NEST DOCUMENTACION SWAGGER

> npm i --save @nestjs/swagger

- En el main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { Logger, ValidationPipe } from '@nestjs/common';
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  const logger = new Logger('bootstrap')

  app.setGlobalPrefix('api')

  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      forbidNonWhitelisted: true
    })
  )

  //SWAGGER----------------------------------------------------
  const config = new DocumentBuilder()
    .setTitle('Teslo RESTFul API')
    .setDescription('Teslo Shop')
    .setVersion('1.0')
    .build()

  const document= SwaggerModule.createDocument(app, config)
  SwaggerModule.setup('api', app, document) //en el endpoint api va a crear la
documentación
 //SWAGGER----------------------------------------------------

  await app.listen(process.env.PORT);
  logger.log(`App running on port ${process.env.PORT}`)
}
bootstrap();
```

> localhost:3000/api

- Evidentemente faltan configuraciones, no hay data de ejemplo, ni referencias...

Tags, ApiProperty, ApiResponse

- Quiero agrupar los endpoints de productos, auth, seed, etc
- Importo **@ApiTags** a nivel de controlador
- En products.controller

```
import { ApiTags } from '@nestjs/swagger';

@ApiTags('Products')
@Controller('products')
export class ProductsController
```

- Hago lo mismo en el resto de controladores (seed, auth, files)
- En el POST me gustaría saber **qué tipo de data** está esperando, **que es obligatorio** o no y **qué tipo de respuestas** hay
- Para ello uso **@ApiResponse** de swagger
- products.controller

```
@Post()
  @Auth(ValidRoles.admin)
  @ApiResponse({status: 201, description: 'Product was created'})
  @ApiResponse({status: 400, description: 'Bad request'})
  @ApiResponse({status: 403, description: 'Forbidden. Token related'})
  create(
    @Body() createProductDto: CreateProductDto,
    @GetUser() user: User,
    ) {
          return this.productsService.create(createProductDto, user);
    }
```

- Pongamos que quiero saber cómo va a lucir la respuesta
- Para ello puedo usar el type
- Si todo sale bien estaría regresando un producto. Lo coloco en type

```
@Post()
@Auth(ValidRoles.admin)
@ApiResponse({status: 201, description: 'Product was created', type: Product})
@ApiResponse({status: 400, description: 'Bad request'})
@ApiResponse({status: 403, description: 'Forbidden. Token related'})
create(
@Body() createProductDto: CreateProductDto,
@GetUser() user: User,
) {
        return this.productsService.create(createProductDto, user);
}
```

- Debo ir a la entity para añadir **@ApiProperty** a cada propiedad
- El usuario no lo decoro porque daría error porque no hay establecida la relación directamente

```
import {Entity, PrimaryGeneratedColumn, Column, BeforeInsert, BeforeUpdate,
OneToMany, ManyToOne} from 'typeorm'
import { ProductImage } from './product-image.entity'
import { User } from 'src/auth/entities/user.entity'
import { ApiProperty } from '@nestjs/swagger'

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

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

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

    @ApiProperty()
    @Column('float',{
        default: 0
    })
    price: number

    @ApiProperty()
    @Column({
        type: 'text',
        nullable: true
    })
    description: string

    @ApiProperty()
    @Column({
        type: 'text',
        unique: true
    })
    slug: string

    @ApiProperty()
    @Column({
        type: 'int',
        default: 0
    })
    stock: number

    @ApiProperty()
    @Column({
        type: 'text',
        array: true
    })
    sizes: string[]
```

```
    @ApiProperty()
    @Column({
        type: 'text',
    })
    gender: string

    @ApiProperty()
    @Column({
        type: 'text',
        array: true,
        default: []
    })
    tags: string[]


    @ApiProperty()
    @OneToMany(
        ()=> ProductImage,
        productImage=> productImage.product,
        {cascade:true, eager: true}
    )
    images?: ProductImage[]

    @ManyToOne(
        ()=>User,
        (user)=>user.product,
        {eager: true}
    )
    user: User

    @BeforeInsert()
    checkSlugInsert(){
        if(!this.slug){
            this.slug = this.title
        }

        this.slug = this.slug
        .toLowerCase()
        .replaceAll(' ', '_')
        .replaceAll("'", "")
    }

    @BeforeUpdate()
    checkSlugUpdate(){
        this.slug = this.slug
        .toLowerCase()
        .replaceAll(' ', '_')
        .replaceAll("'", "")
    }

}
```

- Me gustaría proporcionar más info, por ejemplo que el id no solo es un string, es un UUID, y es único
- Se hace en un objeto dentro de **@ApiProperty**

Expandir el ApiProperty

- En el id puedo añadir un id de ejemplo, una descripción y marcarlo como unique

```
@ApiProperty({
    example: '8da88a62-cd23-4662-a6ab-5a6c85e97bf6',
    description: 'Product ID',
    uniqueItems: true
})
@PrimaryGeneratedColumn('uuid')
id: string
```

- Si voy al Schema, en la documentación, ahora voy a tener más info
- Puedo hacer algo parecido con el título

```
@ApiProperty({
    example: "T-Shirt Teslo",
    description: "Product Title",
    uniqueItems: true
})
@Column('text', {
    unique: true
})
title: string
```

- Y así con el resto de propiedades...

```
import {Entity, PrimaryGeneratedColumn, Column, BeforeInsert, BeforeUpdate,
OneToMany, ManyToOne} from 'typeorm'
import { ProductImage } from './product-image.entity'
import { User } from 'src/auth/entities/user.entity'
import { ApiProperty } from '@nestjs/swagger'

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

    @ApiProperty({
        example: '8da88a62-cd23-4662-a6ab-5a6c85e97bf6',
        description: 'Product ID',
        uniqueItems: true
    })
    @PrimaryGeneratedColumn('uuid')
    id: string
```

```
    @ApiProperty({
        example: "T-Shirt Teslo",
        description: "Product Title",
        uniqueItems: true
    })
    @Column('text', {
        unique: true
    })
    title: string

    @ApiProperty({
        example: 0,
        description: 'Product Price',
    })
    @Column('float',{
        default: 0
    })
    price: number

    @ApiProperty({
        example: "This is a very weird t-shirt with weird colors",
        description: 'Product description',
        default: null
    })
    @Column({
        type: 'text',
        nullable: true
    })
    description: string

    @ApiProperty({
        example: 't_shirt_teslo',
        description: 'slug for SEO',
        uniqueItems: true
    })
    @Column({
        type: 'text',
        unique: true
    })
    slug: string

    @ApiProperty({
        example: '10',
        description:'Product Stock',
        default: 0
    })
    @Column({
        type: 'int',
        default: 0
    })
    stock: number

    @ApiProperty({
        example: ['M', 'S', 'L', 'XL'],
```

```
        description: 'Product Size',
    })
    @Column({
        type: 'text',
        array: true
    })
    sizes: string[]

    @ApiProperty({
        example: 'women',
        description: 'Product gender'
    })
    @Column({
        type: 'text',
    })
    gender: string

    @ApiProperty()
    @Column({
        type: 'text',
        array: true,
        default: []
    })
    tags: string[]


    @ApiProperty()
    @OneToMany(
        ()=> ProductImage,
        productImage=> productImage.product,
        {cascade:true, eager: true}
    )
    images?: ProductImage[]

    @ManyToOne(
        ()=>User,
        (user)=>user.product,
        {eager: true}
    )
    user: User

    @BeforeInsert()
    checkSlugInsert(){
        if(!this.slug){
            this.slug = this.title
        }

        this.slug = this.slug
        .toLowerCase()
        .replaceAll(' ', '_')
        .replaceAll("'", "")
    }

    @BeforeUpdate()
```

```
    checkSlugUpdate(){
        this.slug = this.slug
        .toLowerCase()
        .replaceAll(' ', '_')
        .replaceAll("'", "")
    }

}
```

- Me falta documentar los DTOS. Y cómo documento el update-product.dto si es una expansión de otro DTO?

Documentar DTOS

- Documentar dtos es fundamental, ya que si no el endpoint responderá un error
- Es sencillo. Si los dtos no fueran clases perderiamos la oportunidad de decorar las propiedades
- pagination.dto

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

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()
    @Min(0)
    @Type(()=> Number)
    offset?: number
}
```

- Ahora en la documentación puedo ver en el GET de /api/products el limit y el offset
- En el create-product.dto

```
import { ApiProperty } from "@nestjs/swagger"
import { IsString, MinLength, IsNumber, IsOptional, IsInt, IsPositive, IsArray,
```

```
IsIn } from "class-validator"

export class CreateProductDto {

    @ApiProperty({
        example: 'Blue Trousers',
        description: 'Product Title',
        nullable: false,
        minLength: 1
    })
    @IsString()
    @MinLength(1)
    title: string

    @ApiProperty()
    @IsNumber()
    @IsOptional()
    price?: number

    @ApiProperty()
    @IsString()
    @IsOptional()
    description?: string

    @ApiProperty()
    @IsString()
    @IsOptional()
    slug?: string

    @ApiProperty()
    @IsInt()
    @IsPositive()
    @IsOptional()
    stock?: number

    @ApiProperty()
    @IsString({each: true})
    @IsArray()
    sizes: string[]

    @ApiProperty()
    @IsIn(['men', 'women', 'kid','unisex'])
    gender: string

    @ApiProperty()
    @IsString({each:true})
    @IsArray()
    @IsOptional()
    tags?: string[]

    @ApiProperty()
    @IsString({each:true})
    @IsArray()
    @IsOptional()
```

```
    images?: string[]
}
```

- Pero no tengo el update-product.dto
- En lugar de importar PartialType de @nestjs/mapped-types lo importo de @nestjs/swagger

```
//import { PartialType } from '@nestjs/mapped-types';
import { PartialType } from '@nestjs/swagger';
import { CreateProductDto } from './create-product.dto';

export class UpdateProductDto extends PartialType(CreateProductDto) {}
```