NEST 10

Variables de entorno

- Usualmente las variables de entorno se escriben en un archivo llamado .env en la raíz del proyecto
- Este archivo se añade a .gitignore para que se obvien en github
- Este es el formato:

```
MONGODB=mongodb://localhost:27017/nest-pokemon
```

- tengo que decirle a nest que lea este archivo para poder ejecutarlas
- Para ello se requiere la instalación de un paquete

```
npm i @nestjs/config
```

- Ahora hay que importar el ConfigModule.forRoot en el app.module.ts. Lo coloco al inicio, antes de la variable de entorno para que no de undefined y se pueda conectar

```
import { Module } from '@nestjs/common';
import { join } from 'path';
import {ServeStaticModule} from '@nestjs/serve-static'
import { PokemonModule } from './pokemon/pokemon.module';
import {MongooseModule} from '@nestjs/mongoose'
import { CommonModule } from './common/common.module';
import { SeedModule } from './seed/seed.module';
import {ConfigModule} from '@nestjs/config'

@Module({
  imports: [
    ConfigModule.forRoot(),

    ServeStaticModule.forRoot({
      rootPath: join(__dirname, '..', 'public'),
    }),

    MongooseModule.forRoot(process.env.MONGODB),

    PokemonModule,

    CommonModule,

    SeedModule
  ],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

ConfigurationLoader

- El ConfigModule también ofrece un servicio que me va a permitir hacer inyección de dependencias de las variables de entorno
- En la carpeta /src creo una nueva carpeta llamada /config, y dentro un nuevo archivo llamado env.config.ts
- Lo que quiero es mapear mis variables de entorno a un objeto

```
export const EnvConfiguration =()=>({
    environment: process.env.NODE_ENV || 'dev',
    mongodb: process.env.MONGODB,
    port: process.env.PORT,
    defaultLimit: process.env.DEFAULT_LIMIT || 7
})
```

- De alguna manera tengo que decirle a NEST y al módulo de configuración de las variables de entorno que use este archivo
- Para eso voy a app.module

```
@Module({
  imports: [
    ConfigModule.forRoot({
      load: [EnvConfiguration]
    }),
    ServeStaticModule.forRoot({
      rootPath: join(__dirname, '..', 'public'),
    }),

    MongooseModule.forRoot(process.env.MONGODB),

    PokemonModule,

    CommonModule,

    SeedModule
  ],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

- Cuando quiera hacer inyección de dependencias de variables de entorno no se hará desde el constructor con process.env si no desde ConfigModule

ConfigService

- Uso el ConfigService en el constructor de PokemonService

- pokemon.service:

```
@Injectable()
export class PokemonService {

  constructor(
    @InjectModel(Pokemon.name)
    private readonly pokemonModel: Model<Pokemon>,

    private readonly configService: ConfigService
  ){

  }
```

- Esto da un error, no encuentra la segunda dependencia.
- Hay que importar en el PokemonModule, el ConfigModule

```
import { Module } from '@nestjs/common';
import { PokemonService } from './pokemon.service';
import { PokemonController } from './pokemon.controller';
import { MongooseModule } from '@nestjs/mongoose';
import { Pokemon, PokemonSchema } from './entities/pokemon.entity';
import { ConfigModule } from '@nestjs/config';


@Module({
  controllers: [PokemonController],
  providers: [PokemonService],
  imports:[
    ConfigModule,
    MongooseModule.forFeature([
      {
        name: Pokemon.name,
        schema: PokemonSchema
      }
    ])
  ],
  exports: [MongooseModule]
})
export class PokemonModule {}
```

- Para usar la variable de entorno DEFAULT_LIMIT uso el configService. Pruebo con un console.log

```
@Injectable()
export class PokemonService {

  constructor(
```

```
    @InjectModel(Pokemon.name)
    private readonly pokemonModel: Model<Pokemon>,

    private readonly configService: ConfigService,
  ){
    console.log( configService.get('defaultLimit'))
  }
```

- Esto imprime el valor en consola (como un numero)
- Si pongo el cursor encima del get, veo que es de tipo any. Yo puedo especificarle que sea un numero

```
@Injectable()
export class PokemonService {

  constructor(
    @InjectModel(Pokemon.name)
    private readonly pokemonModel: Model<Pokemon>,

    private readonly configService: ConfigService,
  ){
    const defaultLimit=  configService.get<number>('defaultLimit')
    console.log({defaultLimit})
  }
```

- Puedo crear el defaultLimit como una propiedad de la clase. Es una dependencia

```
import { BadRequestException, Injectable, InternalServerErrorException,
NotFoundException } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
import { InjectModel } from '@nestjs/mongoose';
import { isValidObjectId, Model } from 'mongoose';
import { PaginationDto } from 'src/common/dto/pagination.dto';
import { CreatePokemonDto } from './dto/create-pokemon.dto';
import { UpdatePokemonDto } from './dto/update-pokemon.dto';
import {Pokemon} from './entities/pokemon.entity'

@Injectable()
export class PokemonService {

  private defaultLimit: number

  constructor(
    @InjectModel(Pokemon.name)
    private readonly pokemonModel: Model<Pokemon>,

    private readonly configService: ConfigService,
  ){

    this.defaultLimit=  configService.get<number>('defaultLimit') //
```

```
  'defaultLimit' asi lo mapeé en env.config.ts

    }


    async create(createPokemonDto: CreatePokemonDto) {
      createPokemonDto.name = createPokemonDto.name.toLocaleLowerCase()

     try {
       const pokemon= await this.pokemonModel.create(createPokemonDto)
      return pokemon;

    } catch (error) {
        this.handleException(error)
     }
    }

    findAll(paginationDto: PaginationDto) {

      const {limit=this.defaultLimit, offset=0}= paginationDto //llamo aqui la
  variable de entorno

      return this.pokemonModel.find()
        .skip(offset)
        .limit(limit)
        .sort({
          no:1
        })
        .select('-__v')
    }

    async findOne(term: string) {

      let pokemon: Pokemon
      if( !isNaN(+term)){
        pokemon = await this.pokemonModel.findOne({ no: term })
      }


      if(!pokemon && isValidObjectId(term)){
        pokemon= await this.pokemonModel.findById(term)
      }

      if( !pokemon){
        pokemon = await this.pokemonModel.findOne({name: term.toLowerCase().trim()})
      }
      if(!pokemon) throw new NotFoundException(`Pokemon with id ${term} not exists`)

      return pokemon

    }

    async update(term: string, updatePokemonDto: UpdatePokemonDto) {
```

```
    const pokemon= await this.findOne(term)

    if(updatePokemonDto.name)
      updatePokemonDto.name = updatePokemonDto.name.toLowerCase()

      try {
        await pokemon.updateOne(updatePokemonDto)

        return { ...pokemon.toJSON(), ...updatePokemonDto}

      } catch (error) {
        this.handleException(error)
      }


  }

  async remove(id: string) {

    const {deletedCount} = await this.pokemonModel.deleteOne({_id: id})

    if(deletedCount===0)
      throw new BadRequestException(`Pokemon with id ${id} not found`)

    return
  }

  private handleException(error: any){
    if(error.code === 11000){
      throw new BadRequestException(`Pokemon exists in DB
${JSON.stringify(error.keyValue)}`)
    }
    console.log(error)
    throw new InternalServerErrorException("Can't create Pokemon. Check server
logs")
   }
}
```

- Ahora está todo bien. Con el mapeo de env.config.ts es más que suficiente en la mayoría de apps
- Si la conexión con la variable de entorno a la DB fallara, no tengo un buen manejo de error
- Entonces habría que colocar algunas reglas en las variables de entorno
- Otro punto importante es que en el main no puedo hacer inyección de dependencias, porque esta fuera de los building blocks, para el PORT pero bien lo puedo poner con process.env.PORT ya que está de manera global

```
import { ValidationPipe } from '@nestjs/common';
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function main() {
  const app = await NestFactory.create(AppModule);
```

```
  app.setGlobalPrefix('api/v2')

  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      forbidNonWhitelisted: true,
      transform: true,
      transformOptions:{
        enableImplicitConversion: true
      }
    })
  );

  await app.listen(process.env.PORT);
}
main();
```

Joi ValidationSchema

- Un paquete para validar los objetos. Me va aservir para validar, lanzar errores, revisar que un objeto luzca de la manera esperada
- En este caso las variables de entorno

> npm i joi

- En la carpeta de config creo el archivo joi.validation.ts
- Lo importo con * porque si no no funciona

```
import * as Joi from 'joi'

export const JoiValidationSchema = Joi.object({
    MONGODB: Joi.required(),
    PORT: Joi.number().default(3000),
    DEFAULT_LIMIT:Joi.number().default(10)

})
```

- En app.module puedo decirle que use el schema

```
import { Module } from '@nestjs/common';
import { join } from 'path';
import {ServeStaticModule} from '@nestjs/serve-static'
import { PokemonModule } from './pokemon/pokemon.module';
import {MongooseModule} from '@nestjs/mongoose'
import { CommonModule } from './common/common.module';
import { SeedModule } from './seed/seed.module';
import {ConfigModule} from '@nestjs/config'
```

```
import { EnvConfiguration } from './config/env.config';
import { JoiValidationSchema } from './config/joi.validation';

@Module({
  imports: [
    ConfigModule.forRoot({
      load: [EnvConfiguration],
      validationSchema: JoiValidationSchema,
    }),
    ServeStaticModule.forRoot({
      rootPath: join(__dirname, '..', 'public'),
    }),

    MongooseModule.forRoot(process.env.MONGODB),

    PokemonModule,

    CommonModule,

    SeedModule
  ],

})
export class AppModule {}
```

- Esto hace que por defecto sean 6, porque cuando llega a las variables de entorno ya está trabajado pero lo pasa a STRING. Las variables de entorno siempre son strings
- Debo cambiarlo en el mapeo de las variables de entorno añadiéndole un +

```
export const EnvConfiguration =()=>({
    environment: process.env.NODE_ENV || 'dev',
    mongodb: process.env.MONGODB,
    port: process.env.PORT,
    defaultLimit: +process.env.DEFAULT_LIMIT || 7
})
```