NEST 01

/src

app.module.ts

- ¿Qué son los módulos? Conocidos como building blocks de NEST
  - Agrupan y desacoplan un conjunto de funcionalidad específica por dominio
  - Por ejemplo para la autenticación, voy a tener un módulo autenticación
- El módulo principal, app.module.ts va a tener referencias a todo lo que es mi aplicación.
  - Otros módulos, servicios, dependencias, controladores, etc
- Al fin y al cabo no es más que una clase con el decorador @Module

```
@Module({
  imports: [],
  controllers: [],
  providers: [],
  exports:[]
})
export class AppModule {}
```

main.ts

- Puedo cambiar el nombre de la función bootstrap por main
- Ejecuta la primera instrucción
  - NestFactory.create()
- Luego ejecuta la escucha en el puerto 3000

Controladores

- El decorador convierte la clase en un objeto con cierta funcionalidad específica
- La diferencia entre servicios, controladores, y otros radica en el decorador, pero siguen siendo clases.
- Los **controladores** son los encargados de escuchar la solicitud y emitir una respuesta
- Para crear un controlador usaré el CLI de NEST

```
nest g co <path/nombre>
```

- Para ver todos los comandos

```
nest --help
```

- En NEST hay una estructura de módulos recomendada, donde common es el nombre de mi módulo

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

- Para crear mi modulo en la raíz llamado cars

> nest g mo cars

- Esto me crea la carpeta con un archivo con la clase y su decorador

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

@Module({})
export class CarsModule {}
```

- Si ahora voy al app.module veo que lo ha importado automáticamente

```
import { Module } from '@nestjs/common';
import { CarsModule } from './cars/cars.module';


@Module({
  imports: [CarsModule],
  controllers: [],
  providers: [],
  exports:[]
})
export class AppModule {}
```

- El módulo como tal no hace nada, esta vacío. Creo mi controlador

> nest g co cars

- Incluye el controlador automáticamente en cars + archivo de testing, ya que la inteligencia va a buscar un modulo con ese nombre
- Si ahora voy al cars.module veo que lo ha importado directamente

```
import { Module } from '@nestjs/common';
import { CarsController } from './cars.controller';

@Module({
  controllers: [CarsController]
})
export class CarsModule {}
```

- El controlador (cars.controller.ts) es una simple clase con el decorador @Controller

```
import { Controller } from '@nestjs/common';

@Controller('cars')
export class CarsController {}
```

- Este nombre de 'cars' como parámetro del @Controller significa que responde a la petición del endpoint cars

> http://localhost:3000/cars

- Ahora devuelve un 404 porque la clase de CarsController no contiene nada
- Con añadir sólo el método no es suficiente.
    - Le añado el decorador @Get.
- Eso le dice a NEST que responde así a la petición Get

```
import { Controller, Get } from '@nestjs/common';

@Controller('cars')
export class CarsController {
    @Get()
    getAllCars(){
        return['Toyota', 'Honda', 'Jeep']
    }
}
```

- Notar que el **return** es un arreglo y también que se ha importado Get