NestJS
Página de Atajos

Para iniciar el proyecto:

npm run start:dev

Instalar Nest.js CLI: Command line interface

```
npm i -g @nestjs/cli
```

Nuevo proyecto: en el path actual

```
nest new project-name
```

Comandos útiles del CLI

```
nest generate <comando>
nest g <comando>
```

Mostrar ayuda: en cualquier comando

```
nest -h
nest g -h
nest g s nombre -h
```

Componentes comunes: Añadir -h para extras

```
# Crear una clase
nest g cl <path/nombre>

# Crear un controlador
nest g co <path/nombre>

# Crear un decorador
nest g d <path/nombre>

# Crear un guard
nest g gu <path/nombre>

# Crear un interceptor
nest g in <path/nombre>

# Crear un módulo
nest g mo <path/nombre>

# Crear un pipe
nest g pi <path/nombre>

# Crear un servicio
nest g s <path/nombre>

# Crear un recurso completo
nest g resource <nombre>
```

Pipes integrados por defecto

ValidationPipe	ParseIntPipe
ParseBoolPipe	ParseArrayPipe
ParseFloatPipe	ParseUUIDPipe

Banderas adicionales útiles

```
# Confirmar qué hará el comando
nest g s nombre --dry-run | -d

# No archivo de pruebas automático
nest g s nombre —no-spec
```

Métodos HTTP comunes

```
Import {
Get, Post, Put, Path, Delete
} from '@nestjs/common';
```

Argumentos de Métodos HTTP:

Aplica a cualquier método http

```
# Default Get
@Get()

# Con segmento dinámico
@Get(':id')

# Especificando una ruta
@Get('cats/breed')
@Get(['cats','breed'])

# Paths dinámicos
@Get(':product/:size')
```

Extraer información de la solicitud (**request**)

```
# Obtener parámetros / segmentos
@Param('id')

# Obtener el body de la petición
@Body()

# Obtener los parámetros de query
@Query()

# Obtener respose (Express/Fastify)
# Importarse desde express/fastify
@Res()
```

Convertir **:id** del segmento a entero

```
@Get(':id')
async findOne(
  @Param('id', ParseIntPipe) id: number
){
  return this.catsService.findOne(id);
}
```

NestJS
Página de Atajos

Librerías externas útiles:

```
yarn add class-validator class-transformer
```

Algunos decoradores de Class Validator

IsOptional	IsPositive	IsMongoId
IsArray	IsString	IsUUID
IsDecimal	IsDate	IsDateString
IsBoolean	IsEmail	IsUrl

Configuración global de pipes

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

whiteList: Remueve todo lo que no está incluído en los DTOs
forbidNonWhiteListed: Retorna bad request si hay propiedades en el objeto no requeridas

Estructura de módulo recomendado:

```
src
- common
—— decorators
—— dtos
—— filters
—— guards
—— interceptors
—— middleware
—— pipes
— common.controller.ts
— common.module.ts
— common.service.ts
```

Configuraciones globales:

Que no requieren el “execution context”

```
const app = await NestFactory.create(AppModule);

app.useGlobalFilters( new Filtro1, … );
app.useGlobalGuards( new Guard1, … );
app.useGlobalInterceptors( new Inter1, … );
app.useGlobalPipes( new Pipe1, … );
```

Building Blocks:

Guards:
Usados para permitir o prevenir acceso a una ruta.
Ej: Aquí es donde se debe de autorizar una solicitud.

Interceptors:
Before Interceptor: interceptan la solicitud (request) y la pueden transformar completamente basado en las necesidades.
Ej: caché o logs

Pipes:
Transformar la data recibida en requests, para asegurar un tipo, valor o instancia de un objeto.
Ej: Transformar a números, validaciones, etc.

Controllers:
Controlan rutas, son los encargados de escuchar la solicitud y emitir una respuesta.
Ej: Rutas CRUD

Decoradores: Estos se puede aplicar a cualquier nivel
Expanden la funcionalidad de el método, propiedad o clase a la cual se adjuntan. Nest.js busca aplicar el principio DRY fuertemente con decoradores.
Ej: @Controller(‘usuarios’), @Ip(), @CustomDecorator()

Services:
Alojan la lógica de negocio de tal manera que sea reutilizable mediante inyección de dependencias.
Ej: PeliculasService para todo lo relacionado a obtener, grabar, actualizar o eliminar información de películas.

Interceptors:
After Interceptor: Intercepta la respuesta que emitirá el controlador y la puede transformar completamente basado en las necesidades.
Ej: Estandarizar nuevas necesidades, añadir información adicional o almacenar en caché la respuesta.

Exception Filters:
Maneja los errores de código en mensajes de respuesta http. Usualmente Nest ya incluye todos los casos de uso comunes, pero se pueden expandir basado en las necesidades.

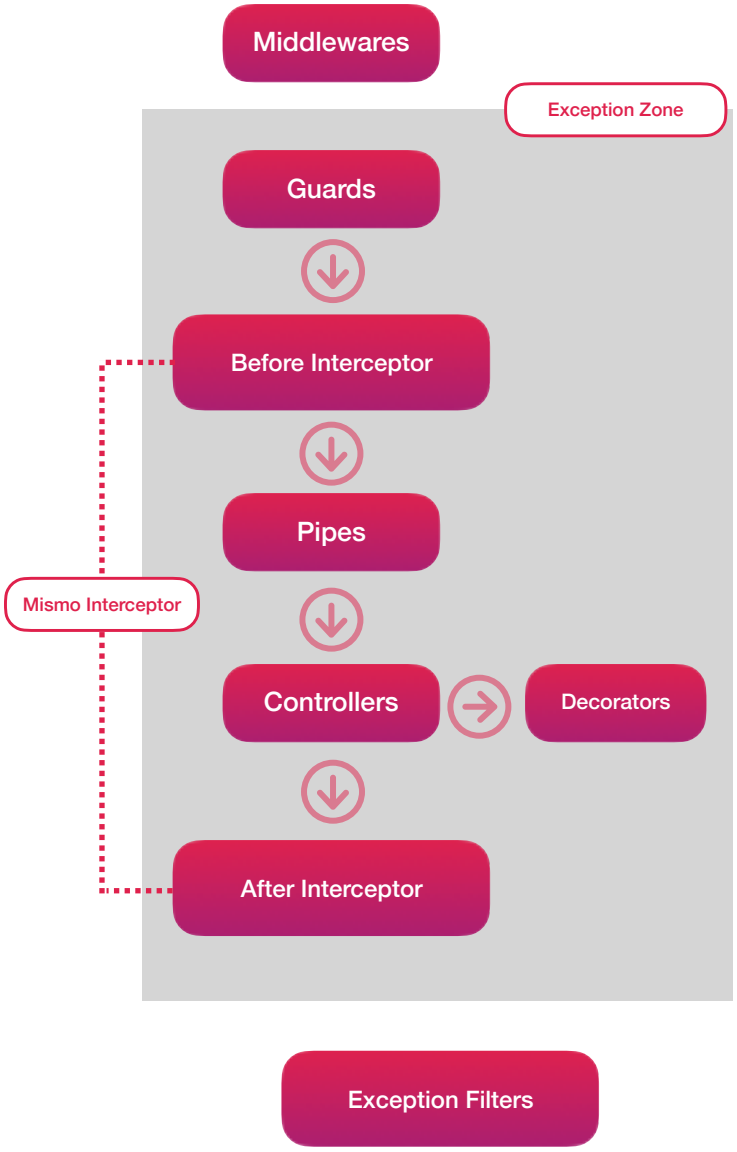
BadRequestException	UnauthorizedException
NotFoundException	ForbiddenException
RequestTimeoutException	GoneException
PayloadTooLargeException	InternalServerErrorException

NestJS
Página de Atajos

Module:
Agrupan y desacoplan un conjunto de funcionalidad específica por dominio.
Ej: auth.module.ts, encargado de todo lo relacionado a la autenticación

Middlewares:
Modifican o mejoran los objetos de solicitudes o respuestas (request / response). Pueden terminar el ciclo de ejecución, pero no tienen acceso al resultado de la ruta.

Ciclo de vida
De forma general, estos son los pasos tradicionales, pero los decoradores pueden ir en cada etapa.



Habilitar CORS:

```
const app = await NestFactory.create(AppModule);

app.enableCors();
app.enableCors( options );

@WebSocketGateway({ cors: true }); // Socket
```

Cookies: Hay que habilitarlas primero con su definición de TS

```
yarn add cookie-parser
yarn add -D @types/cookie-parser

npm i cookie-parser
npm i -D @types/cookie-parser
```

```
import * as cookieParser from 'cookie-parser';
…
…
const app = await NestFactory.create(AppModule);
app.use( cookieParser() );
```

Variables de entorno: .env files
Crear en el root del proyecto el archivo **.env**

```
yarn add @nestjs/config

npm i @nestjs/config
```

app.module.ts

```
import { Module } from '@nestjs/common';
import { ConfigModule } from '@nestjs/config';

@Module({
  imports: [ConfigModule.forRoot()],
})
export class AppModule {}
```

Uso de variables .env: inyectar esto

```
constructor(
private readonly configService:ConfigService
){}
```

Servir contenido estático: crear directorio public

```
yarn add @nestjs/serve-static
```

```
@Module({
  imports: [
    ServeStaticModule.forRoot({
      rootPath: join(__dirname,'..','public'),
    })
  ],
})
export class AppModule {}
```