Pregunta:

Tengo un problema de que me localice los ficheros en nest ya que tengo que escribir las rutas relativas completas(../../mi-path/mi-fichero), en vez de poder hacerlo con @/mi-path-mi-fichero.

¿Cómo puedo configurar nest para obtener el comportamiento deseado?

Respuesta:

¡Hola! Entiendo perfectamente tu frustración. Tener que lidiar con rutas relativas largas como ../../../ es propenso a errores y hace que el código sea más difícil de leer y mantener.

La buena noticia es que lo que quieres hacer es una práctica estándar en el ecosistema de TypeScript/Node.js y es muy sencillo de configurar en un proyecto de NestJS. La solución consiste en configurar los **"Path Aliases"** o alias de ruta en tu fichero de configuración de TypeScript (tsconfig.json).

Esto le dice al compilador de TypeScript (y a tu editor de código como VS Code) cómo resolver esas rutas "mágicas" que empiezan con @/.

Sigue estos dos sencillos pasos:

### Paso 1: Modificar el fichero tsconfig.json

Esta es la parte más importante. Necesitas indicarle a TypeScript dos cosas: una ruta base (baseUrl) y las asignaciones de rutas específicas (paths).

1. Abre el fichero tsconfig.json que se encuentra en la raíz de tu proyecto.
2. Dentro del objeto compilerOptions, añade o modifica las siguientes dos propiedades: baseUrl y paths.

JSON

{  
 "compilerOptions": {  
 // ... otras opciones existentes como "module", "target", etc.  
 "baseUrl": "./", // ¡Esta línea es fundamental!  
 "paths": {  
 "@/\*": ["src/\*"] // Aquí definimos nuestro alias  
 }  
 // ... resto de opciones  
 }  
}

#### ¿Qué significan estas opciones?

* "baseUrl": "./": Esta opción le dice a TypeScript que las rutas base para la resolución de módulos deben comenzar desde el directorio raíz del proyecto (donde se encuentra el tsconfig.json). Es un requisito indispensable para que paths funcione correctamente.
* "paths": { "@/\*": ["src/\*"] }: Aquí es donde ocurre la magia.
  + "@/\*": Define el patrón del alias. Cualquier importación que comience con @/ será interceptada.
  + ["src/\*"]: Define a qué ruta real se debe mapear. El \* en ambos lados actúa como un comodín (wildcard). Esto significa que una importación como @/services/user.service se resolverá como si hubieras escrito src/services/user.service.

**Una vez guardes el fichero, el compilador de NestJS ya sabrá cómo transformar las rutas al compilar tu código de TypeScript a JavaScript.**

#### Ejemplo de uso:

**Antes (con rutas relativas):**

TypeScript

// En src/controllers/user/user.controller.ts  
import { UserService } from '../../services/user.service';  
import { UserEntity } from '../../../database/entities/user.entity';

**Después (con alias de ruta):**

TypeScript

// En src/controllers/user/user.controller.ts  
import { UserService } from '@/services/user.service';  
import { UserEntity } from '@/database/entities/user.entity';

¡Mucho más limpio y robusto si mueves ficheros de sitio!

### Paso 2: Actualizar tu editor de código (VS Code)

Tu editor de código necesita leer esta nueva configuración para poder ofrecerte autocompletado y no marcar las nuevas rutas como errores.

En la mayoría de los casos, especialmente con VS Code, solo necesitas **recargar el editor**.

* Abre la paleta de comandos: Ctrl+Shift+P (o Cmd+Shift+P en Mac).
* Busca y selecciona la opción **"Developer: Reload Window"** (Desarrollador: Recargar ventana).

¡Y ya está! Ahora tu editor debería reconocer los nuevos alias, proporcionarte autocompletado y permitirte navegar a los ficheros con Ctrl+Click (o Cmd+Click).

### Consideración Adicional: Configuración para Jest

Si usas Jest para tus pruebas, te darás cuenta de que Jest no lee la configuración de paths del tsconfig.json por defecto, por lo que tus pruebas fallarán al no poder resolver los módulos.

Para solucionarlo, necesitas replicar la configuración en tu fichero de configuración de Jest (jest.config.js o en la sección "jest" de tu package.json).

1. Abre tu fichero de configuración de Jest.
2. Añade la propiedad moduleNameMapper.

**Ejemplo en jest.config.js:**

JavaScript

module.exports = {  
 // ... otras opciones de Jest  
 moduleNameMapper: {  
 '^@/(.\*)$': '<rootDir>/src/$1',  
 },  
 // ... resto de opciones  
};

Esta configuración le dice a Jest que cualquier importación que coincida con el patrón ^@/(.\*)$ (expresión regular para @/lo-que-sea) debe ser mapeada a <rootDir>/src/lo-que-sea, donde <rootDir> es la carpeta raíz del proyecto.

Con estos pasos, tendrás un sistema de rutas absolutas limpio y funcional tanto para tu aplicación principal como para tus tests en NestJS.