**Proyecto base para el resto de proyectos**

* Crear proyecto

nest new base-project-vx

* Configuro el .gitignore

#files

\*.doc\*

\*.tmp\*

* Comento esto en .eslintrc.js

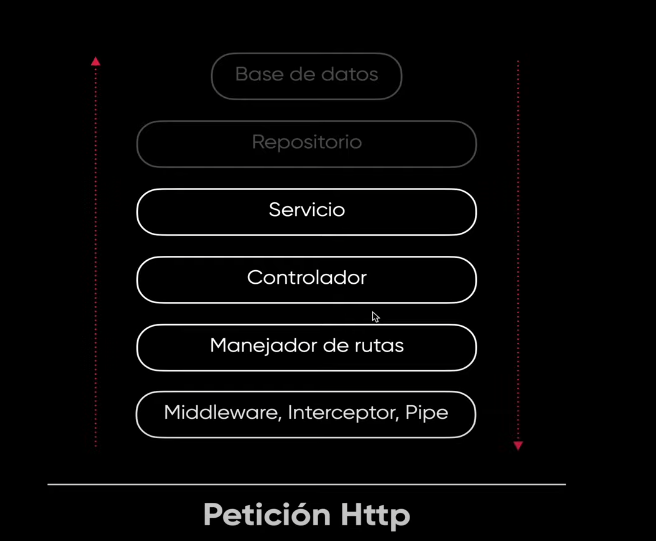
  extends: [

    'plugin:@typescript-eslint/recommended',

    // 'plugin:prettier/recommended',

  ],

* Flujo de una app



* Variables de entorno en desarrollo -> .env

# Server settings

SERVER\_PORT=3000

SERVER\_HOST=localhost

SERVER\_VERSION=v1

SECRET\_KEY=mysecretpassword

#environment

NODE\_ENV=development

# Postgres config

DB\_DIALECT=postgres

DB\_HOST=localhost

DB\_PORT=5432

DB\_USER=postgres

DB\_NAME=base-db

DB\_PASSWORD=admin

DB\_LOGGING=false

AUTO\_LOAD\_ENTITIES = true

* Creo enum de entornos para usarlo más adelante llegado el caso

src\common\enum\environment.enum.ts

export *enum* Env {

  Production = 'production',

  Development = 'development',

}

* Creo enums de constantes para las variables de entorno

src\common\enum\key.enum.ts

export *enum* Key {

  SERVER\_PORT = 'SERVER\_PORT',

  SERVER\_HOST = 'SERVER\_HOST',

  SERVER\_VERSION = 'SERVER\_VERSION',

  NODE\_ENV = 'NODE\_ENV',

}

* Enums para db

src\common\enum\db.enum.ts

export *enum* Db {

  DB\_DIALECT = 'DB\_DIALECT',

  DB\_HOST = 'DB\_HOST',

  DB\_PORT = 'DB\_PORT',

  DB\_USER = 'DB\_USER',

  DB\_PASSWORD = 'DB\_PASSWORD',

  DB\_NAME = 'DB\_NAME',

  AUTO\_LOAD\_ENTITIES = 'AUTO\_LOAD\_ENTITIES',

  DB\_LOGGING = 'DB\_LOGGING',

}

Nota: Para cambiar la ruta del archivo si se desea

ConfigModule.forRoot({

envFilePath: '.development.env',

});

* Exporto los enums que vaya creando con -> src\common\enum\index.ts

export { Db } from './db.enum';

export { Env } from './env.enum';

export { Key } from './key.enum';

export { RoleType } from './role.enum';

* Exporto roletype como enum

src\common\enum\role.enum.ts

export *enum* RoleType {

  SUDO="SUDO",

  ADMIN = 'ADMIN',

  MANAGER = 'MANAGER',

  USER = 'USER',

}

**MODULO DE CONFIGURACIÓN**

<https://docs.nestjs.com/techniques/configuration>

* Instalar librería

npm i @nestjs/config

* Elimino app.controller y el app.service
* src\app.module.ts
  + Limpio el app.controller y el app.service
  + Agrego la configuración de forma global y la aplico en el constructor de la clase

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

import { ConfigModule, ConfigService } from '@nestjs/config';

import { Key } from './common/enum';

import { DatabaseModule } from './database/database.module';

@Module({

  imports: [ConfigModule.forRoot({ isGlobal: true }), DatabaseModule],

})

export *class* AppModule {

  static port: *number*;

  static host: *string*;

  static version: *string*;

*constructor*(private readonly *configService*: ConfigService) {

    AppModule.port = +this.configService.get(Key.SERVER\_PORT);

    AppModule.host = this.configService.get(Key.SERVER\_HOST);

    AppModule.version = this.configService.get(Key.SERVER\_VERSION);

  }

}

**CONFIGURACIÓN DEL SERVIDOR**

* Instalación de dependencias de seguridad en modo permanente + librería de validación y de transformación

npm i helmet@4.6.0 csurf@1.11.0 compression@1.7.4 [express-rate-limit@0.0.0-typescript-beta-7](mailto:express-rate-limit@0.0.0-typescript-beta-7) class-validator class-transformer

* Archivo main.ts
  + Agrego middlewares de seguridad
  + Agrego /api + versión de la aplicación
* Main.ts -> src\main.ts

import { NestFactory } from '@nestjs/core';

import \* as helmet from 'helmet';

import \* as compression from 'compression';

import \* as csurf from 'csurf';

import \* as rateLimit from 'express-rate-limit';

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

import { AppModule } from './app.module';

*const* bootstrap = async () *=>* {

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

*const* { port, host, version } = AppModule;

*const* globalPrefix = `/api/${version}`;

  app.setGlobalPrefix(globalPrefix)

  app.useGlobalPipes(new ValidationPipe({

    transform: true,

    whitelist: true,

    forbidNonWhitelisted: true

  }));

  app.use(helmet());

  app.use(compression());

  app.use(

    rateLimit({

      windowMs: 15 \* 60 \* 1000, // 15 minutes

      max: 100, // Limit each IP to 100 requests per `window` (here, per 15 minutes)

    }),

  );

  await app.listen(port);

*const* serverUrl = `Server listening on http://${host}:${port+globalPrefix}`;

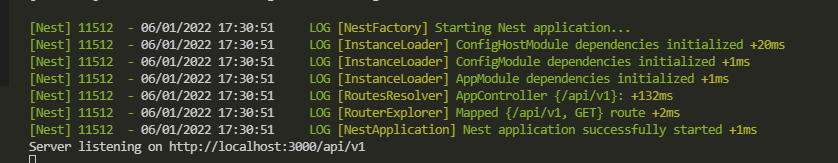
  console.log(`Server listening on ${serverUrl}`);

  app.use(csurf());

}

bootstrap();

* Arranco -> npm run start:dev y ya compila correctamente!!



**CONFIGURAR LA BASE DE DATOS**

* Instalar librería de typeorm

npm install --save @nestjs/typeorm typeorm pg

* Creo módulo para la base de datos

nest generate module database --no-spec

-> src\database\database.module.ts

* Dentro de database creo manualmente database.provider.ts
* src\database\database.provider.ts

**OJO: No es un provider al uso, por lo que llevará imports y exports el módulo para ponerlo disponible en la app**

Nota: Este proveedor inyectará la configuración del módulo de db

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

import { ConfigService } from '@nestjs/config';

import { TypeOrmModule } from '@nestjs/typeorm';

import { ConnectionOptions } from 'typeorm';

import { Key, Env, Db } from 'src/common/enum';

export *const* DatabaseProvider: DynamicModule = TypeOrmModule.forRootAsync({

  inject: [ConfigService],

  async useFactory(*config*: ConfigService) {

*const* isDevelopmentEnv = *config*.get(Key.NODE\_ENV) !== Env.Production;

*const* dbConfig = {

      type: *config*.get(Db.DB\_DIALECT),

      host: *config*.get(Db.DB\_HOST),

      port: +*config*.get(Db.DB\_PORT),

      username: *config*.get(Db.DB\_USER),

      password: *config*.get(Db.DB\_PASSWORD),

      database: *config*.get(Db.DB\_NAME),

      autoLoadEntities: *config*.get(Db.AUTO\_LOAD\_ENTITIES),

      synchronize: isDevelopmentEnv,

      logging: *config*.get(Db.DB\_LOGGING),

    } as ConnectionOptions;

    return dbConfig;

  },

});

* Ahora en src\database\database.module.ts
  + Lo declaro como un imports y exports para que sirva para toda la app.

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

import { DatabaseProvider } from './database.provider';

@Module({

  imports: [DatabaseProvider],

  exports: [DatabaseProvider]

})

export *class* DatabaseModule {}