

TEST NestJS

1. @Post: Registro un Producto.

POST

Query Headers Auth **Body¹** Tests Pre Run ^{New}

Json Xml Text Form Form-encode Graphql Binary

Json Content

```
1 {
2   "title": "Mountains Duo",
3   "description": "Cuadros de 45x30 cm. vinilo pegado sobre
4     madera.",
5   "code": "18012481",
6   "price": 1000,
7   "stock": 10
8 }
9 }
```

Status: **201 Created** Size: 173 Bytes Time: 17 ms

Response Headers⁶ Cookies Results Docs {}

```
1 {
2   "title": "Mountains Duo",
3   "description": "Cuadros de 45x30 cm. vinilo pegado sobre
4     madera.",
5   "code": "18012481",
6   "price": 1000,
7   "stock": 10,
8   "_id": "63e81129862e13c02cfa1053",
9   "_v": 0
10 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL**

```
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RoutesResolver] ProductsController {/products}: +1ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/products, POST} route +1ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/products, GET} route +1ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/products/:id, GET} route +0ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/products/:id, PATCH} route +1ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [RouterExplorer] Mapped {/products/:id, DELETE} route +2ms
[Nest] 7152 - 11/02/2023, 18:45:52 LOG [NestApplication] Nest application successfully started +2ms
```

+ v ... ^

2. @Post: Registro un segundo Producto probando las validaciones

a) Validate @IsNotEmpty.

POST

Query Headers Auth **Body¹** Tests Pre Run ^{New}

Json Xml Text Form Form-encode Graphql Binary

Json Content

```
1 {
2   "title": "Alfombra Frise",
3   "description": "Plumón de alta calidad, 113 cm de
4     ancho x 190 de largo.",
5   "price": 2000,
6   "stock": 20
7 }
```

Status: **400 Bad Request** Size: 79 Bytes Time: 4 ms

Response Headers⁶ Cookies Results Docs

```
1 {
2   "statusCode": 400,
3   "message": [
4     "code should not be empty"
5   ],
6   "error": "Bad Request"
7 }
```

b) Validate @IsNumber.

POST

Query Headers Auth **Body¹** Tests Pre Run ^{New}

Json Xml Text Form Form-encode Graphql Binary

Json Content

```
1 {
2   "title": "Alfombra Frise",
3   "description": "Plumón de alta calidad, 113 cm de
4     ancho x 190 de largo.",
5   "code": "28012482",
6   "price": "2000",
7   "stock": 20
8 }
```

Status: **400 Bad Request** Size: 117 Bytes Time: 5 ms

Response Headers⁶ Cookies Results Docs

```
1 {
2   "statusCode": 400,
3   "message": [
4     "price must be a number conforming to the specified
5     constraints"
6   ],
7   "error": "Bad Request"
8 }
```

c) Registro Correcto.

POST `http://localhost:3000/products` **Send**

Query Headers Auth **Body¹** Tests Pre Run *New*

Json Xml Text Form Form-encode GraphQL Binary

Json Content Format

```
1 {
2   "title": "Alfombra Frise",
3   "description": "Plumón de alta calidad, 113 cm de ancho x 190
4     de largo.",
5   "code": "28012482",
6   "price": 2000,
7   "stock": 20
8 }
```

Status: **201 Created** Size: **182 Bytes** Time: **7 ms**

Response Headers⁶ Cookies Results Docs {} ≡

```
1 {
2   "title": "Alfombra Frise",
3   "description": "Plumón de alta calidad, 113 cm de ancho x 190
4     de largo.",
5   "code": "28012482",
6   "price": 2000,
7   "stock": 20,
8   "_id": "63e8130b862e13c02cfa1055",
9   "_v": 0
10 }
```

2. @Get: Consulto todos los productos registrados.

GET `http://localhost:3000/products` **Send**

Query **Headers** Auth Body¹ Tests Pre Run *New*

Http Headers ☐ Raw

<input type="checkbox"/> Accept	*/*
<input type="checkbox"/> User-Agent	Thunder Client (https://www.thunder
<input type="checkbox"/> header	value

Status: **200 OK** Size: **358 Bytes** Time: **9 ms**

Response Headers⁶ Cookies Results Docs {} ≡

```
1 [
2   {
3     "_id": "63e81129862e13c02cfa1053",
4     "title": "Mountains Duo",
5     "description": "Cuadros de 45x30 cm. vinilo pegado sobre
6       madera.",
7     "code": "18012481",
8     "price": 1000,
9     "stock": 10,
10    "_v": 0
11  },
12  {
13    "_id": "63e8130b862e13c02cfa1055",
14    "title": "Alfombra Frise",
15    "description": "Plumón de alta calidad, 113 cm de ancho x
16      190 de largo.",
17    "code": "28012482",
18    "price": 2000,
19    "stock": 20,
20    "_v": 0
21  }
22 ]
```

3. @Get('id'): Consulto un producto en particular.

GET `http://localhost:3000/products/63e8130b862e13c02cfa1055` **Send**

Query **Headers** Auth Body¹ Tests Pre Run *New*

Http Headers ☐ Raw

<input type="checkbox"/> Accept	*/*
<input type="checkbox"/> User-Agent	Thunder Client (https://www.thunder
<input type="checkbox"/> header	value

Status: **200 OK** Size: **182 Bytes** Time: **6 ms**

Response Headers⁶ Cookies Results Docs {} ≡

```
1 {
2   "_id": "63e8130b862e13c02cfa1055",
3   "title": "Alfombra Frise",
4   "description": "Plumón de alta calidad, 113 cm de ancho x 190
5     de largo.",
6   "code": "28012482",
7   "price": 2000,
8   "stock": 20,
9   "_v": 0
10 }
```

4. @Patch('id'): Actualizo un producto con nuevos datos.

The screenshot shows the Thunder Client interface. The top bar indicates a PATCH request to `http://localhost:3000/products/63e8130b862e13c02cfa1055` with a status of 200 OK, size of 92 Bytes, and time of 10 ms. The 'Body' tab is selected, showing the JSON content of the request:

```
1 {
2   "title": "Blue Vase",
3   "description": "Florero de cerámica esmaltada a mano
4     , 38 cm x 65 cm.",
5   "code": "38012483",
6   "price": 3000,
7   "stock": 30
8 }
```

The 'Response' tab shows the JSON response:

```
1 {
2   "acknowledged": true,
3   "modifiedCount": 1,
4   "upsertedId": null,
5   "upsertedCount": 0,
6   "matchedCount": 1
7 }
```

5. @Delete('id'): Elimino un producto en particular.

The screenshot shows the Thunder Client interface. The top bar indicates a DELETE request to `http://localhost:3000/products/63e81129862e13c02cfa1053` with a status of 200 OK, size of 38 Bytes, and time of 5 ms. The 'Headers' tab is selected, showing the raw headers. The 'Response' tab shows the JSON response:

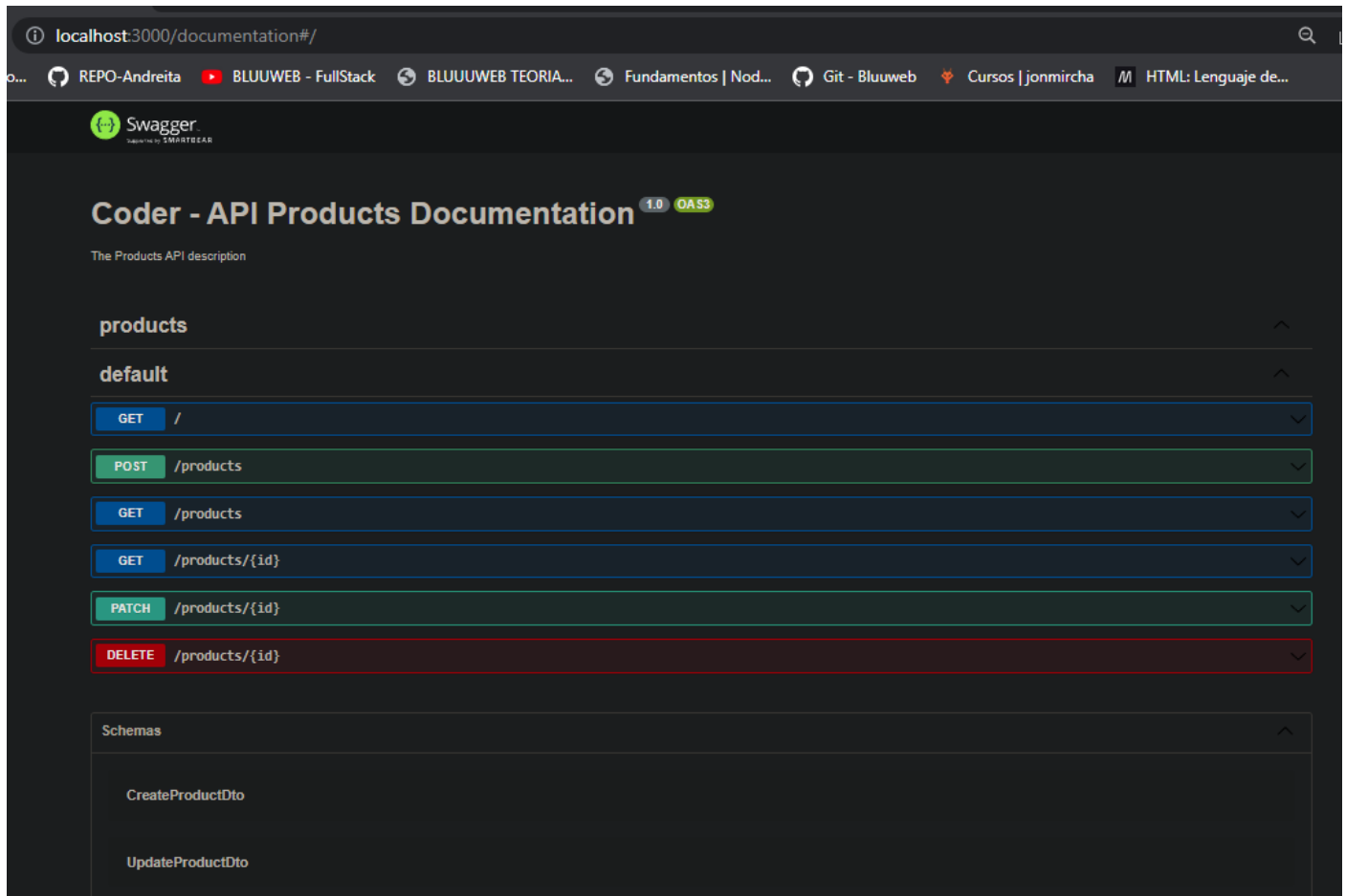
```
1 {
2   "acknowledged": true,
3   "deletedCount": 1
4 }
```

6. @Get: Consulto todos los productos registrados para verificar el producto modificado y que no exista el eliminado.

The screenshot shows the Thunder Client interface. The top bar indicates a GET request to `http://localhost:3000/products` with a status of 200 OK, size of 176 Bytes, and time of 6 ms. The 'Headers' tab is selected, showing the raw headers. The 'Response' tab shows the JSON response:

```
1 [
2   {
3     "_id": "63e8130b862e13c02cfa1055",
4     "title": "Blue Vase",
5     "description": "Florero de cerámica esmaltada a mano, 38
6       cm x 65 cm.",
7     "code": "38012483",
8     "price": 3000,
9     "stock": 30,
10    "__v": 0
11  }
12 ]
```

7. Test documentación con Swagger.



The screenshot shows the Swagger UI for an API titled "Coder - API Products Documentation". The interface is dark-themed. At the top, the browser address bar shows "localhost:3000/documentation#/". Below the Swagger logo, the title "Coder - API Products Documentation" is displayed with version "1.0" and "OAS3" tags. A subtitle "The Products API description" is present. The main section is titled "products" and contains a "default" section. Under "default", there are six endpoints listed with their HTTP methods and paths:

- GET /
- POST /products
- GET /products
- GET /products/{id}
- PATCH /products/{id}
- DELETE /products/{id}

Below the endpoints, there is a "Schemas" section which lists two schemas: "CreateProductDto" and "UpdateProductDto".