como construir una api restfull en nestjs

En **NestJS**, los **middlewares** son funciones que se ejecutan **antes** de que el controlador reciba la petición. Son ideales para:

* **Logs** (registrar peticiones).
* **Autenticación básica**.
* **Validaciones previas**.
* **Modificación del request o response**.
* **Control de acceso**.

NestJS implementa **middleware** de forma muy similar a Express.

Aquí tienes una **guía completa** paso a paso. 🚀

**1. Crear un middleware**

La forma más común es crear un **middleware de clase** que implemente NestMiddleware.

nest g middleware logger

Esto genera algo como:

// src/logger/logger.middleware.ts

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

import { Request, Response, NextFunction } from 'express';

@Injectable()

export class LoggerMiddleware implements NestMiddleware {

use(req: Request, res: Response, next: NextFunction) {

console.log(`[${req.method}] ${req.originalUrl} - ${new Date().toISOString()}`);

next(); // ¡IMPORTANTE! Continua con la siguiente capa

}

}

🔹 **Explicación:**

* req → datos de la petición.
* res → datos de la respuesta.
* next() → pasa al siguiente middleware o al controlador.
* Si **no llamas a next()**, la petición se queda atascada.

**2. Aplicar el middleware en un módulo**

Los middlewares se configuran dentro de un **módulo**, normalmente en app.module.ts o en un módulo específico.

// src/app.module.ts

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

import { LoggerMiddleware } from './logger/logger.middleware';

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

@Module({

imports: [ProductsModule],

})

export class AppModule {

configure(consumer: MiddlewareConsumer) {

consumer

.apply(LoggerMiddleware) // middleware que quieres usar

.forRoutes('\*'); // aplica a TODAS las rutas

}

}

**3. Aplicar a rutas específicas**

En lugar de aplicarlo a toda la app, puedes limitarlo a rutas específicas.

consumer

.apply(LoggerMiddleware)

.forRoutes('products'); // solo para rutas que empiezan con /products

Incluso puedes filtrar por métodos HTTP:

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

consumer

.apply(LoggerMiddleware)

.forRoutes(

{ path: 'products', method: RequestMethod.GET },

{ path: 'products/:id', method: RequestMethod.DELETE },

);

**4. Múltiples middlewares**

Puedes encadenar varios middlewares.

consumer

.apply(LoggerMiddleware, AuthMiddleware, AnotherMiddleware)

.forRoutes('products');

**Orden importante:** se ejecutan en el orden en que los declares.

**5. Middleware funcional (opcional)**

Si necesitas algo simple, puedes crear un middleware como **función** sin clases:

// src/middlewares/simple-logger.middleware.ts

import { Request, Response, NextFunction } from 'express';

export function SimpleLogger(req: Request, res: Response, next: NextFunction) {

console.log(`[Simple Logger] ${req.method} ${req.url}`);

next();

}

Usarlo en el módulo:

consumer

.apply(SimpleLogger)

.forRoutes('products');

**6. Ejemplo con autenticación básica**

Middleware para validar un **token** en el header:

// src/middlewares/auth.middleware.ts

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

import { Request, Response, NextFunction } from 'express';

@Injectable()

export class AuthMiddleware implements NestMiddleware {

use(req: Request, res: Response, next: NextFunction) {

const token = req.headers['authorization'];

if (!token || token !== 'Bearer 12345') {

throw new UnauthorizedException('Invalid or missing token');

}

console.log('Auth OK ✅');

next();

}

}

Aplicarlo solo a ciertas rutas:

consumer

.apply(AuthMiddleware)

.forRoutes('products');

Ahora **todas las peticiones a /products deben tener este header**:

Authorization: Bearer 12345

**7. Estructura final del proyecto**

src/

├── logger/

│ └── logger.middleware.ts

├── middlewares/

│ ├── simple-logger.middleware.ts

│ └── auth.middleware.ts

├── products/

│ ├── products.controller.ts

│ ├── products.module.ts

│ └── products.service.ts

├── app.module.ts

└── main.ts

**8. Resumen rápido**

| **Acción** | **Código** |
| --- | --- |
| Crear middleware | nest g middleware logger |
| Aplicar a toda la app | .forRoutes('\*') |
| Aplicar a rutas específicas | .forRoutes('products') |
| Filtrar por método HTTP | .forRoutes({ path: 'products', method: RequestMethod.GET }) |
| Middleware en cadena | .apply(M1, M2, M3) |

**9. Buenas prácticas**

* Usa middlewares para **procesamiento previo** (logging, auth, parseo).
* Para **validaciones**, mejor usa **Pipes**.
* Para **manejo de errores**, usa **Exception Filters**.
* Centraliza middlewares en una carpeta middlewares/ o common/middleware/.
* Evita lógica de negocio dentro de un middleware.

¿Quieres que te prepare un ejemplo **real** con JWT y un middleware de autenticación completo? 🔐