

# Ejercicio 5

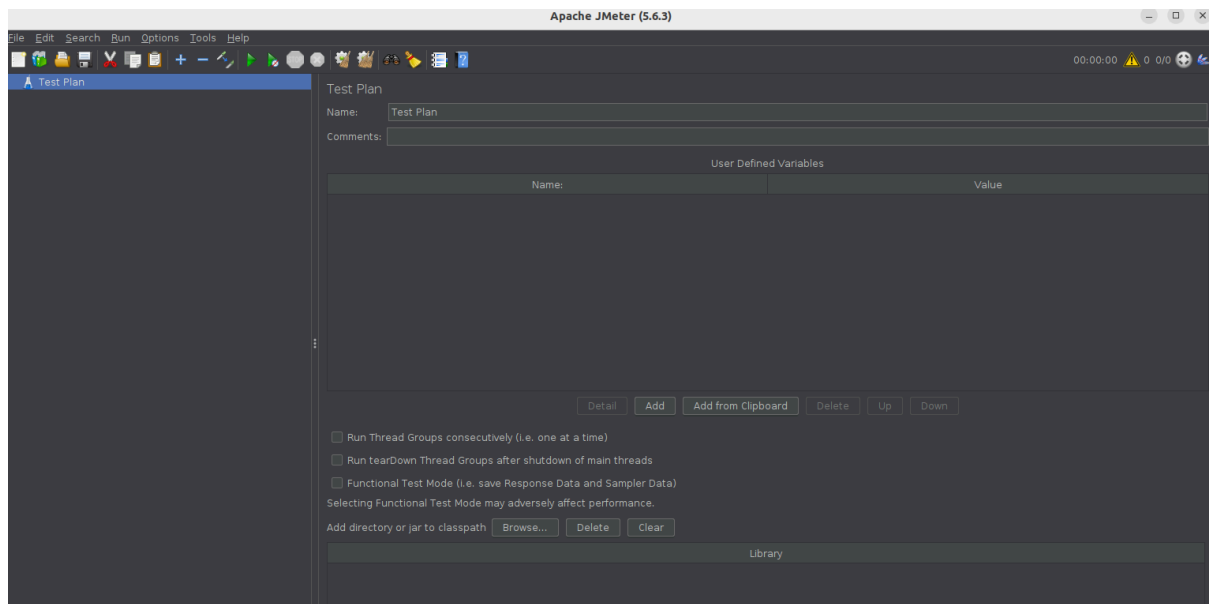
## Rate Limiter

1) Primero, debemos instalar JMeter, en este caso, lo instalare con comandos usando **sudo apt install jmeter**.

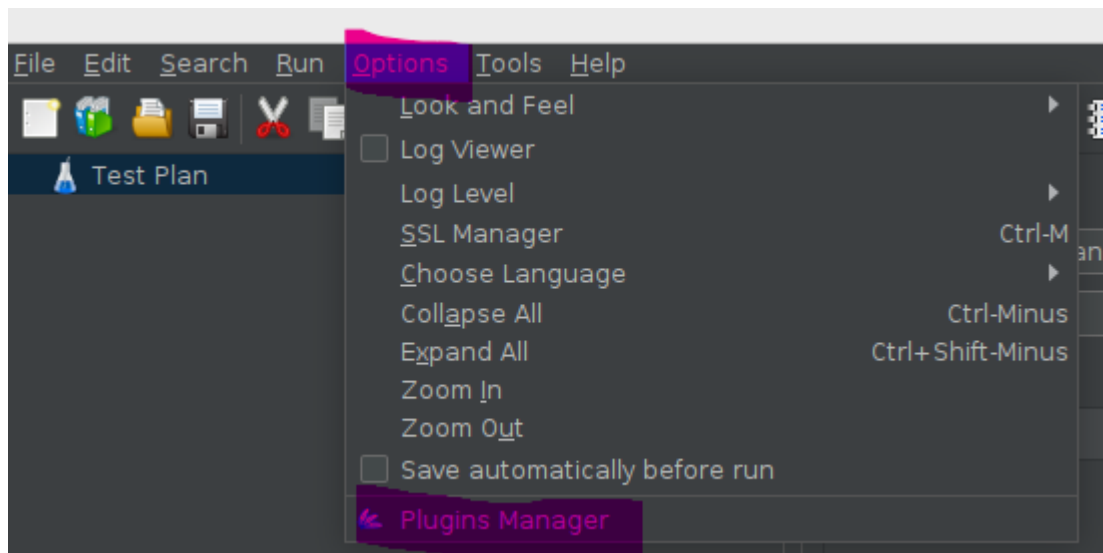
Luego de instalado podemos proceder a instalar los plugins necesarios.

Debemos entrar a <https://jmeter-plugins.org/install/Install/> y descargarlos.

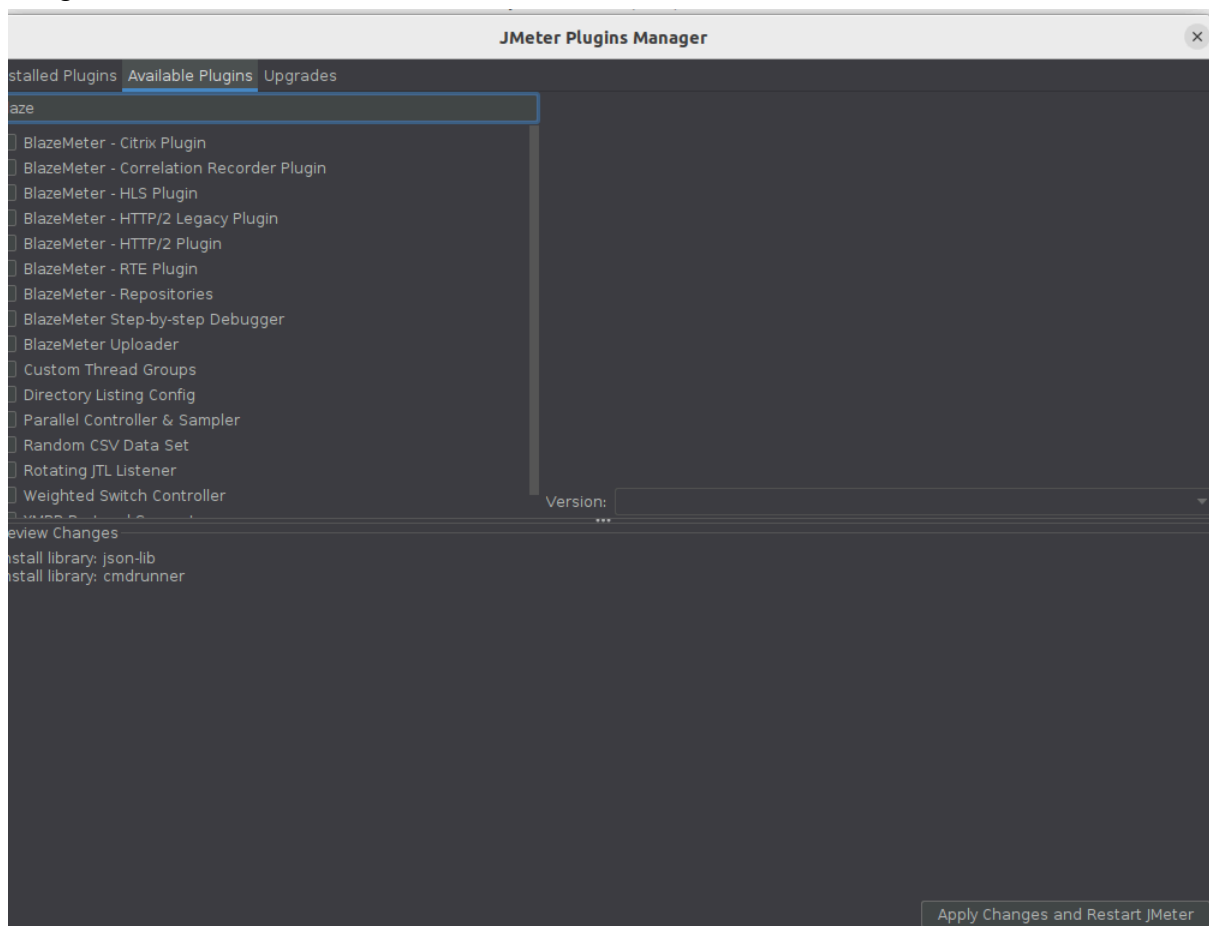
Una vez instalado y configurado, debería quedarnos una ventana como la siguiente.



Desde ahí podemos seleccionar “Options” e irnos al manager de extensiones.



Luego de esto buscamos BlazeMeter. Tambien lo instalaremos.



- 2) Seguido de esto, descargaremos el material que el docente nos provee para continuar con el ejercicio. Descargaremos **03b\_JakartaEESecurity**.

En el archivo del pom.xml podemos ver lo siguiente:

```
*Ejercicio 05/03b_JakartaEESecurity/pom.xml x
29
30  <dependencies>
31    <dependency>
32      <groupId>jakarta.platform</groupId>
33      <artifactId>jakarta.jakartaee-api</artifactId>
34      <version>${jakartaee-api.version}</version>
35      <scope>provided</scope>
36    </dependency>
37
38
39    <dependency>
40      <groupId>jakarta.security.enterprise</groupId>
41      <artifactId>jakarta.security.enterprise-api</artifactId>
42      <scope>provided</scope>
43      <version>3.0.0</version>
44    </dependency>
45
46    <!-- token bucket implemntacion -->
47    <dependency>
48      <groupId>com.bucket4j</groupId>
49      <artifactId>bucket4j-core</artifactId>
50      <version>8.10.1</version>
51    </dependency>
52
53  </dependencies>
54
55  <build>
56    <finalName>03b_JakartaEESecurity</finalName>
57    <plugins>
58      <plugin>
59        <groupId>org.apache.maven.plugins</groupId>
60        <artifactId>maven-compiler-plugin</artifactId>
61        <version>${compiler-plugin.version}</version>
62      </plugin>
63      <plugin>
64        <artifactId>maven-war-plugin</artifactId>
65        <version>${war-plugin.version}</version>
66        <configuration>
67          <failOnMissingWebXml>false</failOnMissingWebXml>
```

Una vez visto esto, procedemos a deployar el servidor para probar los endpoints y abordar los demas puntos.

Para ello deployamos el servidore con mvn package wildfly:dev.

```
root@Artorias: /home/artorias/Desktop/Programming/JavaEE__Practices/Ejercic...
re) values ('grupo2')
18:05:42,309 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Grupo (nomb
re) values ('admin')
18:05:42,310 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Usuario_Gru
po (Usuario_username, grupos_nombre) values ('usr1','grupo1')
18:05:42,310 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Usuario_Gru
po (Usuario_username, grupos_nombre) values ('usr2','grupo2')
18:05:42,311 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Usuario_Gru
po (Usuario_username, grupos_nombre) values ('usr3','grupo1')
18:05:42,311 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Usuario_Gru
po (Usuario_username, grupos_nombre) values ('usr3','grupo2')
18:05:42,312 INFO [stdout] (ServerService Thread Pool -- 39) Hibernate: insert into Usuario_Gru
po (Usuario_username, grupos_nombre) values ('usr4','admin')
18:05:42,620 INFO [org.wildfly.security.soteria.original.CdiExtension] (MSC service thread 1-2)
Activating jakarta.security.enterprise.authentication.mechanism.http.BasicAuthenticationMechani
smDefinition authentication mechanism from ejemplo00.infraestructura.seguridad.SeguridadConfigur
acion class
18:05:42,766 INFO [org.wildfly.security.soteria.original.SamRegistrationInstaller] (Server Servi
ce Thread Pool -- 18) Initializing Soteria 3.0.3.Final for context '/03b_JakartaEESecurity'
18:05:42,804 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 18) WFLYUT0021
: Registered web context: '/03b_JakartaEESecurity' for server 'default-server'
18:05:42,849 INFO [org.jboss.as.server] (management-handler-thread - 1) WFLYSRV0010: Deployed "
03b_JakartaEESecurity.war" (runtime-name : "03b_JakartaEESecurity.war")
```

Luego de eso, podemos hacer un curl para probar los siguientes endpoints.

curl --user usr4:usr4pass -v

[http://localhost:8080/03b\\_JakartaEESecurity/seguero/config/activarRateLimiter?valor=true](http://localhost:8080/03b_JakartaEESecurity/seguero/config/activarRateLimiter?valor=true)

```
artorias@Artorias: ~/Desktop/Programming/JavaEE___Practic...
artorias@Artorias:~/Desktop/Programming/JavaEE___Practices/Ejercicio 05/03b_JakartaEESecurity$ curl --user usr4:usr4pass -v http://localhost:8080/03b_JakartaEESecurity/seguero/config/activarRateLimiter?valor=tru
* Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'usr4'
> GET /03b_JakartaEESecurity/seguero/config/activarRateLimiter?valor=tru HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNyNDp1c3I0cGFzcw==
> User-Agent: curl/7.81.0
> Accept: */*
*
* Mark bundle as not supporting multiuse
< HTTP/1.1 204 No Content
< Date: Wed, 24 Apr 2024 21:05:39 GMT
*
* Connection #0 to host localhost left intact
artorias@Artorias:~/Desktop/Programming/JavaEE___Practices/Ejercicio 05/03b_JakartaEESecurity$
```

Vemos que nos arroja código 204 que es exitoso, pero sin content. Vemos que tambien se autentica con el usr4 y usa Basic auth.

Ahora intentamos con el otro endpoint.

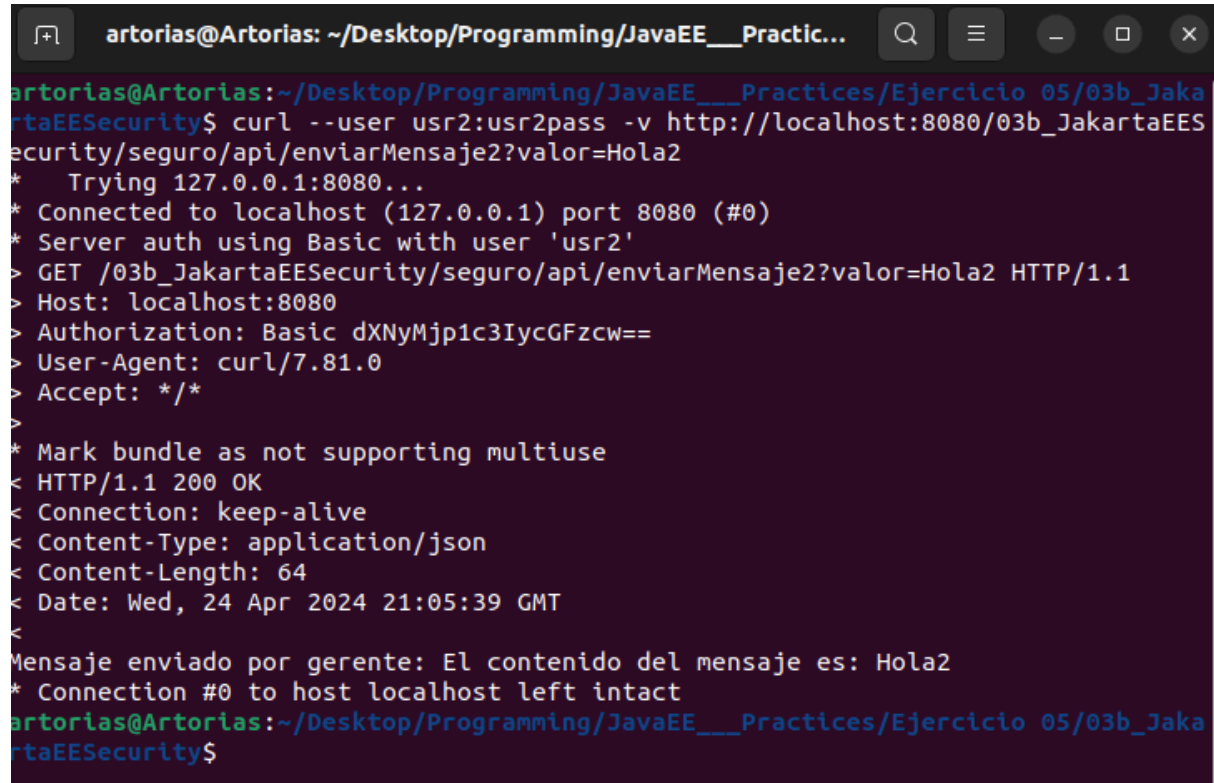
[curl --user usr1:usr1pass -v](#)

[http://localhost:8080/03b\\_JakartaEESecurity/seguero/api/enviarMensaje?valor=Hola](http://localhost:8080/03b_JakartaEESecurity/seguero/api/enviarMensaje?valor=Hola)

```
artorias@Artorias: ~/Desktop/Programming/JavaEE___Practic...
artorias@Artorias:~/Desktop/Programming/JavaEE___Practices/Ejercicio 05/03b_JakartaEESecurity$ curl --user usr1:usr1pass -v http://localhost:8080/03b_JakartaEESecurity/seguero/api/enviarMensaje?valor=Hola
* Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'usr1'
> GET /03b_JakartaEESecurity/seguero/api/enviarMensaje?valor=Hola HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNyMTp1c3IxcGFzcw==
> User-Agent: curl/7.81.0
> Accept: */*
*
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Connection: keep-alive
< Content-Type: application/json
< Content-Length: 63
< Date: Wed, 24 Apr 2024 21:05:39 GMT
*
Mensaje enviado por gerente: El contenido del mensaje es: Hola
* Connection #0 to host localhost left intact
artorias@Artorias:~/Desktop/Programming/JavaEE___Practices/Ejercicio 05/03b_JakartaEESecurity$
```

El mismo caso, respuesta 200, todo ok, esta vez con usr1 usandng basic Auth. Content del mensaje “Hola”.

Siguiente caso con el endpoint `curl --user usr2:usr2pass -v http://localhost:8080/03b_JakartaEESecurity/seguero/api/enviarMensaje2?valor=Hola2`  
Y la respuesta es la siguiente.

A terminal window with a dark background and light-colored text. The window title is 'artorias@Artorias: ~/Desktop/Programming/JavaEE\_\_Practic...'. The prompt is 'artorias@Artorias:~/Desktop/Programming/JavaEE\_\_Practices/Ejercicio 05/03b\_JakartaEESecurity\$'. The command entered is 'curl --user usr2:usr2pass -v http://localhost:8080/03b\_JakartaEESecurity/seguero/api/enviarMensaje2?valor=Hola2'. The output shows the curl process: connecting to localhost:8080, using basic auth with user 'usr2', sending a GET request, and receiving a 200 OK response with headers like 'Content-Type: application/json' and 'Content-Length: 64'. The body of the response is 'Mensaje enviado por gerente: El contenido del mensaje es: Hola2'.

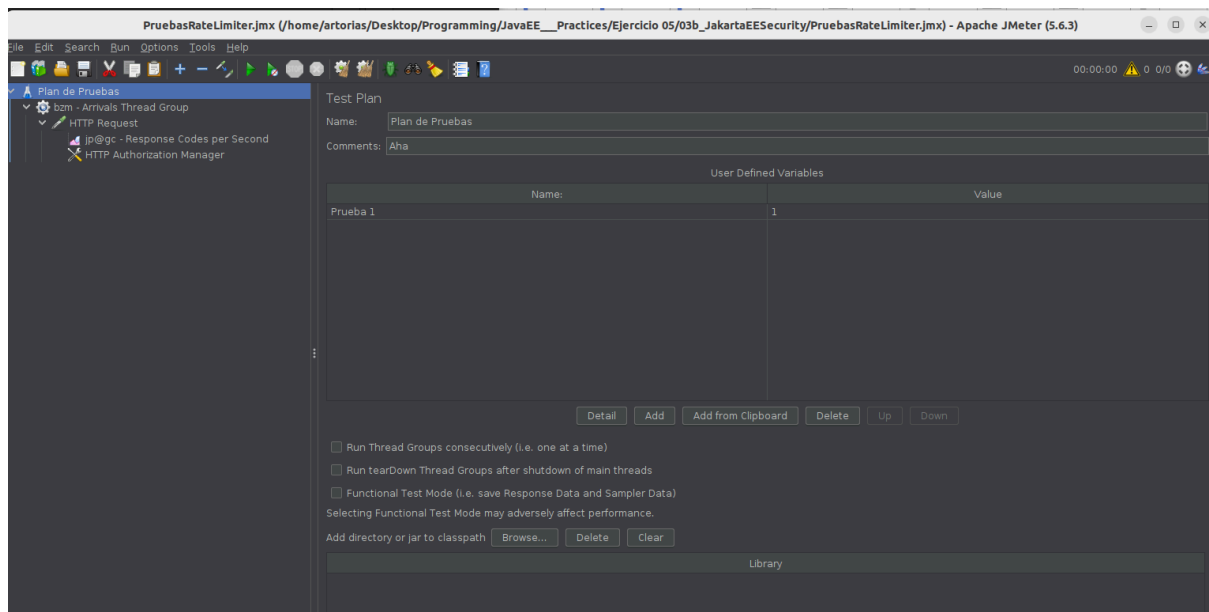
```
artorias@Artorias:~/Desktop/Programming/JavaEE__Practic...
artorias@Artorias:~/Desktop/Programming/JavaEE__Practices/Ejercicio 05/03b_JakartaEESecurity$ curl --user usr2:usr2pass -v http://localhost:8080/03b_JakartaEESecurity/seguero/api/enviarMensaje2?valor=Hola2
* Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* Server auth using Basic with user 'usr2'
> GET /03b_JakartaEESecurity/seguero/api/enviarMensaje2?valor=Hola2 HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNyMjp1c3IycGFzcw==
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Connection: keep-alive
< Content-Type: application/json
< Content-Length: 64
< Date: Wed, 24 Apr 2024 21:05:39 GMT
<
Mensaje enviado por gerente: El contenido del mensaje es: Hola2
* Connection #0 to host localhost left intact
artorias@Artorias:~/Desktop/Programming/JavaEE__Practices/Ejercicio 05/03b_JakartaEESecurity$
```

Respuesta 200, en este caso vemos que el mensaje es “Hola2” y usa basic auth con el usr2.

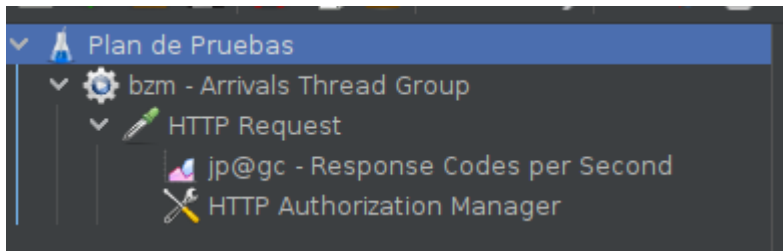
Si vemos la consola del servidor, podemos ver, que al acceder a los endpoints con los usuarios respectivos, vemos que los user se logean y la contraseña es correcta para todos los usuarios.

```
root@Artorias: /home/artorias/Desktop/Programming/JavaEE__Practices/Ejercicio 05/03b_JakartaEESecurity
nfraestructura.seguridad.SeguridadConfiguracion class
18:05:42,766 INFO [org.wildfly.security.soteria.original.SamRegistrationInstaller] (ServerService Thread Pool -- 18) Initi
alizing Soteria 3.0.3.Final for context '/03b_JakartaEESecurity'
18:05:42,804 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 18) WFLYUT0021: Registered web context: '
/03b_JakartaEESecurity' for server 'default-server'
18:05:42,849 INFO [org.jboss.as.server] (management-handler-thread - 1) WFLYSRV0010: Deployed "03b_JakartaEESecurity.war"
(runtime-name : "03b_JakartaEESecurity.war")
19:14:35,602 INFO [stdout] (default task-1) ** IdentityStore en base de datos
19:14:35,636 INFO [stdout] (default task-1) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,g1_1.n
ombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0.username=g
1_0.Usuario_username where u1_0.username=?
19:14:35,662 INFO [stdout] (default task-1) encuentre usuario: usr4
19:14:35,663 INFO [stdout] (default task-1) contrasea correcta
19:14:35,663 INFO [stdout] (default task-1) Lista de grupos:[admin]
19:14:35,928 INFO [stdout] (default task-1) Tokens restantes: 9
19:14:35,948 INFO [stdout] (default task-1) RateLimitir estado: false
19:25:54,345 INFO [stdout] (default task-1) ** IdentityStore en base de datos
19:25:54,346 INFO [stdout] (default task-1) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,g1_1.n
ombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0.username=g
1_0.Usuario_username where u1_0.username=?
19:25:54,348 INFO [stdout] (default task-1) encuentre usuario: usr1
19:25:54,348 INFO [stdout] (default task-1) contrasea correcta
19:25:54,348 INFO [stdout] (default task-1) Lista de grupos:[grupo1]
20:24:15,432 INFO [stdout] (default task-1) ** IdentityStore en base de datos
20:24:15,433 INFO [stdout] (default task-1) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,g1_1.n
ombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0.username=g
1_0.Usuario_username where u1_0.username=?
20:24:15,438 INFO [stdout] (default task-1) encuentre usuario: usr2
20:24:15,439 INFO [stdout] (default task-1) contrasea correcta
20:24:15,439 INFO [stdout] (default task-1) Lista de grupos:[grupo2]
```

Para el siguiente punto, debemos ejecutar **PruebaRateLimiter.jmx**.  
Debemos abrir JMeter. Luego de esto debemos configurarlo correctamente.  
Para esto agregamos varios componentes para captar.



En primera instancia agregamos un Thread Group de llegadas para captar las peticiones, y a su vez dentro de eso, un HTTP request. Dentro de esta request, agregamos response codes per second para captar esas respuestas por segundo y a su vez el auth manager, para poder mandar las credenciales y nos deje generar esas respuestas.



Luego dentro de las request, ponemos las siguientes condiciones:

HTTP Request

Name: HTTP Request

Comments:

Basic Advanced

Web Server

Protocol (http): http Server Name or IP: localhost Port Number: 8080

HTTP Request

GET Path: 03b\_JakartaEESecurity/seguo/api/enviarMensaje Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
valor	Hola	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

Detail Add Add from Clipboard Delete Up Down

Y en el auth manager, tambien dejamos la siguiente configuración.

PruebasRateLimiter.jmx (/home/artorias/Desktop/Programming/JavaEE\_\_Practices/Ejercicio 05/03b\_JakartaEESecurity/PruebasRateLimiter.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

Plan de Pruebas

- bzm - Arrivals Thread Group
  - HTTP Request
    - jp@gc - Response Codes per Second
      - HTTP Authorization Manager

HTTP Authorization Manager

Name: HTTP Authorization Manager

Comments:

Options

☐ Clear auth on each iteration?

☐ Use Thread Group configuration to control clearing

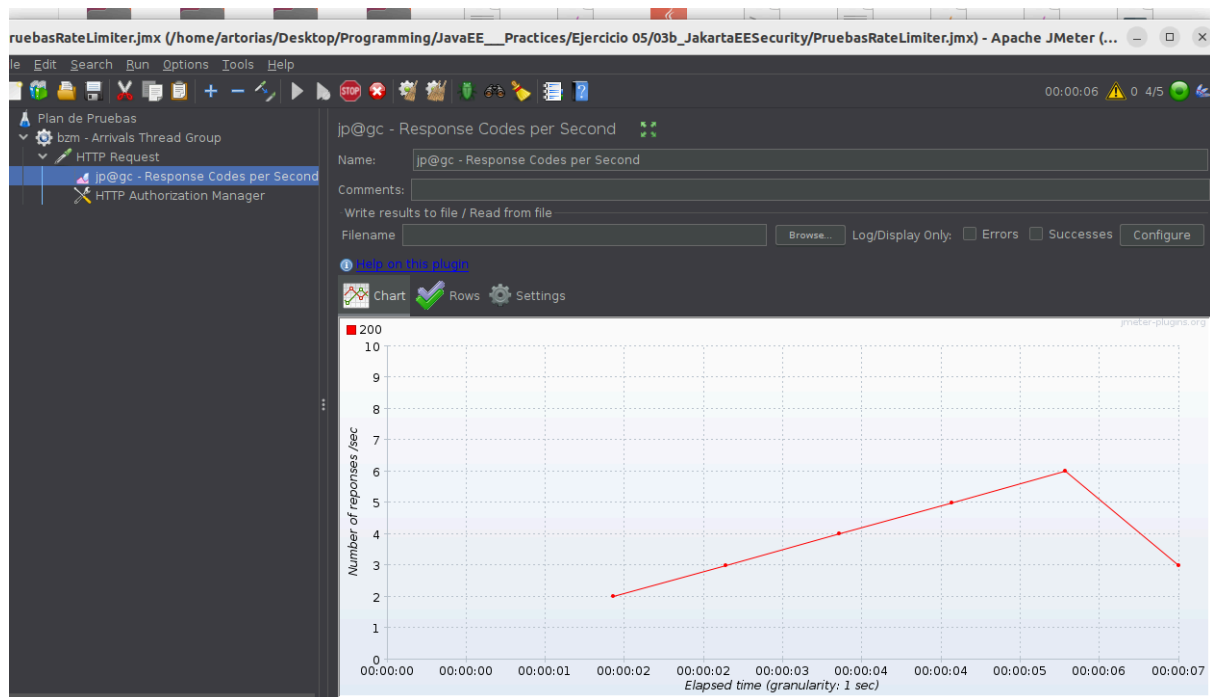
Authorizations Stored in the Authorization Manager

Base URL	Username	Password	Domain	Realm	Mechanism
http://localhost:8080	usr1	usr1.pass			BASIC

Una vez tenemos todo esto andando, podemos volver a las request y correr nuestro JMeter con el servidor corriendo.

Cuando hagamos esto, podremos empezar a ver la gráfica con las request por segundo.





Como vemos en la imagen, llega con código 200, por ende es exitoso, pero si dejamos un rato, podemos ver como limita las peticiones y responde con un 429, pero en el mismo momento que ya libera recurso para responder a esa petición.

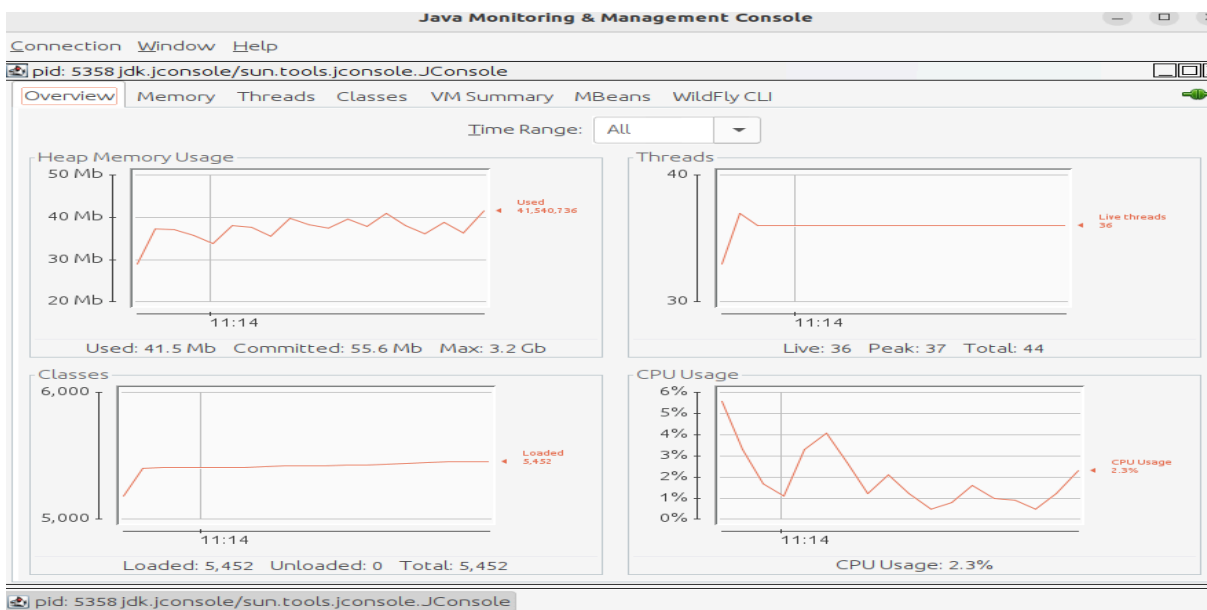
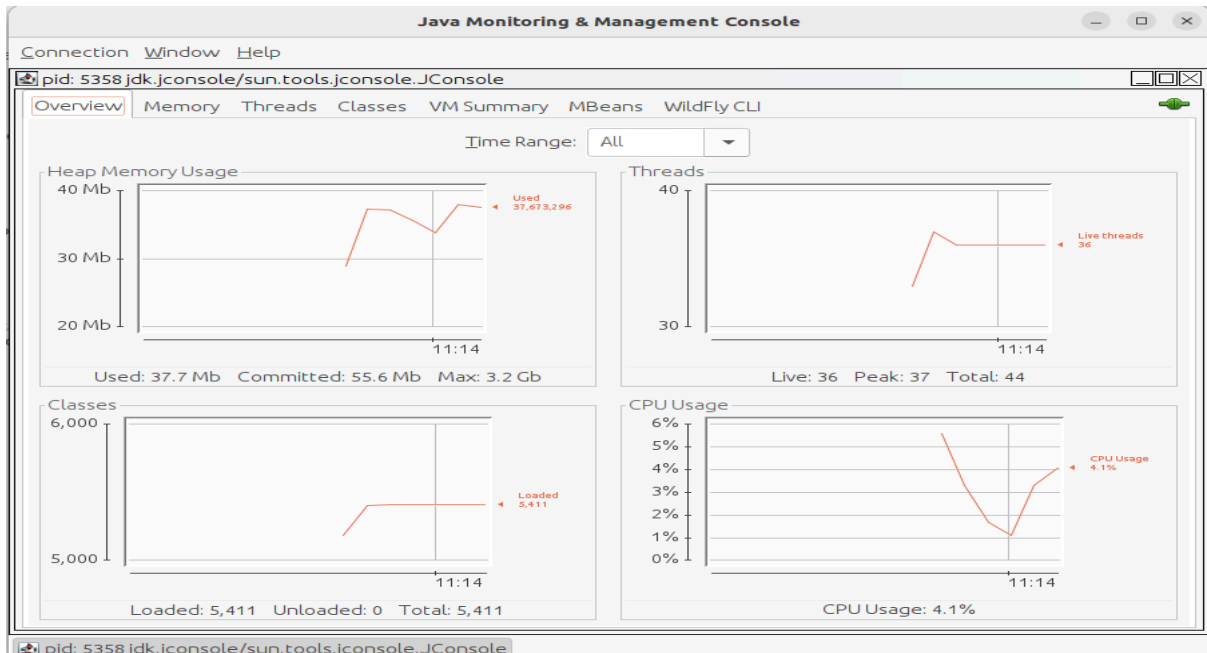
```

root@Artorias: /home/artorias/Desktop/Programming/JavaEE__Practices/Ejercicio 05/03b_JakartaEESecurity
11:30:15,058 INFO [stdout] (default task-6) contrasea correcta
11:30:15,058 INFO [stdout] (default task-6) Lista de grupos:[grupo1]
11:30:15,058 INFO [stdout] (default task-6) Tokens restantes: 0
11:30:15,257 INFO [stdout] (default task-10) ** IdentityStore en base de datos
11:30:15,258 INFO [stdout] (default task-10) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,
g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0
.username=g1_0.Usuario_username where u1_0.username=?
11:30:15,258 INFO [stdout] (default task-10) encuentre usuario: usr1
11:30:15,258 INFO [stdout] (default task-10) contrasea correcta
11:30:15,258 INFO [stdout] (default task-10) Lista de grupos:[grupo1]
11:30:15,258 INFO [stdout] (default task-10) Tokens restantes: 0
11:30:15,258 INFO [stdout] (default task-10) El servidor no acepta mensajes
11:30:15,364 INFO [stdout] (default task-10) ** IdentityStore en base de datos
11:30:15,364 INFO [stdout] (default task-10) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,
g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0
.username=g1_0.Usuario_username where u1_0.username=?
11:30:15,364 INFO [stdout] (default task-10) encuentre usuario: usr1
11:30:15,364 INFO [stdout] (default task-10) contrasea correcta
11:30:15,364 INFO [stdout] (default task-10) Lista de grupos:[grupo1]
11:30:15,365 INFO [stdout] (default task-10) Tokens restantes: 0
11:30:15,365 INFO [stdout] (default task-10) El servidor no acepta mensajes
11:30:15,459 INFO [stdout] (default task-10) ** IdentityStore en base de datos
11:30:15,459 INFO [stdout] (default task-10) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,
g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0
.username=g1_0.Usuario_username where u1_0.username=?
11:30:15,459 INFO [stdout] (default task-10) encuentre usuario: usr1
11:30:15,459 INFO [stdout] (default task-10) contrasea correcta
11:30:15,459 INFO [stdout] (default task-10) Lista de grupos:[grupo1]
11:30:15,460 INFO [stdout] (default task-10) Tokens restantes: 0
11:30:15,460 INFO [stdout] (default task-10) El servidor no acepta mensajes
11:30:15,564 INFO [stdout] (default task-10) ** IdentityStore en base de datos
11:30:15,565 INFO [stdout] (default task-10) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,
g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0
.username=g1_0.Usuario_username where u1_0.username=?
11:30:15,565 INFO [stdout] (default task-10) encuentre usuario: usr1
11:30:15,565 INFO [stdout] (default task-10) contrasea correcta
11:30:15,565 INFO [stdout] (default task-10) Lista de grupos:[grupo1]
11:30:15,565 INFO [stdout] (default task-10) Tokens restantes: 0
11:30:15,565 INFO [stdout] (default task-10) El servidor no acepta mensajes

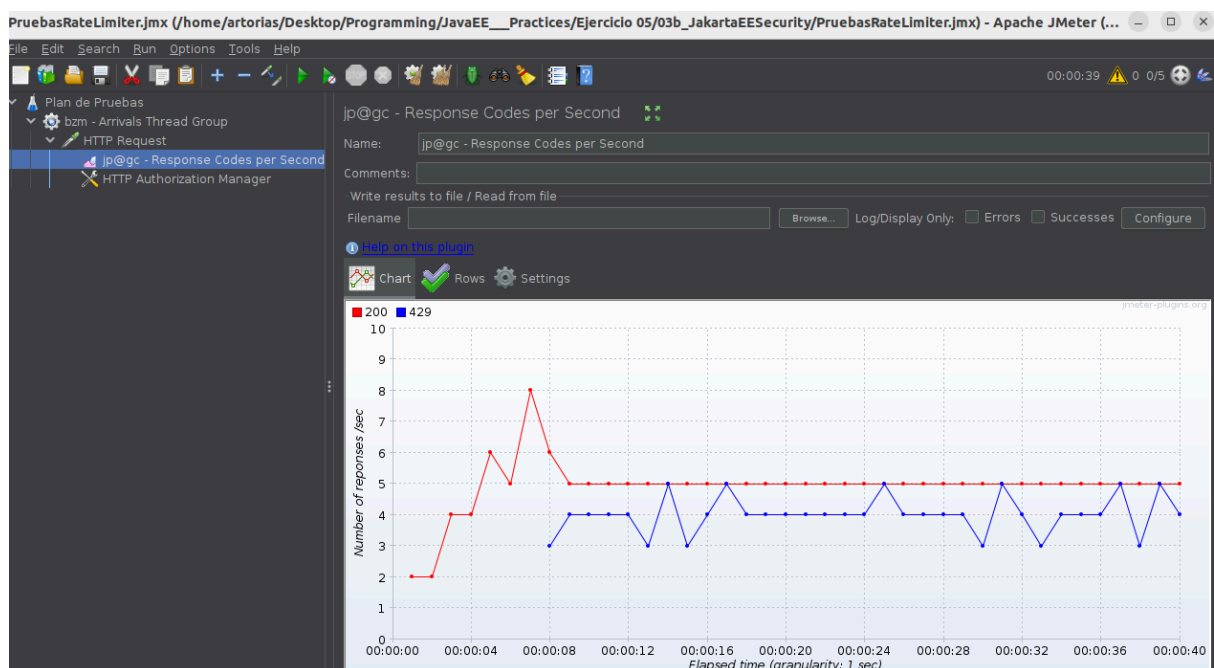
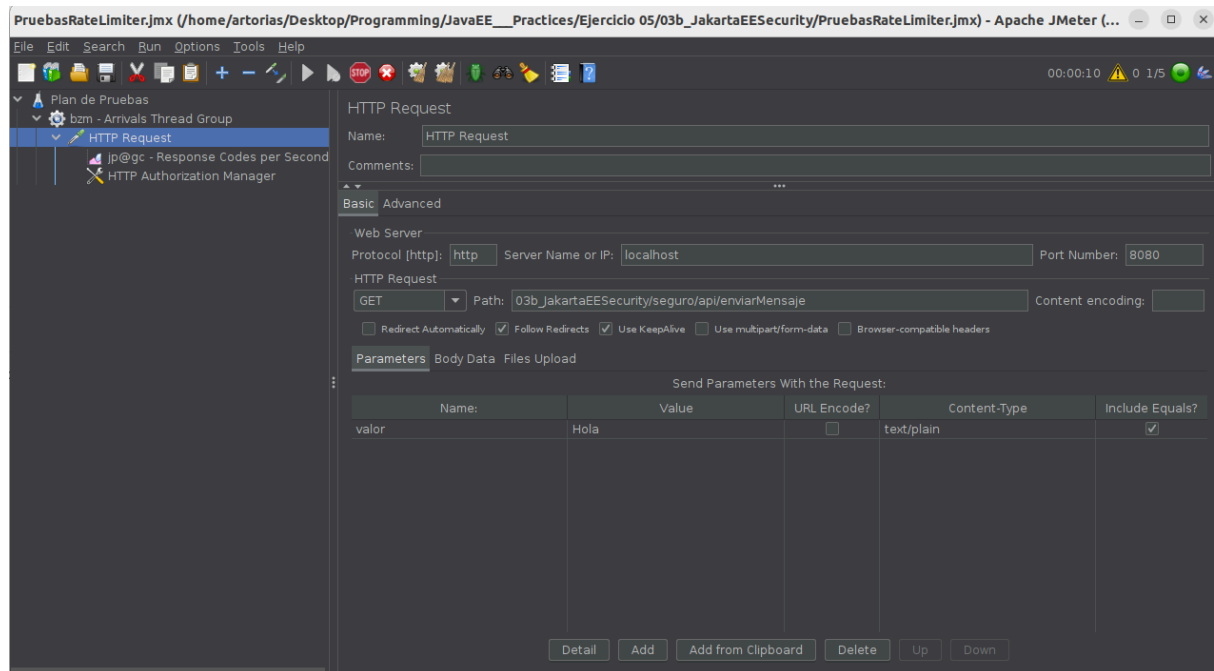
```



Seguidamente. Podemos ir hasta la carpeta server y dentro de server a bin y ejecutar JConsole. Aqui podemos apreciar el uso de recursos como CPU, memory, los threads usados. Mientras



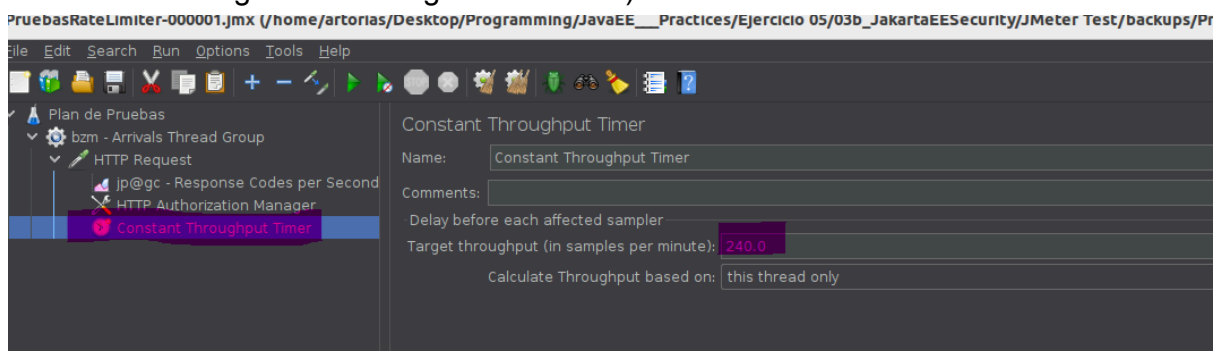
Tambien dentro del RateLimiter ejecutado con el curl que fue proveído



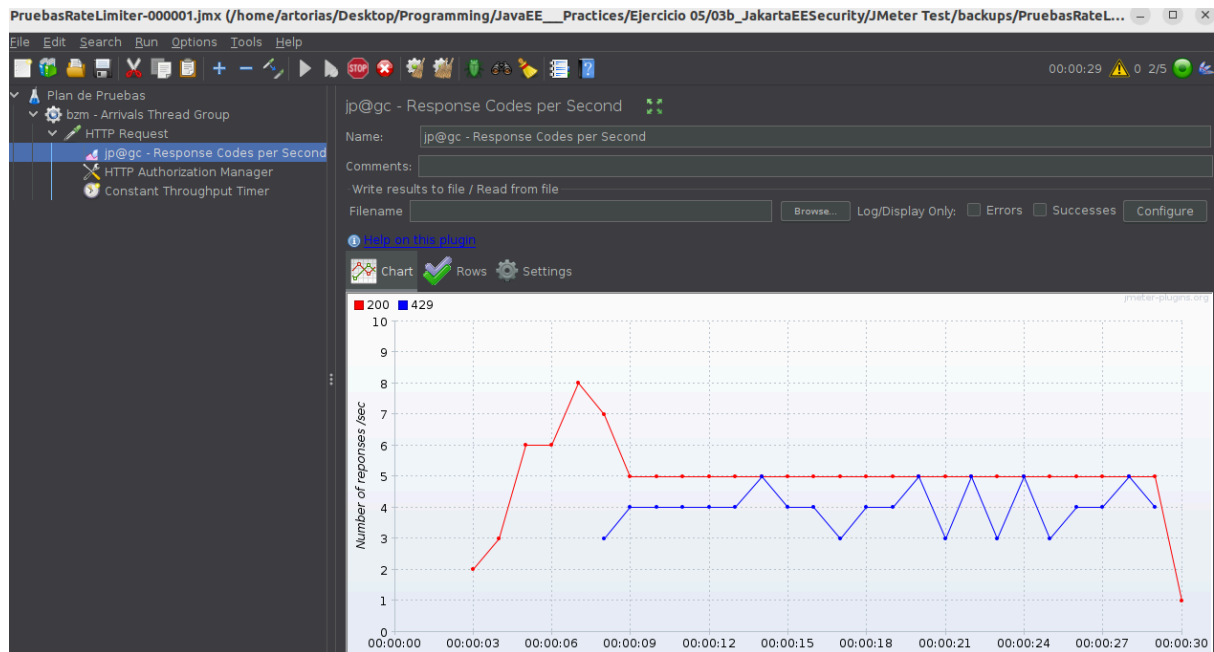
Este es el tipo de RateLimiter que usamos, se puede ver en el código

```
RateLimiter... x HashFunction... UsuarioSiste... Grupo.java ValidadorDeC... 15
14 public class RateLimiter {
15
16     private Bucket bucket;
17     private boolean activo;
18
19     @PostConstruct
20     public void inicializar() {
21         activo = true;
22
23         // el balde tiene un capacidad inicial de 10
24         // cada vez que llega un request, se quita un elemento del balde
25         // si el balde se queda vacío los request serán rechazados
26         // el balde se intentará llenar con 60 tokens en un lapso de 1 minutos
27         // intentando distribuir el llenado en intervalos regulares (un
28         // nuevo token cada 1 segundo)
29
30         //esto se traduce en lo siguiente:
31         //en cualquier momento, el servidor podrá procesar un máximo de 10 transacciones concurrentes
32         //trabajando a su máxima capacidad aceptará una nueva 1 transacción por segundo
33         Bandwidth bucketConf = Bandwidth.builder()
34             .capacity(20)
35             .refillGreedy(60, Duration.ofSeconds(15))
36             .refillIntervally(5, Duration.ofSeconds(1))
37             .build();
38
39         bucket=Bucket.builder().addLimit(bucketConf).build();
40     }
41
42     public boolean consumir() {
43         boolean result = bucket.tryConsume(1);
44         System.out.println("Tokens restantes: " + bucket.getAvailableTokens());
45         return result;
46     }
47
48     public void activarRateLimiter(boolean estado) {
49         System.out.println("RateLimiter estado: " + estado);
50     }
51 }
```

Para el siguiente apartado queremos cambiar la configuración del mismo para que en lugar de procesar dos 2tps sean 4tps. Para esto vamos a agregar una constant throughput timer. Que lo vamos a setear en 240, ya que en 240 (4 transacciones/segundo \* 60 segundos/minuto).



El resultado del mismo es:

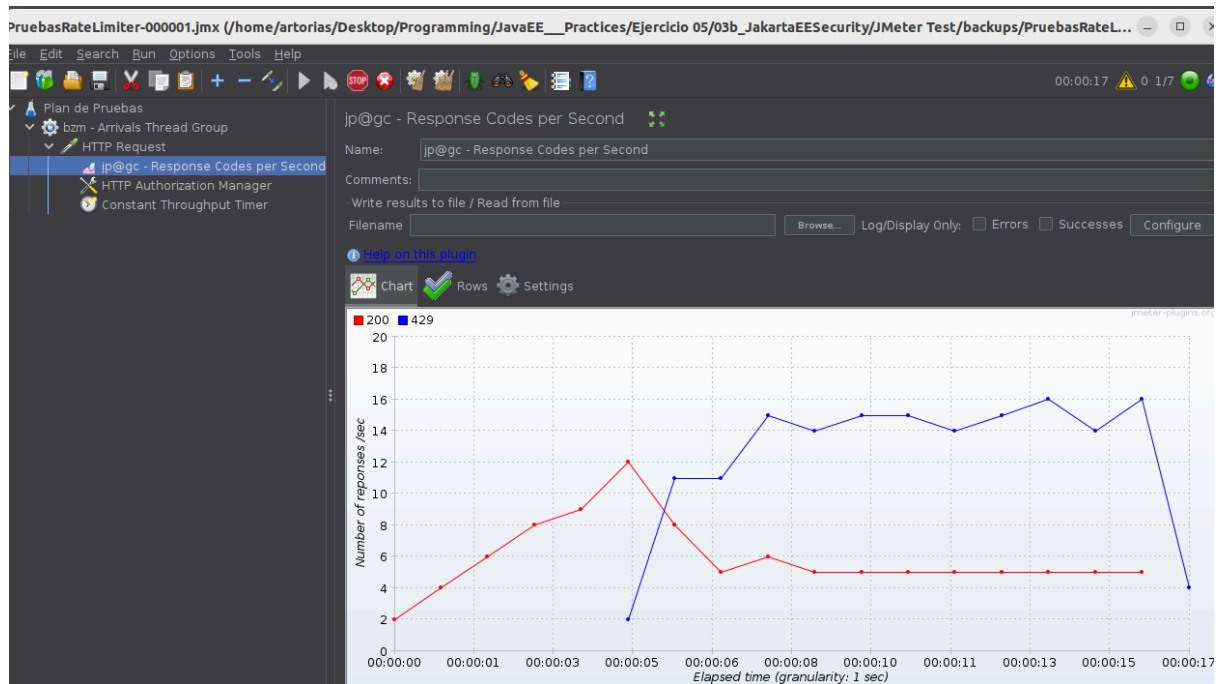


Luego intentaremos aumentar el bucket inicial al doble.

Para esto vamos a nuestro RateLimiter y cambiamos los 10 tokens iniciales, a 20.

```
Ejercicio 05... RateLimiter... RateLimiterC... Red Hat Central
```

```
19 @PostConstruct
20 public void inicializar() {
21     activo = true;
22
23     // el balde tiene un capacidad inicial de 10
24     // cada vez que llega un request, se quita un elemento del balde
25     // si el balde se queda vacío los request serán rechazados
26     // el balde se intentará llenar con 60 tokens en un lapso de 1 minutos
27     // intentando distribuir el llenado en intervalos regulares (un
28     // nuevo token cada 1 segundo)
29
30     //esto se traduce en lo siguiente:
31     //en cualquier momento, el servidor podrá procesar un máximo de 10 transacciones concurrentes
32     //trabajando a su máxima capacidad aceptará una nueva 1 transacción por segundo
33     Bandwidth bucketConf = Bandwidth.builder()
34         .capacity(20)
35         .refillGreedy(60, Duration.ofSeconds(15))
36         .refillIntervally(5, Duration.ofSeconds(1))
37         .build();
38
39     bucket=Bucket.builder().addLimit(bucketConf).build();
40 }
41
42 public boolean consumir() {
43     boolean result = bucket.tryConsume(1);
44     System.out.println("Tokens restantes: " + bucket.getAvailableTokens());
45     return result;
46 }
47
48 public void activarRateLimiter(boolean estado) {
49     System.out.println("RateLimiter estado: " + estado);
50     this.activo = estado;
51 }
52
53 public boolean isActive() {
```



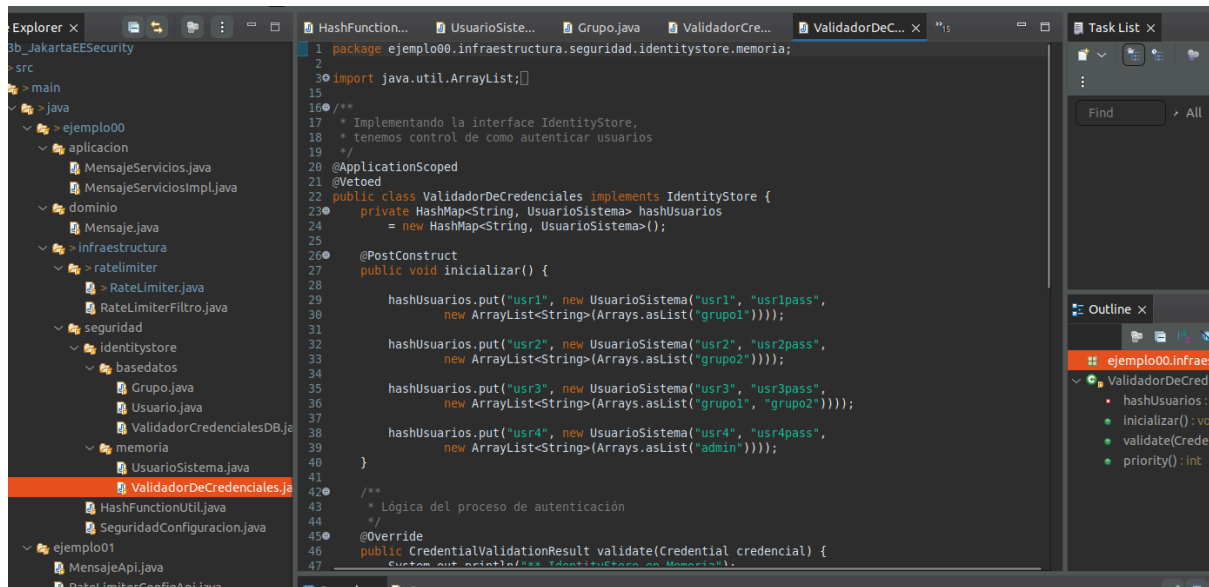
```
root@Artorias: /home/artorias/Desktop/Programming/JavaE...
5:44:40,952 INFO [stdout] (default task-20) Lista de grupos:[grupo1]
5:44:40,953 INFO [stdout] (default task-20) Tokens restantes: 0
5:44:40,953 INFO [stdout] (default task-20) El servidor no acepta mensajes
5:44:40,998 INFO [stdout] (default task-20) ** IdentityStore en base de datos
5:44:40,999 INFO [stdout] (default task-20) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0.username=g1_0.Usuario_username where u1_0.username=?
5:44:40,999 INFO [stdout] (default task-20) encuentre usuario: usr1
5:44:40,999 INFO [stdout] (default task-20) contraseña correcta
5:44:40,999 INFO [stdout] (default task-20) Lista de grupos:[grupo1]
5:44:40,999 INFO [stdout] (default task-20) Tokens restantes: 0
5:44:40,999 INFO [stdout] (default task-20) El servidor no acepta mensajes
5:44:41,047 INFO [stdout] (default task-20) ** IdentityStore en base de datos
5:44:41,047 INFO [stdout] (default task-20) Hibernate: select u1_0.username,u1_0.passwordHash,g1_0.Usuario_username,g1_1.nombre from Usuario u1_0 left join (Usuario_Grupo g1_0 join Grupo g1_1 on g1_1.nombre=g1_0.grupos_nombre) on u1_0.username=g1_0.Usuario_username where u1_0.username=?
5:44:41,048 INFO [stdout] (default task-20) encuentre usuario: usr1
5:44:41,048 INFO [stdout] (default task-20) contraseña correcta
5:44:41,048 INFO [stdout] (default task-20) Lista de grupos:[grupo1]
5:44:41,048 INFO [stdout] (default task-20) Tokens restantes: 0
5:44:41,048 INFO [stdout] (default task-20) El servidor no acepta mensajes
```

También vemos que el comportamiento del lado del cliente es el correcto, si no tiene tokens, no acepta mensajes.

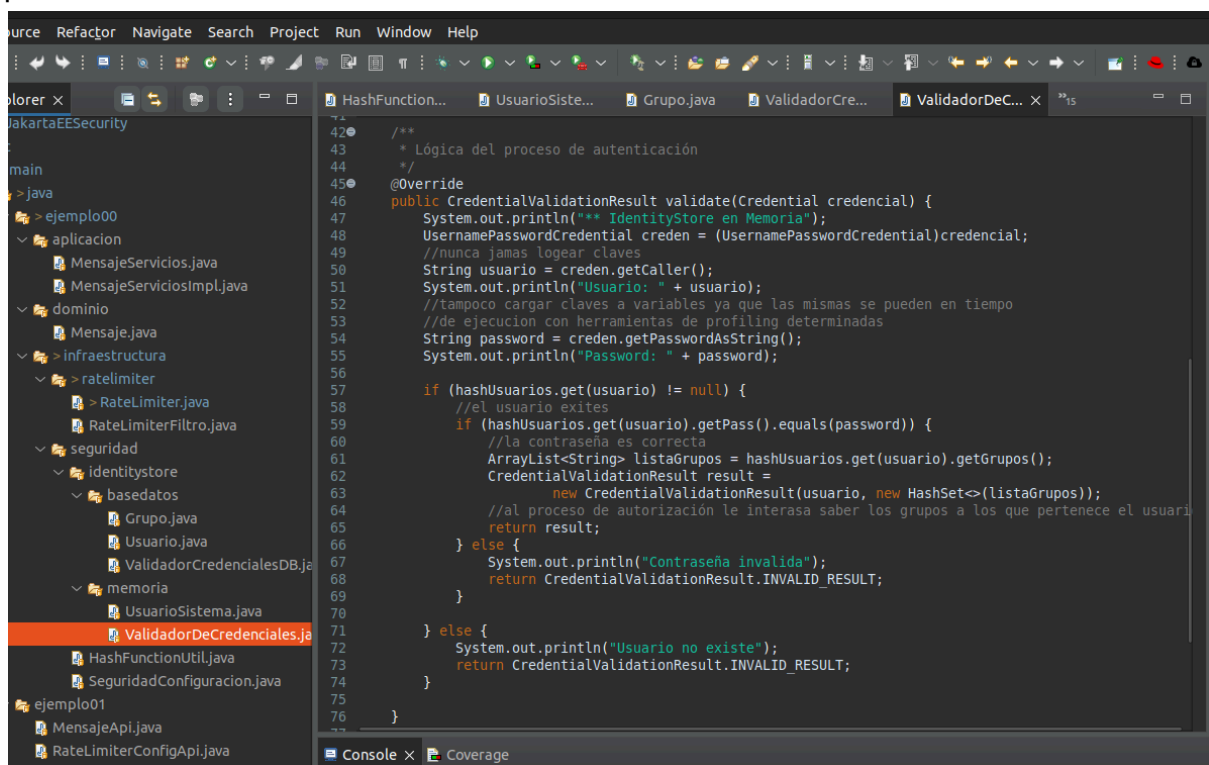
Luego de todo esto, tomaremos una mirada al entity store en memoria.

Como podemos ver en el código, en el entity store, tenemos los usuarios con sus respectivos grupos y sus respectivos chequeos. Como podemos ver en la siguiente

imagen, tenemos al usr1, con su credencial usr1pass y al grupo perteneciente. De la misma manera tenemos a todos los demas.



Y en la siguiente imagen, tenemos el proceso de autenticacion, donde chequea que el usuario sea correcto, luego que su contraseña hasheada sea la esperada y al grupo que pertenece.



Para que el usr1 pueda ejecutar tambien el endpoint del rate limiter, debe poder formar parte del grupo admin, como por ejemplo el usr4.

Para que el usr5 pueda acceder a los cambios de configuración, debemos darle acceso al usr5 el rol de admin, como al usr1, pero a su vez, debemos darle acceso de management role en nuestro servidor (cuando ejecutamos add-user.sh).



```
gate Search Project Run Window Help
HashFunction... UsuarioSiste... Grupo.java ValidadorCre... ValidadorDeCred... x 15
20 @ApplicationScoped
21 @Vetoed
22 public class ValidadorDeCredenciales implements IdentityStore {
23     private HashMap<String, UsuarioSistema> hashUsuarios
24         = new HashMap<String, UsuarioSistema>();
25
26     @PostConstruct
27     public void inicializar() {
28
29         hashUsuarios.put("usr1", new UsuarioSistema("usr1", "usr1pass",
30             new ArrayList<String>(Arrays.asList("grupo1"))));
31
32         hashUsuarios.put("usr2", new UsuarioSistema("usr2", "usr2pass",
33             new ArrayList<String>(Arrays.asList("grupo2"))));
34
35         hashUsuarios.put("usr3", new UsuarioSistema("usr3", "usr3pass",
36             new ArrayList<String>(Arrays.asList("grupo1", "grupo2"))));
37
38         hashUsuarios.put("usr4", new UsuarioSistema("usr4", "usr4pass",
39             new ArrayList<String>(Arrays.asList("admin"))));
40
41         hashUsuarios.put("usr4", new UsuarioSistema("usr5", "usr5pass",
42             new ArrayList<String>(Arrays.asList("admin"))));
43     }
44
45     /**
46      * Lógica del proceso de autenticación
47      */
48     @Override
49     public CredentialValidationResult validate(Credential credencial) {
50         System.out.println("** IdentityStore en Memoria");
51         UsernamePasswordCredential creden = (UsernamePasswordCredential)credencial;
52         //nunca jamas logear claves
53         String usuario = creden.getCaller();
54         System.out.println("Usuario: " + usuario);
55     }
56 }
```

```
2
3 import java.util.ArrayList;
4
5
6 /**
7  * Implementando la interface IdentityStore,
8  * tenemos control de como autenticar usuarios
9  */
10 @ApplicationScoped
11 @Vetoed
12 public class ValidadorDeCredenciales implements IdentityStore {
13     private HashMap<String, UsuarioSistema> hashUsuarios
14         = new HashMap<String, UsuarioSistema>();
15
16     @PostConstruct
17     public void inicializar() {
18
19         hashUsuarios.put("usr1", new UsuarioSistema("usr1", "usr1pass",
20             new ArrayList<String>(Arrays.asList("grupo1", "admin"))));
21
22         hashUsuarios.put("usr2", new UsuarioSistema("usr2", "usr2pass",
23             new ArrayList<String>(Arrays.asList("grupo2"))));
24
25         hashUsuarios.put("usr3", new UsuarioSistema("usr3", "usr3pass",
26             new ArrayList<String>(Arrays.asList("grupo1", "grupo2"))));
27
28         hashUsuarios.put("usr4", new UsuarioSistema("usr4", "usr4pass",
29             new ArrayList<String>(Arrays.asList("admin"))));
30
31         hashUsuarios.put("usr5", new UsuarioSistema("usr5", "usr5pass",
32             new ArrayList<String>(Arrays.asList("admin"))));
33     }
34
35     /**
36      * Lógica del proceso de autenticación
37      */
38 }
```

Si tratamos de darle al endpoint de esta manera, debería funcionar correctamente.

```
curl --user usr?:usr?pass -v
```

```
http://localhost:8080/03b\_JakartaEESecurity/seguro/config/activarRateLimiter?valor=false
```

Todo el código visto se puede encontrar en:

[https://github.com/JuanmaPilon/JavaEE\\_\\_\\_Practices](https://github.com/JuanmaPilon/JavaEE___Practices)