**Documentación detallada de la ejecución y explicación del proyecto desde SpringBoot y Java 8**

1. Instalación de NPM mediante la herramienta NodeJs

[**https://nodejs.org/en/**](https://nodejs.org/en/)

1. Abrimos la consola de comandos y llamamos al instalador de paquetes NPM para que nos instale MAVEN

npm install maven

1. Instalamos OpenApi generator cli para ejecutar comandos de openapi
2. npm install @openapitools/openapi-generator-cli -g
3. Abrimos powershell si estamos en windows 10 como administrador y realizmos el siguiente comando:

**set-executionpolicy unrestricted –force**



1. Nos descargamos el siguiente fichero de ejemplo para darle la información necesaria a OpenApi generator sobre nuestra API. (En nuestro caso hemos usado un fichero petstore.yaml que se encuentra en la documentación oficial de la app.

****

[**https://raw.githubusercontent.com/openapitools/openapi-generator/master/modules/openapi-generator/src/test/resources/3\_0/petstore.yaml**](https://raw.githubusercontent.com/openapitools/openapi-generator/master/modules/openapi-generator/src/test/resources/3_0/petstore.yaml)

1. Creamos una carpeta donde se vaya a alojar el proyecto con el archivo petstore.yaml contenido en ella.
2. Ejecutamos el siguiente comando dentro del directorio creado:

(unicamente modificamos el nombre del fichero .yaml a petstore.yaml o el que hayamos elegido)

openapi-generator-cli generate -g spring --library spring-boot -i petstore.yaml -o ${PWD} -p groupId=com.redhat -p artifactId=todo -p artifactVersion=1.0.0-SNAPSHOT -p basePackage=com.redhat.todo -p configPackage=com.redhat.todo.config -p apiPackage=com.redhat.todo.api -p modelPackage=com.redhat.todo.model -p sourceFolder=src/main/gen -p dateLibrary=java8 -p java8=true

1. Añadimos el siguiente plugin a nuestro pom.xml
2. <plugin>
3. <groupId>org.codehaus.mojo</groupId>
4. <artifactId>build-helper-maven-plugin</artifactId>
5. <version>3.1.0</version>
6. <executions>
7. <execution>
8. <phase>generate-sources</phase>
9. <goals>
10. <goal>add-source</goal>
11. </goals>
12. <configuration>
13. <sources>src/main/gen</sources>
14. </configuration>
15. </execution>
16. </executions>
17. </plugin>

**7.** Añadiremos a plugin spring-boot-maven-plugin el siguiente apartado para que nos deje ejecutar la aplicación maven

 <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

                <executions>

                    <execution>

                       <goals>

                        <goal>run</goal>

                    </goals>

                    </execution>

                </executions>

            </plugin>



8. Abrimos la terminal y ejecutamos el siguiente comando dentro del directorio de nuestro proyecto para limpiar el proyecto e instalar todas sus dependencias modificadas:

mvn clean install

9. No cierre la terminal ahora toca ejecutar el siguiente comando para que se quede la aplicación que hemos creado ejecutandose y poder acceder a ella desde cualquier navegador:

mvn spring:boot run

10. Abrimos el navegador (preferentemente firefox por el poco consumo de recursos y escribimos la siguiente URL):

<http://localhost:8080/swagger-ui.html#/>



**11.** Aquí nos encontraremos la documentación necesaria para hacer peticiones CRUD a nuestras apis



Inlusive se pueden ampliar y probar



12. Modelos expecificados:



**IMPORTANTE:**  este proyecto usa [api\_key](https://soporte.buda.com/es/articles/2805986-que-es-una-api-y-api-key)

**13.** Estructura autogenerada de nuestro proyecto

14. **RECOMENDACIÓN TESTING:**

Recomendamos encarecidamente el uso de postman para probar las apis.

<https://www.postman.com/>