NEST 07

- Creo un nuevo proyecto llamado Pokedex

```
nest new pokedex
```

- Selecciono npm
- Hago limpieza dejando solo el app.module limpio
- Levanto el servidor con

```
npm run start:dev
```

Servir contenido estático

- Creo una carpeta en la raíz (no en /src) llamada public
- Dentro creo un archivo index.html y otra carpeta llamada /css con el archivo styles.css
- Creo un h1 en el html para visualizarlo con algunos estilos (linkeo el css) en la vista
- Para servir contenido estático hay que instalar @nestjs/serve-static, de ahi importar ServeStaticModule y join de 'path' (node) y añadir unas lineas de código en app.module:

```
import { Module } from '@nestjs/common';
import { join } from 'path';
import {ServeStaticModule} from '@nestjs/serve-static'


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

- Son '..', dos puntos después de __dirname, si pones tres puntos no funciona

Global Prefix

- Genero el resource para la API REST que voy a hacer
- g de generar, res de resource, --no-spec para que no genere el archivo de test

```
nest g res pokemon --no-spec
```

- Selecciono REST API

- Ahora lo tengo todo montado. Los endpoints, el módulo importado, los dto, entities, los servicios
- Falta implementar la lógica pero está todo conectado
- Si quiero añadir un prefijo a la url (por ejemplo '/api/loquesea/...') puedo añadirlo en el controlador, pero si tuviera muchos endpoints sería una faena.
    - Puedo usar el app.setGlobalPrefix en el main para esto

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

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

  app.setGlobalPrefix('api/v2')


  await app.listen(3000);
}
main();
```

Docker DockerCompose - MongoDB-

- Creo el archivo en la raíz llamado docker-compose.yaml
- La imagen versión 5
- Conecto el puerto de mongo con el de mi compu
- Nombro la DB
- Para hacer la DB persistente le digo dónde guardará la imagen en mi compu
- En mi proyecto creará una carpeta llamada mongo automaticamente y la conecto con data/db que es la ruta de la imagen que estoy montando
- Ojo con las tabulaciones! Esto descargará la imagen de mongo si no está en disco

```
version: '3'

services:
  db:
    image: mongo:5
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGODB_DATABASE: nest-pokemon
    volumes:
      - ./mongo:/data/db
```

- Para ejecutar y levantar la DB

> docker-compose up -d

- Si abro Docker desktop veo en containers uno llamado Pokedex (running)
- Puedo borrar el container, y volver a ejecutar el comando de docker-compose up -d y descargará la imagen de mongo en docker de nuevo
- Debo realizar la conexión mediante un software como TablePlus mientras la DB esta corriendo (running) en docker
- la conexión es :

> mongodb://localhost:27017/nest-pokemon

Conectar Nest con Mongo

- Es preferible usar los adaptadores que Nest provee para usar inyección de dependencias y otros
- Instalo mongoose y el adaptador

> npm i --save @nestjs/mongoose mongoose

- En app.module creo la referencia a la base de datos
- Sólo hay un forRoot, luego está el forFeature
- Aquí debería colocar el usuario y el password

```
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'

@Module({
  imports: [
    ServeStaticModule.forRoot({
      rootPath: join(__dirname, '..', 'public'),
    }),

    MongooseModule.forRoot('mongodb://localhost:27017/nest-pokemon'),

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

Crear esquemas y modelos

- La entidad antes se usó como una interface. Ahora la voy a trabajar como una clase, y se recomienda que sea así
    - Porque mediante clases se pueden regir reglas

- Usualmente las entidades hacen referencia a como voy a querer grabar en la base de datos
- Es decir, hace la relación con la base de datos
- Una instancia de la clase quiere decir un registro en la base de datos
- pokemon.entity:

```
export class Pokemon {}
```

- El id no lo tengo que especificar porque mongo ya me lo da
- Lo que si necesito es el nombre y el número de pokemon
- Como quiero que sea un documento, tiene que extender de Document (de mongoose)
- Esto le añade todas las funcionalidades
- Exporto el Schema

```
import { SchemaFactory } from "@nestjs/mongoose";
import { Document } from "mongoose";

export class Pokemon extends Document{

    name: string;

    no: number;
}

export const PokemonSchema= SchemaFactory.createForClass(Pokemon)
```

- Necesito especificar el decorador @Schema() (de NEST) para indicar que esto es un esquema de una base de datos
- El nombre tiene que ser único, y el número de pokemon también.
- Todo se hace mediante decoradores
- Le coloco el decorador de @Schema de forma general, para indicar que esto es un schema de una DB

```
import { Prop,Schema, SchemaFactory } from "@nestjs/mongoose";
import { Document } from "mongoose";

@Schema()
export class Pokemon extends Document{

    @Prop({
        unique: true,
        index: true
    })
    name: string;

    @Prop({
        unique: true,
```

```
        index: true
    })
    no: number;
}

export const PokemonSchema= SchemaFactory.createForClass(Pokemon)
```

- Ahora hace falta conectarlo a la base de datos, que mongoose pueda crear las referencias, la colección y me permita la inyección de dependecias para quepueda usar esta clase en mis servicios
- Voy a pokemon.module, imports
- Uso el MongooseModule pero ahora no es forRoot, solo hay un root en mi aplicación. Es el forFeature y me pide los modelos
- Tiene dos propiedades, el name y el schema
- Pokemon.name, este name es de la extensión del Document
- El schema es la exportación
- Ojo, en el forFeature abro corchetes y abro llaves para insertar el objeto

```
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';

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

- Ahora si abro Compass (o TablePlus) en la conexión con nest-pokemon aparece la colección pokemons
- Si tuviera más modelos, podría una coma al objeto y añadiría otro modelo