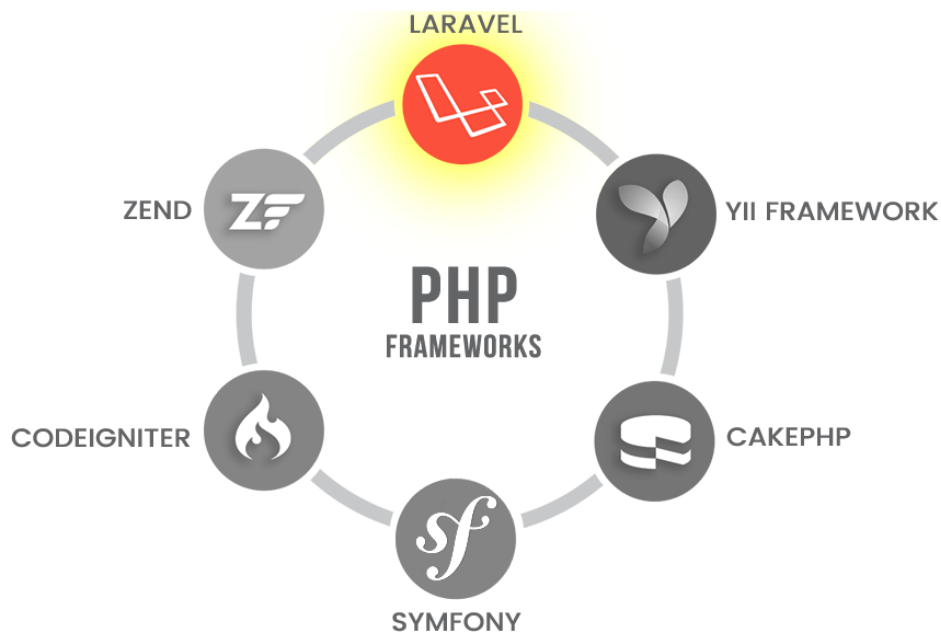


unidad didáctica 7

Framework Laravel - API RESTFULL



1. **crear tabla Productos**
2. **crear controlador ProductoController**
3. **cómo funciona la API REST**
 3. 1. listar todos los productos
 3. 2. producto en concreto
 3. 3. introducir producto nuevo
 3. 4. actualizar un producto existente
 3. 5. eliminar un producto

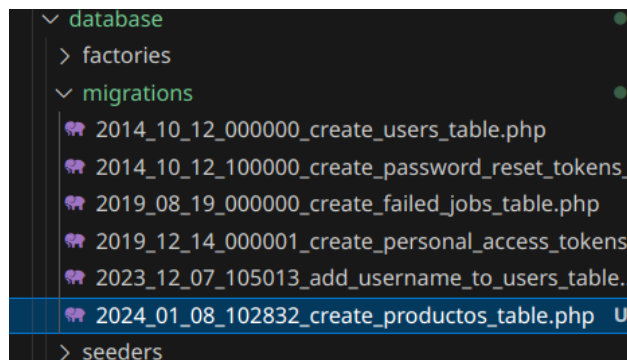
1. crear tabla Productos

1. Crear migración para la tabla `productos`:

```
1 php artisan make:migration create_productos_table
2 # ó
3 # sudo docker-compose exec myapp php artisan make:migration
  create_productos_table
```

```
• abc@jolly-wright:~/Escritorio/IES/DWES/projectes/pru-udemy$ sudo docker-compose exec myapp php artisan make:
migration create_productos_table
[sudo] password for abc:

INFO Migration [database/migrations/2024_01_08_102832_create_productos_table.php] created successfully.
```



2. Añadir al fichero generado (en `migrations`) el resto de campos que se requieren en la tabla `productos`:

```
1 public function up(): void
2 {
3     Schema::create('productos', function (Blueprint $table) {
4         $table->id();
5         $table->string('nombre');
6         $table->text('descripcion');
7         $table->decimal('precio', 8, 2);
8         $table->timestamps();
9     });
10 }
```

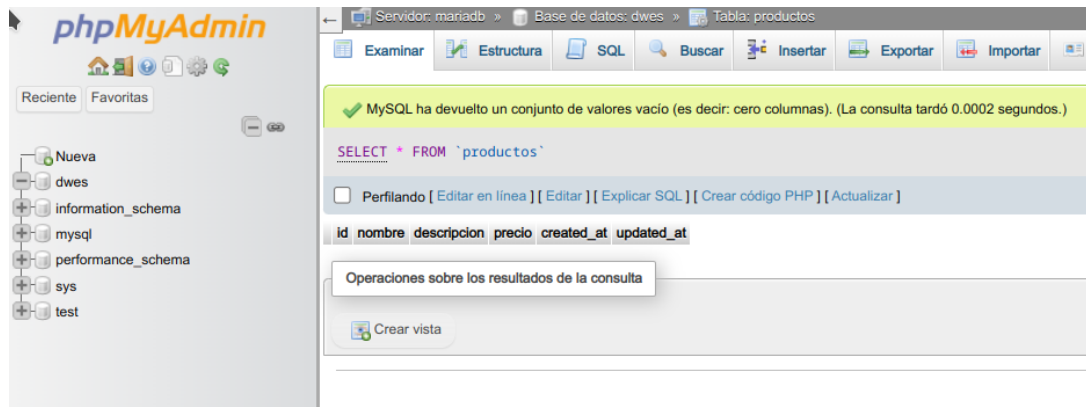
3. Ejecutar migración:

```
1 php artisan migrate
2 # ó
3 # sudo docker-compose exec myapp php artisan migrate
```

```
• abc@jolly-wright:~/Escritorio/IES/DWES/projectes/pru-udemy$ sudo docker-compose exec myapp ph
p artisan migrate

INFO Running migrations.

2024_01_08_102832_create_productos_table ..... 15ms DONE
```



4. Crear un `seeder`:

Introducimos información en esta tabla nueva, creando un fichero en la carpeta `database/seeds` de nombre `ProductoSeeder.php`:

```

1  <?php
2      namespace Database\Seeders;
3      use Illuminate\Database\Seeder;
4      use Illuminate\Support\Facades\DB;
5
6      class ProductoSeeder extends Seeder {
7
8          public function run() {
9              // insertar datos prueba
10             DB::table('productos')->insert([
11                 'nombre' => 'producto prueba 1',
12                 'descripcion' => 'esta es una descripción para el producto
prueba 1',
13                 'precio' => 19.99,
14             ]);
15
16             DB::table('productos')->insert([
17                 'nombre' => 'producto prueba 2',
18                 'descripcion' => 'esta es una descripción para el producto
prueba 2',
19                 'precio' => 29.99,
20             ]);
21         }
22     }

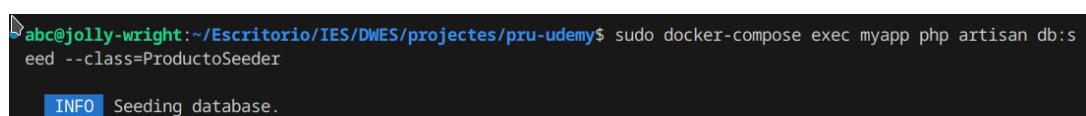
```

5. Ejecutar el `seeder`:

```

1  php artisan db:seed --class=ProductoSeeder
2  # ó
3  # sudo docker-compose exec myapp php artisan db:seed --
class=ProductoSeeder

```



Reciente

Favoritas

Nueva

dws

Nueva

failed_jobs

migrations

password_reset_tokens

personal_access_tokens

productos

users

information_schema

mysql

performance_schema

sys

test

Examinar

Estructura

SQL

Buscar

Insertar

Exportar

Importar

Privilegios

Operaciones

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0.0004 segundos.)

SELECT * FROM `productos`

Perfilando

Editar en línea

Editar

Explicar SQL

Crear código PHP

Actualizar

Mostrar todo

Número de filas: 25

Filtrar filas: Buscar en esta tabla

Ordenar según la clave: Ninguna

Opciones extra

id

nombre

descripcion

precio

created_at

updated_at

1

producto prueba 1

esta es una descripción para el producto prueba 1

19.99

NULL

NULL

2

producto prueba 2

esta es una descripción para el producto prueba 2

29.99

NULL

NULL

Seleccionar todo

Para los elementos que están marcados:

Editar

Copiar

Borrar

Exportar

Mostrar todo

Número de filas: 25

Filtrar filas: Buscar en esta tabla

Ordenar según la clave: Ninguna

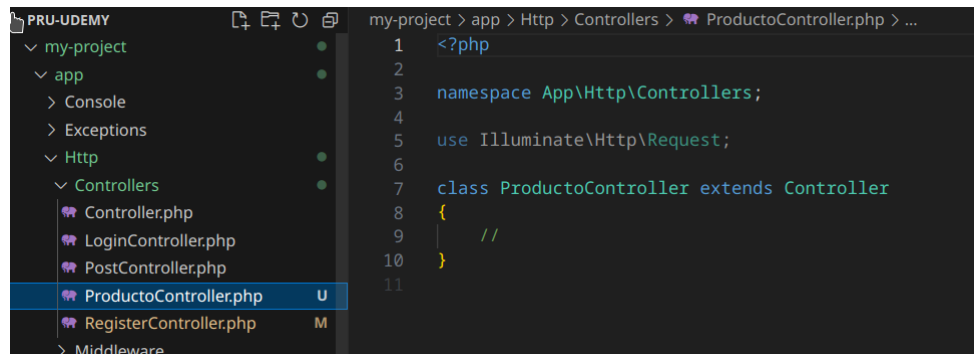
2. crear controlador ProductoController

1. Crear desde consola un controlador para la tabla `producto`:

```
1 php artisan make:controller ProductoController
2 # ó
3 # sudo docker-compose exec myapp php artisan make:controller
  ProductoController
```

```
abc@jolly-wright:~/Escritorio/IES/DWES/proyectos/pru-udemys$ sudo docker-compose exec myapp php artisan make
:controller ProductoController
```

```
INFO Controller [app/Http/Controllers/ProductoController.php] created successfully.
```



2. Como vamos a conectarnos a un modelo para traer la información de dicho modelo añadimos mediante `use`; además creamos la función `index`:

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Http\Request;
5 use App\Models\Producto; // <-- esta linea
6
7 class ProductoController extends Controller
8 {
9     public function index(){
10         return response()->json(Producto::all());
11     }
12 }
```

3. Crear un modelo en la carpeta `Models` de nombre `Producto.php`:

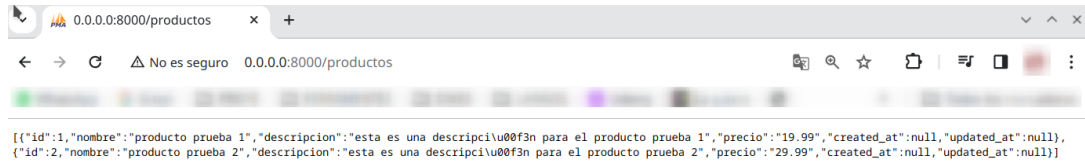
```
1 <?php
2 namespace App\Models;
3
4 use Illuminate\Database\Eloquent\Model;
5
6 class Producto extends Model {
7
8     protected $fillable = ['nombre', 'descripcion', 'precio'];
9 }
```

4. Ir a fichero `web.php` (en la carpeta `routes`) y colocar nuestras rutas:

```

1 // cargar el recurso del controlador ProductoController
2 use App\Http\Controllers\ProductoController
3
4
5 Route::prefix('productos')->group(function(){
6     Route::get('/',[ProductoController::class, 'index']);
7 });

```



```

[{"id":1,"nombre":"producto prueba 1","descripcion":"esta es una descripci\u00f3n para el producto prueba 1","precio":"19.99","created_at":null,"updated_at":null},
{"id":2,"nombre":"producto prueba 2","descripcion":"esta es una descripci\u00f3n para el producto prueba 2","precio":"29.99","created_at":null,"updated_at":null}]

```

La función anterior nos devuelve todos los productos. Pero, qué pasa si queremos un producto en cuestión:

- En `ProductoController.php` añadimos otra función (show) en la que se le pasa por parámetros el `id`:

```

1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Http\Request;
5 use App\Models\Producto; // <-- esta linea
6
7 class ProductoController extends Controller
8 {
9     public function index(){
10         return response()->json(Producto::all());
11     }
12     public function show($id){
13         return response()->json(Producto::find($id));
14     }
15 }

```

- En `web.php` añadimos otra ruta en nuestro grupo:

```

1 Route::prefix('productos')->group(function(){
2
3     Route::get('/',[ProductoController::class, 'index']);
4     Route::get('/{id}',[ProductoController::class, 'show']);
5 });

```



```

{"id":2,"nombre":"producto prueba 2","descripcion":"esta es una descripci\u00f3n para el producto prueba 2","precio":"29.99","created_at":null,"updated_at":null}

```

- Siguiente método, `store`:

- en `ProductoController.php`:

```

1     public function store(Request $request){
2         $producto = Producto::create($request->all());
3         return response()->json($producto, 201);
4     }

```

b) en `web.php`:

```

1     Route::prefix('productos')->group(function(){
2
3         Route::get('/',[ProductoController::class, 'index']);
4         Route::get('/{id}',[ProductoController::class, 'show']);
5
6         Route::post('/',[ProductoController::class, 'store']);
7     });

```

8. Método `update`:

a) en `ProductoController.php`:

```

1     public function update(Request $request, $id){
2         $producto = Producto::findOrFail($id);
3         $producto -> update($request->all());
4
5         return response()->json($producto, 200);
6     }

```

b) en `web.php`:

```

1     Route::prefix('productos')->group(function(){
2
3         Route::get('/',[ProductoController::class, 'index']);
4         Route::get('/{id}',[ProductoController::class, 'show']);
5
6         Route::post('/',[ProductoController::class, 'store']);
7         Route::put('/{id}',[ProductoController::class, 'update']);
8     });

```

9. Método `delete`:

a) en `ProductoController.php`:

```

1     public function destroy($id){
2         Producto::findOrFail($id)->delete();
3
4         return response()->json(null, 204);
5     }

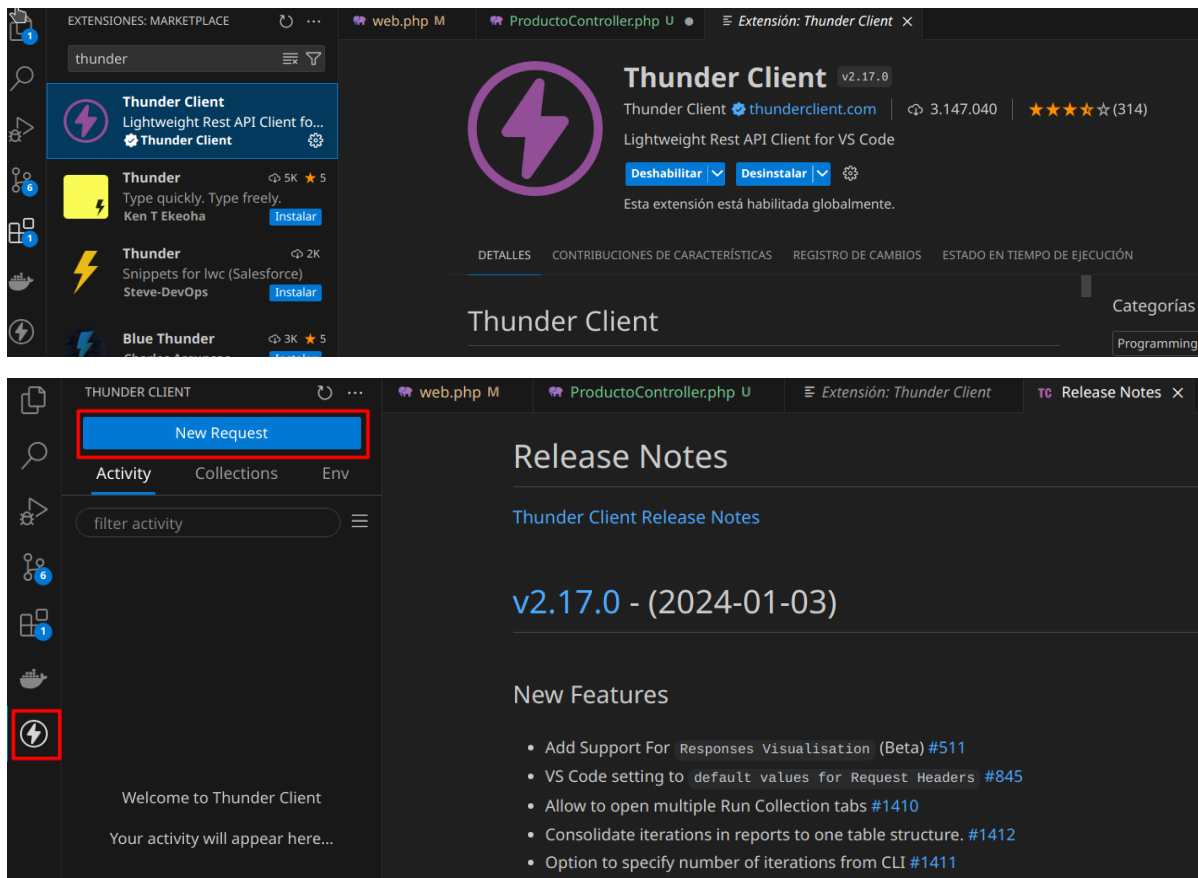
```

b) en `web.php`:

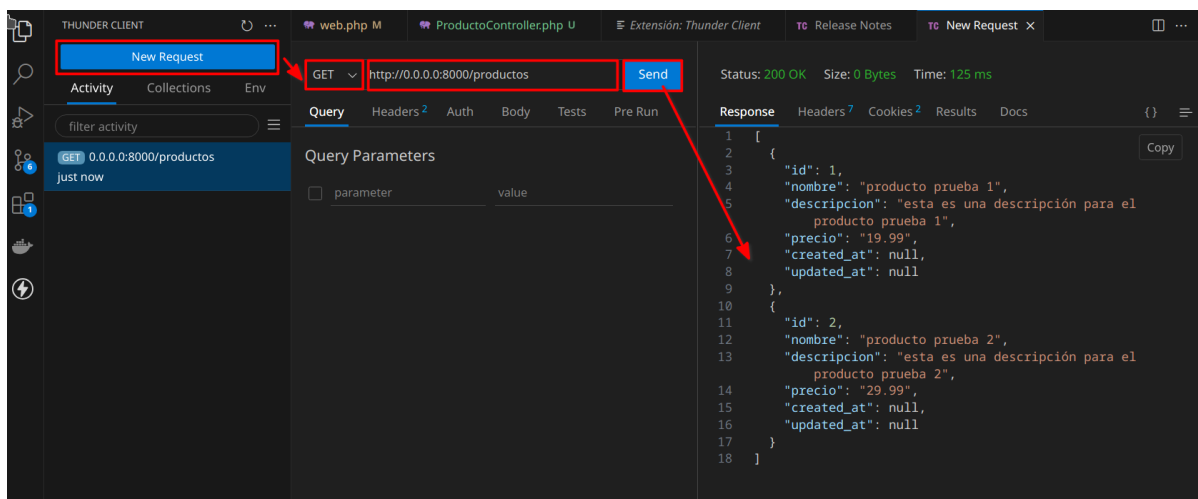

```
1 Route::prefix('productos')->group(function(){
2
3     Route::get('/',[ProductoController::class, 'index']);
4     Route::get('/{id}',[ProductoController::class, 'show']);
5
6     Route::post('/',[ProductoController::class, 'store']);
7     Route::put('/{id}',[ProductoController::class, 'update']);
8     Route::delete('/{id}',[ProductoController::class, 'destroy']);
9 });
```

3. cómo funciona la API REST

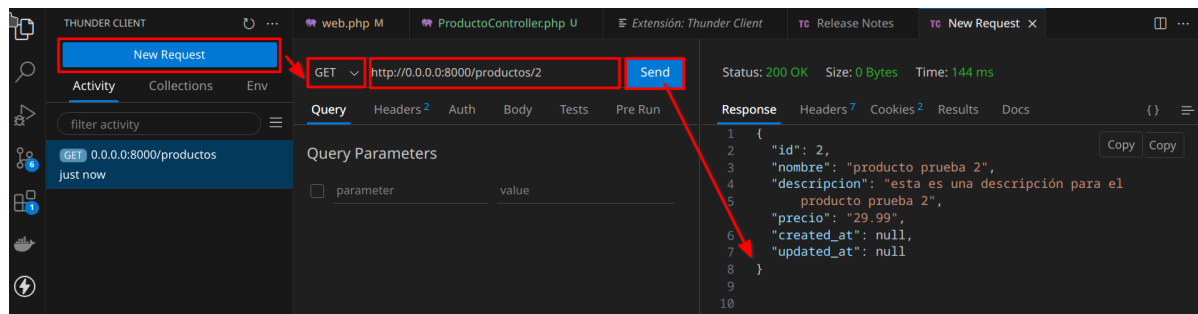
Para ello vamos a utilizar un software que es una extensión de Visual Studio Code, de nombre **Thunder Client**:



3.1. listar todos los productos



3.2. producto en concreto



3.3. introducir producto nuevo

Si realizamos una nueva petición (new request) con método **post** y pasando (desde **body** y en **json**) un nuevo producto, va a dar un error.

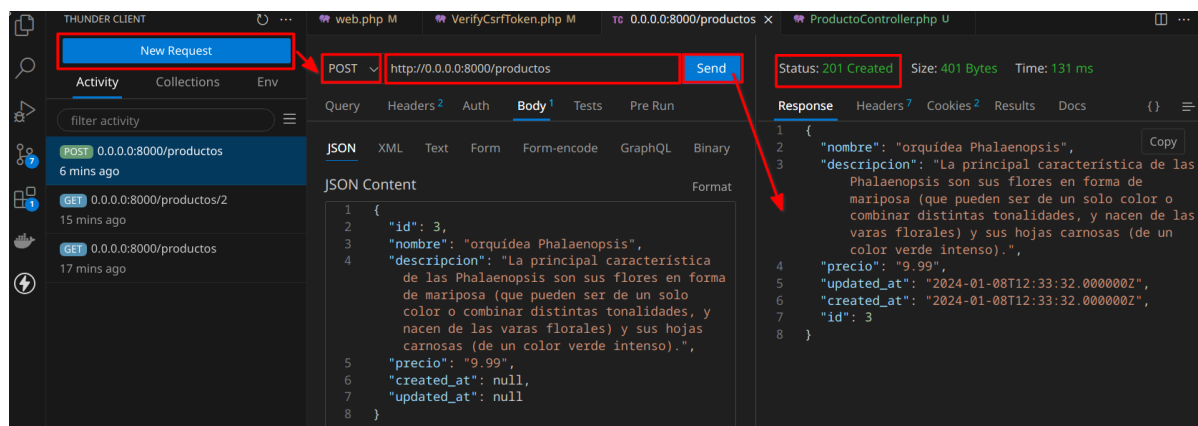
Esto se debe a que Laravel, por sus métodos de seguridad, necesita un *token* llamado **csrf**. Ya que, ahora mismo, estamos realizando pruebas, vamos a indicarle a Laravel que excluya la URL en cuestión de la verificación.

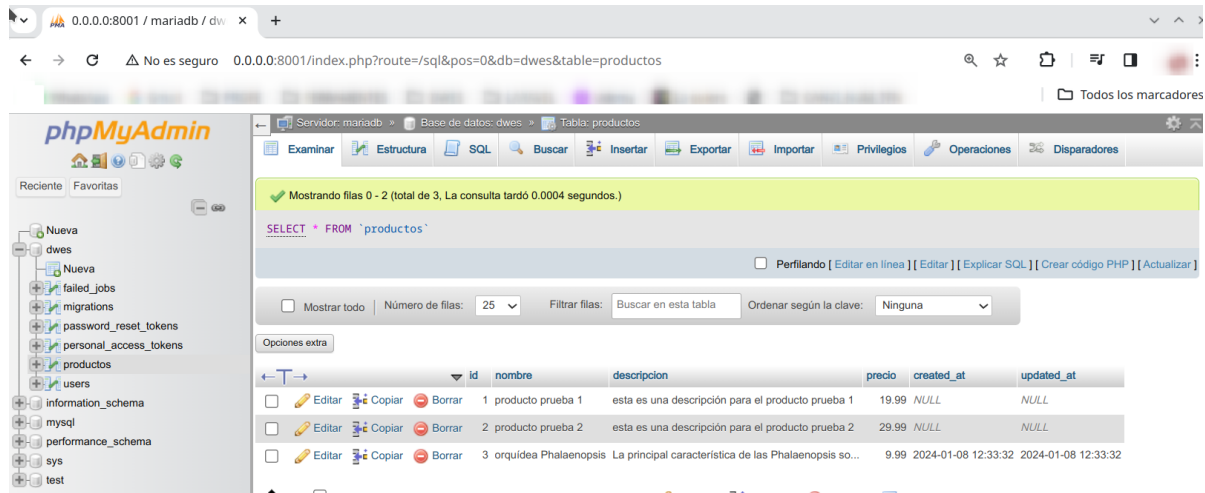
Para esto, accedemos al fichero **VerifyCsrfToken.php** de la carpeta **app\Http\Middleware**:

```

1  <?php
2  namespace App\Http\Middleware;
3
4  use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;
5
6  class VerifyCsrfToken extends Middleware
7  {
8      /**
9       * The URIs that should be excluded from CSRF verification.
10
11      * @var array<int, string>
12      */
13      protected $except = [
14          "http://0.0.0.0:8000/productos", // <-- esta excepción
15      ];
16  }

```





3.4. actualizar un producto existente

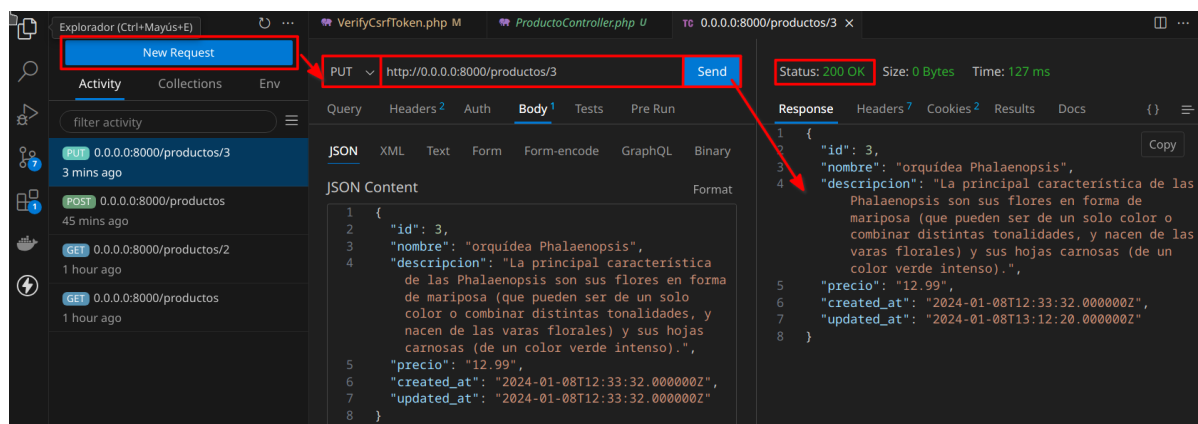
1. Añadir al fichero `VerifyCsrfToken.php` de la carpeta `app\Http\Middleware` la excepción:

```

1  <?php
2  namespace App\Http\Middleware;
3
4  use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as
    Middleware;
5
6  class VerifyCsrfToken extends Middleware
7  {
8      /**
9       * The URIs that should be excluded from CSRF verification.
10      *
11      * @var array<int, string>
12      */
13      protected $except = [
14          "http://0.0.0.0:8000/productos",
15          "http://0.0.0.0:8000/productos/3", // <-- esta nueva
    excepción
16      ];
17  }

```

2. Probar en Thunder Client:



← → ↻ ⚠ No es seguro 0.0.0.0:8001/index.php?route=/sql&pos=0&db=dwes&table=productos

phpMyAdmin

Reciente Favoritas

Nueva
dwes
Nueva
failed_jobs
migrations
password_reset_tokens
personal_access_tokens
productos
users
information_schema
mysql
performance_schema
sys
test

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Disparadores

Mostrando filas 0 - 2 (total de 3, La consulta tardó 0.0004 segundos.)

SELECT * FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	id	nombre	descripcion	precio	created_at	updated_at
<input type="checkbox"/> [Editar] [Copiar] [Borrar]	1	producto prueba 1	esta es una descripción para el producto prueba 1	19.99	NULL	NULL
<input type="checkbox"/> [Editar] [Copiar] [Borrar]	2	producto prueba 2	esta es una descripción para el producto prueba 2	29.99	NULL	NULL
<input type="checkbox"/> [Editar] [Copiar] [Borrar]	3	orquídea Phalaenopsis	La principal característica de las Phalaenopsis so...	12.99	2024-01-08 12:33:32	2024-01-08 13:12:20

Seleccionar todo Para los elementos que están marcados: [Editar] [Copiar] [Borrar] [Exportar]

3.5. eliminar un producto

Explorador (Ctrl+Mayús+E) VerifyCsrfToken.php M ProductoController.php U 0.0.0.0:8000/productos/3 New Request X New Request X 0.0.0.0:8000

New Request

Activity Collections Env

filter activity

DEL 0.0.0.0:8000/productos/3 just now

PUT 0.0.0.0:8000/productos/3 3 mins ago

POST 0.0.0.0:8000/productos 45 mins ago

GET 0.0.0.0:8000/productos/2 1 hour ago

GET 0.0.0.0:8000/productos 1 hour ago

DELETE http://0.0.0.0:8000/productos/3 Send

Status: 204 No Content Size: 0 Bytes Time: 130 ms

Query Headers Auth Body Tests Pre Run

Query Parameters

parameter	value

Response Headers Cookies Results Docs

1

0.0.0.0:8001 / mariadb / dwes

← → ↻ ⚠ No es seguro 0.0.0.0:8001/index.php?route=/sql&pos=0&db=dwes&table=productos

phpMyAdmin

Reciente Favoritas

Nueva
dwes
Nueva
failed_jobs
migrations
password_reset_tokens
personal_access_tokens
productos
users
information_schema
mysql
performance_schema
sys
test

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0.0003 segundos.)

SELECT * FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	id	nombre	descripcion	precio	created_at	updated_at
<input type="checkbox"/> [Editar] [Copiar] [Borrar]	1	producto prueba 1	esta es una descripción para el producto prueba 1	19.99	NULL	NULL
<input type="checkbox"/> [Editar] [Copiar] [Borrar]	2	producto prueba 2	esta es una descripción para el producto prueba 2	29.99	NULL	NULL

Seleccionar todo Para los elementos que están marcados: [Editar] [Copiar] [Borrar] [Exportar]