Cần tải :

PHP

MySQL

Composer : quản lý các thư việ trong php

Laravel Installer:

Docker desktop

Cài đặt : laravel new laravel\_api

Run : php artisan serve

* New database : mysql -uroot -p -e "create database laravel\_api"

php artisan migrate

* Tạo model : php artisan make:model Task –all
* 1.--migration : định nghĩa các thay đổi cơ sở dữ liệu (thêm bảng, cột, khóa ngoại, v.v.)
  + php artisan make:migration create\_{tenbang}\_table
  + Xóa toàn bộ migtate và chạy lại ( no data) : php artisan migrate:fresh
  + Xóa toàn bộ migtate và chạy lại ( yes data) : php artisan migrate:fresh –seed
* 2.--factory : Factory cho phép bạn định nghĩa cách tạo dữ liệu giả cho các trường trong model một cách ngẫu nhiên
  + php artisan make:factory name -m modelName
    - định nghĩa các trường dữ liệu giả trong factories

 return [

            'name' => fake()->sentence(),

            'price' => fake()->randomDigitNotZero(0) \* 100,

            'category\_id' => Category::pluck('id')->random(),

        ];

* + - sử dụng trong seeders

  Product::factory(5)->create();

* --seed : được sử dụng để gọi các factory và tạo dữ liệu giả cho model. Seeder được thường sử dụng để tạo dữ liệu mẫu và kiểm thử cơ sở dữ liệu
  + Tạo file : php artisan make:seeder name
  + Thực thi file : php artisan db:seed --class=name
  + Thực thi trong file chính : php artisan db:seed
* 3.model : đại diện cho các đối tượng dữ liệu trong cơ sở dữ liệu của ứng dụng.
  + php artisan make:model name
    - use HasFactory: áp dụng Factory Pattern cho model đó
      * Factory Pattern : Factory Pattern là một mẫu thiết kế trong lập trình, giúp bạn tạo ra các đối tượng mẫu (instances) của một class mà bạn định nghĩa.
    - $fillable : cho phép truy cập và gán các trường đó

    protected $fillable = ['name', 'price', 'category\_id'];

* $guarded: cho phép truy cập các trường đó
* 4.--resource : chuyển đổi dữ liệu của mô hình (model) thành dạng mà bạn muốn trả về từ API hoặc ứng dụng web của bạn
  + php artisan make:resource name

        return [

            'id' => $this->id,

            'name' => $this->name,

            'price' => $this->price,

        ];

* --controller : Controller là nơi xử lý các yêu cầu từ người dùng và thực hiện các hành động tương ứng với dữ liệu . Controller làm việc với model để truy xuất dữ liệu từ cơ sở dữ liệu và gửi dữ liệu đến view để hiển thị cho người dùng.
  + index() : Được sử dụng để hiển thị danh sách các tài nguyên
  + create() : Gửi dữ liệu đến store ( ít dùng )
  + store(Request $request) : Được sử dụng để lưu tài nguyên mới được tạo vào cơ sở dữ liệu
  + show($id) : hiển thị tài nguyên theo id
  + edit($id) : không cập nhật dữ liệu vào cơ sở dữ liệu gửi vào update ( ít dùng , dùng thẳng update cho lẹ )
  + update($id)
  + destroy($id)
  + php artisan make:controller name
    - invokable : sử dụng để xử lý các yêu cầu HTTP mà không cần định nghĩa các phương thức riêng biệt cho từng hành động (action) của controller(tạo phương thức tự định nghĩa)
* --requests : Requests giúp bạn xử lý validation và đảm bảo dữ liệu hợp lệ trước khi nó được gửi đến Controller
* --policy : Policy thường được sử dụng để kiểm soát quyền truy cập vào các tài nguyên
* Thực thi migrate : php artisan migrate –seed
* Hoàn tác migrate: php artisan migrate:rollback
* Test : php artisan tinker
* Liệt kê danh sách các route: php artisan route:list --path=api
* Trong file Kernel.php của folder http

\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class

* Laravel Sanctum : cho phép bạn dễ dàng xác thực các yêu cầu HTTP từ phía frontend của ứng dụng web
* Trạng thái (Stateful) và Stateless trong xác thực API :
  + Stateless (Không có trạng thái):
    - Không lưu trữ trạng thái: Server không theo dõi hoặc lưu trữ thông tin liên quan đến phiên làm việc của client giữa các yêu cầu.
    - Thông tin xác thực: Client phải cung cấp thông tin xác thực (thông thường là mã thông báo token) trong tiêu đề yêu cầu mỗi lần gửi yêu cầu.
      * mỗi yêu cầu phải cung cấp thông tin xác thực (như token) để xác định người dùng và quyền truy cập.
  + Stateful (Có trạng thái):
    - Lưu trữ trạng thái: Server lưu trữ thông tin về phiên làm việc của client, giúp theo dõi trạng thái và xác thực của người dùng
    - thông tin xác thực (trong trạng thái): Client gửi thông tin xác thực (như token) trong mỗi yêu cầu, nhưng server cũng có thể dựa vào thông tin trạng thái để xác thực người dùng.
      * Ví dụ mỗi lần đăng nhập thì thông tin của phiên làm việc đc lưu trữ vào cookie. Khi có yêu cầu khác thì cookie này sẽ gửi kèm theo
* Middleware EnsureFrontendRequestsAreStateful :
  + dùng để đảm bảo các yêu cầu từ phía frontend của ứng dụng web có trạng thái (stateful).
  + Khi yêu cầu từ frontend được gửi đến server, middleware này sẽ kiểm tra xem có tồn tại cookie chứa thông tin xác thực không. Nếu không có, nó sẽ không cho phép yêu cầu tiếp tục, ngăn cản việc xác thực thành công.
* Tạo user: php artisan tinker
  + app\Models\User::factory()->create()
  + Routes
    - web.php : các tuyến liên quan đến giao diện người dùng (front-end) của ứng dụng. Đây là nơi bạn định nghĩa các tuyến cho các trang, biểu mẫu, xác thực người dùng và các chức năng liên quan đến giao diện
    - api.php : xác định các tuyến (routes) dành riêng cho API của ứng dụng
* redirect : tự động nhảy qua route khác

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

    return redirect("products");

    return view('welcome');

});

* request : lấy ra query

Route::get('/products', function () {

    return request();

    return "<h1>All products</h1>";

});

* dd(request()) : trả về tất cả

  dd(request());

* request(‘name’) : trả về giá trị của name
* view : resource/view -> chứa file php html: name.blade.php

Route::get('/products/create', function () {

    return view("products.create");

});

-> trỏ đến resource/view/products/create.blade.php

Route::get('/products/{id}', function ($id) {

    return view("products.show", ['id' => $id]);

});

# **MVC và Laravel**

* Laravel triển khai mô hình MVC bằng cách sử dụng các thành phần sau
  + Routes: Định nghĩa các tuyến (routing) và kết nối URL với các phương thức trong controller.
  + Controllers: Chứa logic xử lý và điều phối. Mỗi phương thức trong controller thường tương ứng với một hành động của người dùng.
  + Models: Đại diện cho dữ liệu và tương tác với cơ sở dữ liệu.
  + Views: Blade templates để hiển thị giao diện người dùng.

# **Routing và Middleware**

* Routing:
  + Routing là quá trình điều hướng yêu cầu HTTP đến các phần tương ứng của ứng dụng dựa trên URL mà người dùng truy cập.
  + Trong Laravel, bạn định nghĩa các tuyến (routes) để ánh xạ URL đến các hành động (actions) trong controller.
* Middleware:
  + Middleware là các lớp xử lý mà yêu cầu HTTP phải đi qua trước khi đến các hành động trong controller.

# **Cơ sở dữ liệu**

* Cơ sở dữ liệu SQL:
  + Cơ sở dữ liệu SQL (Structured Query Language) sử dụng các bảng và mối quan hệ giữa chúng để lưu trữ dữ liệu
* Cơ sở dữ liệu NoSQL:
  + Cơ sở dữ liệu NoSQL không sử dụng mô hình bảng quan hệ mà thay vào đó sử dụng cấu trúc lưu trữ dữ liệu linh hoạt hơn.
* ORM (Object-Relational Mapping):
  + ORM là một kỹ thuật cho phép tương tác với cơ sở dữ liệu bằng cách sử dụng các đối tượng và mã nguồn, thay vì viết các câu truy vấn SQL trực tiếp.
* Migration
  + Migration là quá trình tạo và quản lý cấu trúc cơ sở dữ liệu, bao gồm việc tạo bảng, thêm cột, chỉnh sửa cấu trúc.
* Seeder
  + Seeder là các tập lệnh để điền dữ liệu mẫu hoặc dữ liệu kiểm thử vào cơ sở dữ liệu.
* Query Builder
  + là một cách để xây dựng và thực thi các truy vấn SQL bằng cách sử dụng mã nguồn thay vì viết trực tiếp các câu truy vấn SQL.

# **Blade Templates**

* tạo và hiển thị giao diện người dùng một cách dễ dàng và linh hoạt trong lavarel

# **Controllers và Models**

* Controllers:
  + nơi bạn định nghĩa các hành động (actions) và logic xử lý cho các yêu cầu HTTP từ người dùng.
  + thường tương tác với Models để lấy hoặc ghi dữ liệu vào cơ sở dữ liệu.
* Models:
  + đại diện cho các bảng trong cơ sở dữ liệu, giúp bạn tương tác với dữ liệu một cách dễ dàng thông qua các truy vấn và thao tác dữ liệu.
* Tương Tác giữa Controllers và Models:
  + Controllers sử dụng các Models để truy vấn dữ liệu từ cơ sở dữ liệu và thực hiện các thao tác liên quan.