

Hacer una API RESTful en Laravel

Contenido

Preparar entorno.....	2
Crear los modelos involucrados.....	2
Definir los modelos y sus relaciones.....	2
Establecer los campos de cada modelo y los tipos de relaciones.....	2
Crear providers para librería Faker (Generar datos de pueba automáticamente).....	3
Crear las migraciones.....	3
Añadir el código de las migraciones.....	3
Instalar y actualizar la librería fzaninotto/faker para crear datos de relleno.....	3
Poblar la base de datos - Creación de seeders y model factories.....	4
Crear los archivos ManufacturerSeeder y VehicleSeeder.....	4
Definir las llamadas en el archivo principal DatabaseSeeder.php.....	4
Ejecutar los seeders.....	4
Deshabilitar validación VerifyCsrfToken.....	4
Crear controladores.....	4
Agregar código a Controladores.....	5
Agregar rutas de los controladores.....	5
Testear la api.....	5
Referencias web adicionales.....	6



Preparar entorno

Información en <https://github.com/riv4wi/document-and-notes/blob/master/nuevo-proyecto-con-laravel-composer-apache.pdf>

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.

Vehicle.php

Manufacturer.php

User.php

Configurar la base de datos en archivo .env

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



Repositorio: <https://github.com/riv4wi/apirf-laravel>

Rama: **Master**

Nombre-documento: [apirf-laravel.master](#)

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

VehicleBrandProvider.php

VehicleModelProvider.php

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

2018_04_11_040105_create_manufacturers_table.php

2018_04_11_041204_create_vehicles_table.php

2014_10_12_000000_create_users_table.php

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
```



Repositorio: <https://github.com/rivqwi/apirf-laravel>

Rama: **Master**

Nombre-documento: [apirf-laravel.master](#)

Añadir factory para los modelos

ModelFactory.php (section Users)

ModelFactory.php (Vehicle)

Añadir el código correspondiente a los seeders

UserTableSeeder.php

ManufacturerTableSeeder.php

VehicleTableSeeder.php

Definir las llamadas en el archivo principal DatabaseSeeder.php

DatabaseSeeder.php

Ejecutar los seeders

```
php artisan db:seed
```

ó

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

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

VehicleController.php

Agregar rutas de los controladores

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



Repositorio: <https://github.com/rivqwi/apirf-laravel>

Rama: **Master**

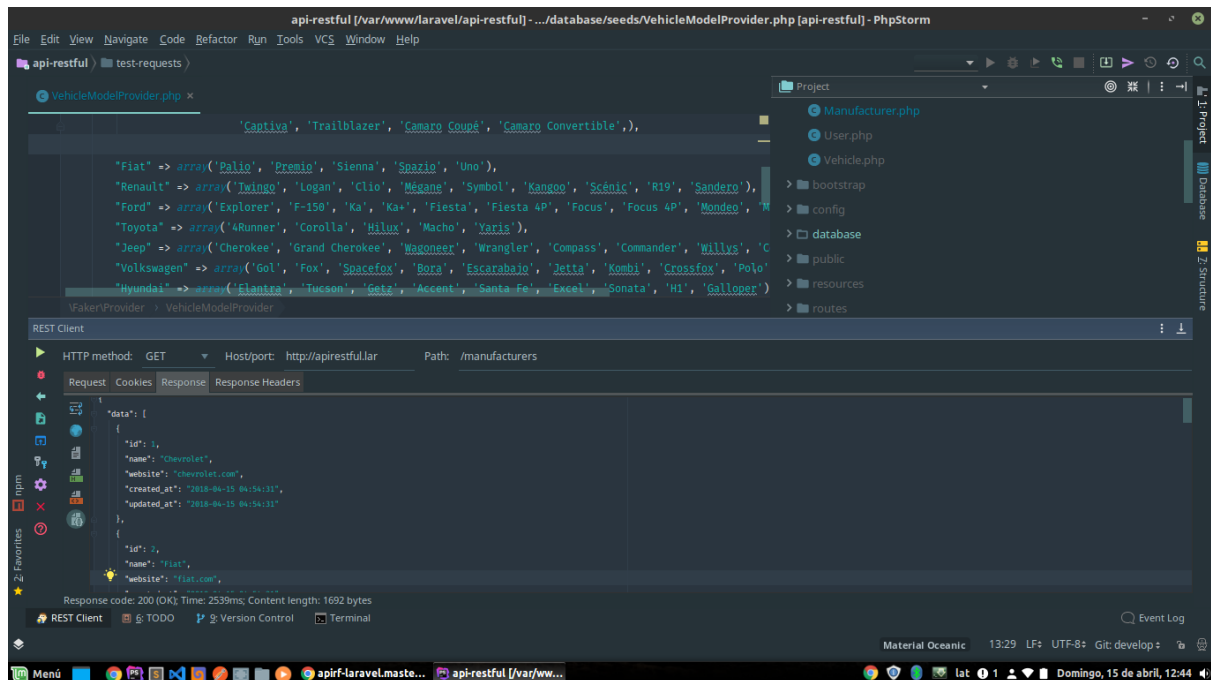
Nombre-documento: [apirf-laravel.master](#)

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

-- En archivo api.php

Testear la api

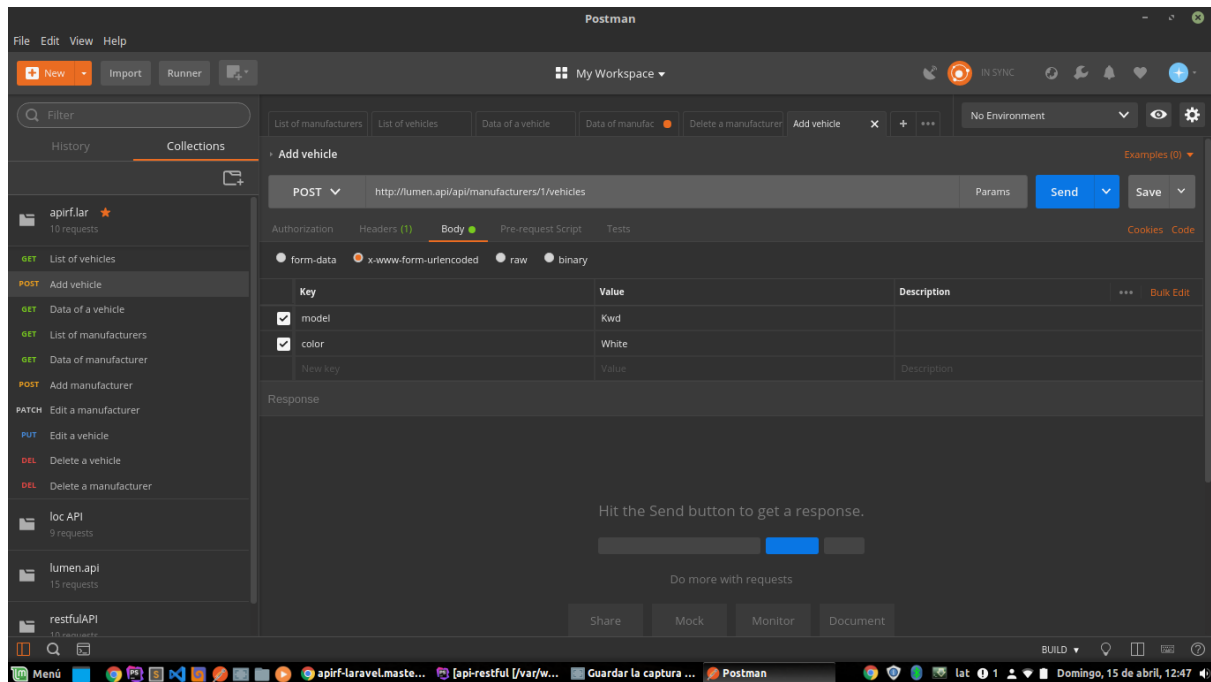
Con herramientas con Postman o similar, o con la que viene integrada en phpstorm. Hay archivos incluidos ya elaborados en el directorio **test-requests**. Sólo queda importar las peticiones desde Postman o el Client REST Tool de phpstorm



Repositorio: <https://github.com/rivqwi/apirf-laravel>

Rama: **Master**

Nombre-documento: **apirf-laravel.master**



Referencias web adicionales

<https://laravel.com/docs/5.6>

<https://stye.net/como-usar-el-componente-de-traduccion-de-laravel/>

<https://medium.com/teknomuslim/multi-language-api-response-messages-in-laravel-5c9029a32e5c>



Repositorio: <https://github.com/rivqwi/apirf-laravel>

Rama: **Master**

Nombre-documento: [apirf-laravel.master](#)