OBJETIVO:

Hacer una API RESTful en Laravel

Contenido

Crear los modelos involucrados

En este caso usaré 3 entidades, user, vehicle y manufacturer. El modelo user ya viene creado por defecto.

```
php artisan make:model Manufacturer
php artisan make:model Vehicle
```

Si quiero que se genere el archivo de migraciones de una vez, en vez del anterior escribo

```
php artisan make:model -m Manufacturer
php artisan make:model -m Vehicle
```

Definir los modelos y sus relaciones

Establecer los campos de cada modelo y los tipos de relaciones

El modelo **User** tendrá los campos: id, first_name, last_name, email, password, role ['client','operator','admin']

El modelo **Vehicle** tendrá los campos: id, manufacturer_id, model, color, created_at, updated_at

Estos 2 últimos campos serán ocultos del json que se devuelve como respuesta. Entonces:

1 Vehicle *pertenece_a* **1 Manufacturer** *(belongsTo)*

1 Manufacturer *fabrica* **N Vehicle** *(hasMany)*

Considerar que un modelo generalmente tiene 3 elementos claves, *protected $table, protected $fillable* y las relaciones definidas de la siguiente forma *public function [ENTIDAD]* donde *ENTIDAD* es la otra entidad con quien está relacionado el modelo. Entonces queda:

Vehicle Model

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Vehicle extends Model
{
    protected $table = 'vehicles';
    protected $fillable = array('manufacturer_id', 'model', 'color');

    /* Relationship: 1 Vehicle BelongsTo 1 Manufacturer */
    public function manufacturer()
    {
        return $this->belongsTo(Manufacturer::class, 'manufacturer_id');
    }
}
```

Manufacturer Model

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Manufacturer extends Model
{
    protected $table = "manufacturers";
    protected $fillable = array('name', 'website');

    // Hide those field in json
    protected $hidden = ['created_at', 'updated_at'];

    // Relationship: 1 Manufacturer hasMany N Vehicles
    public function vehicle()
    {
        return $this->hasMany(Vehicle::class);
    }
}
```

User Model

```php
class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'first_name', 'last_name', 'email', 'password', 'role',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function getFullNameAttribute(){
        return $this->first_name . ' ' . $this->last_name;
    }
}
```

Configurar la base de datos en archivo .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lar_apirestful
DB_USERNAME=riv4wi
DB_PASSWORD=1234
```

Crear providers para libreria Faker (Generar datos de pueba automáticamente)

VehicleBrandProvider.php

```php
<?php
namespace Faker\Provider;

class VehicleBrandProvider extends \Faker\Provider\Base
{

  protected static $brand = array(
      'Chevrolet',
      'Fiat',
      'Renault',
      'Ford',
      'Toyota',
      'Jeep',
      'Volkswagen',
      'Hyundai',
      'Mitsubishi',
      'Honda',
      'Chrysler',
      'Nissan',
      'Peugeot',
      'Citroën',
  );

  public function brand(){
      return static::randomElement(static::$brand);
  }
}
```

VehicleModelProvider.php

```php
<?php
namespace Faker\Provider;

class VehicleModelProvider extends \Faker\Provider\Base
{

  protected static $modelveh = array(
      "Chevrolet" => array('Aveo', 'Corsa', 'Malibú', 'Optra', 'Silverado', 'Spark', 'Onix Joy',
```

```
                    'Onix', 'Onix Activ', 'Prisma Joy', 'Cobalt', 'Cruze 5', 'Cruze', 'Tracker',
                    'Captiva', 'Trailblazer', 'Camaro Coupé', 'Camaro Convertible',),

    "Fiat" => array('Palio', 'Premio', 'Sienna', 'Spazio', 'Uno'),
    "Renault" => array('Twingo', 'Logan', 'Clio', 'Mégane', 'Symbol', 'Kangoo', 'Scénic', 'R19', 'Sandero'),
    "Ford" => array('Explorer', 'F-150', 'Ka', 'Ka+', 'Fiesta', 'Fiesta 4P', 'Focus', 'Focus 4P', 'Mondeo', 'Mustang',),
    "Toyota" => array('4Runner', 'Corolla', 'Hilux', 'Macho', 'Yaris'),
    "Jeep" => array('Cherokee', 'Grand Cherokee', 'Wagoneer', 'Wrangler', 'Compass', 'Commander', 'Willys', 'Comanche'),
    "Volkswagen" => array('Gol', 'Fox', 'Spacefox', 'Bora', 'Escarabajo', 'Jetta', 'Kombi', 'Crossfox', 'Polo'),
    "Hyundai" => array('Elantra', 'Tucson', 'Getz', 'Accent', 'Santa Fe', 'Excel', 'Sonata', 'H1', 'Galloper'),
    "Mitsubishi" => array('Lancer', 'Montero', 'Signo', 'L-300', 'MF', 'Galant', 'Panel', 'MX', 'Eclipse'),
    "Honda" => array('Civic', 'Accord', 'Fit', 'CR-V', 'Odyssey', 'Pilot', 'Prelude', 'CRX', 'Integra', 'Legend'),
    "Chrysler" => array('Neon', 'Sebring', 'Caliber', '300 C', 'Grand Caravan', 'Town & Country', 'Le baron', 'Spirit', 'Stratus'),
    "Nissan" => array('March', 'Kicks', 'Frontier ', 'Sentra', 'Tiida', 'Pathfinder', 'X-Trail', '350Z', 'Patrol', 'Almera', 'Murano'),
    "Peugeot" => array('206', '207', '307', '407', 'Expert', '205', '306', '605'),
    "Citroën" => array('C3 Aircross', 'C4 Cactus', 'C4 Grand Picasso', 'C4 Lounge', 'C4 Picasso', 'C-Elysée'),
  );


  public function modelveh($brand)
  {
    return static::randomElement(static::$modelveh[$brand]);
  }

}
```

Crear las migraciones

Si las migraciones no fueron creadas al momento de crear los modelos, entonces se crean individualmente con:

```
php artisan make:migration create_manufacturers_table --create=manufacturers
php artisan make:migration create_vehicles_table --create=vehicles
php artisan migrate:install
```

--Instala la tabla migrations para el control de las migraciones

Añadir el código de las migraciones

Manufacturers

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```php
class CreateManufacturersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('manufacturers', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('website');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('manufacturers');
    }
}
```

Vehicles

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateVehiclesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('vehicles', function (Blueprint $table) {
            $table->increments('id');
            $table->string('manufacturer_id');
            $table->string('model');
            $table->string('color');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('vehicles');
    }
}
```

User

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('first_name');
            $table->string('last_name');
            $table->string('email')->unique();
            $table->string('password');
            $table->enum('role', ['client', 'operator', 'admin']);
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Instalar y actualizar la librería fzaninotto/faker para crear datos de relleno

```
composer search faker
composer require fzaninotto/faker --dev
```

--dev para que lo instale en el entorno de desarrollo

Poblar la base de datos - Creación de seeders y model factories

Crear los archivos ManufacturerSeeder y VehicleSeeder

```
php artisan make:seeder ManufacturerTableSeeder
php artisan make:seeder VehicleTableSeeder
```

Añadir factory para el modelo User

ModelFactory.php (User)

```php
/* Factory User */
$factory->define(App\User::class, function (Faker\Generator $faker) {
    $faker = Faker\Factory::create('es_ES');  // Init Faker in spanish mode
    return [
        'first_name' => $faker->firstName('male' | 'female'),
        'last_name' => $faker->lastName,
        'email' => $faker->unique()->safeEmail,
        'role' => $faker->randomElement(['client', 'operator', 'admin']),
        'password' => bcrypt(str_random(10)),
        'remember_token' => str_random(10),
    ];
});
```

Añadir el código correspondiente al seeder de User

UserTableSeeder

```php
<?php

use Illuminate\Database\Seeder;

class UserTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        factory(App\User::class)->create([
            'first_name'    => 'Pink',
            'last_name'     => 'Panther',
            'email'         => 'pp@gmail.com',
            'role'      => 'admin',
            'password'      => bcrypt('123'),
        ]);

        factory(App\User::class, 10)->create();
    }
}
```

Añadir el código correspondiente al seeder de Manufacturer

ManufacturerTableSeeder

```
use \App\Manufacturer as Manufacturer;

class ManufacturerTableSeeder extends Seeder
{
    public function run()
    {
        //        With DB not create create_at and updated_at fields
        //        Add: use Illuminate\Support\Facades\DB;
        //        DB::table('manufacturers')->insert([
        //            'name' => 'Chevrolet',
        //            'website' => 'chevrolet.com',
        //        ]);

        Manufacturer::create([
            'name' => 'Chevrolet',
            'website' => 'chevrolet.com',
        ]);

        Manufacturer::create([
            'name' => 'Fiat',
            'website' => 'fiat.com',
        ]);

        Manufacturer::create([
            'name' => 'Renault',
            'website' => 'renault.com',
        ]);

        Manufacturer::create([
            'name' => 'Ford',
            'website' => 'ford.com',
        ]);

        Manufacturer::create([
            'name' => 'Toyota',
            'website' => 'toyota.com',
        ]);
    }
}
```

Añadir factory para el modelo Vehicle

ModelFactory.php (Vehicle)

```php
/* Factory Vehicle */
$factory->define(App\Vehicle::class, function (Faker\Generator $faker) {
    $faker->addProvider(new Faker\Provider\VehicleBrandProvider($faker));
    $faker->addProvider(new Faker\Provider\VehicleModelProvider($faker));
    $brandi = $faker->brand;

    $manufacturer = DB::select("SELECT id FROM manufacturers where name = '".$brandi."'");
    return [
        'manufacturer_id' => $manufacturer[0]->id,
        'model' => $faker->modelveh($brandi),
        'color' => $faker->ColorName(),
    ];
});
```

Añadir el código correspondiente al seeder de Vehicle

VehicleTableSeeder.php

```php
<?php
use Illuminate\Database\Seeder;

class VehicleTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        factory(App\Vehicle::class, 100)->create();
    }
}
```

Definir las llamadas en el archivo principal DatabaseSeeder.php

DatabaseSeeder.php

```php
class DatabaseSeeder extends Seeder
{
    public function run()
    {
        $this->call(UserTableSeeder::class);
        $this->call(ManufacturerTableSeeder::class);
        $this->call(VehicleTableSeeder::class);
    }
```

```
}
```

```
php artisan db:seed
```

ó

```
php artisan migrate:refresh --seed
```

-- Ejecutar los seeders

Deshabilitar validación VerifyCsrfToken

\App\Http\Middleware\VerifyCsrfToken::class,

-- Comentar esta línea en el archivo Http/Kernel.php ya que no se usarán validaciones de formularios porque es una API

Crear controladores

```
php artisan make:controller ManufacturerController --resource
php artisan make:controller VehicleController --resource
```

Agregar código a Controladores

ManufacturerController.php

```
<?php

namespace App\Http\Controllers;

use App\Manufacturer;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

/* Methods create() and edit() are not include because they are call
to the forms to process those requests */

class ManufacturerController extends Controller
{

  /**
   * Display a listing of the resource.
   *
   * @return \Illuminate\Http\Response
   */
  public function index()
  {
```

```
    // Other way, with collection, but slower >>> return response()->json(['data' => Manufacturer::all()], 200);
     return response()->json(['data' =>  DB::select("SELECT * FROM manufacturers")], 200);
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    if (!$request->get('name') || !$request->get('website'))
        return response()->json(['msg' => 'Data not completed'], 422);
    else {
        $manufacturer = Manufacturer::create($request->all());
        return response()->json(['data' => $manufacturer, 'msg' => 'Manufacturer created'], 201);
    }
}

/**
 * Display the specified resource.
 *
 * @param  int $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $manufacturer = Manufacturer::find($id);

    if ($manufacturer)
        return response()->json(['data' => $manufacturer], 200);
    else
        return response()->json(['msg' => 'Manufacturer ' . $id . ' not found'], 404);
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request $request
 * @param  int $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $method = $request->method();
```

```
    $manufacturer = Manufacturer::find($id);
    if (!$manufacturer) {
        return response()->json(['msg' => 'Manufacturer ' . $id . ' not found'], 404);
    }

    $name = $request->get('name');
    $website = $request->get('website');

    /* Method PATCH */
    if ($method === 'PATCH') {
        /* Update name */
        if ($name != null && $name != '') {
            $manufacturer->name = $name;
        }

        /* Update website */
        if ($website != null && $website != '') {
            $manufacturer->website = $website;
        }

        $manufacturer->save();
        return response()->json(['msg' => 'Manufacturer \'s record ' . $id . ' edit with PATCH'], 200);
    }

    /* Method PUT */
    if (!$name || !$website) {
        return response()->json(['msg' => 'Data not completed'], 422);
    } else {
        $manufacturer->name = $name;
        $manufacturer->website = $website;
        $manufacturer->save();
        return response()->json(['msg' => 'Manufacturer \'s record ' . $id . ' edit with PUT'], 200);
    }
}

/**
 * Remove the specified resource from storage.
 *
 * @param  int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    $manufacturer = Manufacturer::find($id);

    if (!$manufacturer) {
```

```
        return response()->json(['msg' => 'Manufacturer ' . $id . ' not found'], 404);
    }

    // Other way, with collection, but slower >>> $vehicles = $manufacturer->vehicle;
    $vehicles = DB::select("SELECT id FROM vehicles where manufacturer_id = '".$id."'");
    if (sizeof($vehicles) > 0) {
        return response()->json(['msg' => 'The manufacturer can not be eliminated because it has associated vehicles.
         Eliminate the vehicles first'], 200);
    }
    $manufacturer->delete();
    return response()->json(['msg' => 'Manufacturer ' . $id . ' eliminated'], 200);
  }
}
```

VehicleController.php

```
<?php

namespace App\Http\Controllers;

use App\Manufacturer;
use App\Vehicle;
use Illuminate\Http\Request;

class VehicleController extends Controller
{

  /**
   * Display a listing of the resource.
   *
   * @return \Illuminate\Http\Response
   */
  public function index($manufacturer_id)
  {
    $manufacturer = Manufacturer::find($manufacturer_id);

    $vehicles = $manufacturer->vehicles;

    if ($vehicles)
        return response()->json(['data' => $vehicles], 200);
    else
        return response()->json(['msg' => 'Manufacturer without vehicles'], 404);
  }

  /**
```

```
    * Store a newly created resource in storage.
    *
    * @param  \Illuminate\Http\Request $request
    * @return \Illuminate\Http\Response
    */
   public function store(Request $request, $manufacturer_id)
   {
       if (!$request->get('model') || !$request->get('color'))
           return response()->json(['msg' => 'Datos incompletos'], 422);
       else {
           $manufacturer = Manufacturer::find($manufacturer_id);
           if (!$manufacturer)
               return response()->json(['msg' => 'Manufacturer is not exist.'], 404);
           else {
               /*Vehicle::create($request->all());*/
               Vehicle::create(['manufacturer_id' => $manufacturer_id, 'model' => $request->get('model'), 'color' => $request-
>get('color')]);
               return response()->json(['mgs' => 'Vehicle of manufacturer ' . $request->get('manufacturer_id') . ' it was created'],
201);
           }
       }
   }

   /**
    * Display the specified resource.
    *
    * @param  int $id
    * @return \Illuminate\Http\Response
    */
   public function show($manufacturer_id, $vehicle_id)
   {
       $vehicle = Vehicle::where('manufacturer_id', '=', $manufacturer_id)
           ->where('id', '=', $vehicle_id)
           ->limit(1)
           ->get();

       if ($vehicle)
           return response()->json(['data' => $vehicle], 200);
       else
           return response()->json(['msg' => 'Manufacturer without vehicles'], 404);
   }

   /**
    * Show the form for editing the specified resource.
    *
    * @param  int $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function edit($manufacturer_id, $vehicle_id)
    {
        return "Editando el vehiculo $vehicle_id del fabricante $manufacturer_id";
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $manufacturer_id, $vehicle_id)
    {
        $manufacturer = Manufacturer::find($manufacturer_id);
        if (!$manufacturer) {
            return response()->json(['msg' => 'Manufacturer ' . $manufacturer_id . ' not found'], 404);
        }

        $vehicle = Vehicle::find($vehicle_id);
        if (!$vehicle) {
            return response()->json(['msg' => 'Vehicle ' . $vehicle_id . ' of manufacturer ' .
                $manufacturer_id . ' not found'], 404);
        }

        $model = $request->get('model');
        $color = $request->get('color');
        $method = $request->method();

        /* Method PATCH */
        if ($method === 'PATCH') {

            $edited = false;

            /* Update model */
            if ($model != null && $model != '') {
                $vehicle->model = $model;
                $edited = true;
            }

            /* Update color */
            if ($color != null && $color != '') {
                $vehicle->color = $color;
                $edited = true;
```

```
        }

        if ($edited) {
            $vehicle->save();
            return response()->json(['msg' => 'Vehicle ' . $vehicle_id . ' of manufacturer ' . $manufacturer_id .
                ' edited with PATCH'], 200);
        }
        else{
            return response()->json(['msg' =>'There wasn\'t changes']);
        }
    }

    /* Method PUT */
    if (!$model || !$color) {
        return response()->json(['msg' => 'Data not completed'], 422);
    } else {
        $vehicle->model = $model;
        $vehicle->color = $color;
        $vehicle->save();
        return response()->json(['msg' => 'Vehicle ' . $vehicle_id . ' of manufacturer ' . $manufacturer_id .
            ' edited with PUT'], 200);
    }
}

/**
 * Remove the specified resource from storage.
 *
 * @param  int $id
 * @return \Illuminate\Http\Response
 */
public function destroy($manufacturer_id, $vehicle_id)
{
    $manufacturer = Manufacturer::find($manufacturer_id);
    if (!$manufacturer) {
        return response()->json(['msg' => 'Manufacturer ' . $manufacturer_id . ' not found'], 404);
    }

    $vehicle = Vehicle::find($vehicle_id);
    if (!$vehicle) {
        return response()->json(['msg' => 'Vehicle ' . $vehicle_id . ' of manufacturer ' .
            $manufacturer_id . ' not found'], 404);
    }

    /* Method DELETE */
    $vehicle->delete();
    return response()->json(['msg' => 'Vehicle ' . $vehicle_id . ' of manufacturer ' . $manufacturer_id .
```

```
        'eliminated'], 200);
    }
}
```

Agregar rutas de los controladores

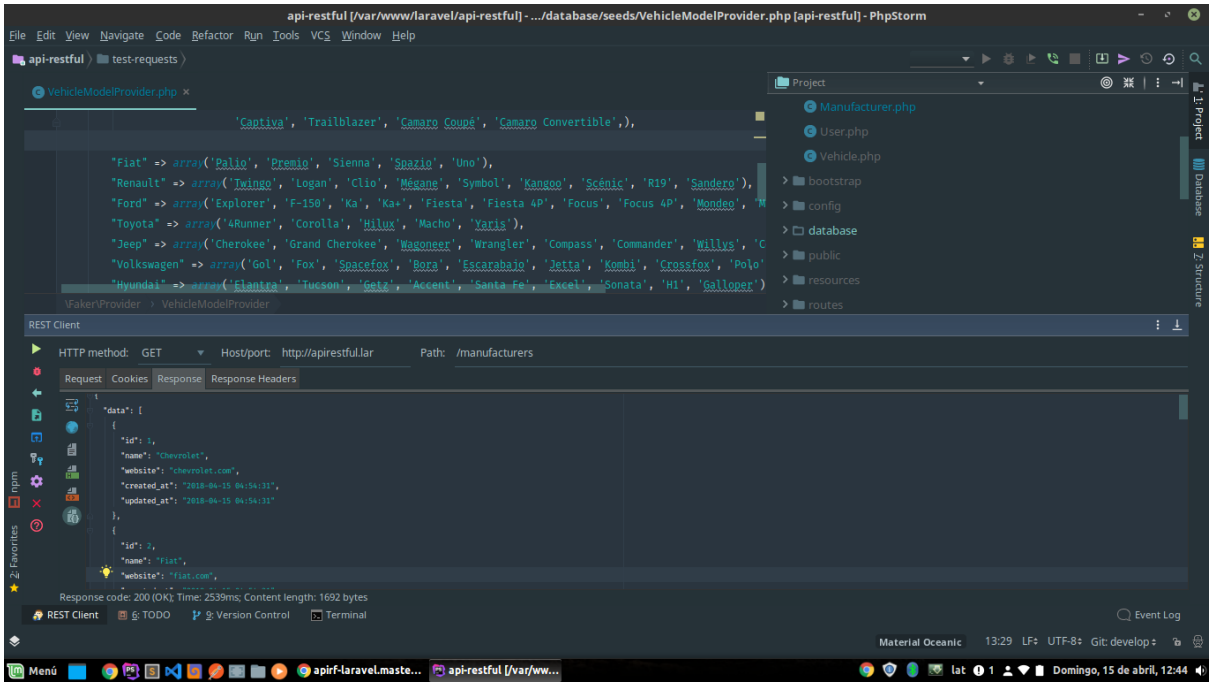
Route::resource('manufacturers', 'ManufacturerController');

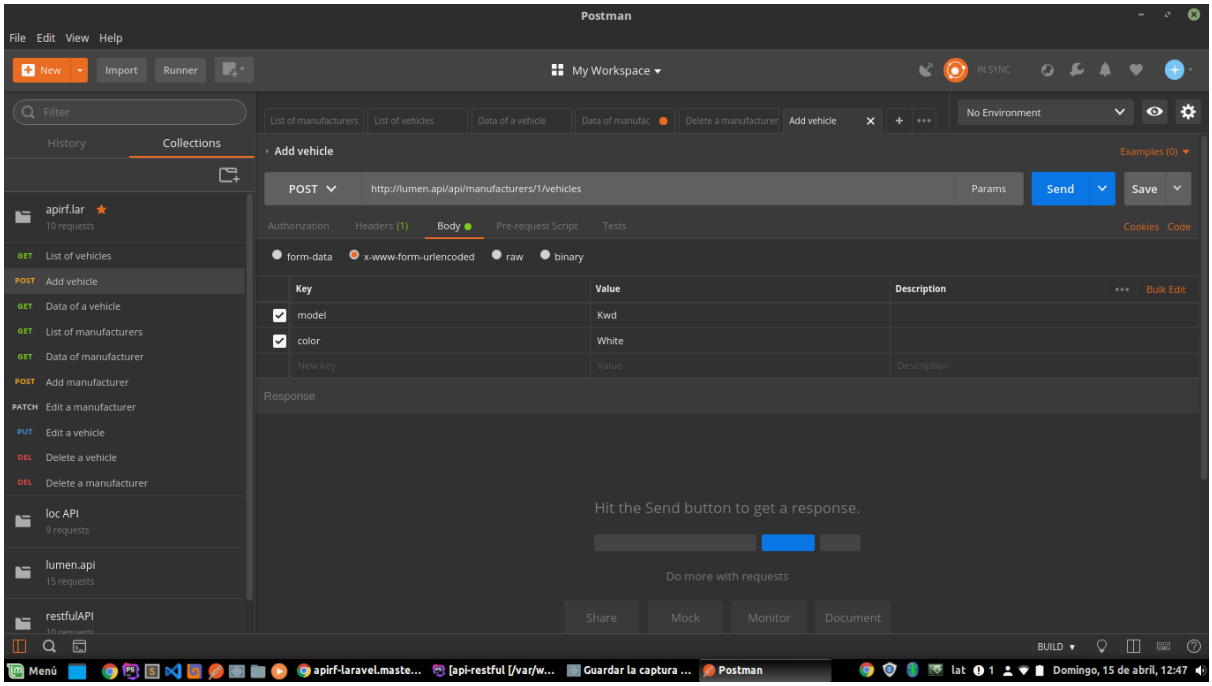
Route::resource('manufacturers.vehicles', 'VehicleController');

-- En archivo web.php

Testear la api

Con herramientas con Postman o similar, o con la que viene integrada en phpstorm. Hay archivos incluidos ya elaborados en el directorio **test-requests**. Sólo queda importar las peticiones desde Postman o el Client REST Tool de phpstorm




Postman
File Edit View Help
New
Import
Runner
My Workspace
IN SYNC
Filter
History
Collections
apirf.lar
10 requests
GET List of vehicles
POST Add vehicle
GET Data of a vehicle
GET List of manufacturers
GET Data of manufacturer
POST Add manufacturer
PATCH Edit a manufacturer
PUT Edit a vehicle
DEL Delete a vehicle
DEL Delete a manufacturer
loc API
9 requests
lumen.api
15 requests
restfulAPI
List of manufacturers
List of vehicles
Data of a vehicle
Data of manufac
Delete a manufacturer
Add vehicle
No Environment
Add vehicle
Examples (0)
POST
http://lumen.api/api/manufacturers/1/vehicles
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
Key
Value
Description
Bulk Edit
model
Kwd
color
White
New key
Value
Description
Response
Hit the Send button to get a response.
Do more with requests
Share
Mock
Monitor
Document
BUILD
Menú
apirf-laravel.maste...
[api-restful [/var/w...
Guardar la captura ...
Postman
lat
Domingo, 15 de abril, 12:47