



## PROYECTOS CON LARAVEL 9.0

En esta serie vamos a manejar lo siguiente:

1. Practica. Migraciones y modelos para el proyecto Shopcenter
2. Practica. Registrando usuarios en el sistema
3. Practica. Login de usuario
4. Practica. Creación del CRUD de category

### 1. Practica. Creación del proyecto Shopcenter 2022 en Laravel

La empresa Shopcenter Mercado virtual necesita desarrollar un sistema de información web que le permita realizar un sistema de ventas en línea, registrando los usuarios, ordenes, sitio donde se deben llevar las ordenes del pedido.

#### Historias de Usuario

- Necesito una LandingPage para dar a conocer la información de los sitios y los servicios que promocionan
- Registrar usuarios con sus datos básicos de nombre, email, contraseña
- Permitirle generar una reservación al usuario de un sitio y los servicios prestados
- Permitirle dejar los comentarios de los sitios visitados

#### Tareas

Crear las migraciones para generar la base de datos del proyecto junto con los modelos

#### Crear un nuevo proyecto en Laravel

```
composer create-project --prefer-dist laravel/laravel blog
```

#### Crear la base de datos en XAMMP

Llamar la base de datos en el archivo del proyecto .env

Crear las migraciones en Laravel. **Los nombres deben estar en inglés y singular y la primera letra en mayúsculas.** Para no tener dificultades al momento de pasar el proyecto a producción.

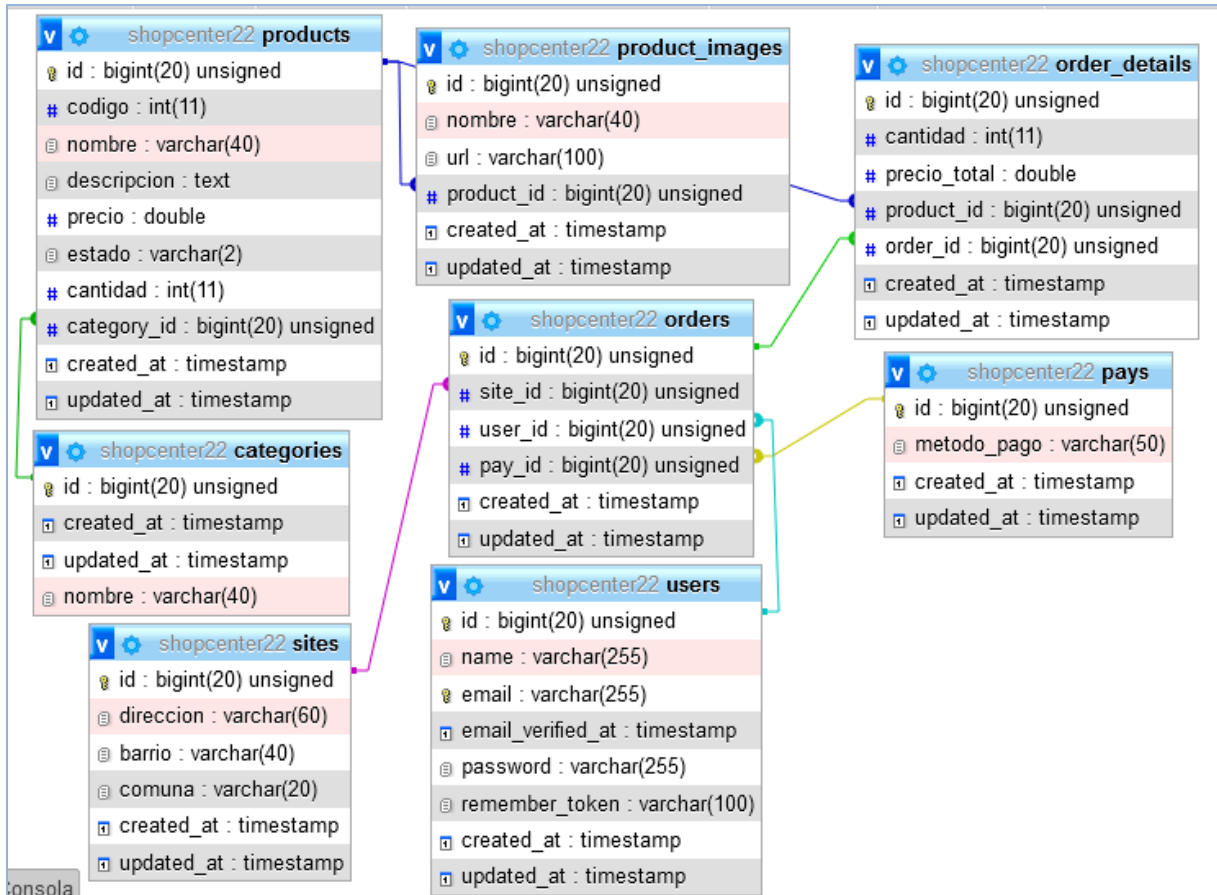
```
Php artisan make:model Category -mcr
```

Php artisan migrate

Si la base de datos tenía información registrada es mejor digitar el comando

Php artisan **migrate:fresh**

<https://laravel.com/docs/9.x/eloquent-relationships>



## Migración User

```
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

### Migración Category

```
public function up()
{
    Schema::create('categories', function (Blueprint $table) {
        $table->id();
        $table->timestamps();
        $table->string('nombre',40);
    });
}
```

### Migración Site

```
public function up()
{
    Schema::create('sites', function (Blueprint $table) {
        $table->id();
        $table->string('direccion',60);
        $table->string('barrio',40);
        $table->string('comuna',20);
        $table->timestamps();
    });
}
```

### Migración Pay

```
public function up()
{
    Schema::create('pays', function (Blueprint $table) {
        $table->id();
        $table->string('metodo_pago',50);
        $table->timestamps();
    });
}
```

### Migración Order

```
public function up()
{
    Schema::create('orders', function (Blueprint $table) {
        $table->id();
        $table->unsignedBigInteger('site_id');
        $table->unsignedBigInteger('user_id');
        $table->unsignedBigInteger('pay_id');
        $table->foreign("site_id")->references("id")->on("sites")->
        >onDelete("cascade")->onUpdate("cascade");
    });
}
```

```

        $table->foreign("user_id")->references("id")->on("users")->
        >onDelete("cascade")->onUpdate("cascade");
        $table->foreign("pay_id")->references("id")->on("pays")->
        >onDelete("cascade")->onUpdate("cascade");

        $table->timestamps();
    });
}

```

### Migración Product

```

public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->id();
        $table->integer('codigo')->length(10);
        $table->string('nombre',40);
        $table->text('descripcion');
        $table->double('precio',12.2);
        $table->string('estado',2);
        $table->integer('cantidad')->length(10);
        $table->unsignedBigInteger('category_id');
        $table->foreign("category_id")->references("id")->
        >on("categories")->onDelete("cascade")->onUpdate("cascade");

        $table->timestamps();
    });
}

```

### Migración producto\_image

```

public function up()
{
    Schema::create('product_images', function (Blueprint $table) {
        $table->id();
        $table->string('nombre',40);
        $table->string('url',100);
        $table->unsignedBigInteger('product_id');
        $table->foreign("product_id")->references("id")->on("products")->
        >onDelete("cascade")->onUpdate("cascade");
        $table->timestamps();
    });
}

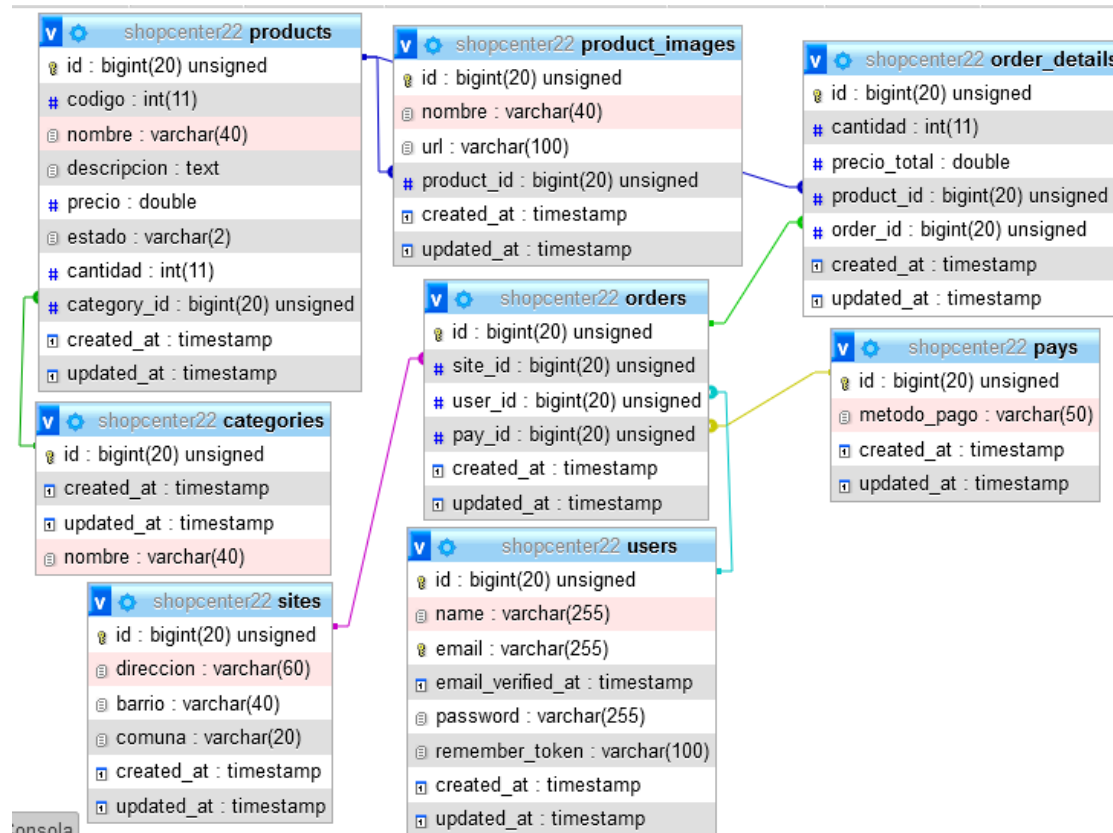
```

## Migración order\_detail

```
public function up()
{
    Schema::create('order_details', function (Blueprint $table) {
        $table->id();
        $table->integer('cantidad')->length(10);
        $table->double('precio_total',10.2);
        $table->unsignedBigInteger('product_id');
        $table->unsignedBigInteger('order_id');
        $table->foreign("product_id")->references("id")->on("products")->onDelete("cascade")->onUpdate("cascade");
        $table->foreign("order_id")->references("id")->on("orders")->onDelete("cascade")->onUpdate("cascade");

        $table->timestamps();
    });
}
```

Verificamos en XAMPP el diseño de la base de datos



## 2. Practica adicionando los roles de usuario en Laravel

Los roles de usuario son algo fundamental en cualquier aplicación que haga uso de autenticación. Almacena los permisos y los roles y permite tener un usuario con un rol o con varios roles y los permisos quedaran guardados también en la base de datos.

Añadiendo librería de roles en Laravel

Debemos instalar **spatie** para comenzar a trabajar con roles en la aplicación con Laravel.

### Comandos para instalar roles y permisos

```
composer require spatie/laravel-permission
```

### Dejar visibles las migraciones creadas para roles

```
Php artisan vendor:publish --provider="Spatie\PermissionServiceProvider"
```



### Php artisan optimize:clear

```
Php artisan migrate:fresh
```

Nota: Cuando no es posible crear con spatie las migraciones debes ir a la ruta

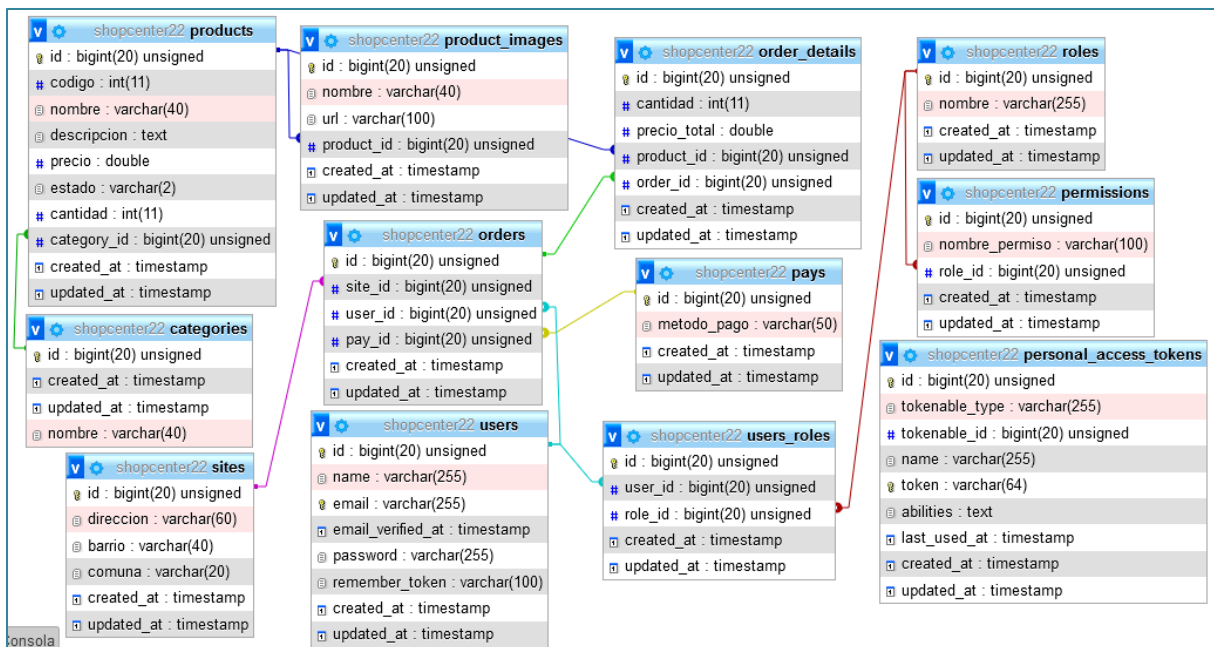
```
C:\xampp\htdocs\Laravel\Tienda22M\vendor\spatie\laravel-permission\database\migrations
```

<< Disco local (C:) > xampp > htdocs > Laravel > Tienda22M > vendor > spatie > laravel-permission > database > migrations

Nombre	Fecha de modificación	Tipo	Tamaño
 add_teams_fields.php.stub	11/01/2022 10:06 a. m.	Archivo STUB	4 KB
 create_permission_tables.php.stub	11/01/2022 10:06 a. m.	Archivo STUB	7 KB

Para pasar este archivo hasta la carpeta de migraciones quitar el stub y ejecutar nuevamente migraciones.

Ahora podemos verificar las nuevas migraciones creadas en MySQL



### 3. Sistema de autenticación con Sanctum

Laravel Sanctum proporciona un sistema de autenticación ligero para SPA (aplicaciones de una sola página), aplicaciones móviles y API simples basadas en tokens. <https://www.ouлуб.com/es-ES/Laravel/sanctum>

Instalación

`composer require laravel/sanctum`

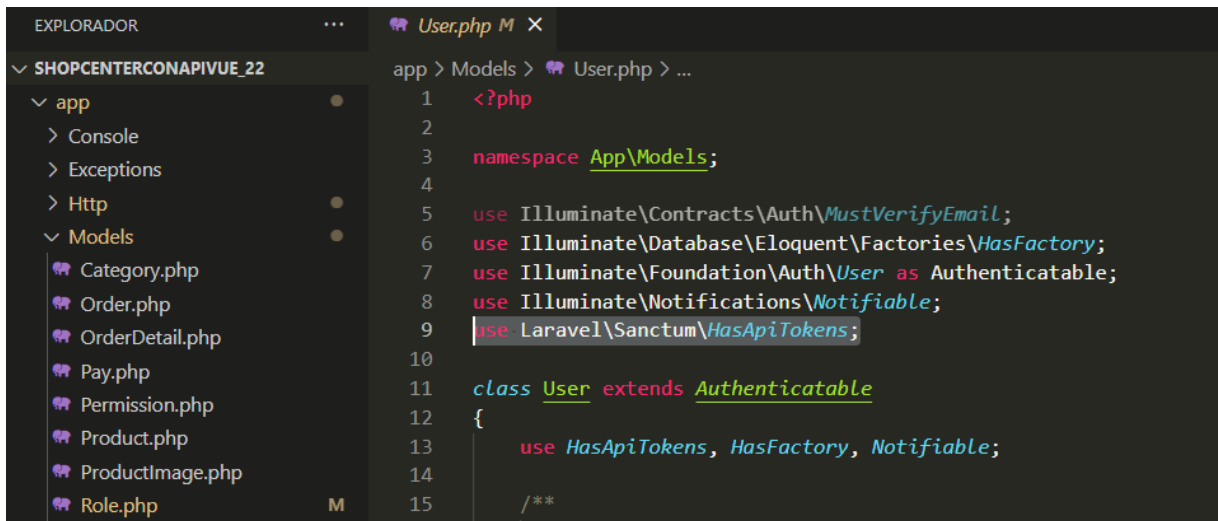
Ejecutamos las migraciones

`php artisan migrate`

aparece ahora la migración creada por sanctum

shopcenter22 personal_access_tokens
id : bigint(20) unsigned
tokenable_type : varchar(255)
# tokenable_id : bigint(20) unsigned
name : varchar(255)
token : varchar(64)
abilities : text
last_used_at : timestamp
created_at : timestamp
updated_at : timestamp

Verificamos en el modelo que se encuentre el archivo



```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Sanctum\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     /**

```

Crear un controlador normal

```
php artisan make:controller AuthController
```

## Practica 2. Registrar usuarios en el sistema

*Creamos un método register*

Tener en cuenta el llamado a

```
use App\Models\User;
```

**bcrypt** método para encriptar el **password** de usuario

**createToken** Librería propia de Laravel donde valida que el dato recibido sea efectivamente del cliente

```

public function register(RegisterRequest $request){
    $user = new User();
    $user->name = $request->name;
    $user->email = $request->email;
    $user->password = bcrypt($request->password);
    $user->save();
    if (Auth::attempt($request->only('email','password'))){
        return response()->json([
            'message'=>'usuario creado',
            'token'=>$request->user()->createToken($request->email)-
>plainTextToken,

```

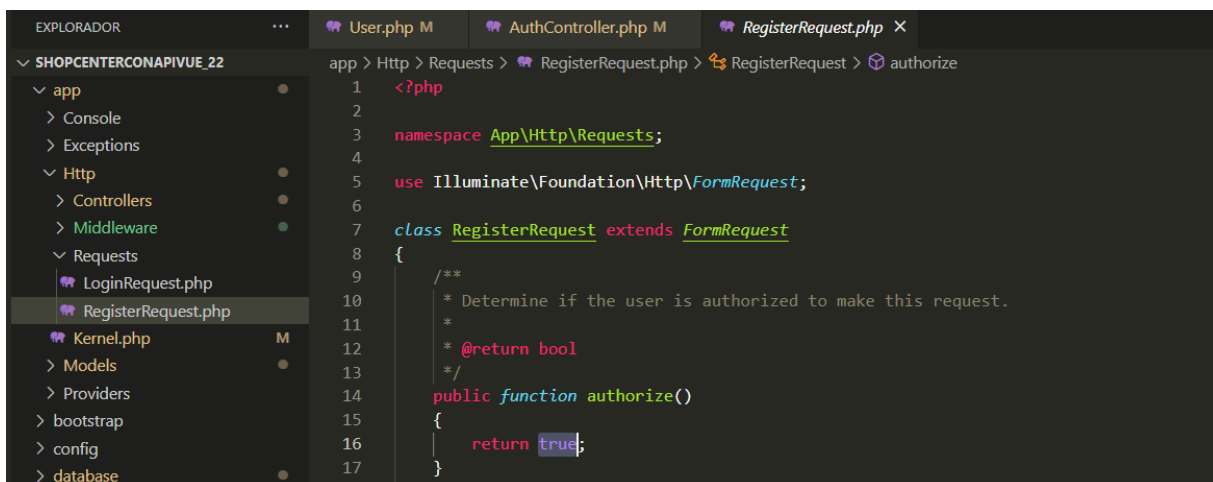


```
    });  
}  
}
```

### Crear un request

`php artisan make:request RegisterRequest`

cambiamos el método `authorize` a `true`



```
1  <?php  
2  
3  namespace App\Http\Requests;  
4  
5  use Illuminate\Foundation\Http\FormRequest;  
6  
7  class RegisterRequest extends FormRequest  
8  {  
9  
10     /**  
11      * Determine if the user is authorized to make this request.  
12      *  
13      * @return bool  
14      */  
15     public function authorize()  
16     {  
17         return true;  
18     }  
19 }
```

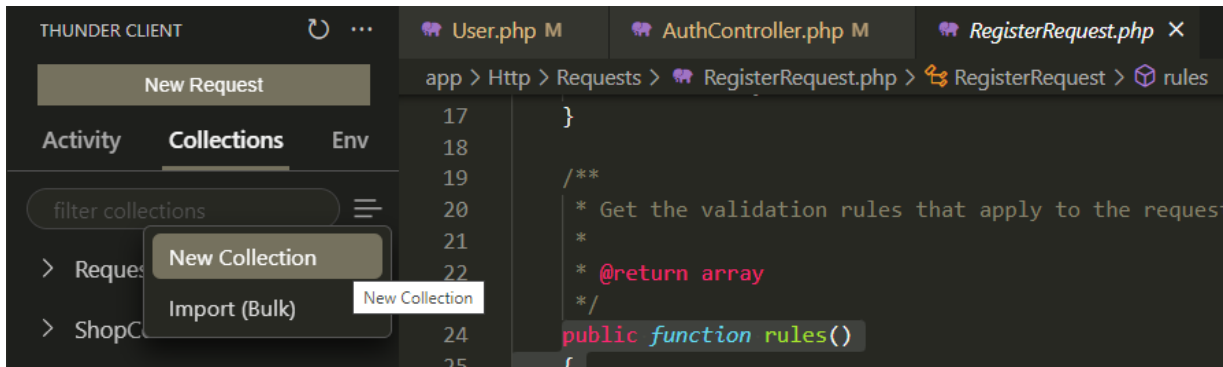
En el método `rules` creamos las **reglas de validación** para evitar que se puedan registrar correos diferentes del cliente, tamaño del campo

```
public function rules()  
{  
    return [  
        //crear las reglas de validación  
        'name' => 'required|min:3',  
        'email' => 'required|email|unique:users',  
        'password' => 'required|min:6'  
    ];  
}
```

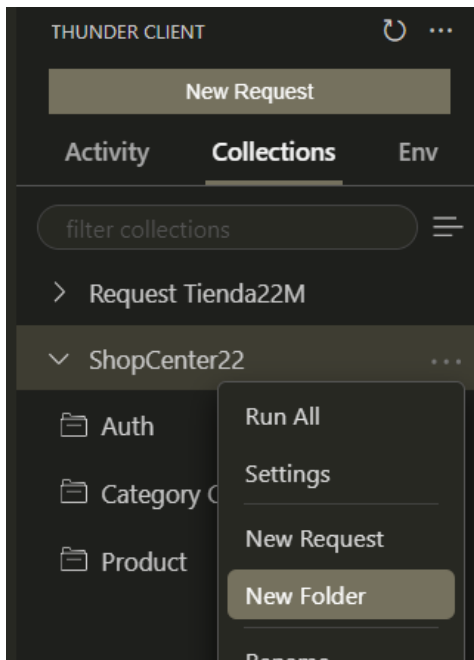
### Creamos la ruta

Ahora en **Thunder Client** verificamos el funcionamiento del registro

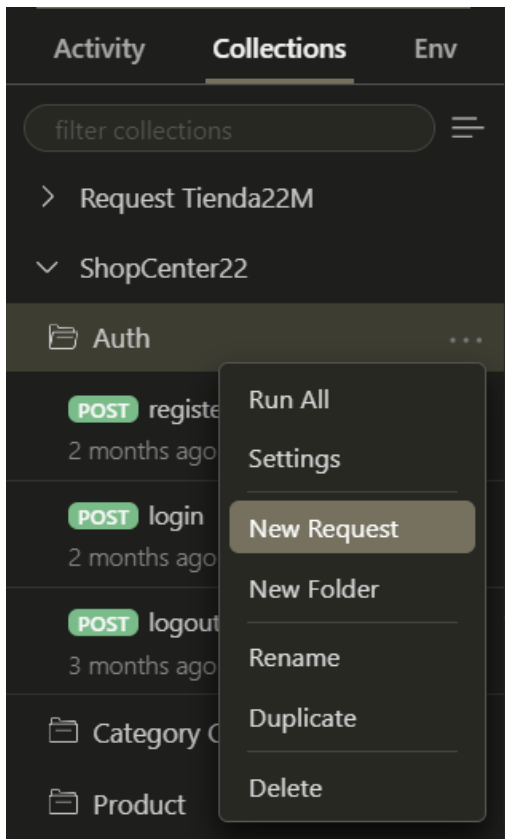
Creamos una carpeta para el proyecto Shopcenter22



Creamos la carpeta **Auth**



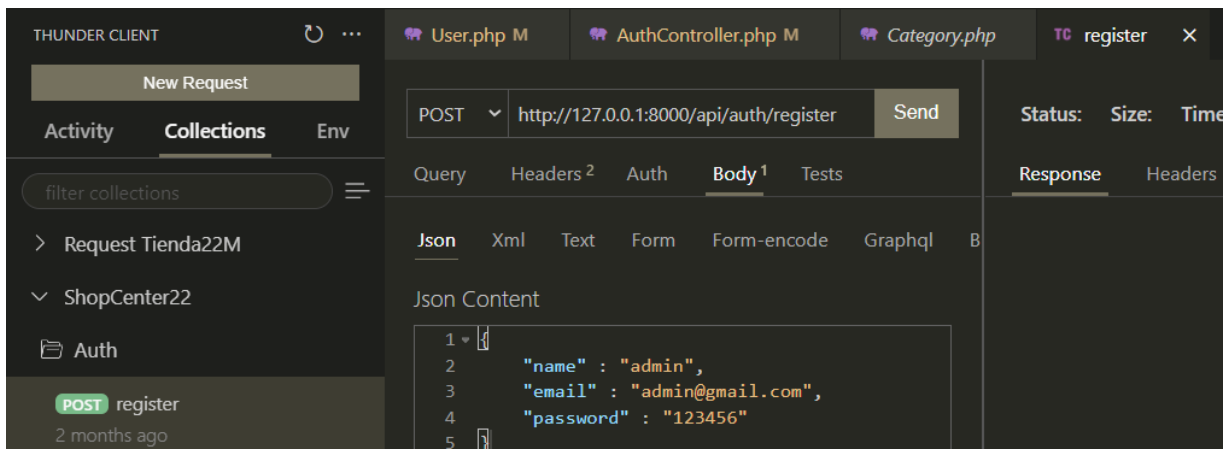
Creamos un nuevo **Request**



Seleccionamos el tipo POST

Adicionamos la ruta y hacemos un envío **SEND**

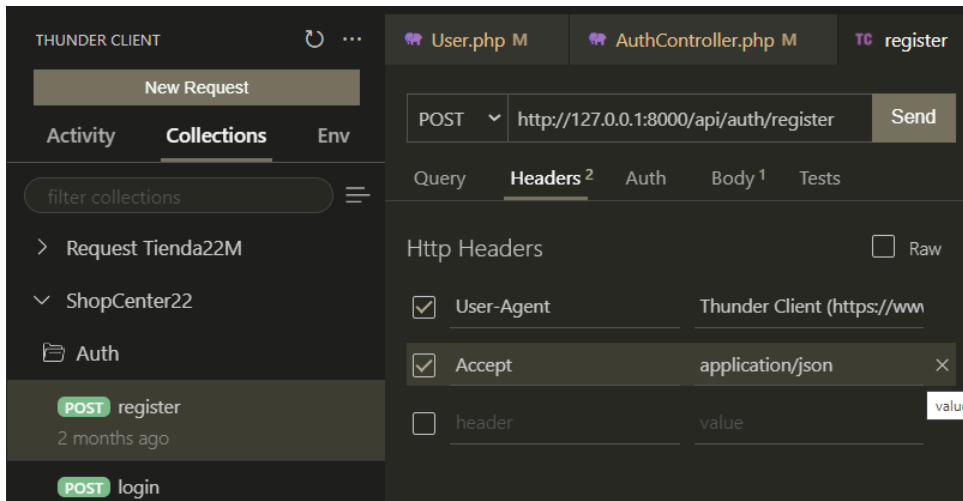
Ctrl + s guarda la consulta



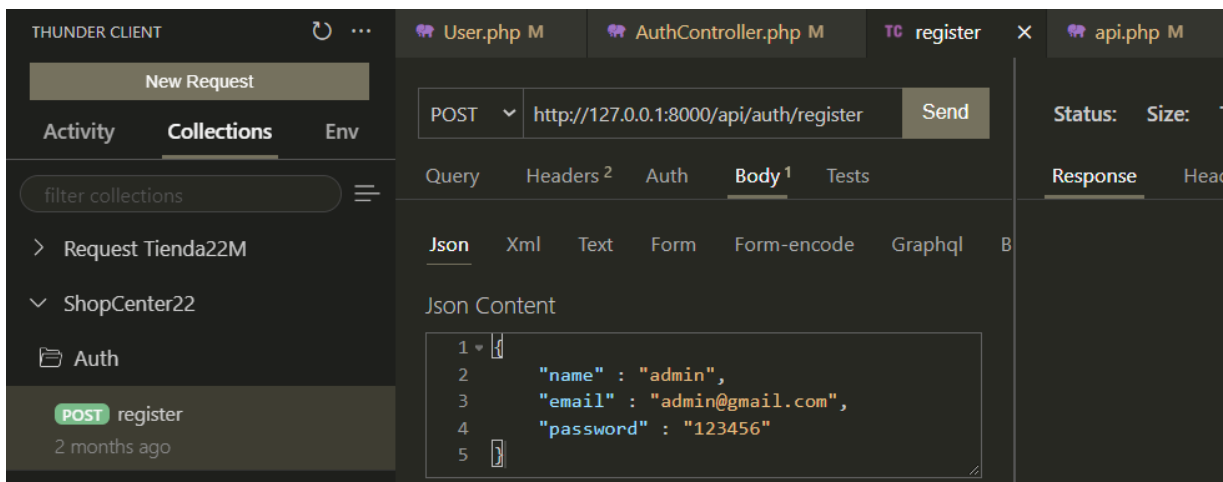
Creamos la ruta

```
Route::post('/auth/register',[AuthController::class, 'register']);
```

Creamos un Headers llamado Accept de tipo application/json



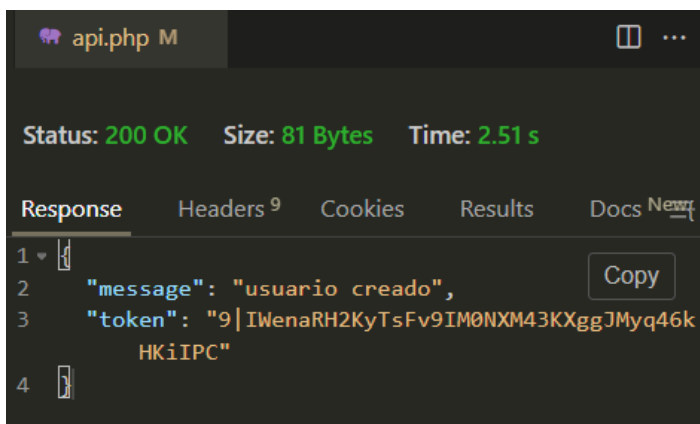
Realizamos el registro en el **Body** en formato Json



Iniciamos el servidor de PHP

Creamos un nuevo registro

Ejecutamos el servicio



Verificamos en la BD

Programa Análisis y Desarrollo de Sistemas de Información – ADSI

Página 12

☐ Mostrar todo
 Número de filas: 25
 Filtrar filas: 
 Sort by key: Ninguna

				id	name	email	email_verified_at	password	remember_token
<input type="checkbox"/>	Editar	Copiar	Borrar	1	german	german@gmail.com	NULL	\$2y\$10\$9JTxYGJc4.DaEeeNXMc9cu.oZ/WdpubW3IP51KiwQ7h...	NULL
<input type="checkbox"/>	Editar	Copiar	Borrar	2	admin	admin@gmail.com	NULL	\$2y\$10\$Ifp6p.3xYv1TxXKZKcoed.58pqkQDxWqmXuSUGLRQk...	NULL
<input type="checkbox"/>	Editar	Copiar	Borrar	3	julian	julian@gmail.com	NULL	\$2y\$10\$XhjHF76o/yidYavHDpWeSeUy2YQqxVwjjjPEdGepbP...	NULL

## 2. Practica. Login de usuario

Creamos el método *login*

```

EXPLOADOR
...
User.php M
AuthController.php M
api.php M

SHOPCENTERCONAPIVUE_22
app > Http > Controllers > AuthController.php > ...
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

public function login(LoginRequest $request){
    /** negamos el Auth si los datos no coinciden, devolviendo el msg no esta autorizado */
    if (!Auth::attempt($request->only('email','password'))) {
        return response()->json([
            'message'=>'Datos incorrectos',
            'success'=>false
        ], 200);
    }
    //verifico si el usuario tiene un token creado, si es asi procedo a eliminar el token
    //procedo a crear un nuevo token
    $userToken = Token::where('name',$request->email)->first();
    if ($userToken) {
        $userToken->delete();
    }
    return response()->json([
        'success'=>true,
        'token'=>$request->user()->createToken($request->email)->plainTextToken,
    ], 200);
}
  
```

Creamos un nuevo request para validar los datos de entrada de login

```

EXPLOADOR
...
User.php M
AuthController.php M
LoginRequest.php
api.php M

SHOPCENTERCONAPIVUE_22
app > Http > Requests > LoginRequest.php > ...
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

/**
 * @return bool
 */
public function authorize()
{
    return true;
}

/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'email' => 'required|email',
        'password' => 'required|min:6'
    ];
}
  
```

## Los códigos de estado de respuesta HTTP

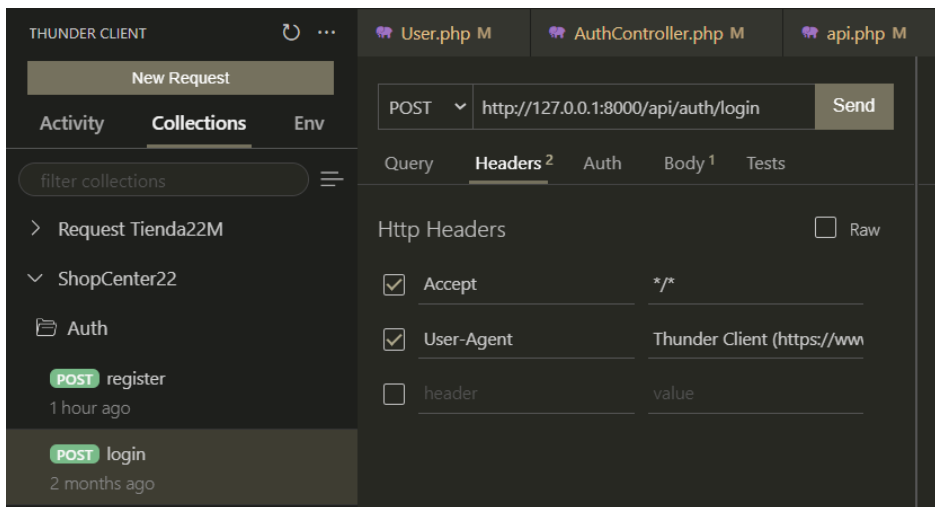
<https://developer.mozilla.org/es/docs/Web/HTTP/Status>

1. Respuestas informativas ( 100 – 199 ),
2. Respuestas satisfactorias ( 200 – 299 ),
3. Redirecciones ( 300 – 399 ),
4. Errores de los clientes ( 400 – 499 ),
5. y errores de los servidores ( 500 – 599 ).

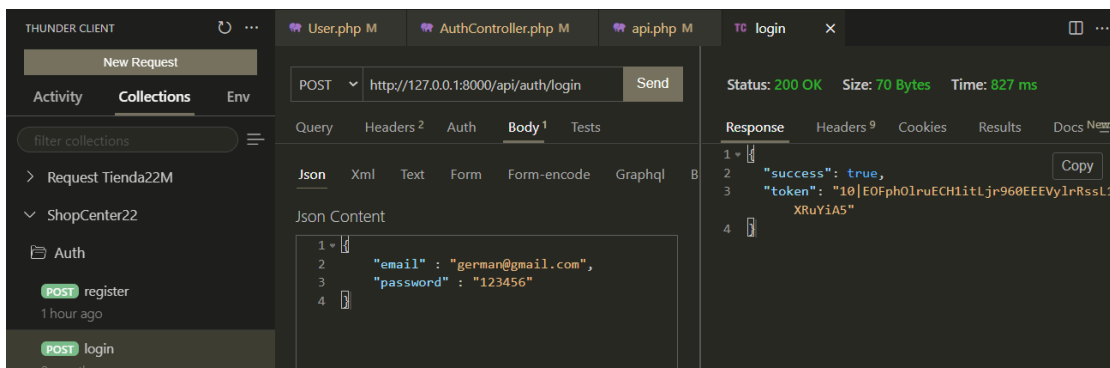
Creamos la ruta para login

```
Route::post('/auth/login',[AuthController::class, 'login']);
```

Creamos una nueva request en **Thunder client** para verificar



Enviamos los datos de inicio de sesión y verificamos



Al verificar en la base de datos observamos que las entradas se repiten cada vez que realizo ingreso

id	tokenable_type	tokenable_id	name	token	abilities	last_used
1	App\Models\User	1	pedro21@gmail.com	ab94bde3749e84c4f6252c868e7badfe2c79d841943525a0e...	["*"]	NULL
2	App\Models\User	2	oscar21@gmail.com	010838a5465e401de27fbd10ef381534fab3b5756e30fd541d...	["*"]	NULL
3	App\Models\User	3	oscar23@gmail.com	4655e721402fd838a90b5f4939f8f99c5edcb8ec96dc35b71...	["*"]	NULL
4	App\Models\User	4	oscar22@gmail.com	3b66b7567d081e38c80a048866066046104cd277304be479...	["*"]	NULL
5	App\Models\User	5	oscar22@gmail.com	0da56620e063a52e8d84549c4c1d6628e083ed9e14b63c073...	["*"]	NULL
6	App\Models\User	3	oscar23@gmail.com	a3f787953c6dd9622da28cb50e3af146ac323b19d44f684a4e...	["*"]	NULL

Por lo que habría que validar si tiene un token el usuario creado no generar nuevos

Creación de cierre de sesión de usuario a través del método **logout**

Con el método **currentAccessToken** valido eliminar la generación de token para el mismo usuario

```

THUNDER CLIENT
New Request
Activity Collections Env
filter collections
Request Tienda22M
ShopCenter22
Auth

app > Http > Controllers > AuthController.php > AuthController > login

49
50 public function logout(Request $request){
51     $request->user()->currentAccessToken()->delete();
52     return response()->json([
53         'message'=>'Token eliminado correctamente',
54     ], 410);
55 }
56
57
58

```

Creamos la ruta de logout

Nota: La cual esta protegida para su acceso; lo hacemos con **middleware y sanctum**

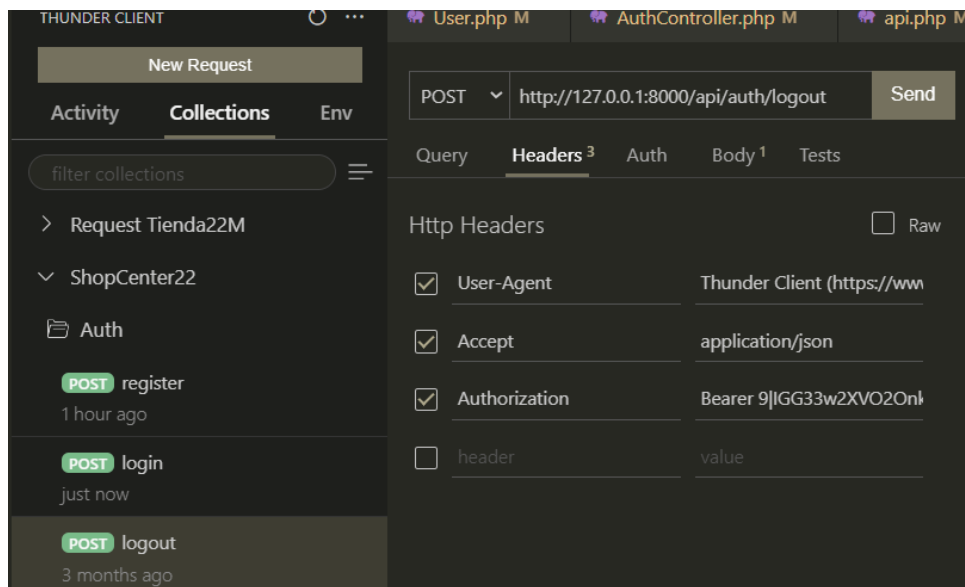
Esto lo hacemos para cada ruta que deseamos este protegida por login

```

Route::post('/auth/logout',[AuthController::class, 'logout'])-
>middleware('auth:sanctum');

```

Verificamos la funcionalidad en Thunder client



Creo la cabecera Authorization con el valor **Bearer** y token creado en la base de datos.

#### 4. Practica. Creación del CRUD de category

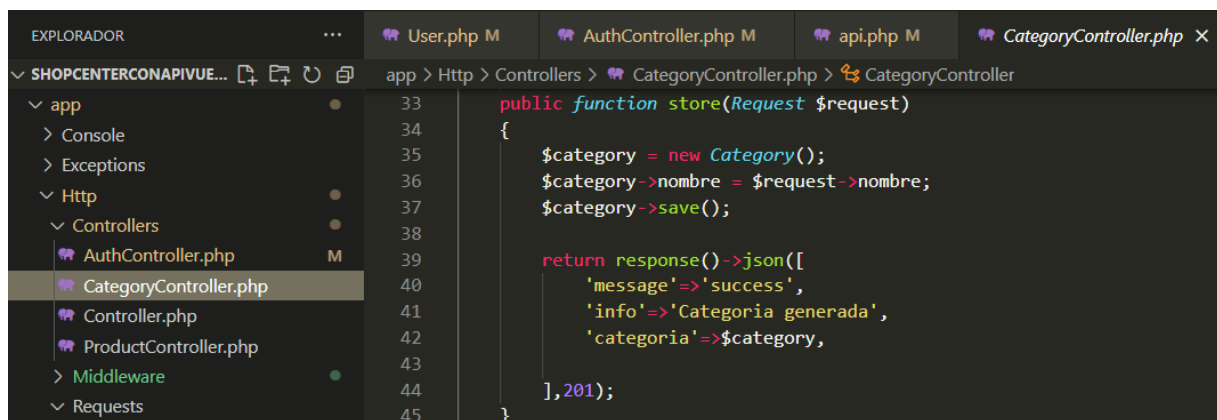
Creación del controlador

```
php artisan make:controller CategoryController --api
```

Creamos la ruta en el archivo API para el controlador

```
Route::apiResource('/product',ProductController::class);
```

Iniciamos trabajando con el método **store**



Verificamos en **Thunder client**

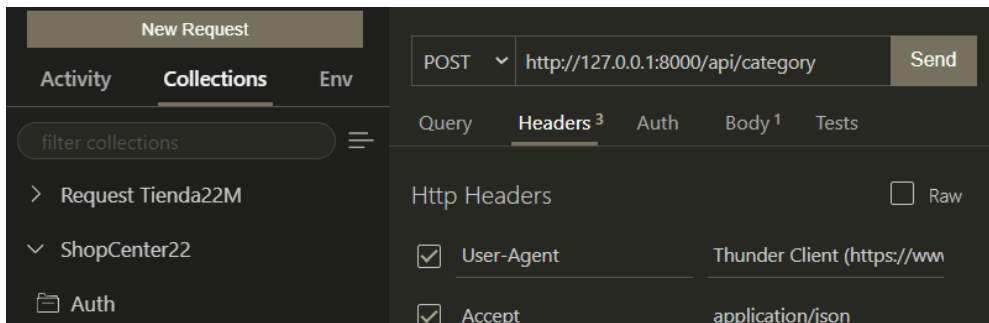
**Creamos la carpeta Category CRUD**

Seleccionamos el método POST pegamos la URL y la terminación api/category de acuerdo al archivo de

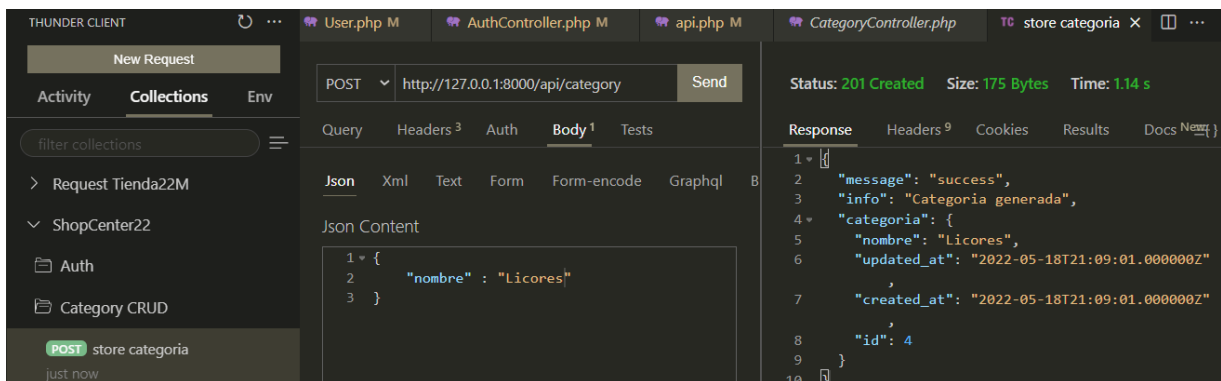


rutas

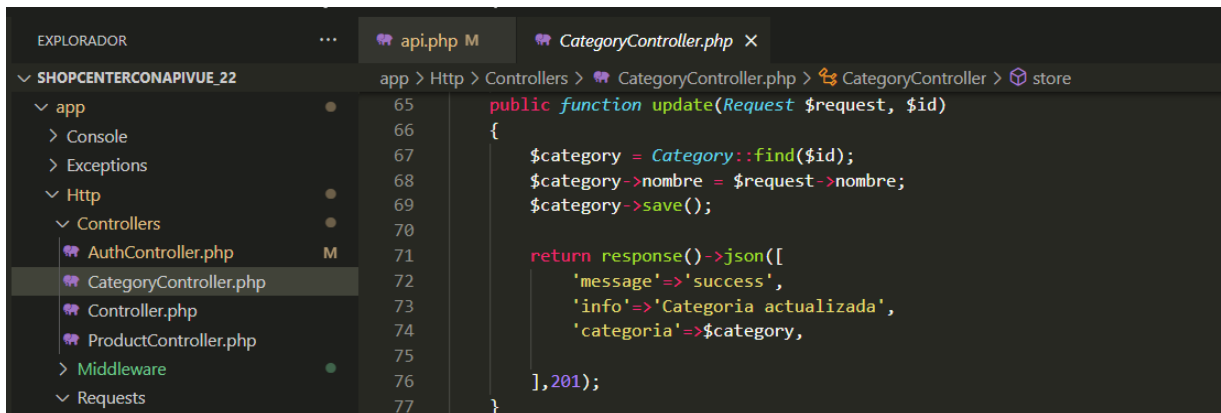
En cabecera creamos el Accept de value application/json



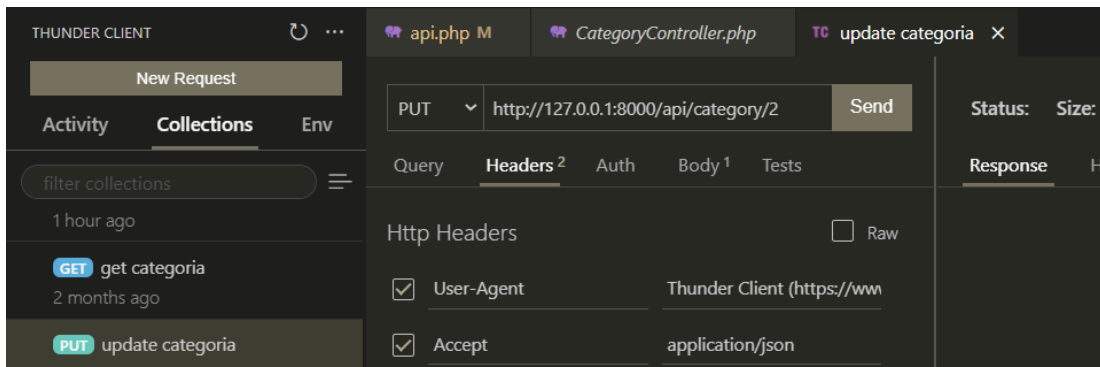
En body en formato Json creamos la categoría que vamos a almacenar



Creación del método modificar category



Verificamos en Thunder client



## 1. Practica. Creación del CRUD de Product

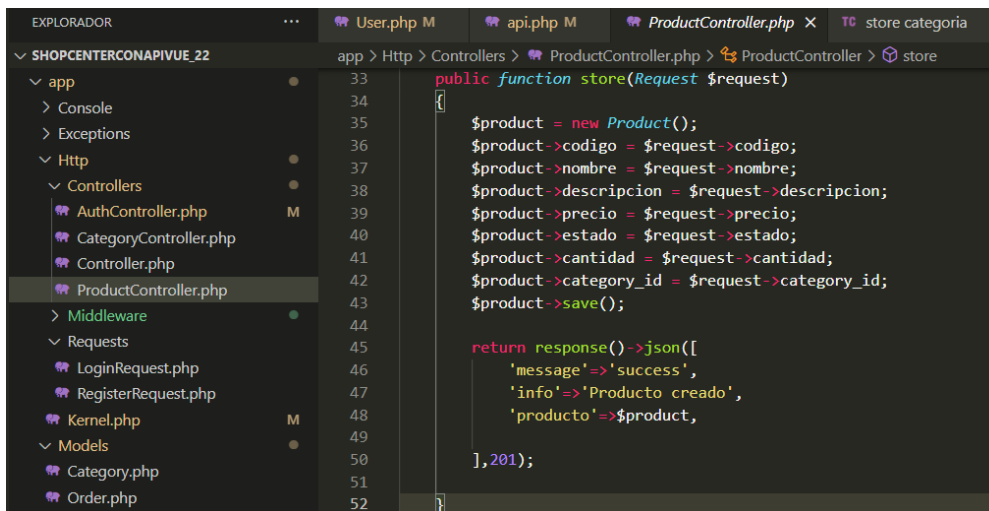
Creamos un controlador de tipo API RESOURCE

```
php artisan make:controller ProductController --api
```

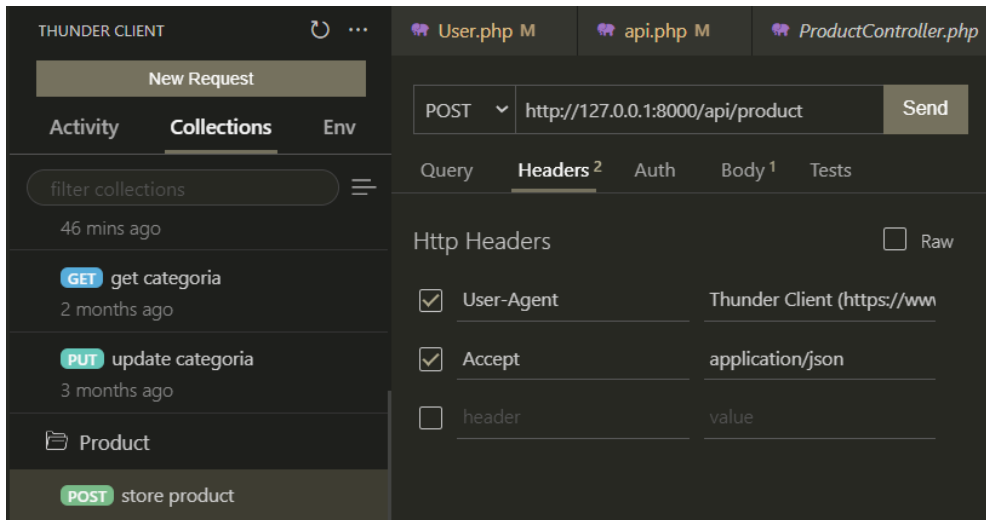
Creamos la ruta en el archivo API para el controlador

```
Route::apiResource('/product',ProductController::class);
```

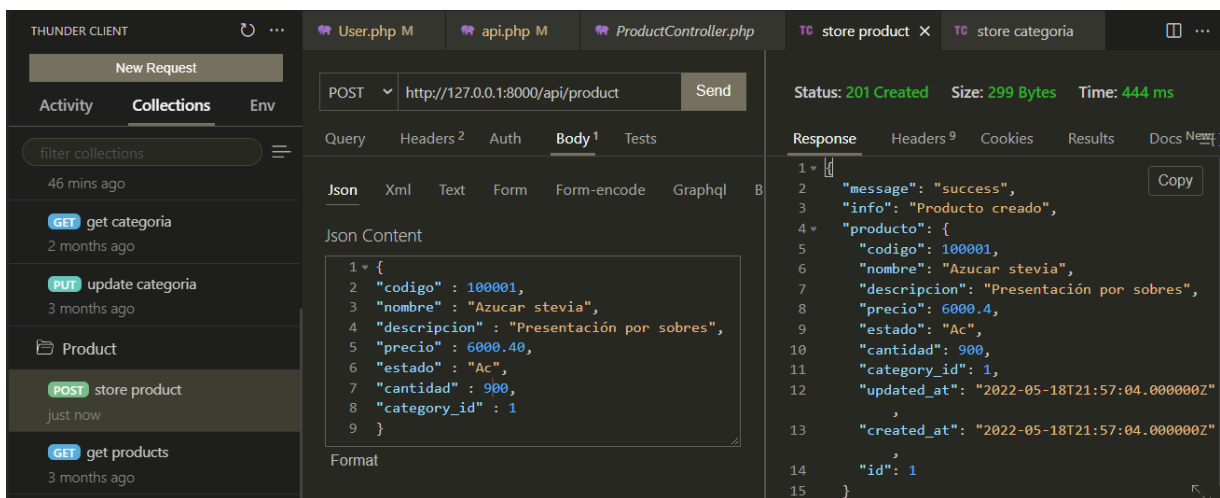
Iniciamos trabajando con el método **store**



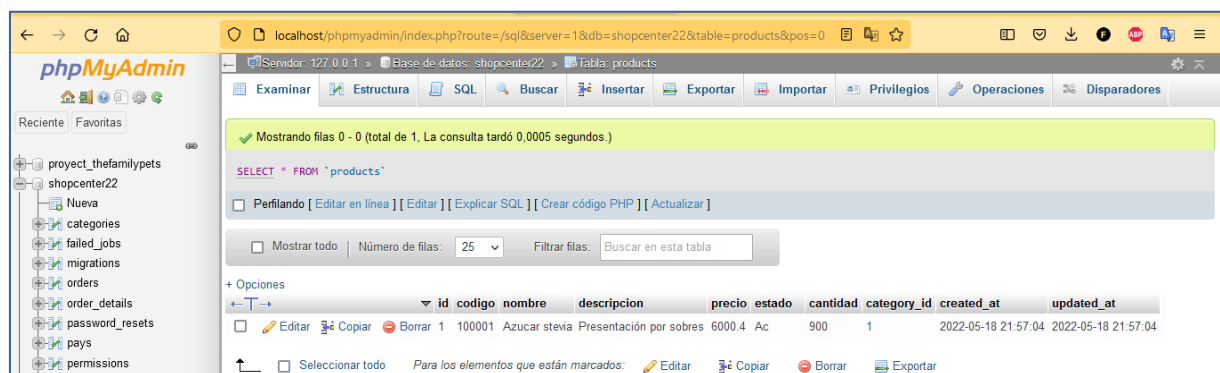
Verificamos en **Thunder client** el envío de datos




En el body ajustamos el contenido de envío en formato Json



Verificación en la base de datos



Creamos el método Index para listar los productos creados

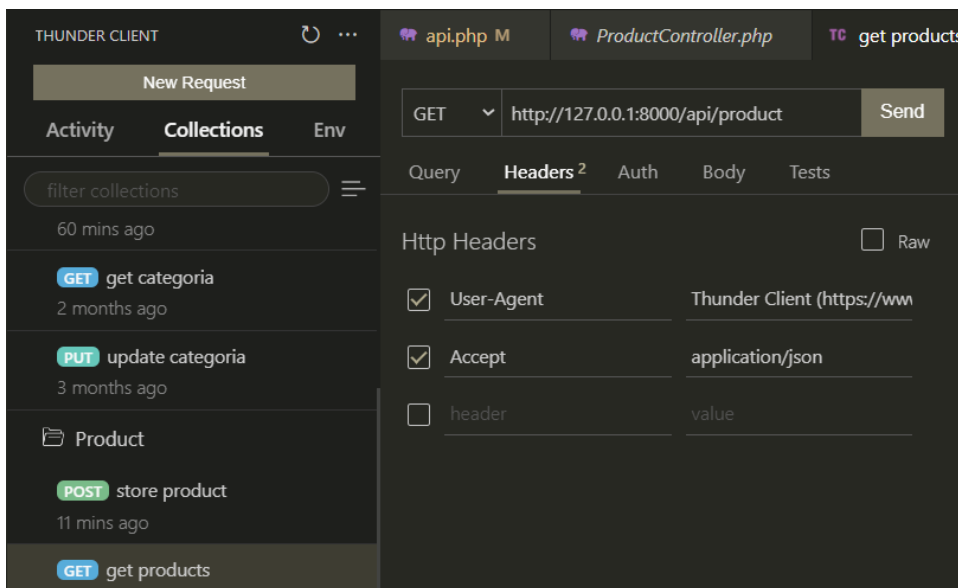


```

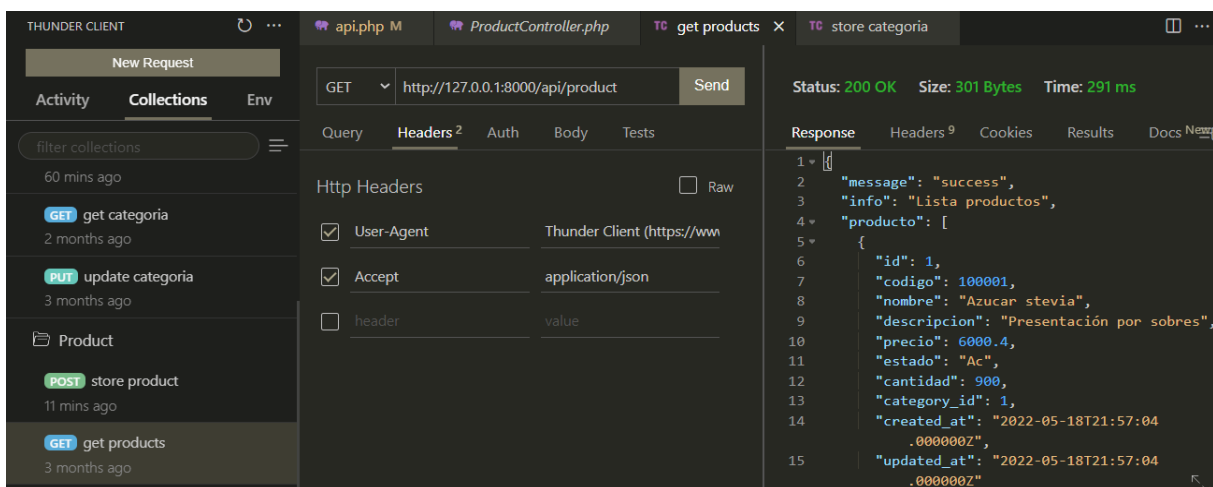
15 public function index()
16 {
17     $products = Product::all();
18
19     return response()->json([
20         'message'=>'success',
21         'info'=>'Lista productos',
22         'producto'=>$products,
23     ],200);
24 }

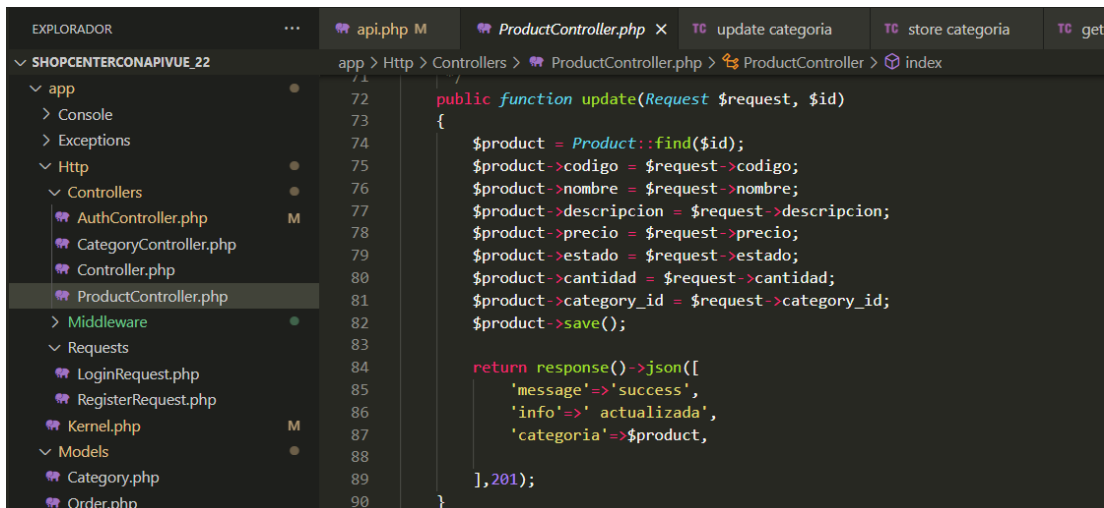
```

Verificamos en Thunder client su funcionamiento



En el método GET enviamos la ruta





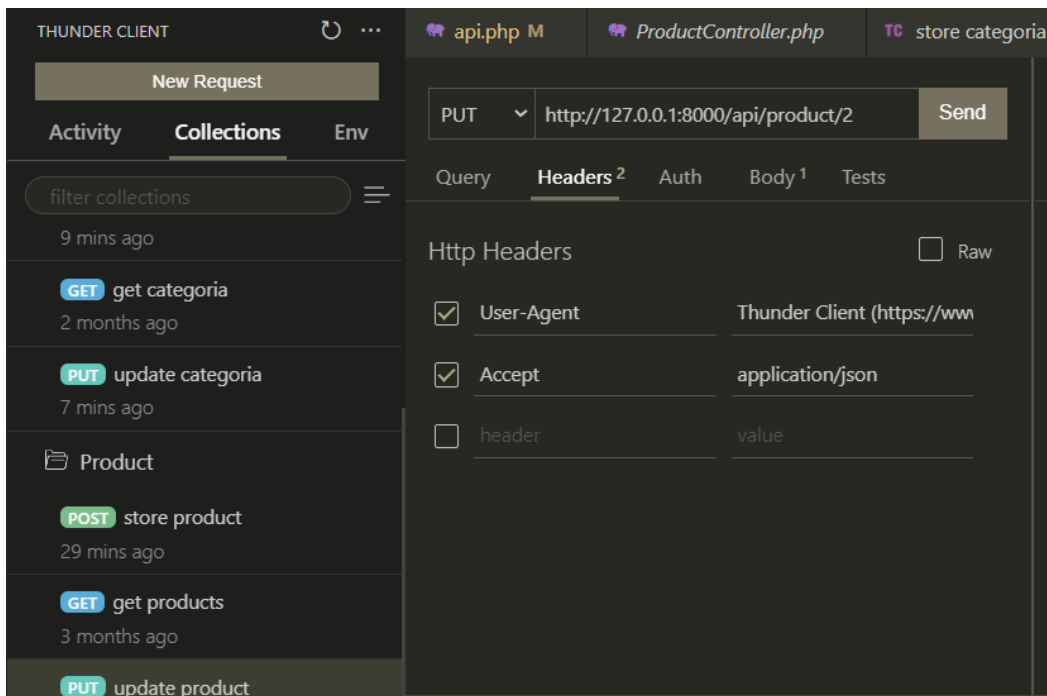
```

public function update(Request $request, $id)
{
    $product = Product::find($id);
    $product->codigo = $request->codigo;
    $product->nombre = $request->nombre;
    $product->descripcion = $request->descripcion;
    $product->precio = $request->precio;
    $product->estado = $request->estado;
    $product->cantidad = $request->cantidad;
    $product->category_id = $request->category_id;
    $product->save();

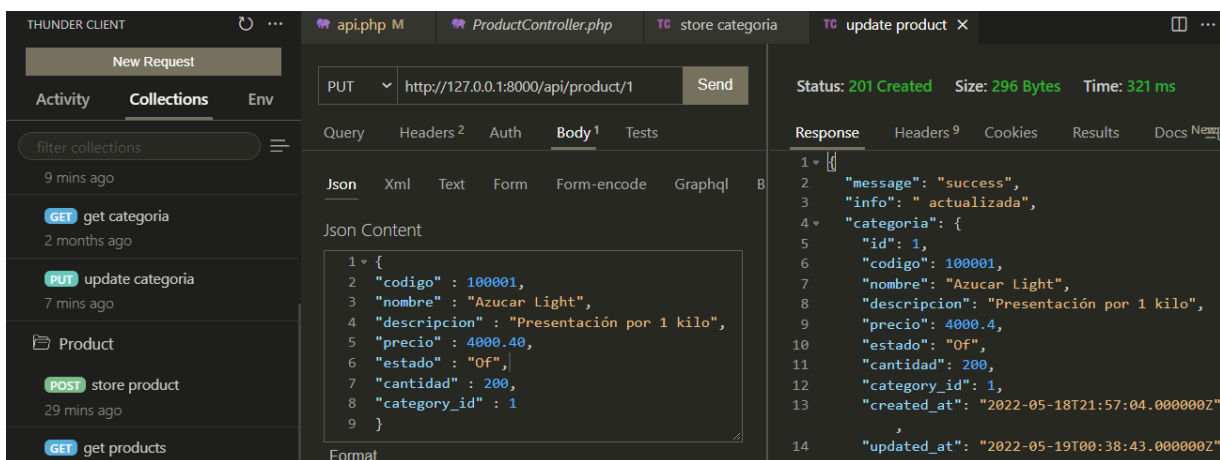
    return response()->json([
        'message'=>'success',
        'info'=>' actualizada',
        'categoria'=>$product,
    ],201);
}

```

Validamos en Thunder client su ejecución



Envío desde la pestaña body



## Bibliografía

<https://laravel.com/docs/8.x>

<https://www.youtube.com/watch?v=KKpXpWCTlbo&list=PLPI81lqbj-4KHPEGngoy5PSjjxcwnpCdb>

<https://aprendible.com/series/laravel-desde-cero>

<https://imacreste.com/bootstrap-colores-fuentes-y-tipografias/#Small>

<https://bluuweb.github.io/tutorial-laravel/bases-datos/>

**W3C The World Wide Web Consortium (2016).** HTML 5.1 W3C Recommendation.

Recuperado de: <https://www.w3.org/TR/html>

**W3C The World Wide Web Consortium (2017).** CSS Snapshot 2017. Recuperado de:

<https://www.w3.org/TR/CSS>

<https://www.cursosdesarrolloweb.es/cursos-gratuitos/>

## CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor(es)	Franco Reina	Instructor	CTPI	Junio 2022