NEST 15

- Swagger es agnóstico, sirve para cualquier lenguaje, pero NEST tiene su propio paquete

```
npm i --save @nestjs/swagger
```

- En el main, antes del app.listen, debo colocar un bloque de código extraído de la documentación

```
  const config = new DocumentBuilder()
    .setTitle('Cats example')
    .setDescription('The cats API description')
    .setVersion('1.0')
    .addTag('cats')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api', app, document);
```

- Le quito el tag, renombro el titulo y la descripción. Hago la importación de DocumentBuilder y SwaggerModule de @nestjs/swagger

```
  const config = new DocumentBuilder()
  .setTitle('Teslo Shop')
  .setDescription('RESTFul API Teslo')
  .setVersion('1.0')
  .build();
const document = SwaggerModule.createDocument(app, config);
SwaggerModule.setup('api', app, document);
```

- Si voy al localhost:3000/api en el navegador veo los primeros pasos de una documentación
- Todavía le faltan configuraciones

Tags, ApiProperty y ApiResponse

- Estaría bien agrupar todos los endpoints que tengan en común (products, seed, etc) tuvieran su propia etiqueta
- Para esto haré uso de decoradores que vienen en @nestjs/swagger
- products.controller:

```
@ApiTags('Products')
@Controller('products')
export class ProductsController {
  constructor(private readonly productsService: ProductsService) {}
```

- Lo mismo con el resto (seed, files, auth)
- Me interesa cuando visito en la documentación el endpoint Post de creación de producto, que tipo de data espera y que respuesta obtengo
- Para ello se hará uso de un nuevo decorador, usualmente se coloca debajo del tipo de petición en el controller
- Le puedo poner como luce la respuesta con type: Product (entity). Va a regresar algo de tipo Product
- La data de retorno así luce como un objeto vacío

```
@Post()
  @ApiResponse({status: 201, description: 'Product was created', type: Product})
  @ApiResponse({status: 400, description: 'Bad Request'})
  @ApiResponse({status: 403, description: 'Token is not valid'})
  @Auth()
  create(
    @Body() createProductDto: CreateProductDto,
    @GetUser() user: User
    ) {
    return this.productsService.create(createProductDto, user);
```

- Debo anotar en la entidad que propiedades quiero en la respuesta con @ApiProperty

```
@Entity({ name: 'products'})
export class Product {

    @ApiProperty()
    @PrimaryGeneratedColumn('uuid')
    id: string

    @ApiProperty()
    @Column('text',{
        unique: true
    })
    title: string
```

- Para expandir esta información hay varias opciones dentro del objeto de @ApiProperty

Expandir el ApiProperty

- Puedo poner un ejemplo de como luce un ID copiando un ID de un producto, añadirle una descripción

```
    @ApiProperty({
        example: 'cd54663-4344-665-ad34-57s8',
        description: 'Product ID',
        uniqueItems: true
    })
```

```
    @PrimaryGeneratedColumn('uuid')
    id: string
```

- Y así ir especificando los campos de la entidad para que luzca en el ejemplo
- Si yo miro en el controllador el Get, veo que esta esperando el PaginationDto
- Cómo hago para que quede indicado en la documentación? Y los valores del limite y el offset ene l GetProducts?
- En el patch del producto tampoco aparece nada porque es la expansión de otro Dto
- Documentar un Dto es importante para saber porqué es que responde un BadRequest

Documentar Dto

- El decorador se implementa en el Dto, es por eso que todos los Dtos son clases y no interfaces
- Usaré el @ApiProperty en el PaginationDto

```
import { IsNumber, IsOptional, IsPositive } from "class-validator";
import {Type} from 'class-transformer'
import { ApiProperty } from "@nestjs/swagger";

export class PaginationDto{

    @ApiProperty({
        default: 10,
        description: "How many rows do you need"
    })
    @IsOptional()
    @IsPositive()
    @Type(()=> Number)
    limit?: number;

    @ApiProperty({
        default: 0,
        description: "How many rows do you want to skip"
    })
    @IsOptional()
    @IsPositive()
    @Type(()=> Number)
    offset?: number

}
```

- Voy al createProductDto

```
export class CreateProductDto {

    @ApiProperty({
        description: "Product title (unique)",
```

```
        nullable: false,
        minLength: 1
    })
    @IsString()
    @MinLength(1)
    title: string;

    @ApiProperty()
    @IsNumber()
    @IsPositive()
    @IsOptional()
    price?: number;
```

- Ahora no tengo forma de especificar en el UpdateProductDto porque el PartialType no lo permite
- El equipo de swagger sabe bien esto y proporciona su propio PartialType, lo importo de swagger en lugar de map-types