trong logic làm việc của ứng dụng web, điều phối hoạt động của **Model** và **View**.



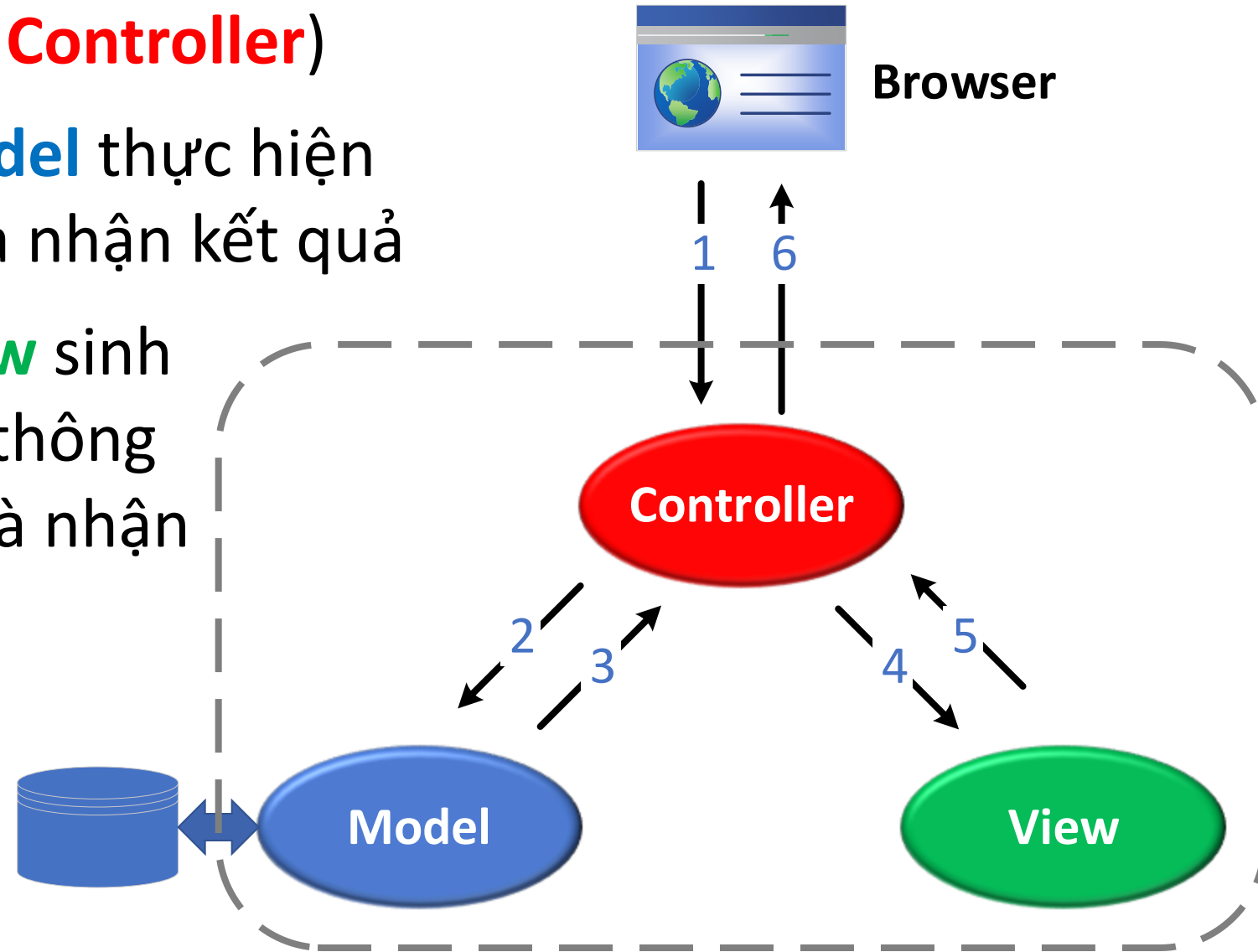


**Có thể có nhiều phương án  
khác nhau về luồng tương  
tác giữa 3 thành phần**



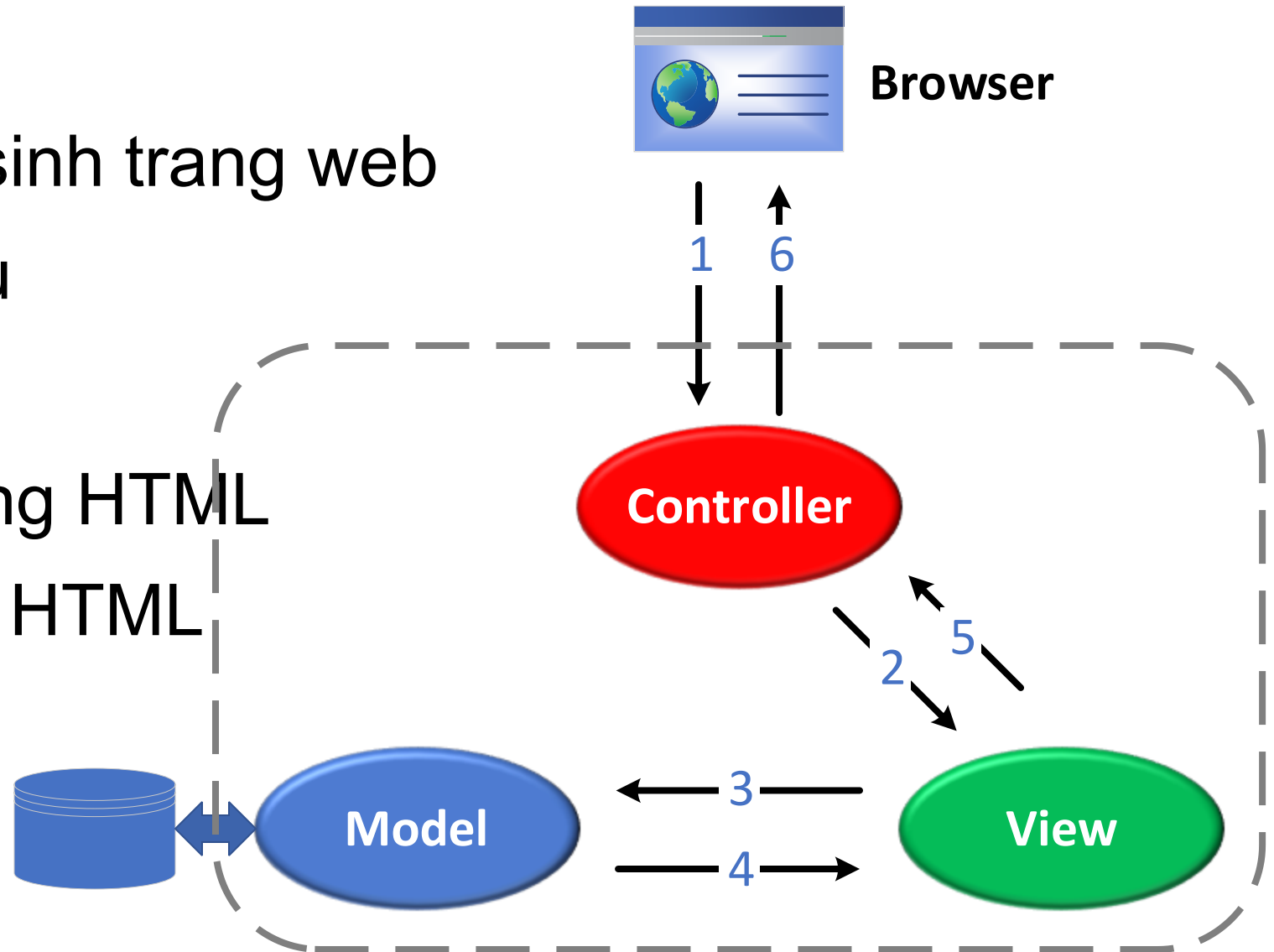
# MVC workflow (Ví dụ 1)

- 1: **Client** gửi truy vấn (tới **Controller**)
- 2,3: **Controller** yêu cầu **Model** thực hiện thao tác trên dữ liệu và nhận kết quả
- 4,5: **Controller** yêu cầu **View** sinh trang HTML (sử dụng thông tin mới có từ **Model**) và nhận kết quả
- 6: **Controller** gửi trang HTML cho **Client**



# MVC workflow (Ví dụ 2)

1. **Client** gửi truy vấn
2. **Controller** yêu cầu sinh trang web
3. **View** yêu cầu dữ liệu
4. **Model** gửi dữ liệu
5. **View** sinh và gửi trang HTML
6. **Controller** gửi trang HTML



# MVC workflow (Post/Redirect/Get pattern)

1. **POST**

2. Create/Update/Delete

3. **Redirect**

4. **GET**

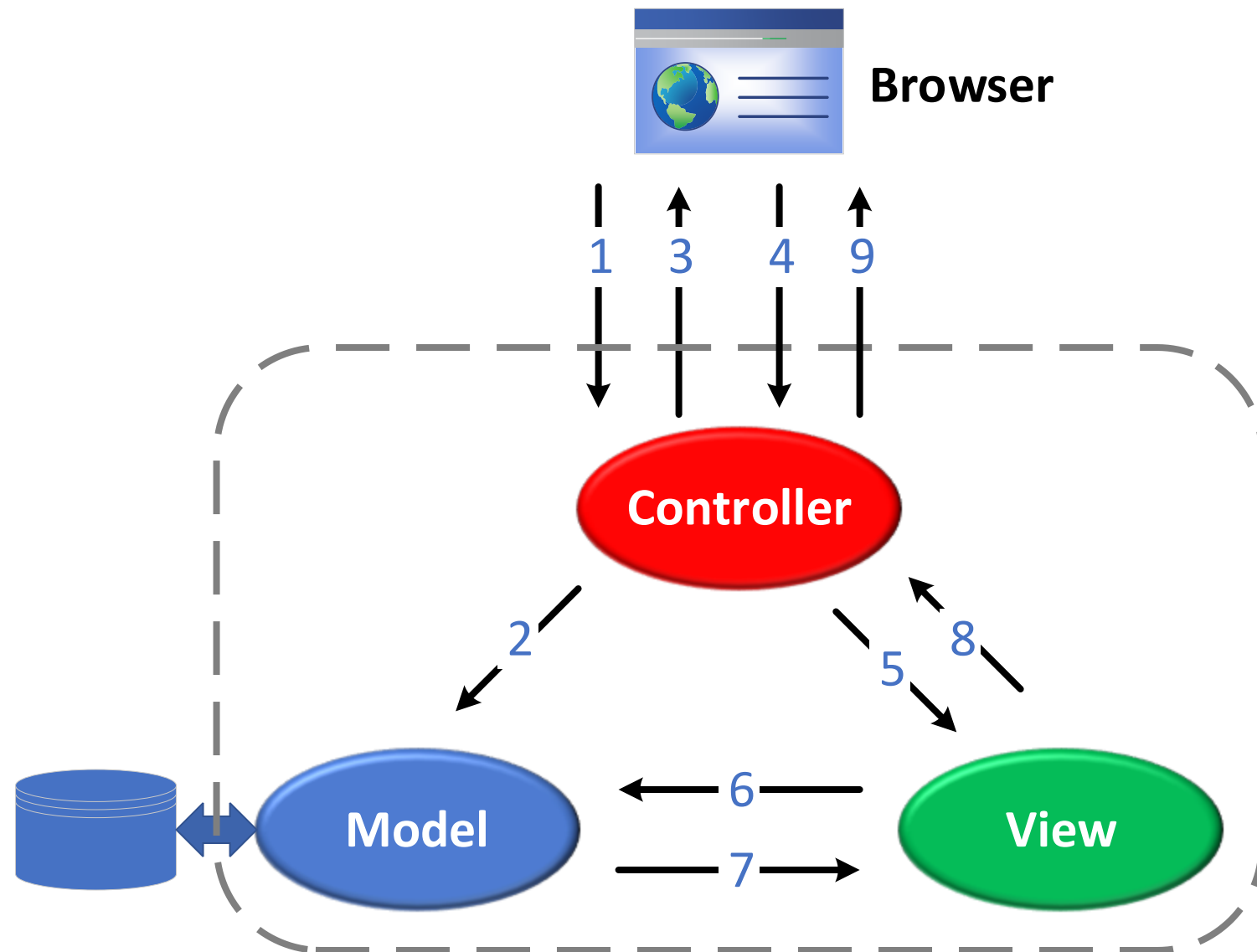
5. Yêu cầu trang HTML

6. Yêu cầu dữ liệu

7. Cung cấp dữ liệu

8. Gửi trang HTML

9. Chuyển trang HTML



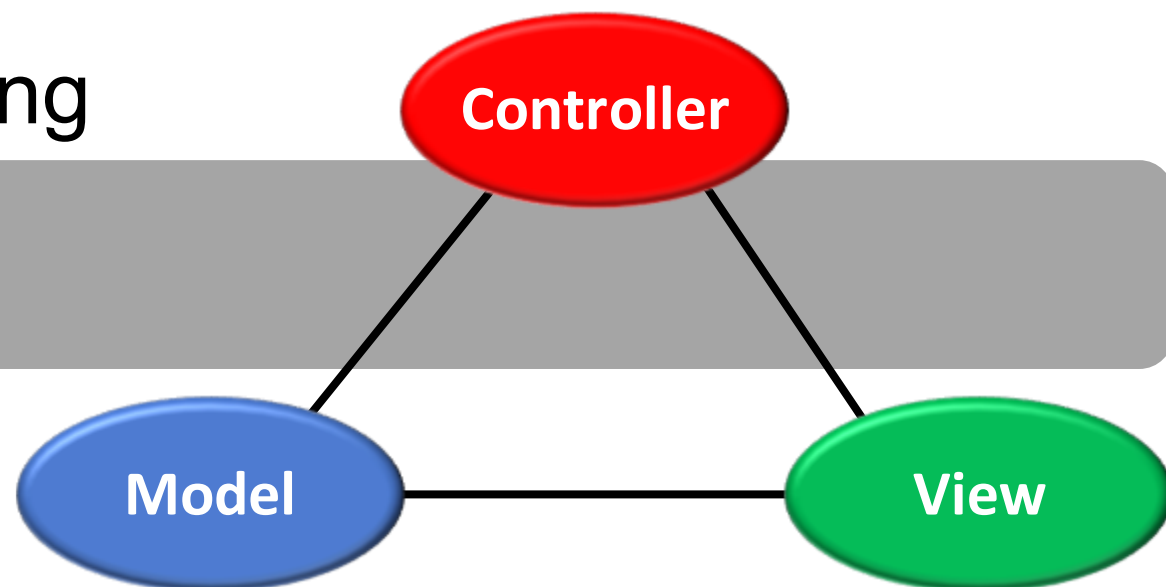
# Ưu/nhược điểm của MVC

## Ưu điểm

- Mô-đun hóa nên mã trong sáng: một ứng dụng có nhiều C, nhiều M, nhiều V; mỗi mô-đun thường khá ngắn.
- Chuyên môn hóa công việc
- Cộng tác, phát triển song song

## Nhược điểm

- Khó debug
- Khó unit test



# Một số mẫu khác

---

- Hierarchical MVC
- Model-View-ViewModel
- Model-View-Presenter
- (còn nữa)

1

Mô hình MVC

2

Khung phát triển ứng dụng web

3

Giới thiệu Laravel

4

Ứng dụng CRUD với Laravel

# Web Framework

❑ **Khung phát triển ứng dụng web** (web framework, web application framework) là khung phát triển phần mềm dành cho việc phát triển web, bao gồm web application, web services, web resources, and web APIs





# Web Framework

❑ Web Framework giúp tự động hóa, đơn giản hóa một số công đoạn trong phát triển web:

- Truy cập CSDL
- Quản lý phiên
- Quản lý template
- Mẫu thiết kế và các lớp cơ sở (MVC chẳng hạn)
- Lọc dữ liệu
- Kiểm soát truy cập
- ...

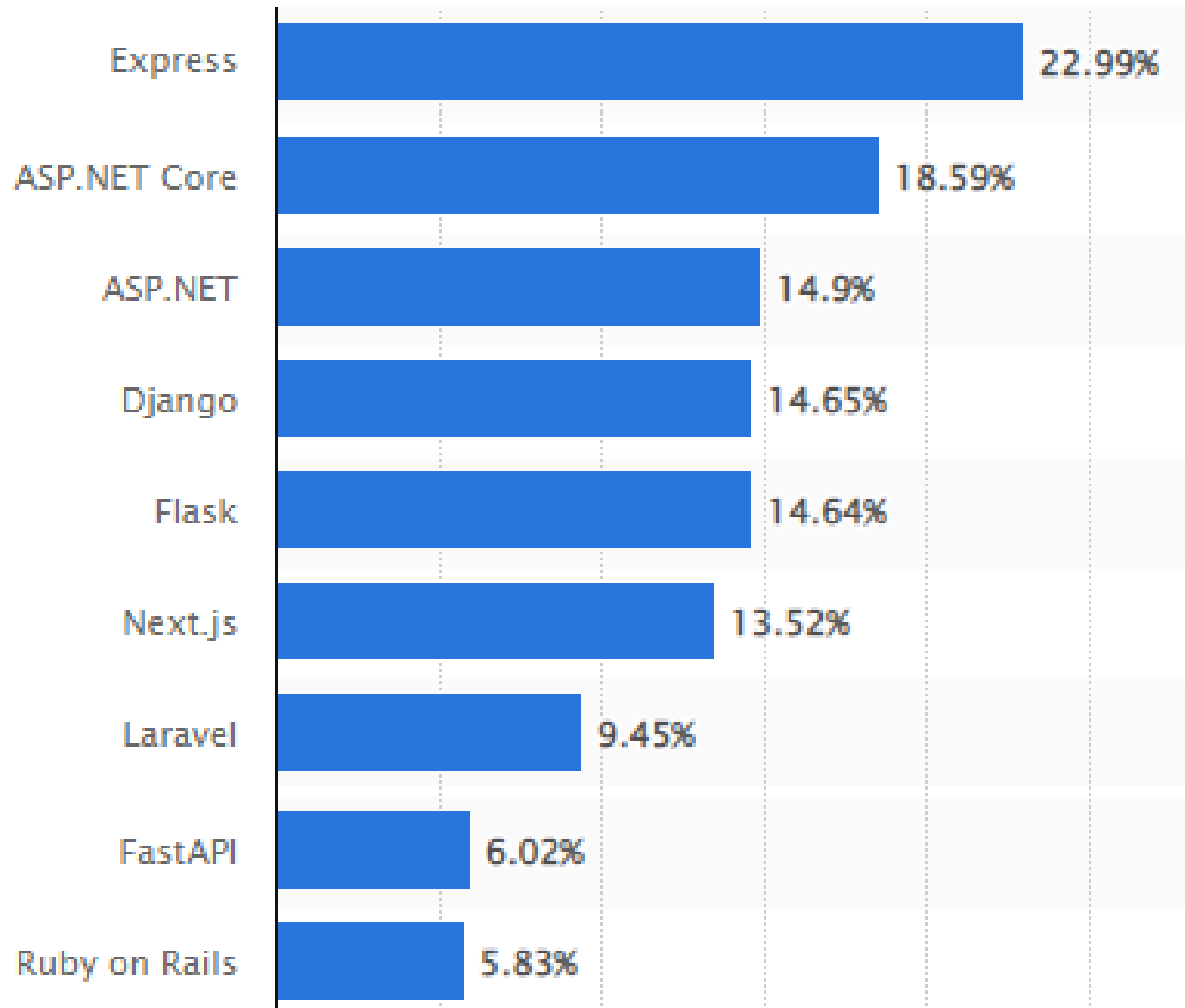


# Phân loại Web Framework

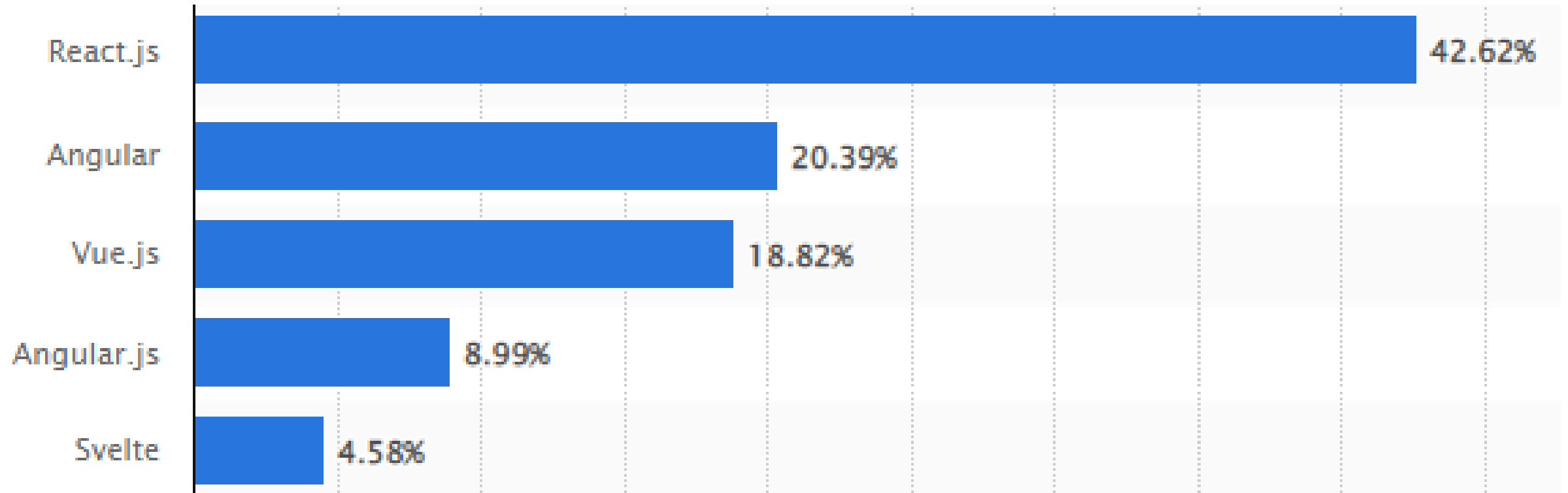
- Nơi áp dụng: Server-side, Client-side
- Ngôn ngữ: PHP, Java, C#, Python, JavaScript...
- Mẫu thiết kế: MVC, khác
- Mức độ hỗ trợ
  - Tối thiểu: CodeIgnitor, FastAPI...
  - Rất nhiều: Laravel, Spring MVC, ASP.NET...



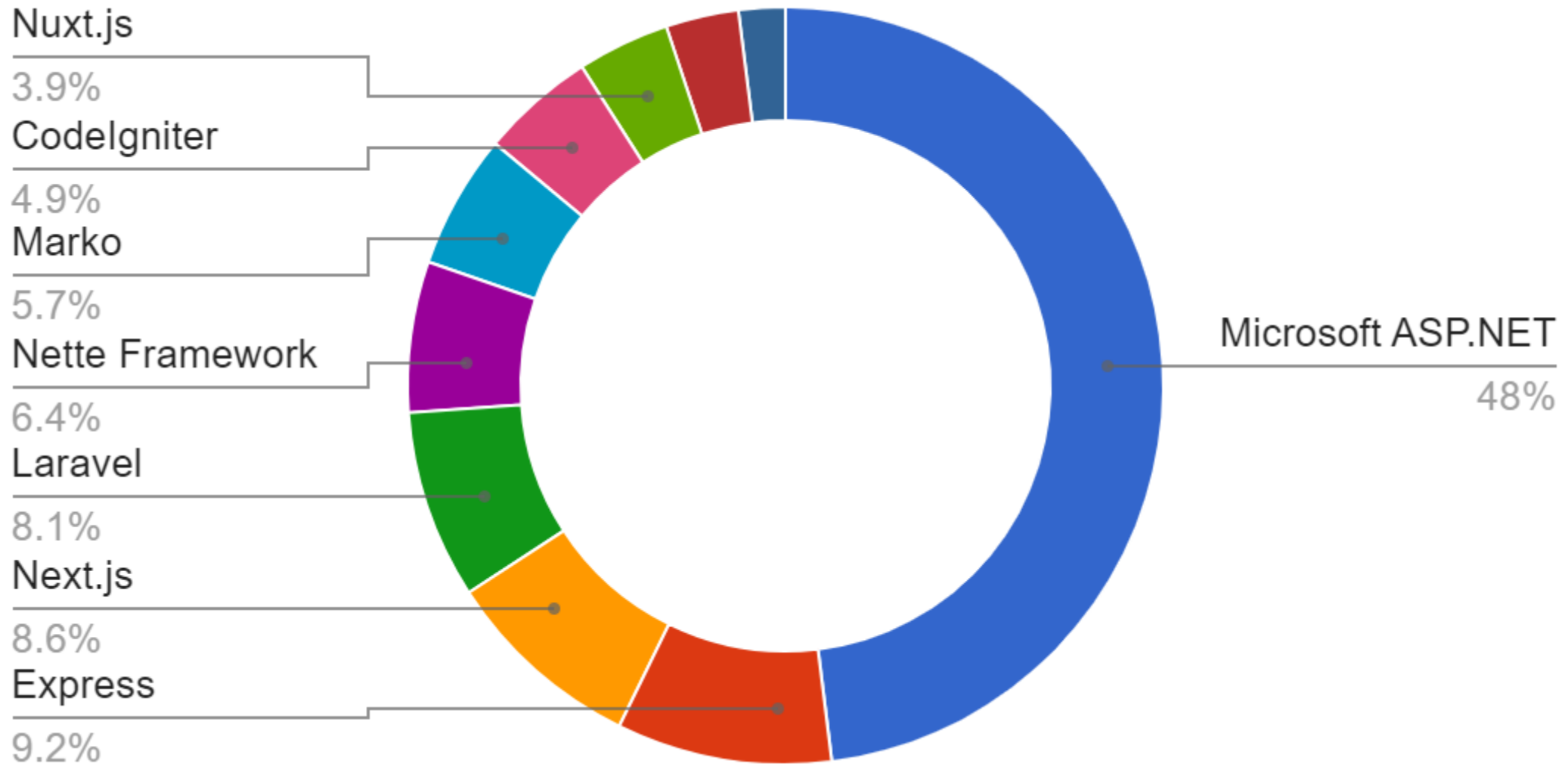
# Server-Side Web Frameworks



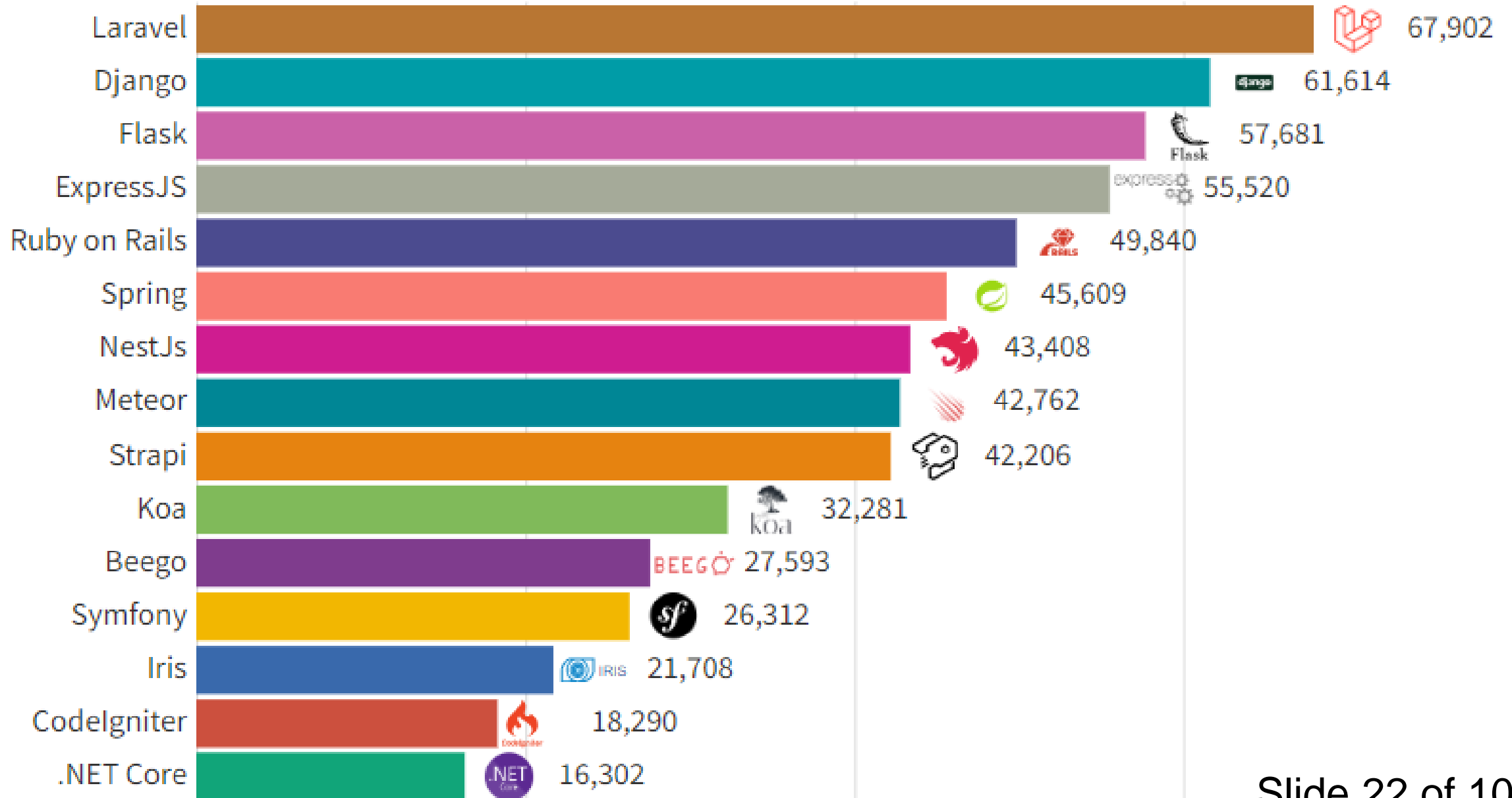
# Client-Side Web Frameworks

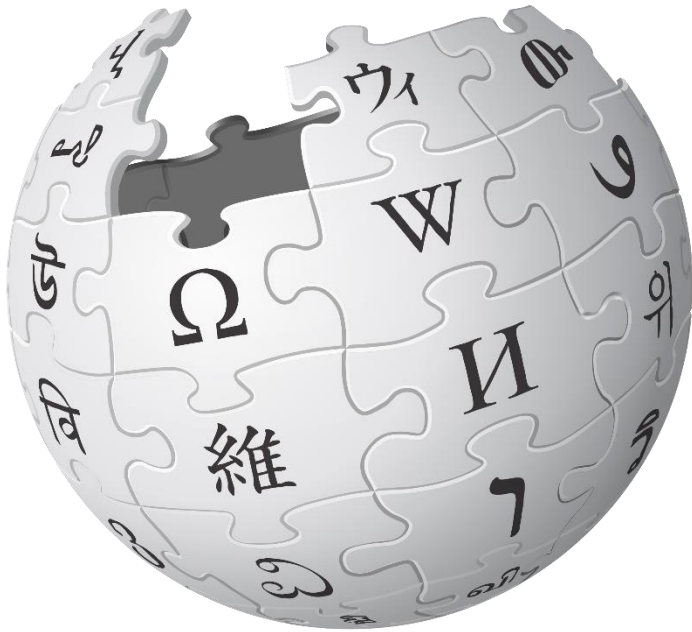


# Web frameworks technologies market share



# Open source Backend Frameworks by "GitHub star"





# Comparison of web frameworks

- **Server-side**

[https://en.wikipedia.org/wiki/Comparison\\_of\\_server-side\\_web\\_frameworks](https://en.wikipedia.org/wiki/Comparison_of_server-side_web_frameworks)

- **Client-side**

[https://en.wikipedia.org/wiki/Comparison\\_of\\_JavaScript-based\\_web\\_frameworks](https://en.wikipedia.org/wiki/Comparison_of_JavaScript-based_web_frameworks)

# Lợi ích của Web Framework

- Tái sử dụng mã
  - Tốc độ
  - Hiệu quả
  - An toàn
- Chia sẻ phong cách (style) phát triển
- Sự hỗ trợ từ cộng đồng





1

Mô hình MVC

2

Khung phát triển ứng dụng web

3

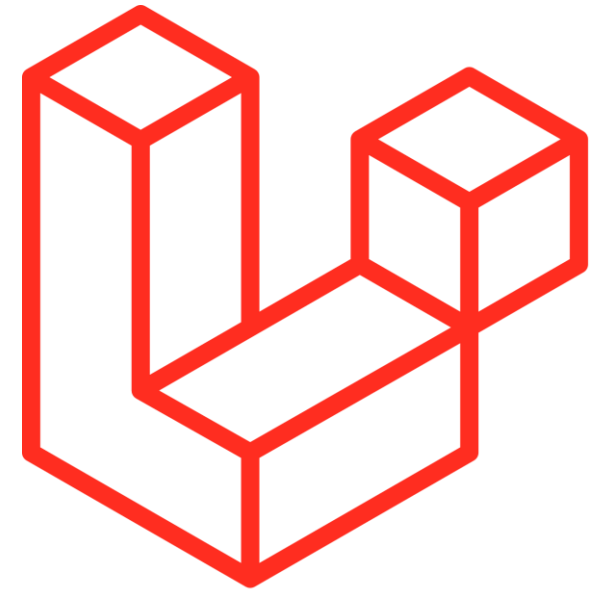
**Giới thiệu Laravel**

4

Ứng dụng CRUD với Laravel

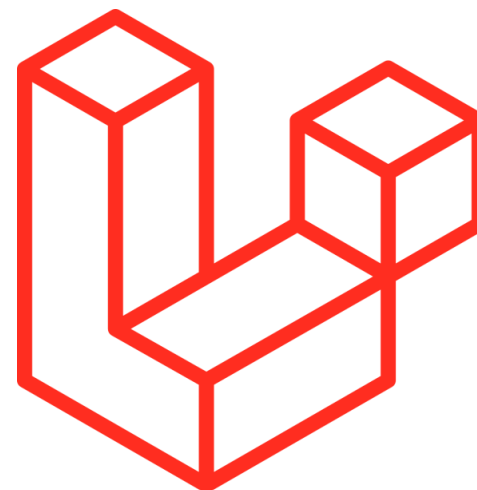
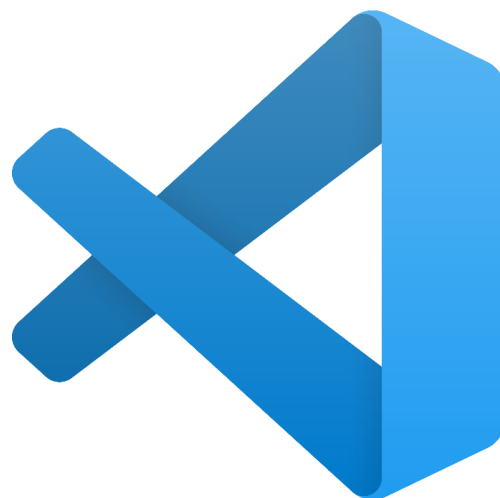
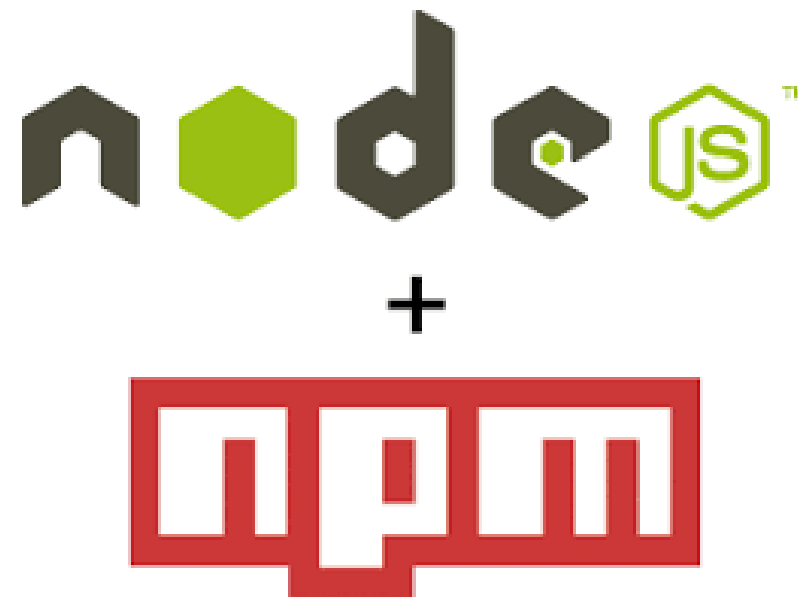
# Giới thiệu Laravel

- PHP MVC web framework
- Hỗ trợ
  - ORM (Object-relational mapping)
  - Database migration
  - Template
- Module mở rộng
  - Xác thực, Phân quyền
  - ....



# Thiết lập môi trường

- Web server, PHP, DBMS (XAMPP)
- Visual Studio Code (vscode)
- Composer
- Node.js và npm



# Kiểm tra môi trường

```
C:\xampp\htdocs>php -v
```

```
PHP 8.1.6 (cli) (built: May 11 2022 08:55:59) (ZTS Visual C++ 2019 x64)  
Copyright (c) The PHP Group  
Zend Engine v4.1.6, Copyright (c) Zend Technologies
```

```
C:\xampp\htdocs>composer --version
```

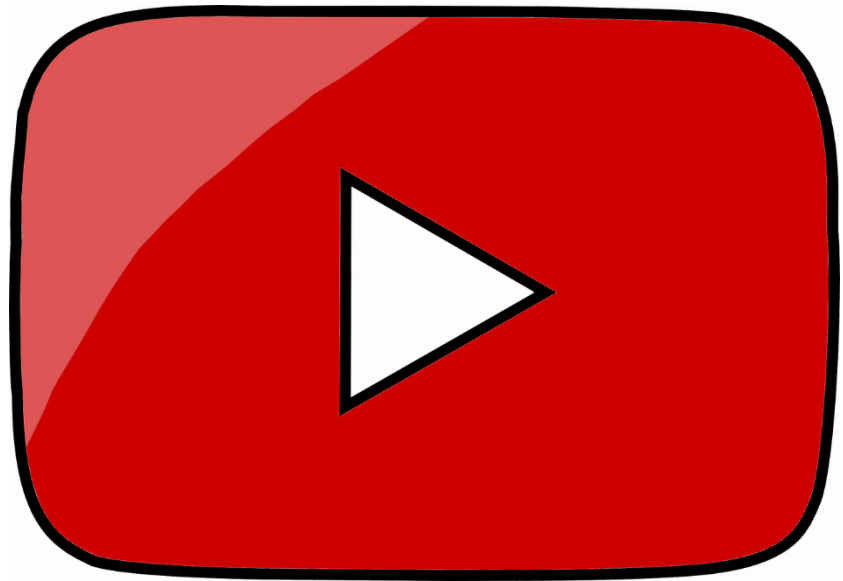
```
Composer version 2.3.10 2022-07-13 15:48:23
```

```
C:\xampp\htdocs>npm -v
```

```
8.15.0
```

```
C:\xampp\htdocs>node -v
```

```
v16.17.0
```



# How to setup VS Code for Laravel

<https://www.youtube.com/watch?v=Z2atp-ZUukQ>

# Khởi tạo một Laravel project

```
C:\xampp\htdocs>composer create-project laravel/laravel hellolaravel
```

```
Creating a "laravel/laravel" project at "./hellolaravel"
```

```
Installing laravel/laravel (v9.3.7)
```

```
- Installing laravel/laravel (v9.3.7): Extracting archive
```

```
Created project in C:\xampp\htdocs\hellolaravel
```

```
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
```

```
Loading composer repositories with package information
```

```
Updating dependencies
```

```
Lock file operations: 105 installs, 0 updates, 0 removals
```

- Locking brick/math (0.10.2)
- Locking dflydev/dot-access-data (v3.0.1)

# Bổ sung 'bootstrap' vào project

---

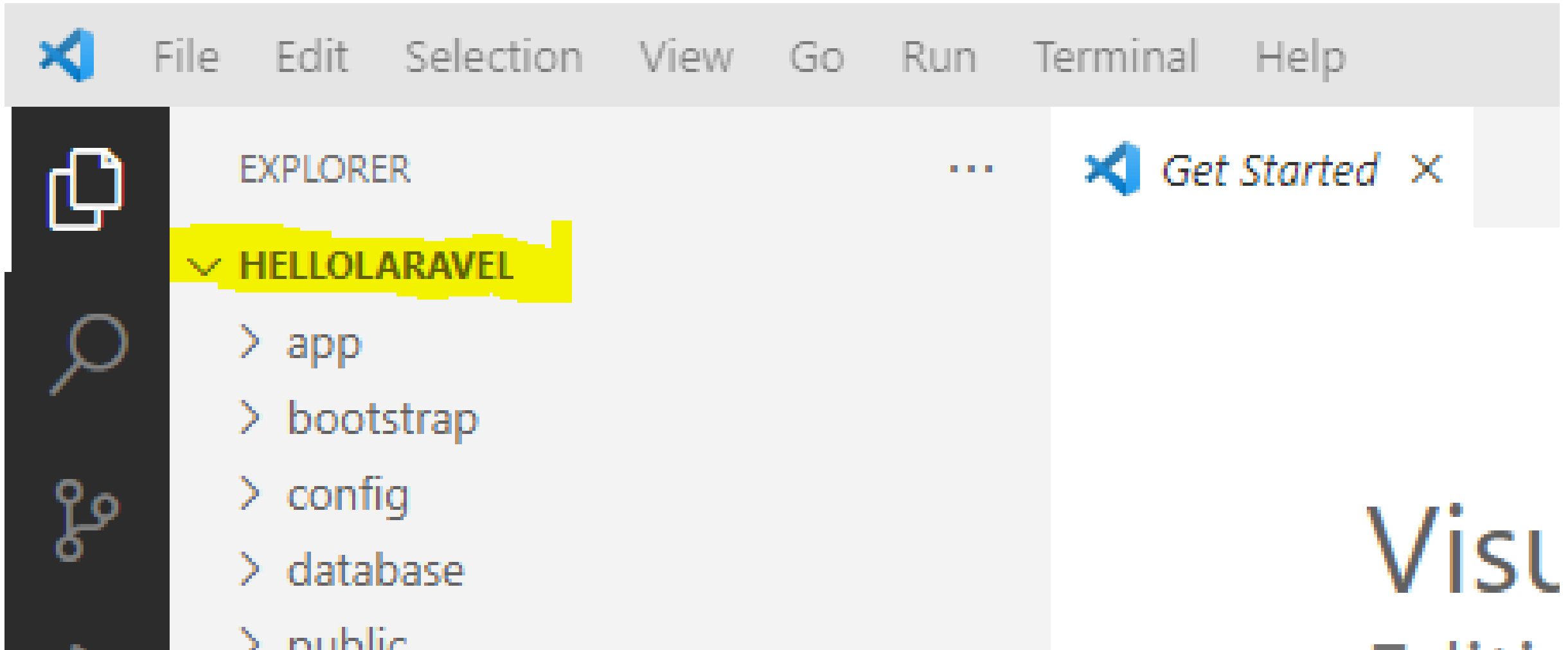
```
C:\xampp\htdocs\hellolaravel> composer require ui
```

```
C:\xampp\htdocs\hellolaravel> php artisan ui bootstrap
```

```
C:\xampp\htdocs\hellolaravel> npm install
```

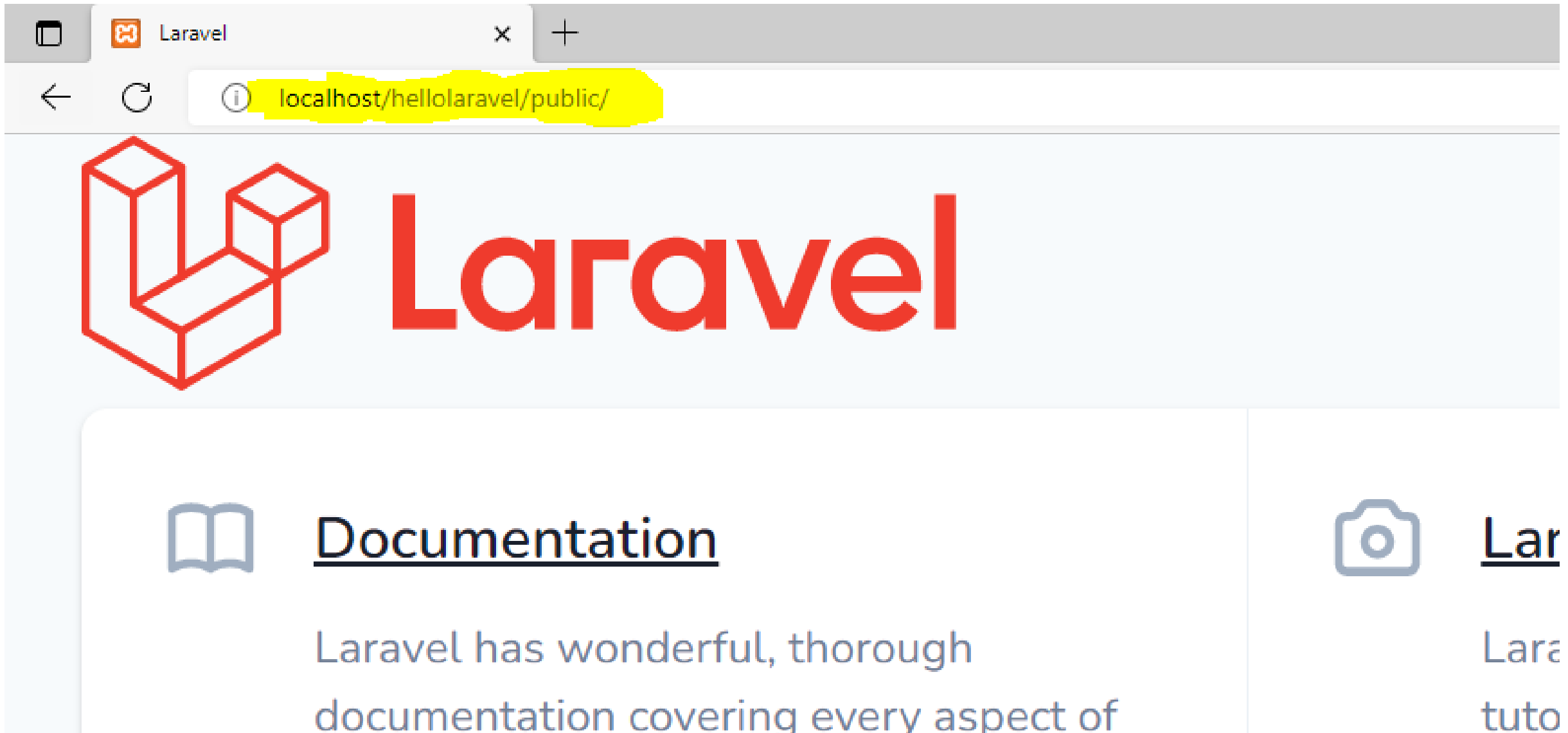
```
C:\xampp\htdocs\hellolaravel> npm run build
```

# Mở project trong vscode





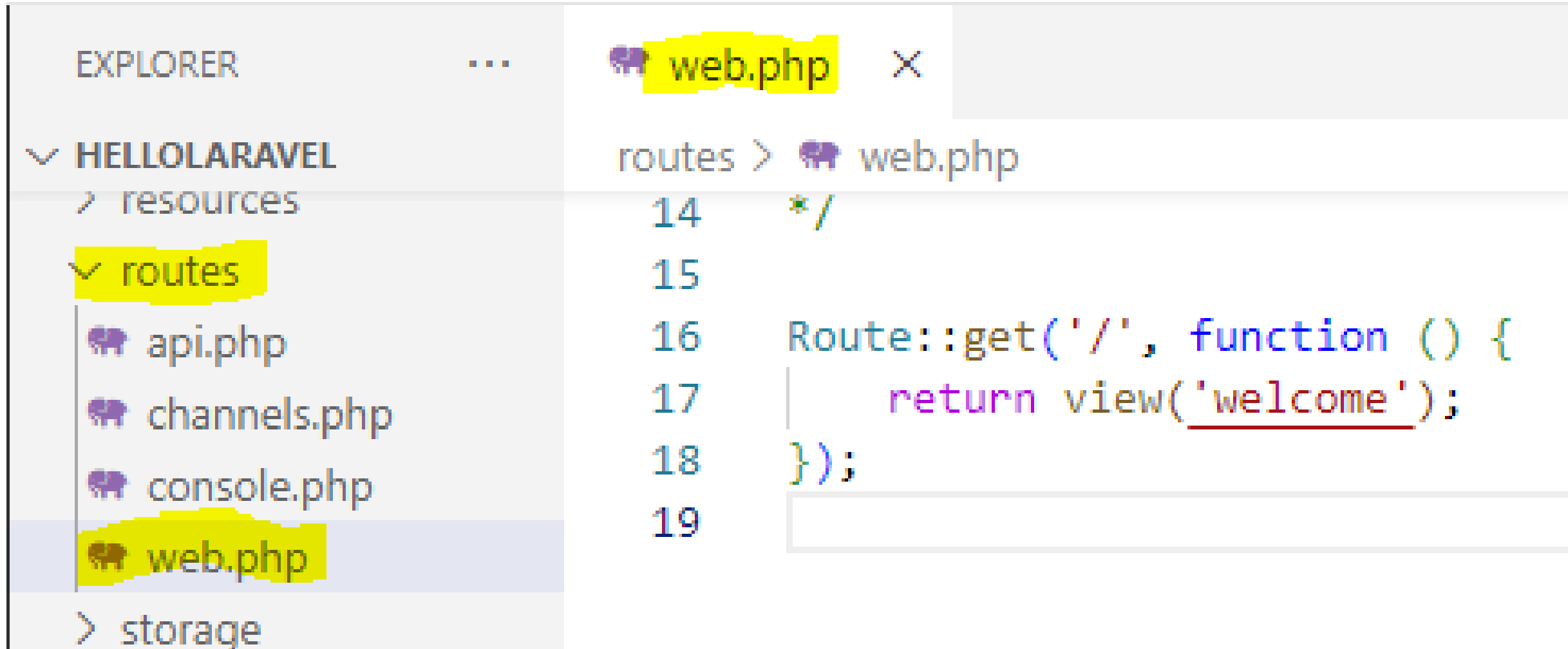
# Giao diện trang Laravel mặc định





**Trang mặc định được sinh ra  
thế nào?**

# \routes\web.php



The image shows a screenshot of an IDE interface. On the left is the 'EXPLORER' sidebar showing a project structure. The project is named 'HELLOLARAVEL'. It contains a 'resources' directory, which in turn contains a 'routes' directory. The 'routes' directory is expanded, showing four files: 'api.php', 'channels.php', 'console.php', and 'web.php'. The 'web.php' file is selected and highlighted. On the right is the main editor window, titled 'web.php'. It shows the following code:

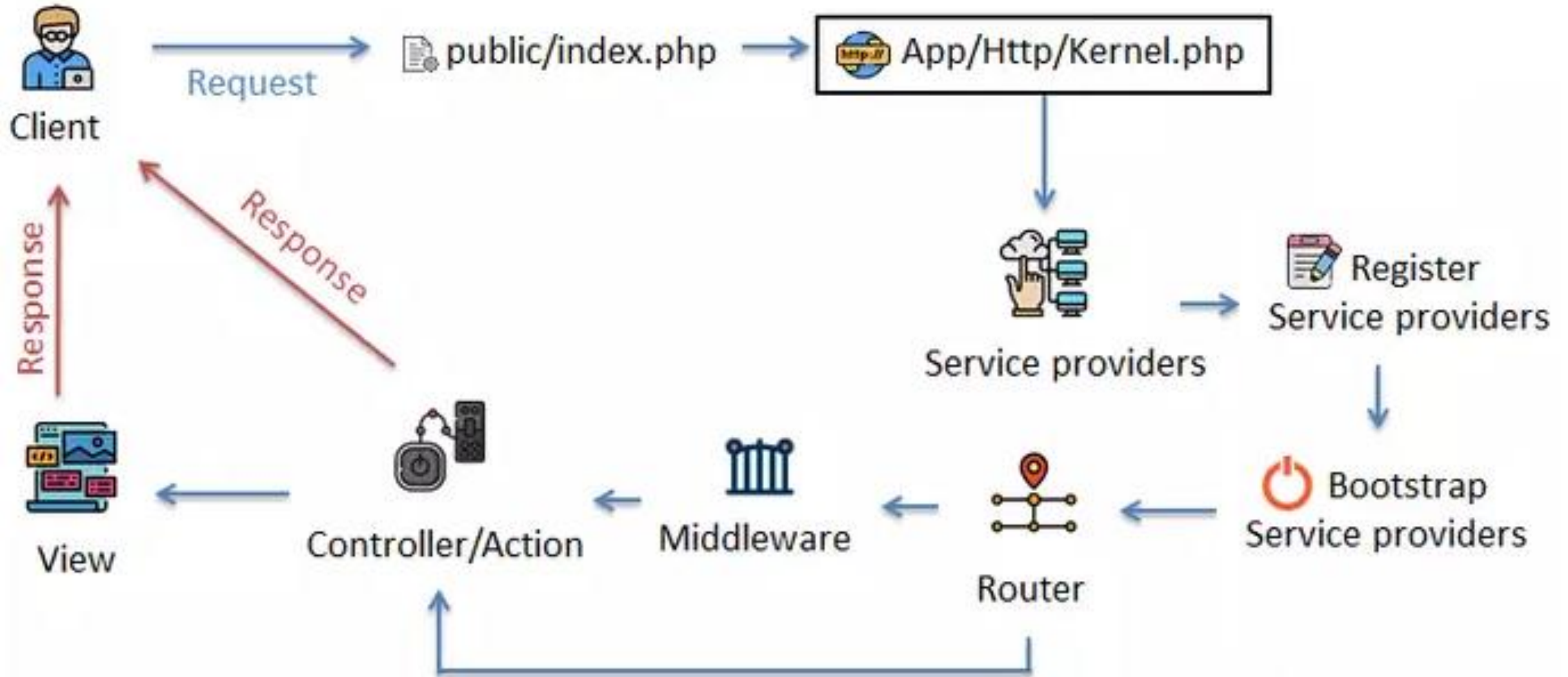
```
14 */
15
16 Route::get('/', function () {
17     return view('welcome');
18 });
19
```

# \resources\views\welcome.blade.php

resources > views > welcome.blade.php

```
1  <!DOCTYPE html>
2  <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3      <head>
4          <meta charset="utf-8">
5          <meta name="viewport" content="width=device-width, initial-sca
6
7          <title>Laravel</title>
8
9          <!-- Fonts -->
10         <link href="https://fonts.googleapis.com/css2?family=Nunito:wght=400&family=Roboto:wght=400">
11
12         <!-- Styles -->
```

# Laravel Request Lifecycle



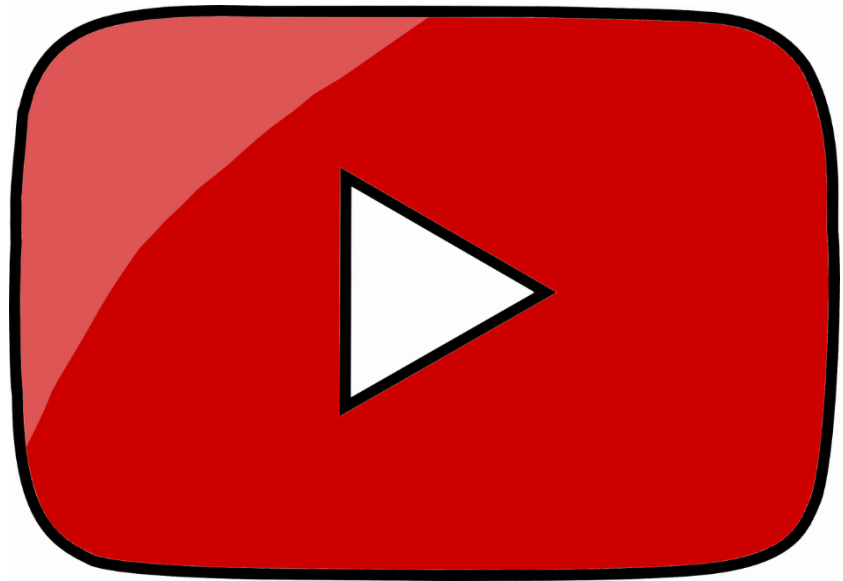


# Cơ chế hoạt động của Laravel

- [Request Lifecycle](#)
- [Routing](#)

# Debug Laravel

- Debugger là công cụ hữu ích, cần có khi lập trình
- Debug ứng dụng web phức tạp hơn so với debug ứng dụng thông thường
- Công cụ
  - PHP: `var_dump()`, `print_r()`
  - Laravel: `dd()`
  - Chuyên dụng: `xdebug`, `debugbar`



# How to Use Laravel Debugbar

<https://www.youtube.com/watch?v=DCoYynZ45Ws>





# Cài đặt Xdebug cho VSCode trên Windows

<https://tanakma.blogspot.com/2022/09/php-xdebug-vscode.html>

1

Mô hình MVC

2

Khung phát triển ứng dụng web

3

Giới thiệu Laravel

4

Ứng dụng CRUD với Laravel

# Mô tả ứng dụng CRUD

---

- Một ứng dụng quản lý danh bạ (Contacts). Thông tin mỗi người bao gồm:
  - họ
  - tên
  - email
- Thao tác: Create, Read, Update, Delete
- Không thực hiện xác thực



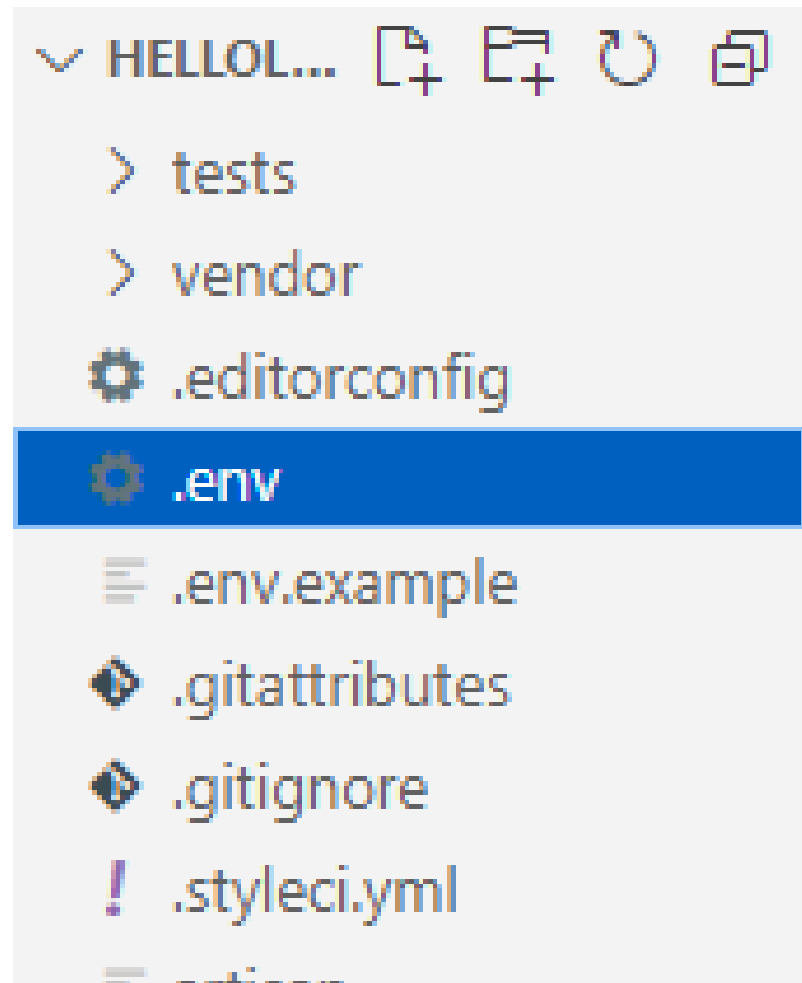
# Laravel naming conventions

- [Naming conventions](#)
- [Best practices](#)

# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'

# Cấu hình CSDL



```
.env
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=hellolaravel
15 DB_USERNAME=test
16 DB_PASSWORD=
17
```

# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'

# Tạo model "Contact"

```
PS C:\xampp\htdocs\hellolaravel> php artisan make:model Contact --migration
Model created successfully.
Created Migration: 2022_09_19_024937_create_contacts_table
```



# Sườn của migration 'contacts'

EXPLORER

- HELLOLARAVEL
  - app
  - bootstrap
  - config
  - database
    - factories
    - migrations
      - 2014\_10\_12\_000000\_create\_users\_table.php
      - 2014\_10\_12\_100000\_create\_password\_resets\_t...
      - 2019\_08\_19\_000000\_create\_failed\_jobs\_table.p...
      - 2019\_12\_14\_000001\_create\_personal\_access\_to...
      - 2022\_09\_19\_024937\_create\_contacts\_table.php
    - seeders
  - .gitignore
  - public

2022\_09\_19\_024937\_create\_contacts\_table.php X

database > migrations > 2022\_09\_19\_024937\_create\_contacts\_ta

```
1 use Illuminate\Database\Migrations\Migration;
2 use Illuminate\Database\Schema\Blueprint;
3 use Illuminate\Support\Facades\Schema;
4
5 class CreateContactsTable extends Migration
6 {
7     /**
8      * Run the migrations.
9      *
10     * @return void
11     */
12     public function up()
13     {
14         Schema::create('contacts', function (Blueprint $table) {
15             $table->id();
16             $table->timestamps();
17         });
18     }
19 }
```

# Chỉnh lại cấu trúc bảng 'contacts' (trong tập tin migration)

```
public function up()
{
    Schema::create('contacts', function (Blueprint $table) {
        $table->increments('id');
        $table->timestamps();
        $table->string('first_name');
        $table->string('last_name');
        $table->string('email');
    });
}
```

# Thực hiện migrate

```
C:\xampp\htdocs\hellolaravel>php artisan migrate
```

```
Migration table created successfully.
```

```
Migrating: 2014_10_12_000000_create_users_table
```

```
Migrated: 2014_10_12_000000_create_users_table (34.67ms)
```

```
Migrating: 2014_10_12_100000_create_password_resets_table
```

```
Migrated: 2014_10_12_100000_create_password_resets_table (21.69ms)
```

```
Migrating: 2019_08_19_000000_create_failed_jobs_table
```

```
Migrated: 2019_08_19_000000_create_failed_jobs_table (26.91ms)
```

```
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
```

```
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (38.24ms)
```

```
Migrating: 2022_09_19_024937_create_contacts_table
```

```
Migrated: 2022_09_19_024937_create_contacts_table (14.72ms)
```

# Thực hiện migrate

← Server: 127.0.0.1 » Database: hellolaravel	
☰	
Table	Action
<input type="checkbox"/> contacts	★ Browse Structure Search
<input type="checkbox"/> failed_jobs	★ Browse Structure Search
<input type="checkbox"/> migrations	★ Browse Structure Search
<input type="checkbox"/> password_resets	★ Browse Structure Search
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search
<input type="checkbox"/> users	★ Browse Structure Search
6 tables	Sum


# Thực hiện migrate

← Server: 127.0.0.1 » Database: hellolaravel » Table: contacts

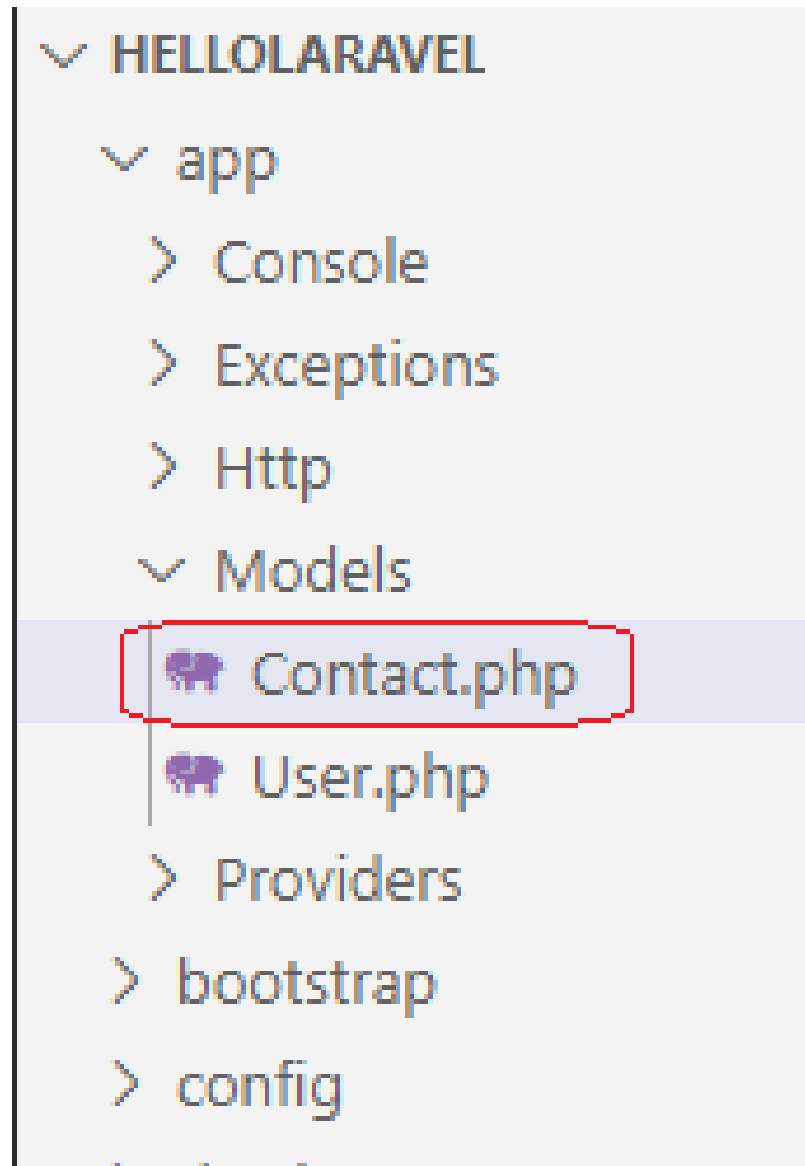
☰


id	created_at	updated_at	first_name	last_name	email
----	------------	------------	------------	-----------	-------

Query results operations

 Create view

# Sườn của model 'Contact'



app > Models >  Contact.php

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent
6  use Illuminate\Database\Eloquent
7
8  class Contact extends Model
9  {
10     use HasFactory;
11 }
12
```

# Hoàn thiện model 'Contact'

app > Models >  Contact.php

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Contact extends Model
9  {
10     use HasFactory;
11     protected $fillable = ['first_name', 'last_name', 'email'];
12 }
```

# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'



# Khởi tạo controller 'ContactController' và định nghĩa các route

```
C:\xampp\htdocs\hellolaravel>php artisan make:controller ContactController --resource  
Controller created successfully.
```

- Controller file: app/Http/Controllers/PhotoController.php
- --resource: tạo sẵn các phương thức CRUD trong controller để về sau định tuyến nhanh bằng Route::resource()

```
routes > web.php
```

```
14  */  
15  use App\Http\Controllers\ContactController;  
16  
17  Route::get('/', function () {  
18      |   return view('welcome');  
19  });  
20  Route::resource('contacts', ContactController::class);
```

# Sườn của 'ContactController'

app > Http > Controllers > ContactController.php

```
11
12     * @return \Illuminate\Http\Response
13     */
14     public function index()
15     {
16         //
17     }
18
19     /**
20      * Show the form for creating a new resource.
21      *
22      * @return \Illuminate\Http\Response
23      */
24     public function create()
25     {
26         //
```

# Routing các tác vụ CRUD

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/PATCH	/contacts/{contact}	update()
DELETE	/contacts/{contact}	destroy()

# Routing các tác vụ CRUD

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/PATCH	/contacts/{contact}	update()
DELETE	/contacts/{contact}	destroy()

## Read:

1. Đọc danh sách các contact trong CSDL
2. Đọc thông tin chi tiết một contact

# Routing các tác vụ CRUD

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/		
DELETE		

## Create:

1. Lấy form để nhập thông tin một contact
2. Nhận một contact (mới) và ghi vào CSDL

# Routing các tác vụ CRUD

Method	URL	Controller method
GET	/contacts	index()
GET	/contacts/show/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/PATCH	/contacts/{contact}	update()
DELETE	/contacts/{contact}	destroy()

## Update:

1. Lấy form để chỉnh sửa một contact
2. Nhận thông tin mới về một contact và cập nhật vào CSDL

# Routing các tác vụ CRUD

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/PATCH	/contacts/{contact}	update()
DELETE	/contacts/{contact}	destroy()

**Delete:** Xóa một contact

# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'



# Lập trình tác vụ 'Create'

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT/		
DELETE		

## Create:

1. Lấy form để nhập thông tin một contact
2. Nhận một contact (mới) và ghi vào CSDL

# Lập trình tác vụ 'Create'

```
use App\Http\Model\Contact;
class ContactController extends Controller{
    public function create(){
        return view('contacts.create');
    }
    public function store(Request $request) {
        /*Thực tế cần sanitize dữ liệu trước khi lưu*/
        $contact = new Contact([
            'first_name'=>$request->get('first_name'),
            'last_name'=>$request->get('last_name'),
            'email'=>$request->get('email')]);
        $contact->save();
        return redirect('/contacts')->with('success', 'Contact saved');
    }
    ...
}
```

# Sơ lược về template trong Laravel

- **Thông thường trong ứng dụng Laravel:**
  - các trang HTML được sinh ra bởi Blade Template;
  - một ứng dụng thường có nhiều Template;
  - có một Template đóng vai trò là khung (base) cho mọi View; các Template khác sẽ mở rộng Template khung này;
  - template khung cung cấp phần chung cho mọi view (cấu trúc trang, menu, sidebar, footer,...); các template khác sẽ bổ sung các phần chi tiết vào template khung (main content,...)

# Sơ lược về template trong Laravel

## ❑ Đối với ứng dụng đang xây dựng

[resources/views]

+contacts

|- create.blade.php

|- edit.blade.php

|- index.blade.php

+layouts

|-base.blade.php

# base.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <title>Demo Laravel CRUD app</title>
    @vite(['resources/sass/app.scss'])
</head>
<body>
    <div class="container"> @yield('main') </div>
    @vite(['resources/js/app.js'])
</body>
</html>
```

# base.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <title>Demo Laravel CRUD app</title>
  @vite(['resources/sass/app.scss'])
</head>
```

- Vite là công cụ phát triển front-end
- Chỉ thị **@vite** sinh ra các thẻ HTML để sử dụng các tài sản (asset) như CSS, JavaScript, images...
- Kết quả ở đây là:  
<link rel="stylesheet"  
href="http://localhost/testvite/public/build/assets/xxxx.css" />

# base.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <script type="module"
    src="http://localhost/testvite/public/build/assets/yyyy.js">
    </script>
    @vite(['resources/js/app.js'])
</head>
<body>
    <div class="container"> @yield('main') </div>
</body>
</html>
```

# base.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Demo Laravel 5.4</title>
  @vite(['resources/css/app.css', 'resources/js/app.js'])
</head>
<body>
  <div class="container"> @yield('main') </div>
  @vite(['resources/js/app.js'])
</body>
</html>
```

Nội dung này sẽ được thay thế bởi  
**@section('main')** trong các template con

**@yield('main')**



# create.blade

```
@extends('layouts.base')
@section('main')
<div class="row"> <div class="col-sm-8 offset-sm-2">
    <form method="post" action="{{ route('contacts.store') }}">
        @csrf
        <div class="form-group">
            <label for="first_name">Họ, đệm:</label>
            <input type="text" class="form-control" name="first_name"/> </div>
        <div class="form-group">
            <label for="last_name">Tên:</label>
            <input type="text" class="form-control" name="last_name"/> </div>
        <div class="form-group">
            <label for="email">Email:</label>
            <input type="text" class="form-control" name="email"/> </div>
        <button type="submit" class="btn btn-primary">Add contact</button>
    </form>
</div> </div>
@endsection
```

# create.blade

```
@extends('layouts.base')
```

```
@section('main')
```

```
<div class="row"> <div class="col-sm-8 offset-sm-2">
```

```
<form method="POST" action="{{ route('contacts.store') }}">
```

```
@csrf
```

'create' kế thừa 'base'

```
<div class="form-group">
```

```
<label for="first_name">Họ, đệm:</label>
```

```
<input type="text" class="form-control" name="first_name"/> </div>
```

```
<div class="form-group">
```

```
<label for="last_name">Tên:</label>
```

```
<input type="text" class="form-control" name="last_name"/> </div>
```

```
<div class="form-group">
```

```
<label for="email">Email:</label>
```

```
<input type="text" class="form-control" name="email"/> </div>
```

```
<button type="submit" class="btn btn-primary">Add contact</button>
```

```
</form>
```

```
</div> </div>
```

```
@endsection
```

# create.blade

```
@extends('layouts.base')
@section('main')
<div class="row"> <div class="col-sm-8 offset-sm-2">
  <form method="post" action="{{ route('contacts.store') }}">
    @csrf
    <div class="form-group">
      <label for="first_name">First Name</label>
      <input type="text" class="form-control" name="first_name"/>
    <div class="form-group">
      <label for="last_name">Last Name</label>
      <input type="text" class="form-control" name="last_name"/>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="text" class="form-control" name="email"/>
    <button type="submit" class="btn btn-primary">Add contact</button>
  </form>
</div> </div>
@endsection
```

- **create** định nghĩa lại vùng **main** trong **base**
- tức là **@yield('main')** trong **base** sẽ được thay thế bởi nội dung của **@section('main')** trong template con (ở đây là **create**)

# create.blade

```
@extends('layouts.base')
@section('main')
<div class="row"> <div class="col-sm-8 offset-sm-2">
  <form method="post" action="{{ route('contacts.store') }}">
    @csrf
    <div class="form-group">
      <label>
        <input type="text" class="form-control" name="last_name" /> </div>
      <div class="form-group">
        <label>
        <input type="text" class="form-control" name="first_name" /> </div>
      <div class="form-group">
        <input type="text" class="form-control" name="email" /> </div>
      <div class="form-group">
        <label for="email">Email:</label>
        <input type="text" class="form-control" name="email" /> </div>
      <button type="submit" class="btn btn-primary">Add contact</button>
    </form>
  </div> </div>
@endsection
```

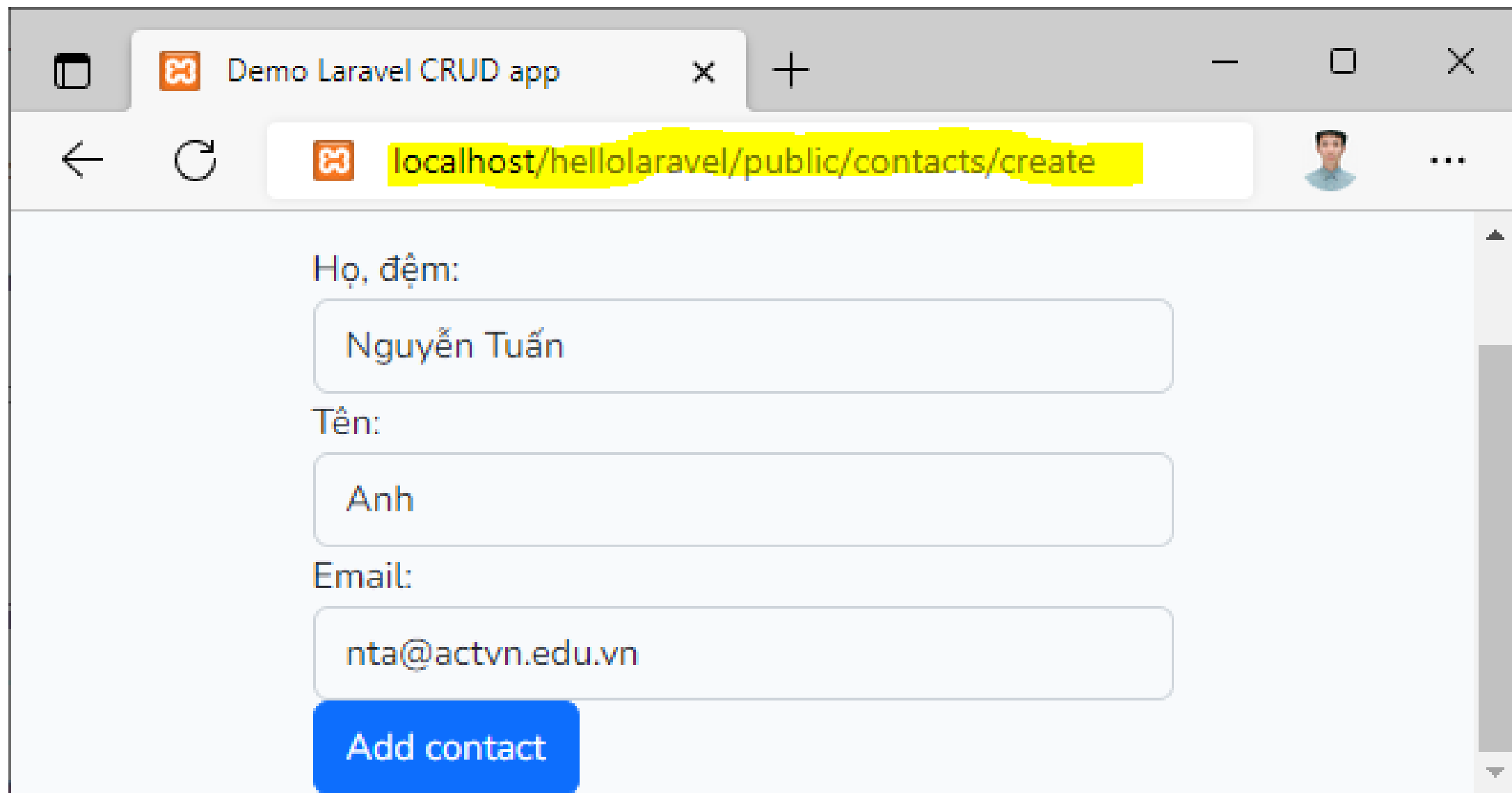
Trả về URL có dạng  
http(s)://domain/public/contacts

# create.blade

```
@extends('layouts.base')
@section('main')
<div class="row"> <div class="col-sm-8 offset-sm-2">
  <form method="post" action="{{ route('contacts.store') }}">
    @csrf
    <div class="form-group">
      <label for="first_name">Họ, đệm:</label>
      <input type="text" class="form-control" name="first_name"/> </div>
      <label>
      <input type="text" class="form-control" name="last_name"/> </div>
    </div>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="text" class="form-control" name="email"/> </div>
    <button type="submit" class="btn btn-primary">Add contact</button>
  </form>
</div> </div>
@endsection
```

Sinh hidden input chứa mã chống CSRF

# Kết quả Create: lấy form



The screenshot shows a web browser window with a single tab titled "Demo Laravel CRUD app". The address bar displays the URL "localhost/hello-laravel/public/contacts/create", which is highlighted in yellow. The browser interface includes standard navigation buttons (back, forward, refresh) and window controls (minimize, maximize, close). The page content is a form for creating a new contact. It consists of three text input fields with labels "Họ, đệm:", "Tên:", and "Email:". The first field contains the text "Nguyễn Tuấn", the second contains "Anh", and the third contains "nta@actvn.edu.vn". Below these fields is a blue button with the text "Add contact". A vertical scrollbar is visible on the right side of the page content area.

Họ, đệm:

Nguyễn Tuấn

Tên:

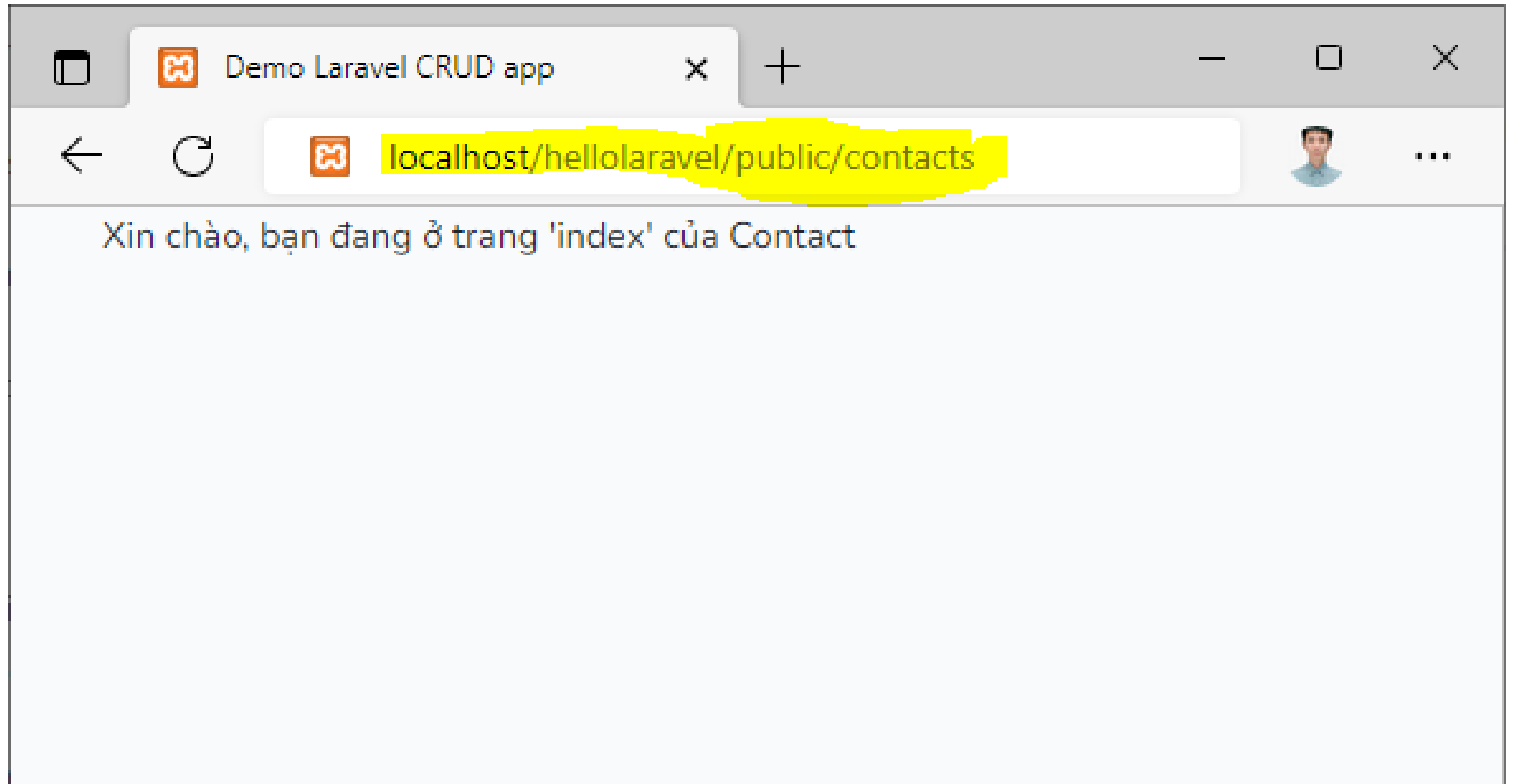
Anh

Email:

nta@actvn.edu.vn

Add contact

# Kết quả Create: redirect



# Kết quả Create: cập nhật CSDL

id	created_at	updated_at	first_name	last_name	email
7	2022-09-21 08:08:36	2022-09-21 08:08:36	Nguyễn Tuấn	Anh	nta@actvn.edu.vn



# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'

# Lập trình tác vụ 'Read'

Method	Route	Controller method
GET	/contacts	index()
GET	/contacts/{contact}	show()
GET	/contacts/create	create()
POST	/contacts	store()
GET	/contacts/{contact}/edit	edit()
PUT		
DELETE		

## Read:

1. Lấy danh sách contact
2. Lấy thông tin chi tiết về một contact

Ghi chú: chỉ triển khai index(), còn show() tương tự

# Lập trình tác vụ 'Create': controller

```
use App\Models\Contact;  
class ContactController extends Controller  
{  
    public function index()  
    {  
        $all_contacts = Contact::all();  
        return view('contacts.index', compact('all_contacts'));  
    }  
    //.....  
}
```

Truyền dữ liệu vào View

# index.blade.php (1/4)

```
@extends('layouts.base')
@section('main')
<div class="row"><div class="col-sm-12">
    <h1 class="display-3">Contacts</h1>
    <table class="table table-striped">
        <thead>
            <tr>
                <td>ID</td>
                <td>Họ và tên</td>
                <td>Email</td>
                <td colspan = 2></td>
            </tr>
        </thead>
```

## index.blade.php (2/4)

```
<tbody>
  @foreach($all_contacts as $contact)
    <tr>
      <td>{{ $contact->id }}</td>
      <td>{{ $contact->first_name }} {{ $contact->last_name }}</td>
      <td>{{ $contact->email }}</td>
```

## index.blade.php (3/4)

```
<td><a href="{{ route('contacts.edit',$contact->id) }}" class="btn btn-primary">Edit</a>
</td>
<td><form action="{{ route('contacts.destroy', $contact->id) }}"
method="post">
    @csrf
    @method('DELETE')
    <button class="btn btn-danger" type="submit">Delete</button>
</form></td>
```

- HTML form chỉ hỗ trợ GET và POST
- Chỉ thị này tạo hidden input để lưu tên của phương thức (DELETE) mà Laravel sẽ xử lý

# index.blade.php (4/4)

@endforeach

</tbody>

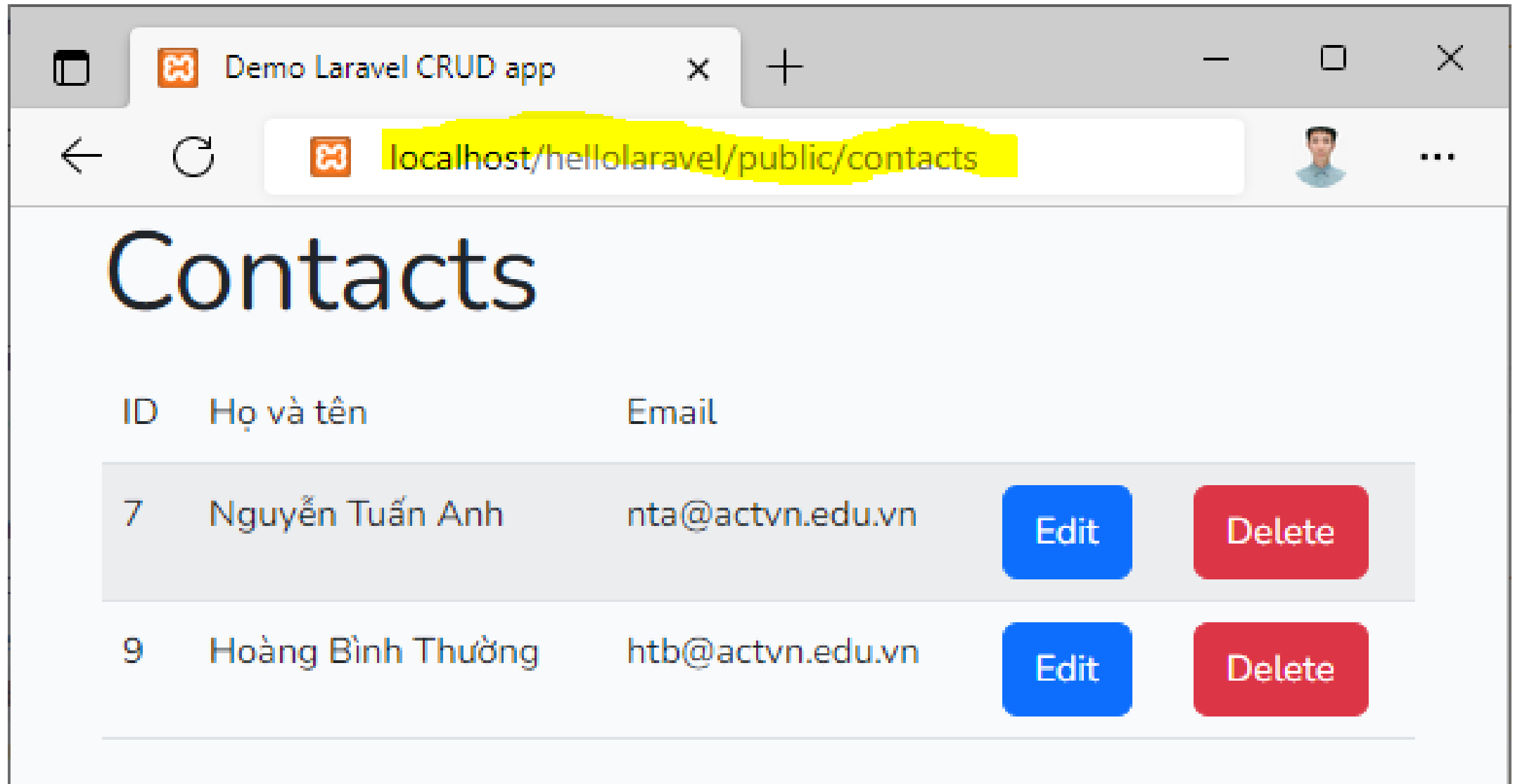
</table>

</div>

</div>

@endsection

# Lập trình tác vụ 'Create': lấy danh sách





# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'

# Lập trình tác vụ 'Update'

Meth	Update:		method
GET	1. Lấy form để edit một contact 2. Gửi thông tin để cập nhật một contact		
GET			
GET	/contacts/create		create()
POST	/contacts		store()
GET	/contacts/{contact}/edit		edit()
PUT/PATCH	/contacts/{contact}		update()
DELETE	/contacts/{contact}		destroy()

# Lập trình tác vụ 'Update': controller

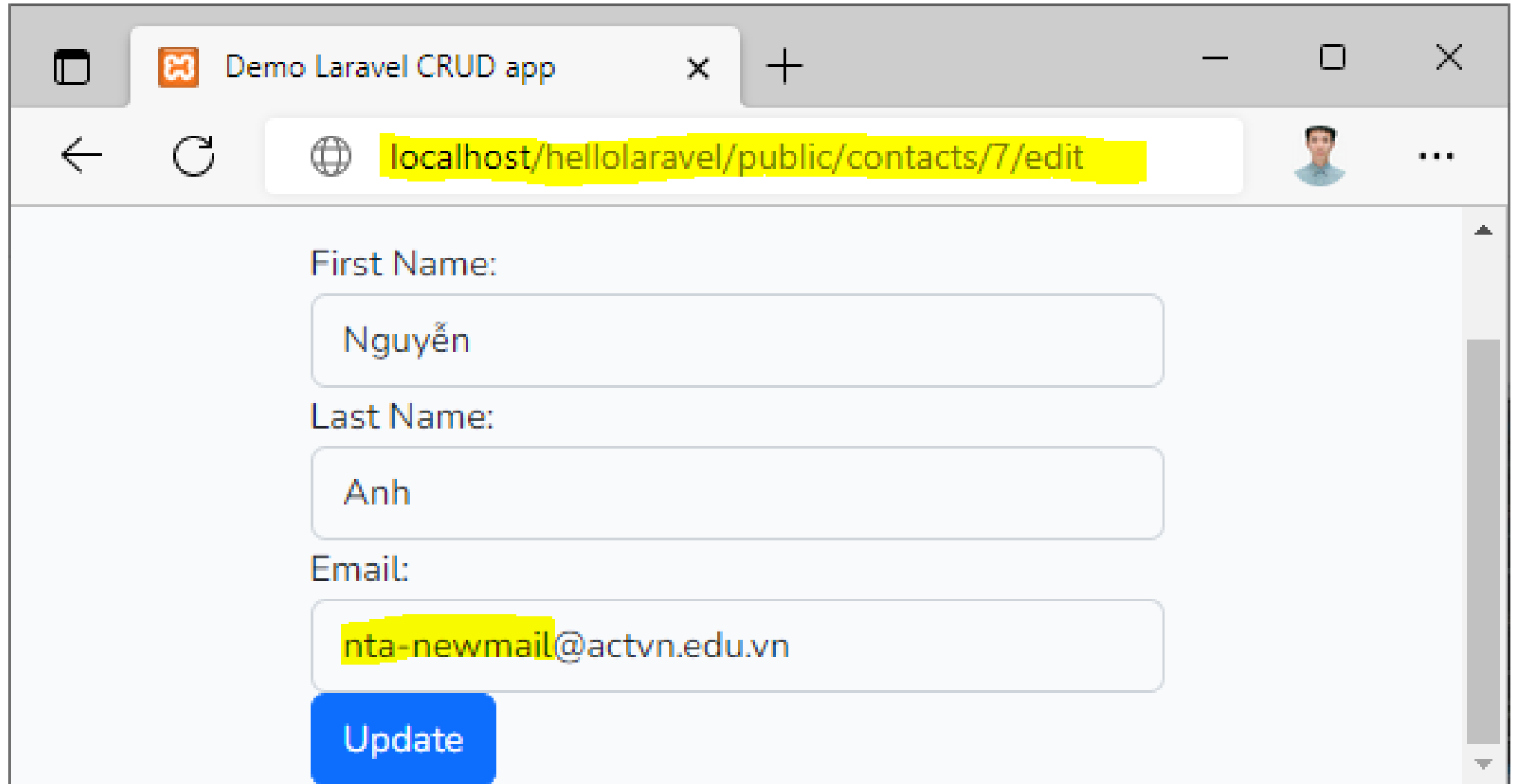
```
class ContactController extends Controller{
  public function edit($id){
    $contact = Contact::find($id);
    return view('contacts.edit', compact('contact'));
  }
  public function update(Request $request, $id) {
    $contact = Contact::find($id);
    $contact->first_name = $request->get('first_name');
    $contact->last_name = $request->get('last_name');
    $contact->email = $request->get('email');
    $contact->save();
    return redirect('/contacts')->with('success', 'Updated!');
  }
}
```

# edit.blade.php

```
@extends('layouts.base')
@section('main')
<div class="row"> <div class="col-sm-8 offset-sm-2">
<form method="post" action="{{ route('contacts.update',
                                $contact->id) }}">

    @method('PATCH')
    @csrf
    <div class="form-group">
        <label for="first_name">First Name:</label>
        <input type="text" class="form-control" name="first_name"
                value="{{ $contact->first_name }}" />
    </div>
    <!--thêm 2 form-group nữa cho 'last_name' và 'email' --!>
    <button type="submit" class="btn btn-primary">Update</button>
</form>
</div></div>
@endsection
```

# Lập trình tác vụ 'Update': edit



The screenshot shows a web browser window with the title "Demo Laravel CRUD app". The address bar displays the URL "localhost/hellolaravel/public/contacts/7/edit". The page content includes three form fields: "First Name:" with the value "Nguyễn", "Last Name:" with the value "Anh", and "Email:" with the value "nta-newmail@actvn.edu.vn". A blue "Update" button is located below the email field.

First Name:

Nguyễn

Last Name:

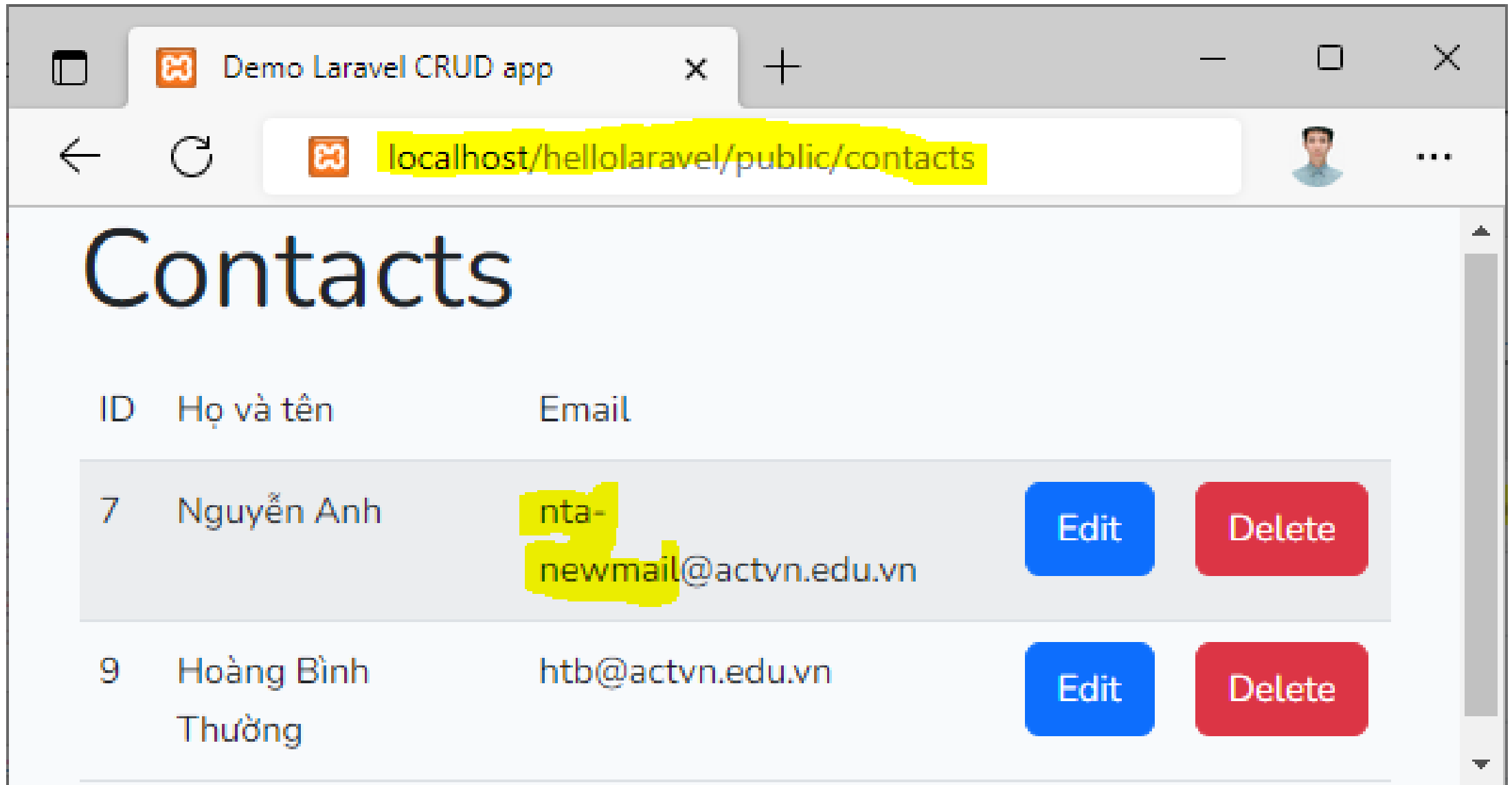
Anh

Email:

nta-newmail@actvn.edu.vn

Update

# Lập trình tác vụ 'Update': redirect



# Các bước thực hiện

- 1 Cấu hình CSDL
- 2 Tạo model 'Contact'
- 3 Khởi tạo controller 'ContactController' và thiết lập định tuyến
- 4 Lập trình tác vụ 'Create'
- 5 Lập trình tác vụ 'Read'
- 6 Lập trình tác vụ 'Update'
- 7 Lập trình tác vụ 'Delete'

# Lập trình tác vụ 'Delete'

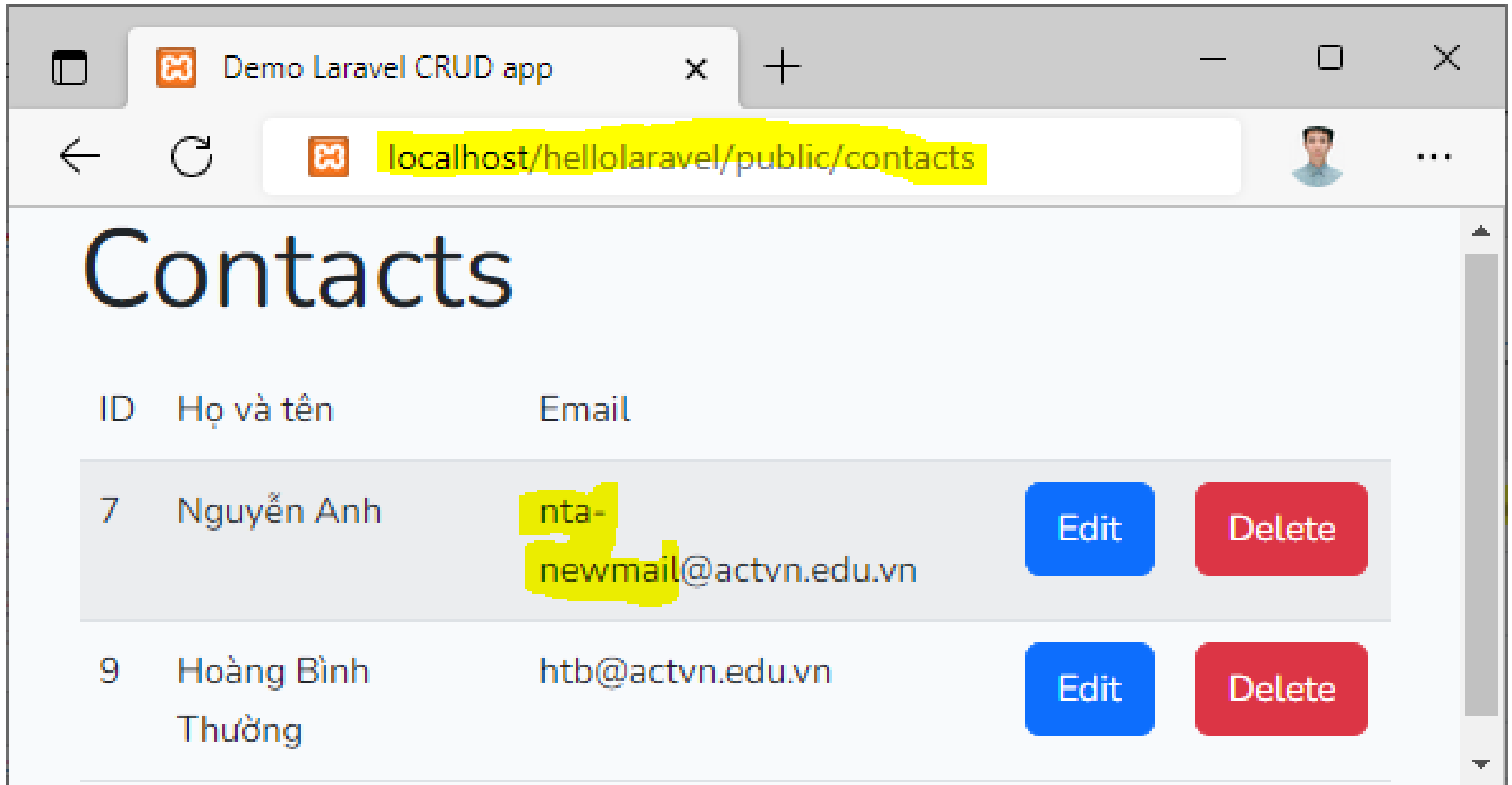
Method	Route	Controller method
GET	/contacts	index()
GET	<div>Delete: Xóa một contact</div>	
GET		
POST		
GET	/contacts/{contact}/edit	edit()
PUT/PATCH	/contacts/{contact}	update()
DELETE	/contacts/{contact}	destroy()



# Lập trình tác vụ 'Delete': controller

```
class ContactController extends Controller{  
  public function destroy($id)  
  {  
    $contact = Contact::find($id);  
    $contact->delete();  
    return redirect('/contacts')->with('success', 'Deleted!');  
  }  
  //...
```

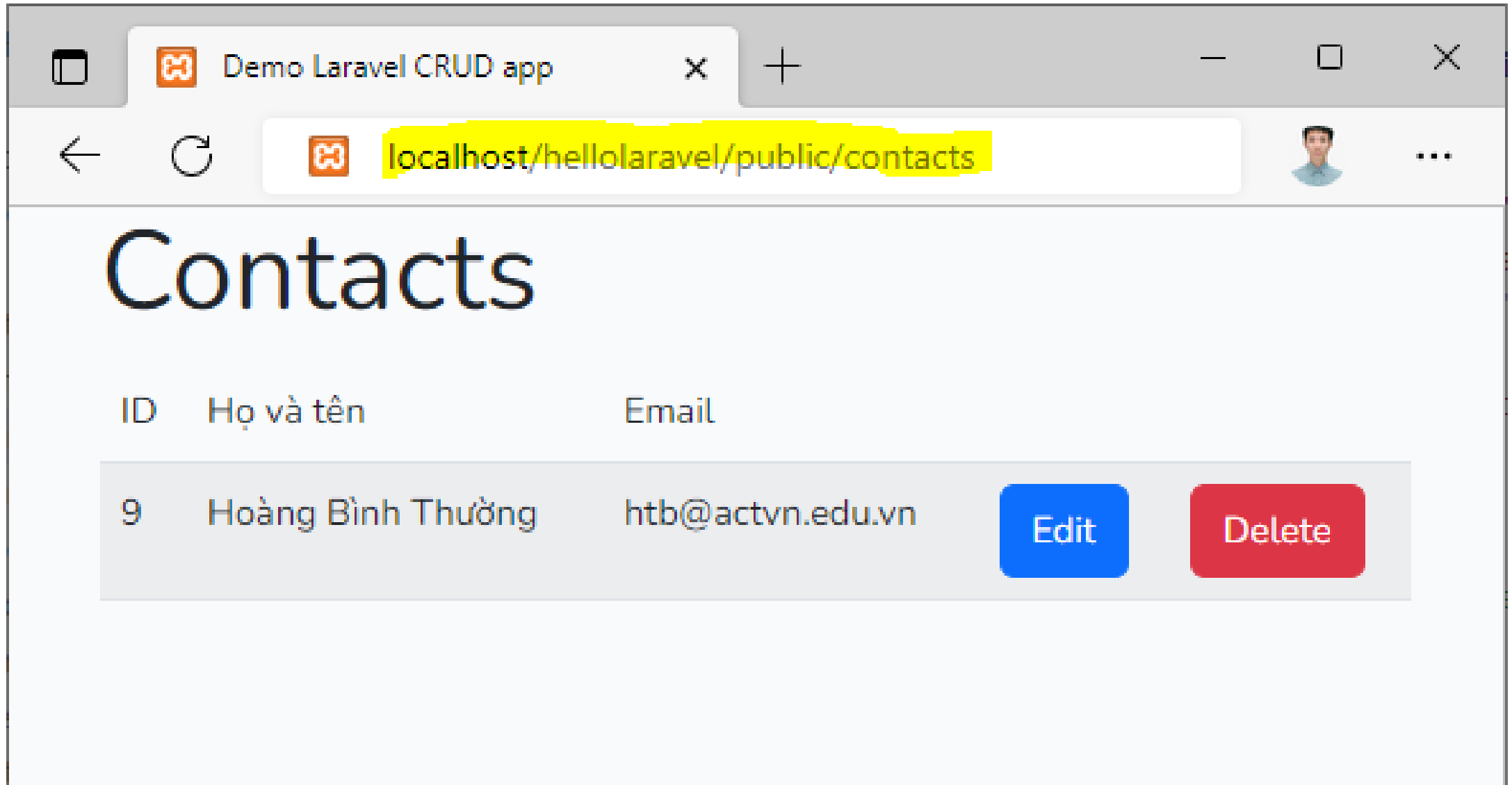
# Thực thi tác vụ 'Delete'



The screenshot shows a web browser window titled 'Demo Laravel CRUD app'. The address bar displays 'localhost/hellolaravel/public/contacts'. The page content is titled 'Contacts' and features a table with two rows of contact data. The first row is highlighted in yellow.

ID	Họ và tên	Email		
7	Nguyễn Anh	nta-newmail@actvn.edu.vn	Edit	Delete
9	Hoàng Bình Thường	htb@actvn.edu.vn	Edit	Delete

# Lập trình tác vụ 'Delete': redirect



1

Mô hình MVC

2

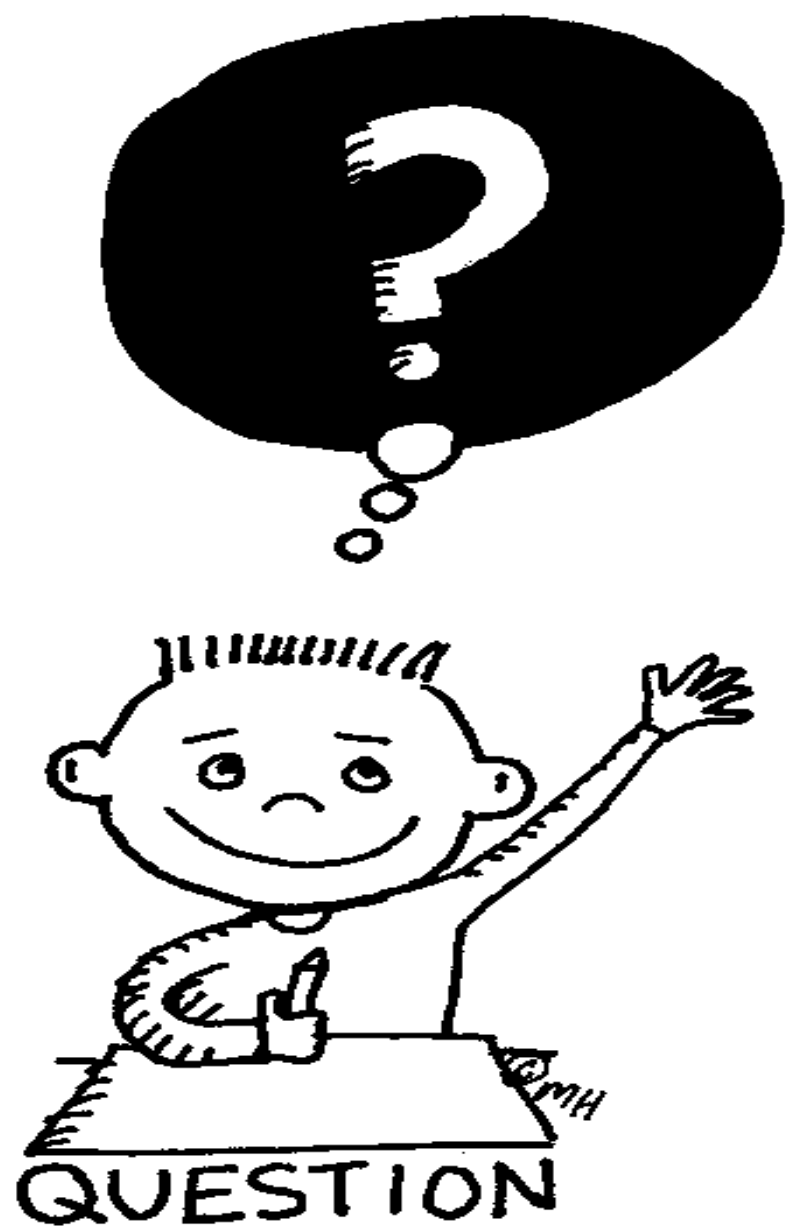
Khung phát triển ứng dụng web

3

Giới thiệu Laravel

4

Ứng dụng CRUD với Laravel



# Bài tập về nhà

1. Ở view 'index', hãy thêm nút "Thêm contact" vào cuối danh sách (để gọi tác vụ 'create')
2. Ở view 'index', hãy bổ sung mã để hiển thị hộp thoại xác nhận (confirmation) mỗi khi người dùng chọn tác vụ Delete
3. Tìm hiểu việc bổ sung Authentication Scaffolding vào một ứng dụng Laravel
4. Tìm hiểu việc triển khai ứng dụng Laravel lên môi trường vận hành