1.- dependencias de RestEasy y Gson en el archivo pom.xml

<dependencies>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-web-api</artifactId>

<version>6.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.jboss.resteasy</groupId>

<artifactId>resteasy-jaxrs</artifactId>

<version>2.3.7.Final</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.2.4</version>

</dependency>

</dependencies>

//modificar el archivo web.xml para poder utilizar RestEasy y dejarlo así:

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0" xmlns="<a class="vglnk" href="<http://java.sun.com/xml/ns/javaee>" rel="nofollow"><span>http</span><span>://</span><span>java</span><span>.</span><span>sun</span><span>.</span><span>com</span><span>/</span><span>xml</span><span>/</span><span>ns</span><span>/</span><span>javaee</span></a>" xmlns:xsi="<a class="vglnk" href="<http://www.w3.org/2001/XMLSchema-instance>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>w3</span><span>.</span><span>org</span><span>/</span><span>2001</span><span>/</span><span>XMLSchema</span><span>-</span><span>instance</span></a>" xsi:schemaLocation="<a class="vglnk" href="<http://java.sun.com/xml/ns/javaee>" rel="nofollow"><span>http</span><span>://</span><span>java</span><span>.</span><span>sun</span><span>.</span><span>com</span><span>/</span><span>xml</span><span>/</span><span>ns</span><span>/</span><span>javaee</span></a> <a class="vglnk" href="<http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd>" rel="nofollow"><span>http</span><span>://</span><span>java</span><span>.</span><span>sun</span><span>.</span><span>com</span><span>/</span><span>xml</span><span>/</span><span>ns</span><span>/</span><span>javaee</span><span>/</span><span>web</span><span>-</span><span>app</span><span>\_</span><span>3</span><span>\_</span><span>0</span><span>.</span><span>xsd</span></a>">

<display-name>Restful Web Application</display-name>

<!-- Auto scan REST service -->

<context-param>

<param-name>resteasy.scan</param-name>

<param-value>true</param-value>

</context-param>

<!-- this need same with resteasy servlet url-pattern -->

<context-param>

<param-name>resteasy.servlet.mapping.prefix</param-name>

<param-value>/rest</param-value>

</context-param>

<listener>

<listener-class>

org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap

</listener-class>

</listener>

<servlet>

<servlet-name>resteasy-servlet</servlet-name>

<servlet-class>

org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>resteasy-servlet</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

</web-app>

//con esto terminamos la parte de la configuración

 //Ahora toca crear una nueva clase llamada «SumaRestService» dentro del package rest // del proyecto:

package rest;

import clases.Numeros;

import com.google.gson.Gson;

import javax.ws.rs.Consumes;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

@Path("/suma")

public class SumaRestService {

@GET

@Path("/sumar/{param}/{param2}")

public int sumarNumerosGet(@PathParam("param") int msg, @PathParam("param2") int msg2) {

return msg+msg2;

}

@POST

@Path("/sumar")

@Consumes(MediaType.APPLICATION\_JSON)

@Produces(MediaType.TEXT\_PLAIN)

public String sumarNumerosPost(String json) {

Gson gson = new Gson();

Numeros nums = gson.fromJson(json, Numeros.class);

String responseString = String.valueOf(nums.getNro1()+nums.getNro2());

return responseString;

}

@PUT

@Path("/sumar")

@Consumes(MediaType.APPLICATION\_JSON)

@Produces(MediaType.TEXT\_PLAIN)

public String sumarNumerosPut(String json) {

Gson gson = new Gson();

Numeros nums = gson.fromJson(json, Numeros.class);

String responseString = String.valueOf(nums.getNro1()+nums.getNro2());

return responseString;

}

}

//  creo un package clases y dentro del package una clase llamada Numeros:

package clases;

 public class Numeros {

private int nro1;

private int nro2;

public int getNro1() {

return nro1;

}

public void setNro1(int nro1) {

this.nro1 = nro1;

}

public int getNro2() {

return nro2;

}

public void setNro2(int nro2) {

this.nro2 = nro2;

}

}

2: - DockerFile

server {

listen 80;

listen 443 http2 ssl;

listen [::]:443 http2 ssl;

server\_name localhost;

ssl\_certificate /etc/ssl/certs/nginx-selfsigned.crt;

ssl\_certificate\_key /etc/ssl/private/nginx-selfsigned.key;

ssl\_dhparam /etc/ssl/certs/dhparam.pem;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_prefer\_server\_ciphers on;

ssl\_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";

ssl\_ecdh\_curve secp384r1;

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_tickets off;

ssl\_stapling on;

ssl\_stapling\_verify on;

resolver 8.8.8.8 8.8.4.4 valid=300s;

resolver\_timeout 5s;

add\_header Strict-Transport-Security "max-age=63072000; includeSubdomains";

add\_header X-Frame-Options DENY;

add\_header X-Content-Type-Options nosniff;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

docker exec -it test01 sh

$ docker run -d --name test01 nginx:números

docker stop test01

docker start test01

docker commit test01 numeros/nginx:autocertificado

3. Bashero de nginx

#!/bin/bash

RESPONSE="HTTP/1.1 200 OK\r\nConnection: keep-alive\r\n\r\n${2:-"OK"}\r\n"

while { echo -en "$RESPONSE"; } | nc -l "${1:-8080}"; do

echo "================================================"

done

./numeros.sh [port] [response]