Documentación de API REST

Tecnologías usadas
PHP con Composer y Artisan
MySQL
Laravel con ORM Eloquent

Software para testing
Postman

Repositorio y manejo de versión
GIT y GITHUB

Link de Repositorio: https://github.com/aleunsj5/mailUp.git

La API desarrollada permite realizar las distintas acciones de un CRUD que persiste en una base de datos gestionada por el motor de MYSQL utilizando el entorno gráfico de phpMyAdmin. Las acciones que realiza la API se enumeran a continuación:

1. Crear un producto
2. Obtener todos los productos (Implementar una lógica de paginación y de búsqueda por nombre)
3. Obtener información de un producto
4. Modificar información de un producto
5. Eliminar un producto

Migration:

```php
<?php

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

class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->increments('product_id');
            $table->string('name','500');
            $table->text('description')->nullable();
            $table->string('image','500');
            $table->string('brand','200');
            $table->integer('price');
            $table->integer('price_sale');
            $table->string('category','200');
            $table->integer('stock');
            $table->timestamps();      You, an hour ago • Consumo de api y persistencia de datos
        });
    }

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

Seeder:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Products;
use Carbon\Carbon;

class ProductsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Products::truncate();

        $faker = \Faker\Factory::create();

        for ($i = 0; $i < 50; $i++) {
            Products::create([
                'name' => $faker->name,
                'description' => $faker->paragraph,
                'image'=>$faker->imageUrl(600,400),
                'brand'=>$faker->randomElement(['Apple','Asus','HP','Lenovo','Acer']),      You, an hour ago • Consumo de api y persi
                'price'=>$faker->numberBetween($min = 1000, $max = 9000),
                'price_sale'=>$faker->numberBetween($min = 1350, $max = 11500),
                'category'=>$faker->randomElement(['MacBook Air', 'MacBook Pro', 'Nitro', 'ROG','Envy','IThink','ThinkPad']),
                'stock'=>$faker->numberBetween($min = 0, $max = 100),
                'creation_date'=>Carbon::now(),
            ]);
        }
    }
}
```

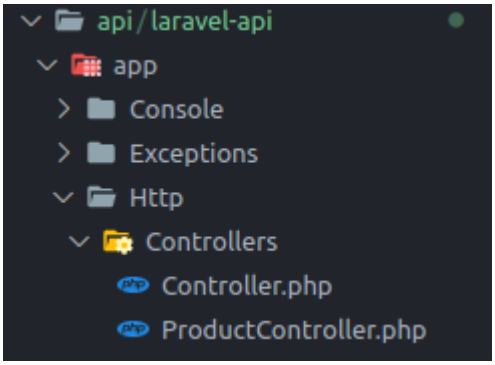
Configuración y conexión de base de datos desde archivo .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mailUp
DB_USERNAME=root
DB_PASSWORD=password
```

Endpoints correspondientes:

```php
Route::get('products/getAll', [ProductController::class,'getAllProducts']);
Route::get('products/getProduct/{id}', [ProductController::class,'getProductForId']);
Route::post('products/insert', [ProductController::class,'insertProduct']);
Route::post('products/searchProduct', [ProductController::class,'searchProductForName']);
Route::put('products/update/{id}', [ProductController::class,'updateProduct']);
Route::delete('products/delete/{id}', [ProductController::class,'deleteProduct']);
```

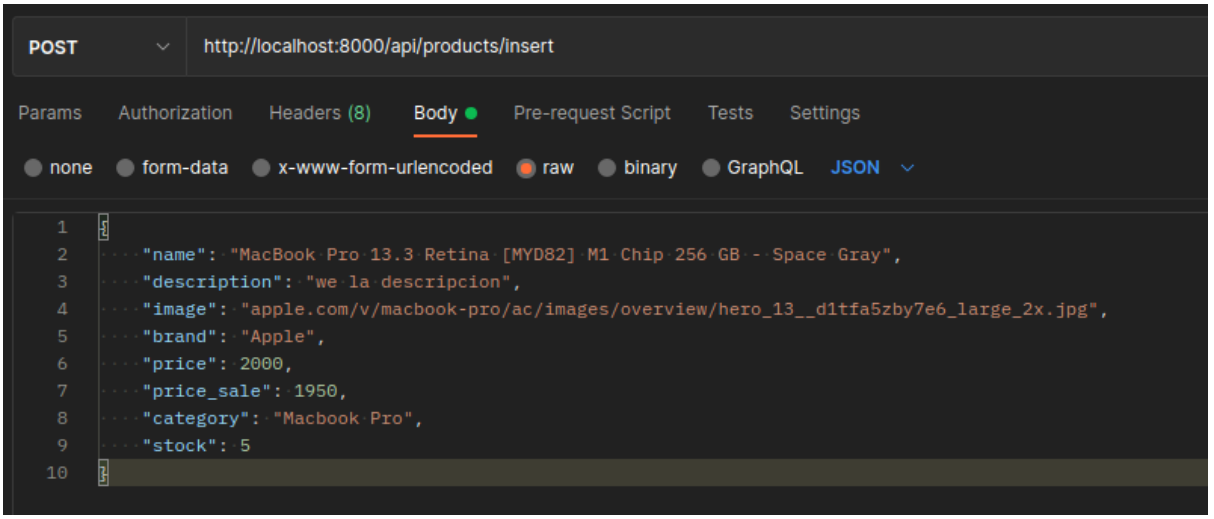
Cada petición es recibida en el controlador definido para los productos:



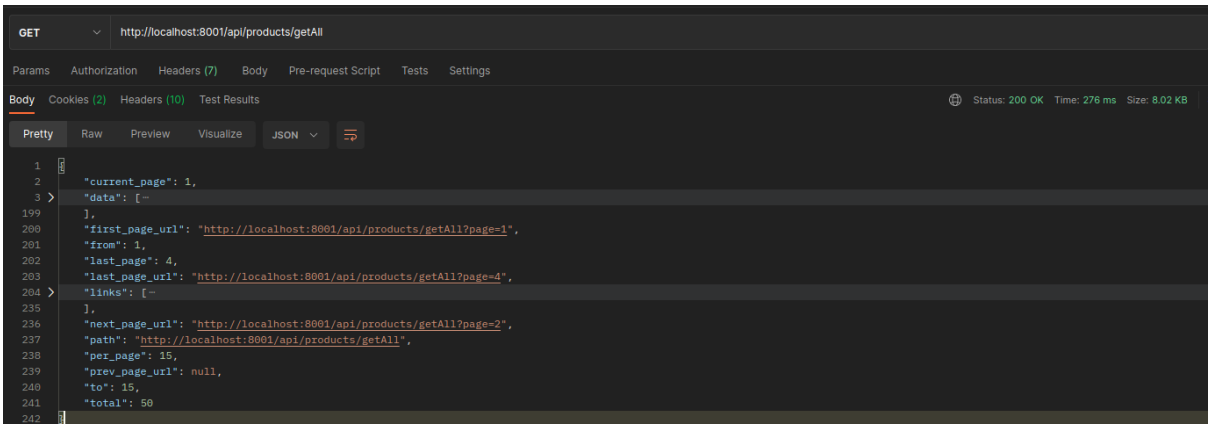
```php
<?php

namespace App\Http\Controllers;

use App\Models\Products;
use Illuminate\Http\Request;
use App\Exceptions\Handler;
use Carbon\Carbon;

class ProductController extends Controller
{       You, 5 hours ago • Implementacion de Api Rest con Laravel 8
    public function getAllProducts()
    {…
    }

    public function getProductForId($id){…
    }

    public function searchProductForName(Request $request){…
    }

    public function insertProduct(Request $request){…
    }

    public function updateProduct(Request $request, int $id){…
    }

    public function deleteProduct($id){…
    }
}

```

Objetos request y response de cada petición:

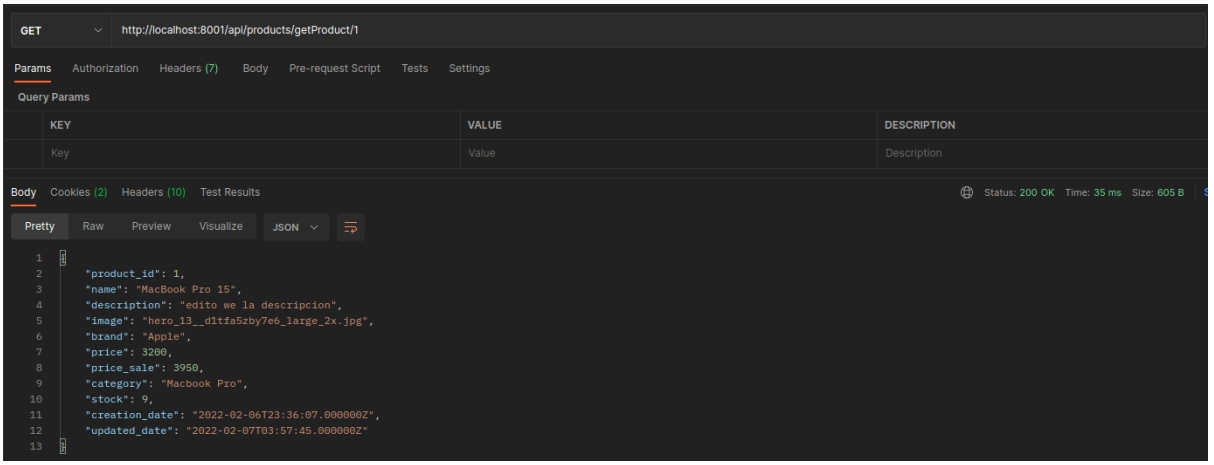
1. Crear un producto



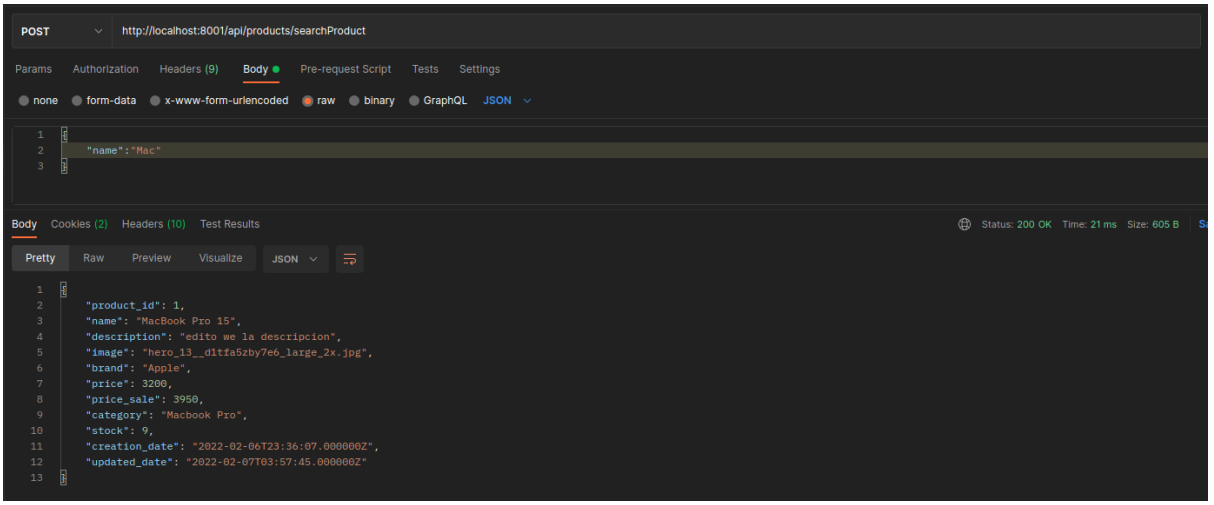
2. Obtener todos los productos (Implementar una lógica de paginación y de búsqueda por nombre)
   2.1 Obtener todos los productos: El objeto response tiene incorporado los atributos para la paginación. El atributo “data” contiene los datos.



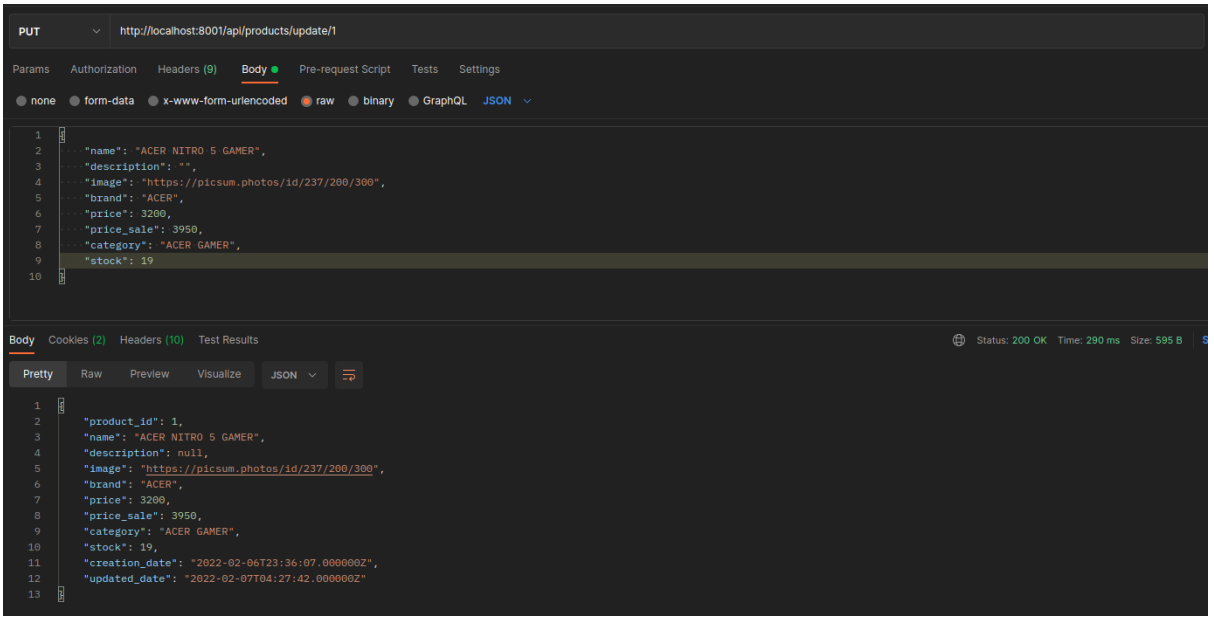
2.2 Obtener un producto por ID:



2.3 Buscar un producto por su nombre:



3. Modificar información de un producto:



4. Eliminar un producto