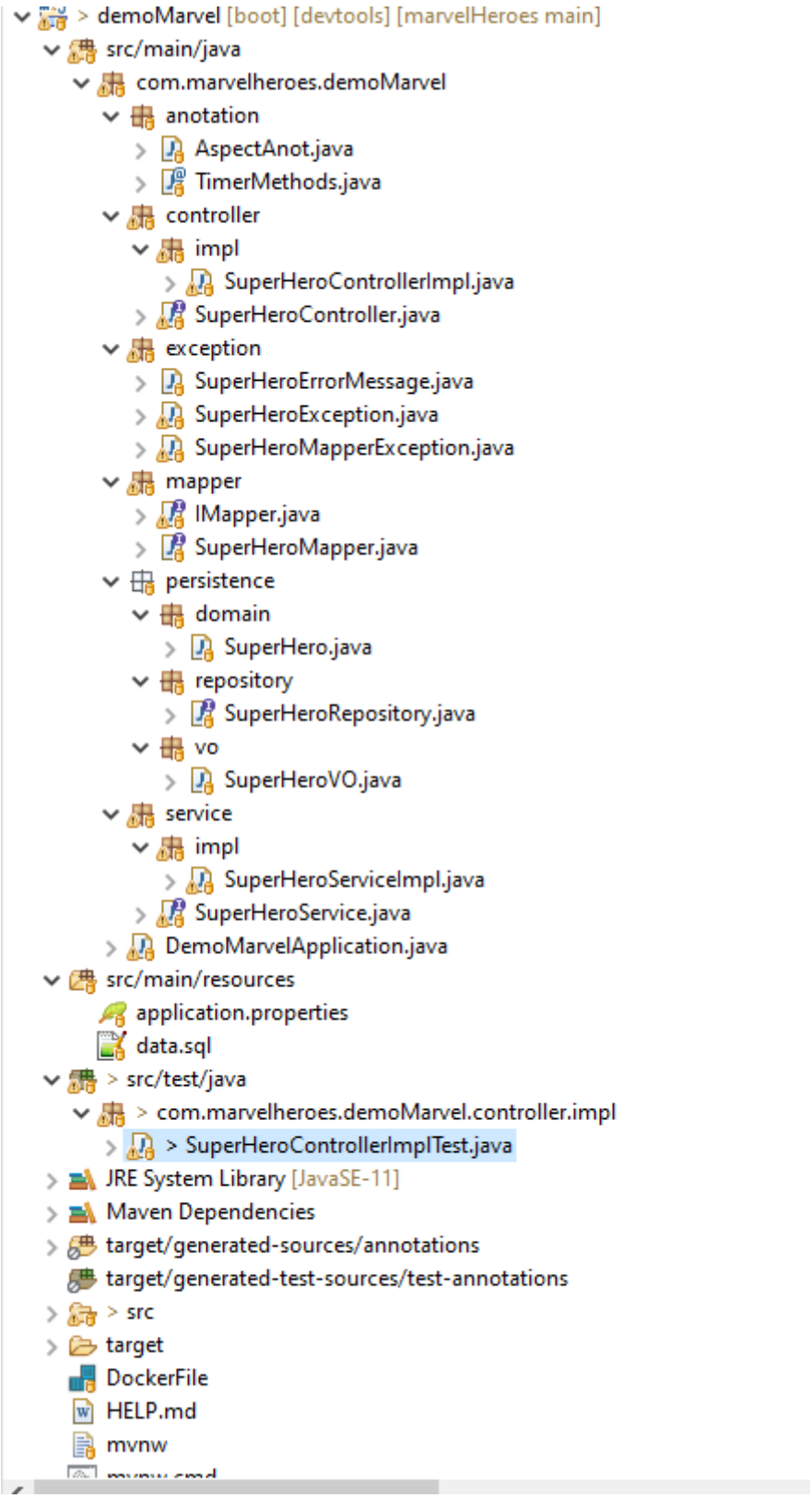
ocumentacion api super heroes

Esqueleto:



com.marvelheroes.demoMarvel.anotation:

Aqui guardamos la anotacion creada ligada a una clase que funciona como aspecto, en ella controlamos el tiempo de los metodos del controler que tengan esta anotacion creada(Esta en todos pero podemos quitarlo del metodo que queramos)

com.marvelheroes.demoMarvel.exception
Aqui tenemos guardadas al excepciones para cada uno de los casos capturados en la aplicación restfull(junto a las costantes usadas para los mensajes de error)

com.marvelheroes.demoMarvel.mapper
Aqui hemos creado un mapper para cambiar entre la entidad y el VO, he usado uno generico por si la aplicación “mejorara” ya solo se extenderia de imapper por si se metieran mas entidades

com.marvelheroes.demoMarvel.persistence
Aqui hemos creado tres paquetes, uno para la entidad, otro para el VO de salida para usar en los contratos rest y otro el repository JPA en el que he usado distintas querys y varias formas(no ha hecho falta ni usar native ni un entity manager para querys complejas porque el ejemplo es sencillo)

com.marvelheroes.demoMarvel.controller
com.marvelheroes.demoMarvel.service

sigue la lógica de interfaces para separar la implementacion del contrato

En el application.yml he puesto la base de datos h2 con los datos de seguridad

La seguridad esta simplificada, se podría usar auth2 o configurar con spring security para configurar una seguridad mas compleja e incluso usar keycloak para mejorar los grupos y demás

El fichero data.sql esta creada la tabla y los datos por defecto creados para la prueba

Con el docker file estaria la configuracion basica para desplegar nuestro microservicio en docker.

Los test, he creado los del controller, ahi he usado mockito con junit con ellos he comprobado la covertura y los test de algunos servicios para ver diferentes ejemplos, usando desde asserts a verify para los servicios void.

He usado un ejemplo de cache muy simple, en el cual cacheamos las peticiones de consulta y limpiamos cache cuando cambiamos o borramos un servicio para que se vean reflejado los cambios en los datos.

He usado jotadoc para facilitar la documentacion del codigo y he usado patrones solid y separando bien cada funcionalidad dentro de su lugar correspondiente(controller-servicio-repository) recibiendo el objetoVO y operando con la entidad y devolviendo ese mismo vo con ayuda del mapper

Normalmente uso lombok pero aquí he optado por usar getter y setter para evitar poner el archivo y etc.

El codigo esta subido en un repositorio git publicado, la ruta es la siguiente:

https://github.com/zapapoptero/marvelHeroes