

Taller: Desarrollo de una API RESTful con NestJS, TypeORM y Buenas Prácticas

Objetivo

Desarrollar una API RESTful en NestJS que implemente:

- ☒ Buenas prácticas en la estructura del proyecto
- ☒ CRUD completo (Create, Read, Update, Delete)
- ☒ Validaciones con class-validator y class-transformer
- ☒ Manejo de errores con HttpException y HttpStatus
- ☒ Conexión a base de datos con TypeORM y al menos 4 entidades relacionadas
- ☒ Documentación de las rutas con Postman o Insomnia
- ☒ Código subido y organizado en un repositorio de GitHub

Descripción del trabajo

El tema del proyecto será libre: pueden elegir el dominio (por ejemplo: gestión de libros, inventarios, usuarios, pedidos, etc.). La API debe incluir mínimo 4 entidades relacionadas con claves foráneas (por ejemplo: Usuario, Pedido, Producto, Categoría). Implementar un CRUD completo para al menos 3 de las entidades. Las relaciones pueden ser:

- OneToMany,
- ManyToOne
- ManyToMany

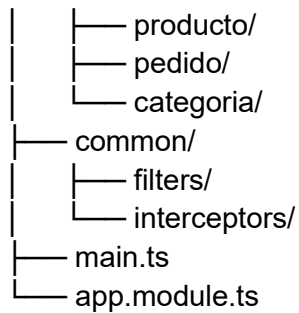
.El backend debe seguir la arquitectura modular de NestJS. Usar DTOs y servicios para separar lógica y validaciones. Subir el código a un repositorio público o privado en GitHub con un README.md que explique cómo correr el proyecto.

Requisitos técnicos

- 1 NestJS con TypeScript (última versión estable)
- 2 TypeORM como ORM + base de datos (MySQL o PostgreSQL)
- 3 Uso de DTOs para entrada de datos + validaciones con class-validator
- 4 Transformación de datos con class-transformer cuando corresponda
- 5 Respuestas y errores gestionados con HttpException y HttpStatus
- 6 Implementar pruebas de las rutas en Postman o Insomnia y exportar la colección

Recomendaciones de estructura

```
src/  
├── modules/  
│   └── usuario/
```



- Usar ConfigModule y variables de entorno (.env) para la conexión a la base de datos.
- Definir las entidades y relaciones en la carpeta de cada módulo.
- Separar DTOs en archivos: create-*.dto.ts, update-*.dto.ts.

Entregables

- ☒ Código completo en GitHub con un README.md que contenga:
 - Requisitos
 - Pasos para levantar el servidor
- ☒ Colección de Postman o Insomnia:
 - Exportada y agregada al repositorio o compartida mediante link.
 - Debe incluir al menos las rutas de CRUD implementadas y funcionales

Criterios de evaluación

Criterio	Puntos
Estructura del proyecto y uso de módulos	20
Uso correcto de DTOs y validaciones	20
Relaciones entre entidades bien implementadas	20
Manejo de errores adecuado con HttpException y HttpStatus	15
Código subido y documentado en GitHub	15
Colección Postman o Insomnia completa	10