Table of Contents

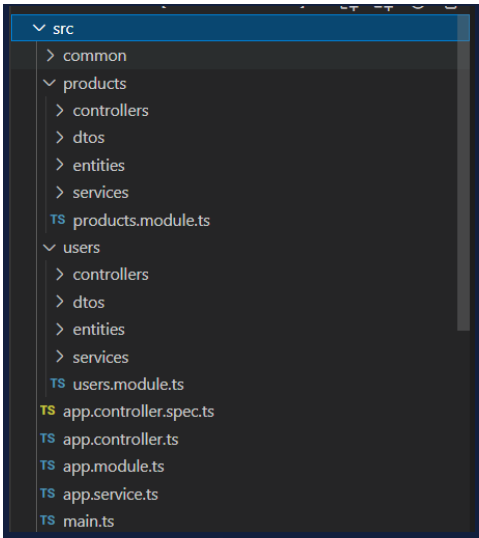
Curso de NestJS: Programación Modular, Documentación con Swagger y Deploy

1. Encapsular Logica en Modulos

Comando para Generar Modulos

nest g mo users
nest g mo products

La aplicación debería quedar organizada así:



Donde los módulos deberían quedar asi:

```
// src/products/products.module.ts
import { Module } from '@nestjs/common';

import { ProductsController } from './controllers/products.controller';
import { BrandsController } from './controllers/brands.controller';
import { CategoriesController } from
'./controllers/categories.controller';
import { ProductsService } from './services/products.service';
import { BrandsService } from './services/brands.service';
import { CategoriesService } from './services/categories.service';

@Module({
  controllers: [ProductsController, CategoriesController,
BrandsController],
  providers: [ProductsService, BrandsService, CategoriesService],
  exports: [ProductsService],
})
export class ProductsModule {}
// src/users/users.module.ts
import { Module } from '@nestjs/common';

import { CustomerController } from './controllers/customers.controller';
import { CustomersService } from './services/customers.service';
import { UsersController } from './controllers/users.controller';
import { UsersService } from './services/users.service';

import { ProductsModule } from '../products/products.module';

@Module({
  imports: [ProductsModule],
  controllers: [CustomerController, UsersController],
  providers: [CustomersService, UsersService],
})
export class UsersModule {}
// src/app.module.ts
```

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { UsersModule } from './users/users.module';
import { ProductsModule } from './products/products.module';

@Module({
  imports: [UsersModule, ProductsModule],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

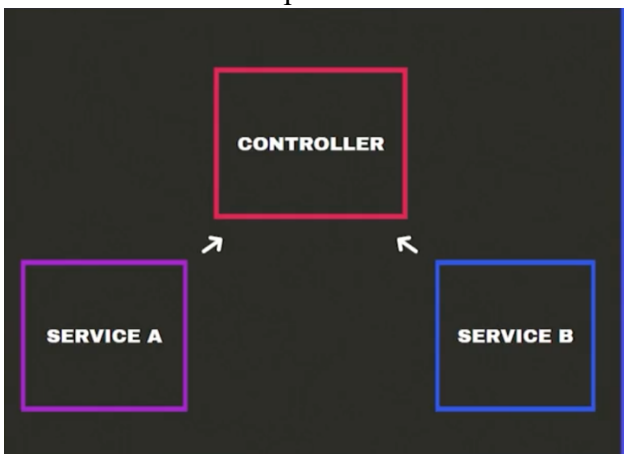
2.

3. Interaccion entre modulos

```
// src/users/entities/order.entity.ts
import { User } from './user.entity';
import { Product } from './../../products/entities/product.entity';

export class Order { //  // 👈 new entity
  date: Date;
  user: User;
  products: Product[];
}
// src/users/controllers/users.controller.ts
  @Get(':id/orders') //  👈 new endpoint
  getOrders(@Param('id', ParseIntPipe) id: number) {
    return this.usersService.getOrderByUser(id);
  }
// src/users/services/users.service.ts

...
import { Order } from '../entities/order.entity';

import { ProductsService } from
'./../../products/services/products.service';


@Injectable()
export class UsersService {
  constructor(private productsService: ProductsService) {}
  ...

  getOrderByUser(id: number): Order { // 👈 new method
    const user = this.findOne(id);
    return {
      date: new Date(),
      user,
      products: this.productsService.findAll(),
    };
  }
```

```
}
// src/products/products.module.ts

import { Module } from '@nestjs/common';

....

@Module({
  controllers: [ProductsController, CategoriesController,
BrandsController],
  providers: [ProductsService, BrandsService, CategoriesService],
  exports: [ProductsService], // 👈 Export ProductsService
})
export class ProductsModule {}
// src/users/users.module.ts
import { Module } from '@nestjs/common';

...

import { ProductsModule } from '../products/products.module';

@Module({
  imports: [ProductsModule], // 👈 Import ProductsModule
  controllers: [CustomerController, UsersController],
  providers: [CustomersService, UsersService],
})
export class UsersModule {}
```

4. Patron de Independencia



5. useValue y useClass

```
// src/app.module.ts
...
```

```
const API_KEY = '12345634';
const API_KEY_PROD = 'PROD1212121SA';

@Module({
  imports: [UsersModule, ProductsModule],
  controllers: [AppController],
  providers: [
    AppService,
    {
      provide: 'API_KEY',
      useValue: process.env.NODE_ENV === 'prod' ? API_KEY_PROD : API_KEY,
    },
  ],
})
export class AppModule {}
// src/app.service.ts
import { Injectable, Inject } from '@nestjs/common';

@Injectable()
export class AppService {
  constructor(@Inject('API_KEY') private apiKey: string) {} // 👈 Inject
API_KEY
  getHello(): string {
    return `Hello World! ${this.apiKey}`;
  }
}
// src/app.controller.ts

@Controller()
export class AppController {

  @Get()
  getHello(): string { // 👈 new enpoint
    return this.appService.getHello();
  }

}
```

6.

NODE_ENV=prod npm run start:dev

7. useFactory

```
// src/app.module.ts
import { Module, HttpModule, HttpService } from '@nestjs/common';  // 👈
imports

@Module({
  imports: [HttpModule, UsersModule, ProductsModule],
  controllers: [AppController],
  providers: [
```

```
    imports: [HttpModule, UsersModule, ProductsModule], // 👈 add
HttpModule
    ...,
    {
      provide: 'TASKS',
      useFactory: async (http: HttpService) => { // 👈 implement
useFactory
        const tasks = await http
          .get('https://jsonplaceholder.typicode.com/todos')
          .toPromise();
        return tasks.data;
      },
      inject: [HttpService],
    },
  ],
})
export class AppModule {}
// src/app.service.ts

import { Injectable, Inject } from '@nestjs/common';

@Injectable()
export class AppService {
  constructor(
    @Inject('API_KEY') private apiKey: string,
    @Inject('TASKS') private tasks: any[], // 👈 inject TASKS
  ) {}
  getHello(): string {
    console.log(this.tasks); // 👈 print TASKS
    return `Hello World! ${this.apiKey}`;
  }
}
```

8. Global Module

```
// src/database/database.module.ts

import { Module, Global } from '@nestjs/common';

const API_KEY = '12345634';
const API_KEY_PROD = 'PROD1212121SA';

@Global()
@Module({
  providers: [
    {
      provide: 'API_KEY',
      useValue: process.env.NODE_ENV === 'prod' ? API_KEY_PROD : API_KEY,
    },
  ],
  exports: ['API_KEY'],
})
```

```
export class DatabaseModule {}
// src/app.module.ts
...
import { DatabaseModule } from './database/database.module';

@Module({
  imports: [
    HttpModule,
    UsersModule,
    ProductsModule,
    DatabaseModule // 👈 Use DatabaseModule like global Module
   ],
  ...
})
export class AppModule {}
// src/users/services/users.service.ts

import { Injectable, NotFoundException, Inject } from '@nestjs/common';
..

@Injectable()
export class UsersService {
  constructor(
    private productsService: ProductsService,
    @Inject('API_KEY') private apiKey: string, // 👈 Inject API_KEY
  ) {}

}
```

9. Modulo de Configuracion

```
npm i --save @nestjs/config
// .gitignore
*.env
// .env
DATABASE_NAME=my_db
API_KEY='1234'
// src/app.module.ts
...
import { ConfigModule } from '@nestjs/config';

@Module({
  imports: [
    ConfigModule.forRoot({ // 👈 Implement ConfigModule
      envFilePath: '.env',
      isGlobal: true,
    }),
    ...
  ],
})
export class AppModule {}
// src/users/services/users.service.ts
```

```
import { ConfigService } from '@nestjs/config';
...

@Injectable()
export class UsersService {
  constructor(
    private productsService: ProductsService,
    private configService: ConfigService, // 👈 inject ConfigService
  ) {}
  ...

  findAll() {
    const apiKey = this.configService.get('API_KEY'); // 👈 get API_KEY
    const dbName = this.configService.get('DATABASE_NAME');  // 👈 get
DATABASE_NAME
    console.log(apiKey, dbName);
    return this.users;
  }

  ...
}
```

10. Tipado en configuracion

```
// .env
DATABASE_NAME=my_db_prod
API_KEY=999
DATABASE_PORT=8091 // 👈
// .stag.env
DATABASE_NAME=my_db_stag
API_KEY=333
DATABASE_PORT=8091 // 👈
// .prod.env
DATABASE_NAME=my_db_prod
API_KEY=999
DATABASE_PORT=8091 // 👈
// src/config.ts // 👈 new file
import { registerAs } from '@nestjs/config';

export default registerAs('config', () => { // 👈 export default
  return {
    database: {
      name: process.env.DATABASE_NAME,
      port: process.env.DATABASE_PORT,
    },
    apiKey: process.env.API_KEY,
  };
});
// src/app.module.ts
import config from './config'; // 👈

@Module({
```

```
  imports: [
    ConfigModule.forRoot({
      envFilePath: enviroments[process.env.NODE_ENV] || '.env',
      load: [config], // 👈
      isGlobal: true,
    }),
    ...
  ],
  ...
})
export class AppModule {}
// src/app.service.ts
import { ConfigType } from '@nestjs/config'; // 👈 Import ConfigType
import config from './config'; // 👈 config file

@Injectable()
export class AppService {
  constructor(
    @Inject('TASKS') private tasks: any[],
    @Inject(config.KEY) private configService: ConfigType<typeof config>,
// 👈
  ) {}
  getHello(): string {
    const apiKey = this.configService.apiKey; // 👈
    const name = this.configService.database.name; // 👈
    return `Hello World! ${apiKey} ${name}`;
  }
}
```

11. Validación de esquemas en .envs con Joi

```
npm install --save joi
// src/app.module.ts

import * as Joi from 'joi';  // 👈

@Module({
  imports: [
    ConfigModule.forRoot({
      envFilePath: enviroments[process.env.NODE_ENV] || '.env',
      load: [config],
      isGlobal: true,
      validationSchema: Joi.object({ // 👈
        API_KEY: Joi.number().required(),
        DATABASE_NAME: Joi.string().required(),
        DATABASE_PORT: Joi.number().required(),
      }),
    }),
    ...
  ],
  ...
```

```
})
export class AppModule {}
```

12. Integrando Swagger y PartialType con Open API

```
npm install --save @nestjs/swagger swagger-ui-express
// src/main.ts

import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';

async function bootstrap() {
  ...
  const config = new DocumentBuilder()
    .setTitle('API')
    .setDescription('PLATZI STORE')
    .setVersion('1.0')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('docs', app, document);
  ...
  await app.listen(3000);
}
bootstrap();

# nest-cli.json

{

  "collection": "@nestjs/schematics",

  "sourceRoot": "src",

  "compilerOptions": {

    "plugins": ["@nestjs/swagger/plugin"]

  }

}

// src/products/dtos/brand.dtos.ts
import { PartialType } from '@nestjs/swagger';
// src/products/dtos/category.dtos.ts
import { PartialType } from '@nestjs/swagger';
// src/products/dtos/products.dtos.ts
import { PartialType } from '@nestjs/swagger';
// src/users/dtos/customer.dto.ts
import { PartialType } from '@nestjs/swagger';
// src/users/dtos/user.dto.ts
import { PartialType } from '@nestjs/swagger';
```

13. Extendiendo la Documentacion

```
// src/products/dtos/products.dtos.ts
```

```ts
import { PartialType, ApiProperty } from '@nestjs/swagger';

import {
  IsString,
  IsNumber,
  IsUrl,
  IsNotEmpty,
  IsPositive,
} from 'class-validator';
import { PartialType, ApiProperty } from '@nestjs/swagger'; // 👈

export class CreateProductDto {
  @IsString()
  @IsNotEmpty()
  @ApiProperty({ description: `product's name` }) // 👈
  readonly name: string;

  @IsString()
  @IsNotEmpty()
  @ApiProperty() // 👈
  readonly description: string;

  @IsNumber()
  @IsNotEmpty()
  @IsPositive()
  @ApiProperty() // 👈
  readonly price: number;

  @IsNumber()
  @IsNotEmpty()
  @ApiProperty() // 👈
  readonly stock: number;

  @IsUrl()
  @IsNotEmpty()
  @ApiProperty() // 👈
  readonly image: string;
}

export class UpdateProductDto extends PartialType(CreateProductDto) {}
// src/products/controllers/products.controller.ts
import { ApiTags, ApiOperation } from '@nestjs/swagger'; // 👈

@ApiTags('products') // 👈
@Controller('products')
export class ProductsController {
  constructor(private productsService: ProductsService) {}

  @Get()
  @ApiOperation({ summary: 'List of products' }) // 👈
  getProducts(
    @Query('limit') limit = 100,
```

```
    @Query('offset') offset = 0,
    @Query('brand') brand: string,
  ) {
    // return {
    //   message: `products limit=> ${limit} offset=> ${offset} brand=>
${brand}`,
    // };
    return this.productsService.findAll();
  }
}
// src/products/controllers/brands.controller.ts
import { ApiTags } from '@nestjs/swagger';


@ApiTags('brands') // 👈
@Controller('brands')
export class BrandsController {
  ...
}
```

14. Deploy de NestJS en Heroku

```
heroku local web
git checkout master
git merge 14-step
git remote -v
git push heroku master
heroku logs --tail
```

También puedes usar el comando `heroku config:set APP_KEY=12345` para configurar variables de ambiente desde el CLI y esto hace que la app se reinicie sin la necesidad de enviar un push.
Notas:

- Cuidado con los typos 🙂
- No dejes comments en producción 📣

15. Fdfd

16. Comandos solucionar

```
nest update -f -t latest
```