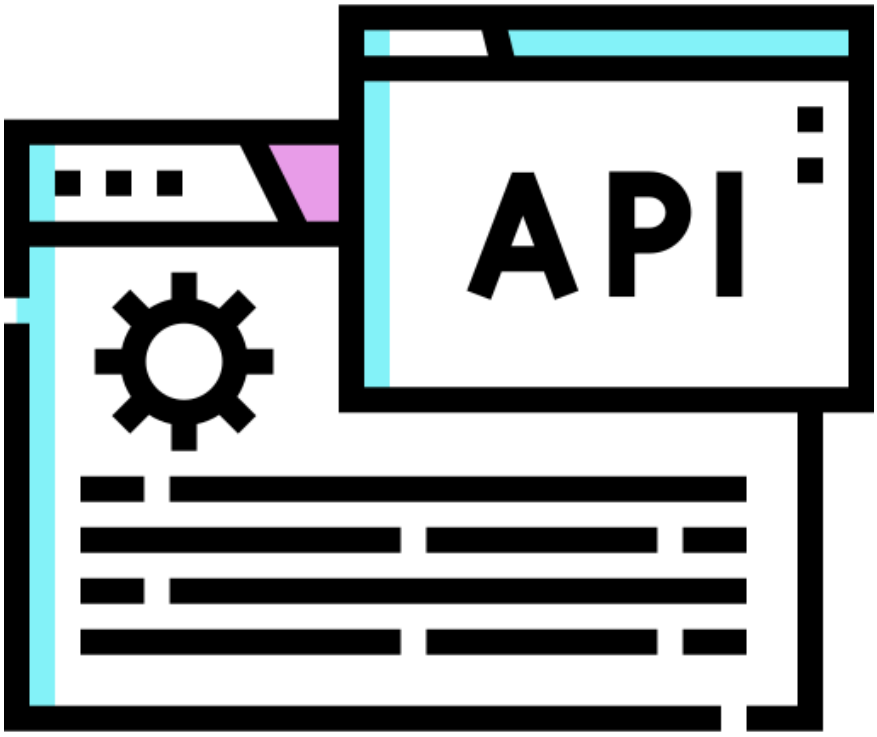
Desarrollo de API

Especialidad PHP



Felipe Izquierdo Romero

Índice

1. Descripción del problema

Desarrollar una API que permita visualizar el listado de productos con los siguientes datos: nombre, descripción, precio y stock.

Además, se podrán eliminar productos crear y modificar los datos de los mismos.

2. Desarrollo

Este desarrollo de API se ha usado una base de datos MySQL y tres frameworks diferentes que hemos aprendido en laboratorios anteriores, Symfoy, Laravel y CodeIgniter. Lo único que varia es el controlador que no retorna una vista sino datos en formato JSON. Por lo que será lo que se muestre en este documento rutas que serían los EndPoints de mi API y los controladores.

Empezando con Symfony las rutas se han definido con "Route" encima de cada función en el controlador "ProductorControler.php". Las funciones son las siguientes:

- Index

```php
/**
 * @Route("/", name="index")
 */
public function index(): JsonResponse
{
    $products = $this->entityManager->getRepository(Productos::class)->findAll();

    $productsArray = [];
    foreach ($products as $product) {
        $productsArray[] = [
            'id' => $product->getId(),
            'name' => $product->getName(),
            'price' => $product->getPrice(),
            'stock' => $product->getStock()
        ];
    }

    return new JsonResponse($productsArray, 200);
}
```

Ilustración 1- Index Symfony

- Créate

```php
/**
 * @Route("/create", name="create")
 */
public function create(Request $request): JsonResponse
{
    $data = $request->getContent();

    $product = $this->serializerInterface->deserialize($data,Productos::class,'json');

    $this->entityManager->persist($product);
    $this->entityManager->flush();

    return new JsonResponse(['message','Producto añadido correctamente'], 200);
}
```

Ilustración 2- Create Symfony

- Show One

```php
    /**
     * @Route("/showOne/{id}", name="showOne")
     */
    public function showOne($id): JsonResponse
    {
        $product = $this->entityManager->getRepository(Productos::class)->find($id);

        if (!$product) {
            return new JsonResponse(['error' => 'Producto no encontrado'],404);
        }

        $productSerialized = $this->serializerInterface->serialize($product,'json');

        return new JsonResponse($productSerialized,200);
    }
```

Ilustración 3- ShowOne Symfony

- Delete

```php
    /**
     * @Route("/delete/{id}", name="delete")
     */
    public function delete($id): JsonResponse
    {
        $product = $this->entityManager->getRepository(Productos::class)->find($id);

        if (!$product) {
            return new JsonResponse(['error' => 'Producto no encontrado'],404);
        }

        $this->entityManager->remove($product);
        $this->entityManager->flush();

        return new JsonResponse(['message','Producto borrado correctamente'],200);
    }
```

Ilustración 4- Delete Symfony

- Modify

```php
    /**
     * @Route("/modify/{id}",name="modify")
     */
    public function modify($id,Request $request)
    {
        $data = $request->getContent();
        $dataModify = $this->serializerInterface->deserialize($data,Productos::class,'json');

        $product = $this->entityManager->getRepository(Productos::class)->find($id);

        if(!$product)
        {
            return new JsonResponse(['error','Producto no encontrado'],404);
        }

        if($dataModify->getName())
        {
            $product->setName($dataModify->getName());
        }
        if($dataModify->getDescription())
        {
            $product->setDescription($dataModify->getDescription());
        }
        if($dataModify->getPrice())
        {
            $product->setPrice($dataModify->getPrice());
        }
        if($dataModify->getStock())
        {
            $product->setStock($dataModify->getStock());
        }

        $this->entityManager->flush();

        return new JsonResponse(['status' => 'Producto modificado'], JsonResponse::HTTP_OK);
    }
```

Ilustración 5- Modify Symfony

Ahora vamos con Laravel primero veremos las rutas de la API:

```php
Route::get('/', [ProductosController::class, 'index'])
    ->name('index');

Route::get('/{id}',[ProductosController::class,'showOne'])
    ->name('show_one');

Route::post('/', [ProductosController::class, 'store'])
    ->name('store');

Route::delete('/delete/{id}', [ProductosController::class, 'delete'])
    ->name('delete');

Route::put('/modify/{id}/{field}/{value}', [ProductosController::class, 'modify'])
    ->name('modify');

Route::put('/modify_json/{id}', [ProductosController::class, 'modify_json'])
    ->name('modify_json');
```

Ilustración 6- Rutas Laravel

Después las funciones del controlador que tienen nombres similares.

- Index

```php
15      public function index()
16      {
17          $products = Producto::all();
18          return json_encode($products);
19      }
```

Ilustración 7- Index Laravel

- Show One

```php
21      public function showOne($id)
22      {
23          $product = Producto::find($id);
24          return json_encode($product);
25      }
```

Ilustración 8- Show One Laravel

- Store

```php
27      public function store(Request $request)
28      {
29
30          $data = $request->validate(
31              [
32                  'name' => 'required',
33                  'description' => 'required',
34                  'price' => 'required',
35                  'stock' => 'required'
36              ]);
37
38
39          $product = Producto::create($data);
40
41          $product['message']= 'Producto añadido correctamente';
42          return new JsonResponse($product,201);
43
44      }
```

Ilustración 9- Store Laravel

- Delete

```php
public function delete($id)
{
    $product = Producto::find($id);

    if($product)
    {
        $product->delete();

        return new JsonResponse(['message'=>'Producto eliminado con exito'],201);
    }
    else
    {
        return new JsonResponse(['message'=>'Producto no encontrado'],404);
    }
}
```

Ilustración 10- Delete Laravel

- Modify

```php
public function modify(Request $request, $id, $field, $value)
{

    $allowedFields = ['name', 'description', 'price', 'stock'];

    // Validar que el campo solicitado sea uno de los campos permitidos
    if (!in_array($field, $allowedFields)) {
        return new JsonResponse(['message' => 'Campo no permitido para modificar'], 400);
    }

    $producto = Producto::find($id);

    if (!$producto) {
        return new JsonResponse(['message' => 'Producto no encontrado'], 404);
    }

    $producto->{$field} = $value;

    $producto->save();

    return new JsonResponse([
        'message' => 'Campo ' . $field . ' modificado correctamente',
        'product' => $producto,
    ]);
}
```

Ilustración 11- Modify Laravel

- Modify JSON

```php
public function modify_json(Request $request, $id)
{
    $producto = Producto::find($id);

    if (!$producto) {
        return new JsonResponse(['message' => 'Producto no encontrado'], 404);
    }

    // Aplicar cada cambio recibido
    foreach ($request->input('changes') as $change) {
        $field = $change['field'];
        $value = $change['value'];

        $producto->{$field} = $value;
    }

    $producto->save();

    return new JsonResponse([
        'message' => 'Campos modificados correctamente',
        'product' => $producto,
    ]);
}
```

Ilustración 12- Modify JSON Laravel

A continuación, las rutas usadas en Codeigniter.

```php
/**
 * @var RouteCollection $routes
 */
$routes->get('/', 'ProductosController::index');
$routes->get('/(:num)', 'ProductosController::showOne/$1');
$routes->post('/create', 'ProductosController::create');
$routes->delete('/delete/(:num)','ProductosController::delete/$1');
$routes->put('modify/(:num)','ProductosController::modify/$1');
```

Ilustración 13- Rutas Codeigniter

Las funciones del controlador:

- Index

```php
public function index()
{
    $product = new Producto();
    $data = $product->findAll();
    return json_encode($data);
}
```

Ilustración 14- Index CodeIgniter

- Show One

```php
public function showOne($id)
{
    $product = new Producto();
    $data = $product->find($id);
    return json_encode($data);
}
```

Ilustración 15- Show One CodeIgniter

- Create

```php
public function create()
{
    $product = new Producto();

    $data = $this->request->getJSON(true);//Para devolver el json como array asociativo TRUE

    if($product->insert($data))
    {
        return $this->response->setJSON(['message' => 'Producto creado correctamente']);
    }
    else
    {
        return $this->response->setJSON(['message' => 'Error al crear el producto']);
    }
}
```

Ilustración 16- Create CodeIgniter

- Delete

```php
public function delete($id)
{
    $product = new Producto();
    $data = $product->find($id);
    if(!$data)
    {
        return $this->response->setJSON(['message' => 'Producto no existe']);
    }
    if($product->delete($id))
    {
        return $this->response->setJSON(['message' => 'Producto borrado correctamente']);
    }
    else
    {
        return $this->response->setJSON(['message' => 'Fallo al borrar producto']);
    }

}

```

Ilustración 17- Delete CodeIgniter

- Modify

```php
public function modify($id)
{
    $request = $this->request;
    $dataRequest = $request->getJSON(true);

    $product = new Producto();
    $dataProduct = $product->find($id);

    if(!$dataProduct)
    {
        return $this->response->setJSON(['message' => 'Producto no existe']);
    }
    if ($product->update($id,$dataRequest)) {
        return $this->response->setJSON(['message' => 'Producto modificado correctamente']);
    } else {
        return $this->response->setJSON(['message' => 'Error al modificar producto']);
    }
}
```

Ilustración 18- ModifyCodeIgniter