

RESIFY

Testing de
rendimiento

SPRINT 4

TESTING DE RENDIMIENTO

A continuación, mostraremos para cada Historia de Usuario, los reportes obtenidos con Gatling para la medición del rendimiento de las Historias de Usuario de nuestra aplicación.

También añadiremos otros pantallazos según sea conveniente, como es el caso de la consola y del rendimiento de la CPU en los casos donde se produzca mayor cuello de botella en nuestros equipos.

Cabe mencionar que cada componente ha realizado las pruebas de rendimiento de las Historias de Usuario que ha implementado, y como cada uno ha implementado 5, cada uno habrá realizado un total de 5 pruebas de rendimiento, siendo así la suma total 20 pruebas de rendimiento, que es lo que nos piden en el entregable, ya que es el total de las historias de usuario de nuestra aplicación.

También es importante destacar que cada miembro ha implementado las pruebas de rendimiento en su propio equipo, y cada equipo difiere de otro por su capacidad y por su potencia, luego especificaremos qué pruebas de rendimiento ha hecho cada componente y sobre qué equipo las ha realizado:

Miembro	Id HU probadas	CPU del equipo
Alfonso Rodríguez	01, 02, 03, 04 y 17	Intel® Core™ i5-4200U CPU @ 1.60GHz
Yassin Lalj	05, 06, 13, 15 y 18	Intel® Core™ i7-4510U CPU @ 2.00GHz
Javier Navarro	07, 11, 12, 14 y 19	Intel® Core™ i5-7200U CPU @ 2.50GHz
Jesús Gámez	08, 09, 10, 16 y 20	Intel® Core™ i7-3630QM CPU @ 2.40GHz

Test de Rendimiento de la HU1

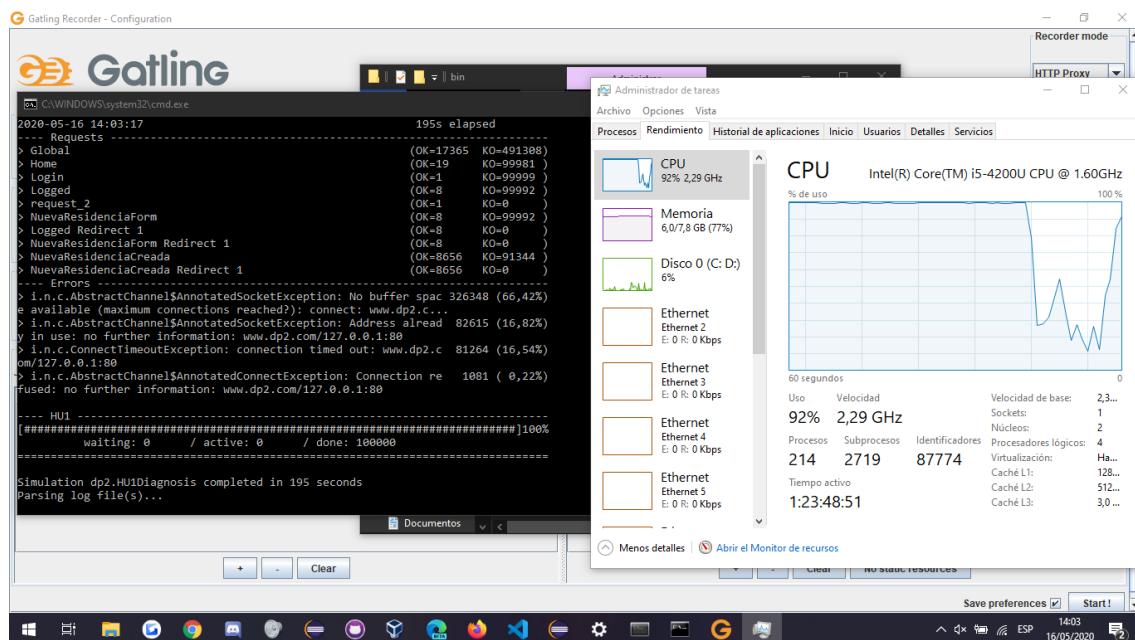
Analizando la Historia de Usuario 1 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(
    scn.inject(rampUsers(100000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:

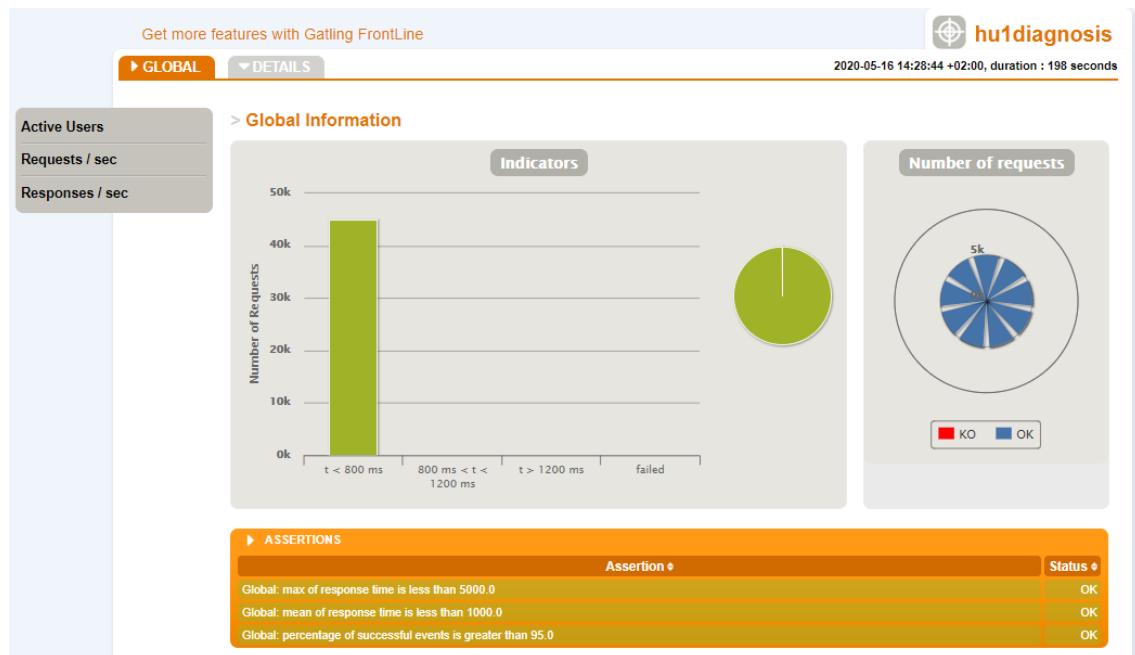


2)

En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(
    scn.inject(rampUsers(5000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95)
    )
```

El sistema funciona correctamente:



Test de Rendimiento de la HU2

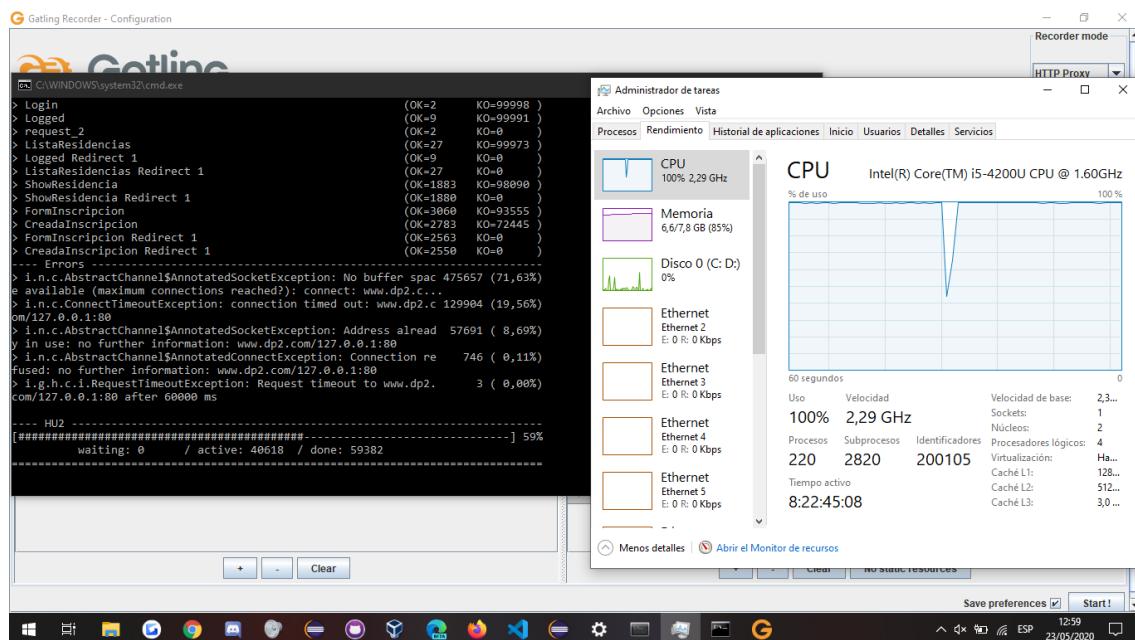
Analizando la Historia de Usuario 2 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(
    scn.inject(rampUsers(100000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(
    scn.inject(rampUsers(5000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95)
    )
)
```

El sistema funciona correctamente:



Test de Rendimiento de la HU3

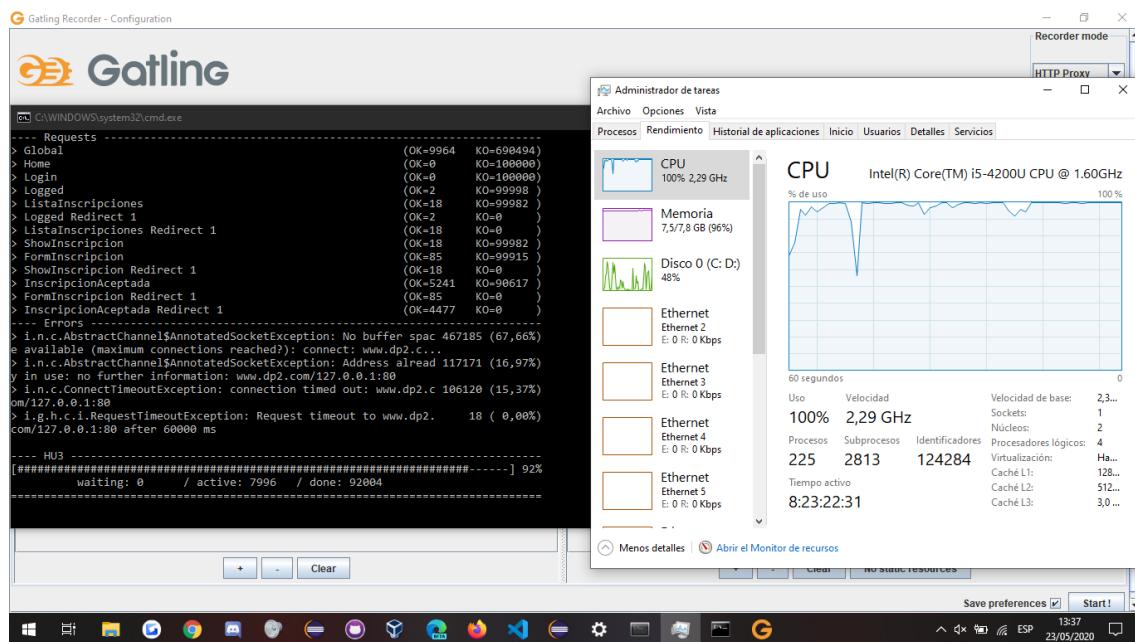
Analizando la Historia de Usuario 3 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(
    scn.inject(rampUsers(100000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(
    scn.inject(rampUsers(5000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95)
    )
)
```

El sistema funciona correctamente:



Test de Rendimiento de la HU4

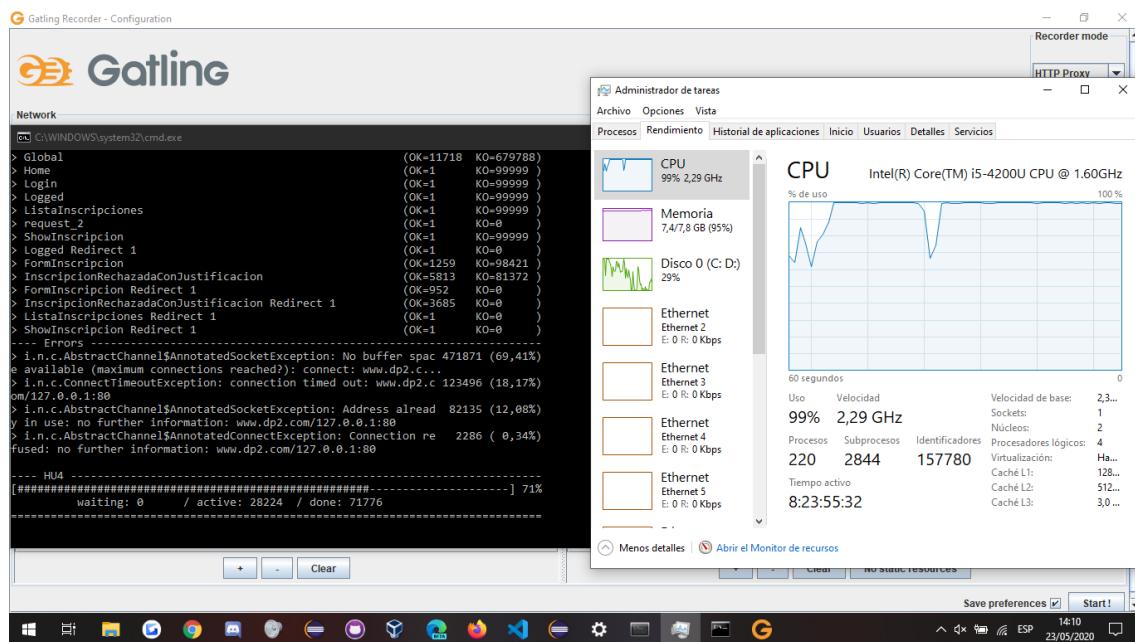
Analizando la Historia de Usuario 4 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(
    scn.inject(rampUsers(100000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(
    scn.inject(rampUsers(5000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95)
    )
```

El sistema funciona correctamente:



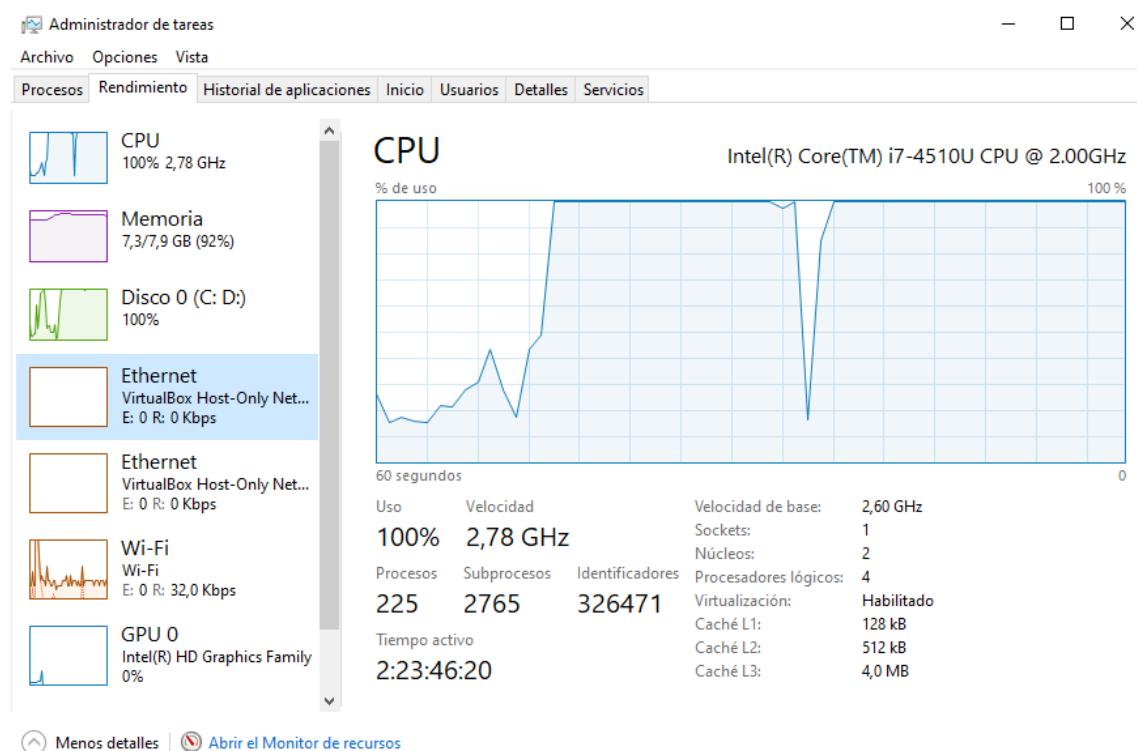
Test de Rendimiento de la HU5

Analizando la Historia de Usuario 5 con Gatling obtenemos los siguientes resultados:

- 1) Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(HU05 scn.inject(rampUsers(100000) during (10 seconds))  
).protocols(httpProtocol)
```

El sistema no funciona correctamente:



```
---- Requests ----
> Global                                     (OK=9485   KO=495231)
> Home                                       (OK=53     KO=99947  )
> Login                                      (OK=53     KO=99947  )
> Logged                                     (OK=53     KO=99947  )
> NuevaBuenaAccionForm                      (OK=53     KO=99947  )
> NuevaBuenaAccionCreada                    (OK=4557   KO=95443  )
> NuevaBuenaAccionCreada Redirect 1          (OK=4557   KO=0      )
> request_2                                  (OK=53     KO=0      )
> Logged Redirect 1                         (OK=53     KO=0      )
> NuevaBuenaAccionForm Redirect 1            (OK=53     KO=0      )
---- Errors -----
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer spac 338215 (68,29%)
e available (maximum connections reached?): connect: www.dp2.c...
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.c 71346 (14,41%)
om/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedSocketException: Address alread 55957 (11,30%)
y in use: no further information: www.dp2.com/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedConnectException: Connection re 29713 ( 6,00%)
fused: no further information: www.dp2.com/127.0.0.1:80

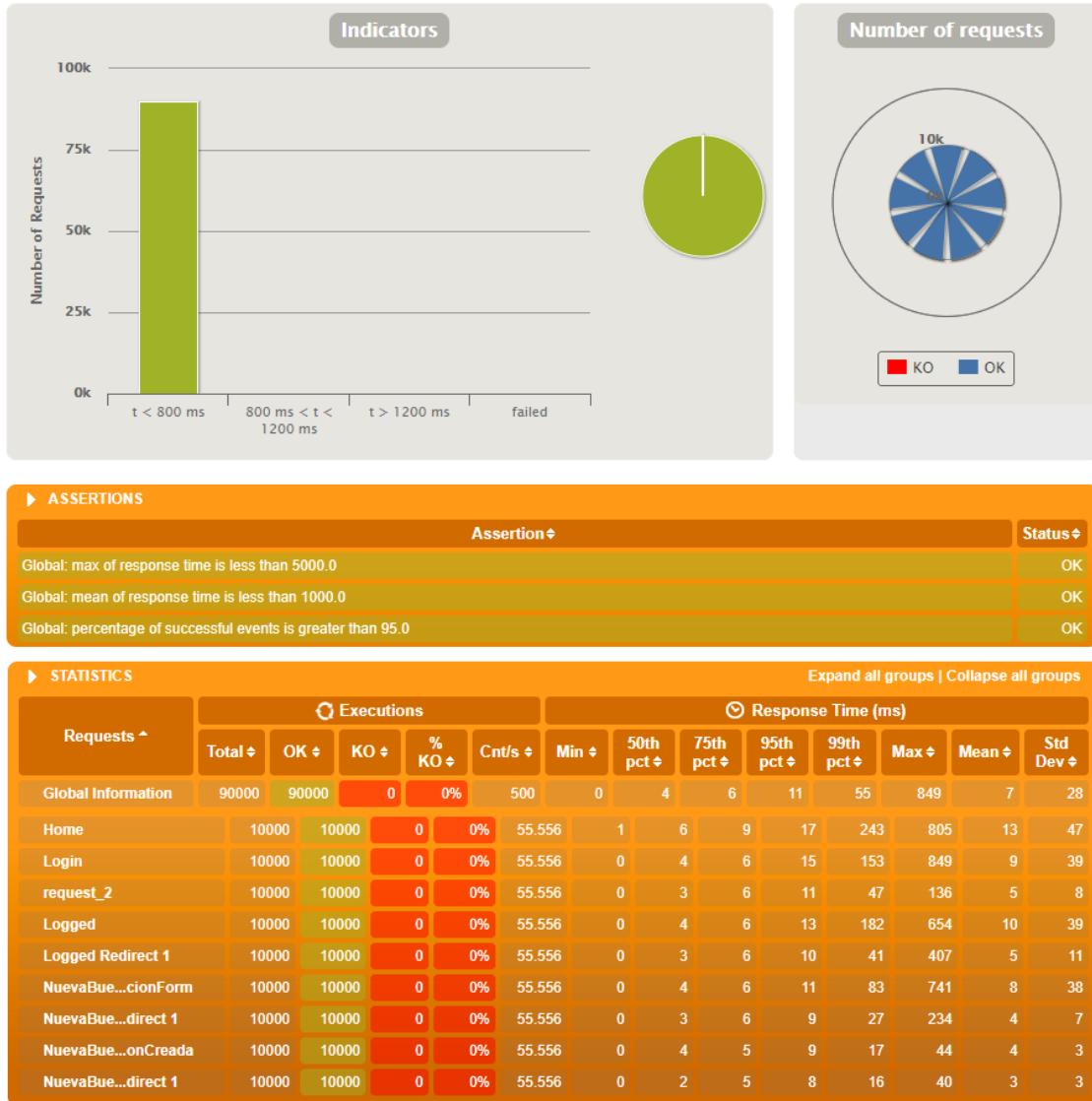
---- HU05 -----
[########################################]100%
      waiting: 0      / active: 0      / done: 100000
=====

Simulation dp2.HU05Diagnosis completed in 194 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
```

- 2) En cambio, con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU05scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente:



3) Probamos a estresar un poco más el sistema aumentando 5 mil usuarios:

```
setUp(HU05scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

Vemos que no responde correctamente pues hay varias peticiones hechas al servidor que terminan en KO, luego estimamos que el número de usuarios aquí establece el cuello de botella en el sistema.



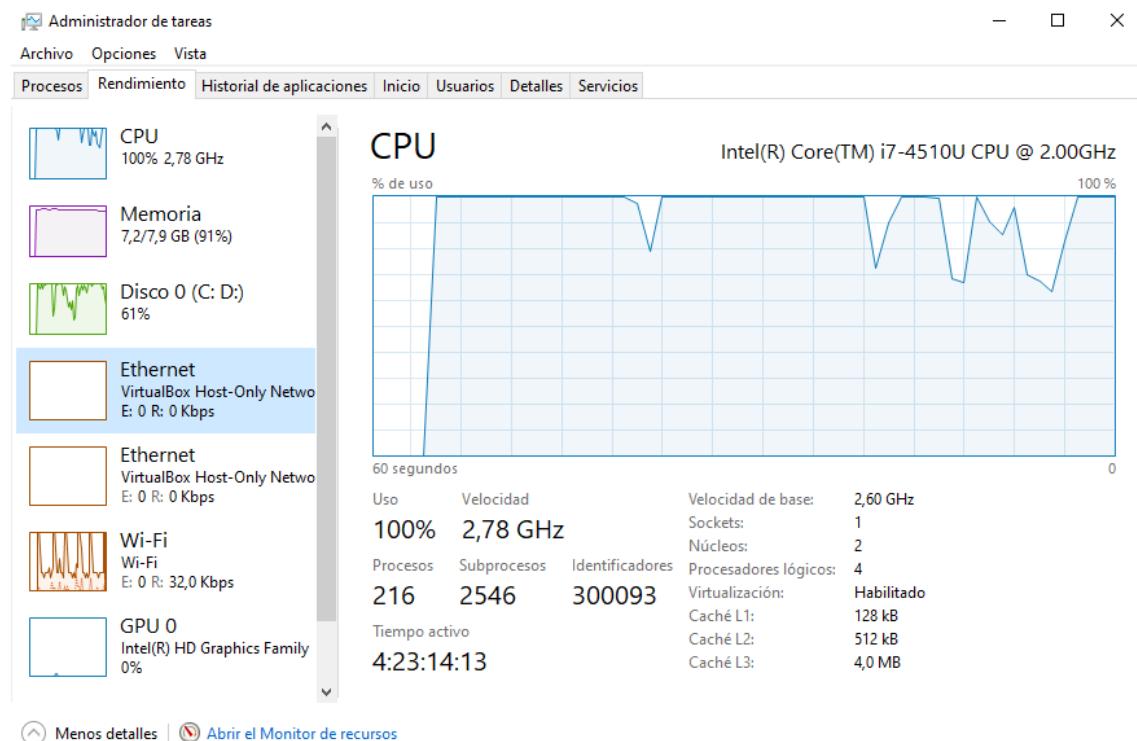
Test de Rendimiento de la HU6

Analizando la Historia de Usuario 6 con Gatling obtenemos los siguientes resultados:

- 1) Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(HU06scn.inject(rampUsers(100000) during (10 seconds))
).protocols(httpProtocol)
```

El sistema no funciona correctamente:



```
---- Requests ----
> Global                                     (OK=1989  KO=499004)
> Home                                       (OK=3    KO=99997  )
> Login                                      (OK=1    KO=99999  )
> Logged                                     (OK=3    KO=99997  )
> request_2                                  (OK=1    KO=0    )
> NuevaIncidenciaForm                        (OK=6    KO=99994  )
> Logged Redirect 1                          (OK=3    KO=0    )
> NuevaIncidenciaForm Redirect 1            (OK=6    KO=0    )
> NuevaIncidenciaCreada                      (OK=983   KO=99017  )
> NuevaIncidenciaCreada Redirect 1          (OK=983   KO=0    )
---- Errors -----
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer spac 329540 (66,04%)
e available (maximum connections reached?): connect: www.dp2.c...
> i.n.c.AbstractChannel$AnnotatedSocketException: Address already 99746 (19,99%)
y in use: no further information: www.dp2.com/127.0.0.1:80
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.c 69718 (13,97%)
om/127.0.0.1:80

---- HU06 -----
[########################################################################]100%
      waiting: 0      / active: 0      / done: 100000
=====
Simulation dp2.HU06 completed in 165 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
```

- 2) En cambio, con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU06scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente:

> Global Information



> ASSERTIONS

Assertion	Status
Global: max of response time is less than 5000.0	OK
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

> STATISTICS

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	90000	90000	0	0%	552.147	0	5	7	69	341	1119	18	63
Home	10000	10000	0	0%	61.35	2	8	11	122	525	1118	29	89
Login	10000	10000	0	0%	61.35	0	5	8	97	442	902	23	74
request_2	10000	10000	0	0%	61.35	0	4	7	56	214	820	13	42
Logged	10000	10000	0	0%	61.35	0	5	8	65	449	1119	22	80
Logged Redirect 1	10000	10000	0	0%	61.35	0	4	7	36	242	908	12	42
NuevaIncidenciaForm	10000	10000	0	0%	61.35	1	5	7	65	303	942	17	58
NuevaInc...direct 1	10000	10000	0	0%	61.35	0	4	6	24	185	455	10	33
NuevaInc...iaCreada	10000	10000	0	0%	61.35	0	4	6	84	482	1006	21	81
NuevaInc...direct 1	10000	10000	0	0%	61.35	0	3	6	34	210	448	11	38

3) Probamos a estresar un poco más el sistema aumentando 5 mil usuarios:

```
setUp(HU06scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

Vemos que no responde correctamente pues hay varias peticiones hechas al servidor que terminan en KO, luego estimamos que el número de usuarios aquí establece el cuello de botella en el sistema.

> Global Information



> ASSERTIONS

Assertion♦	Status♦
Global: max of response time is less than 5000.0	KO
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

> STATISTICS

Requests^	Executions						Response Time (ms)								
	Total ♦	OK ♦	KO ♦	% KO ♦	Cnt/s ♦	Min ♦	50th pct ♦	75th pct ♦	95th pct ♦	99th pct ♦	Max ♦	Mean ♦	Std Dev ♦		
Global Information	133208	128261	4947	4%	797.653	0	16	130	2133	2815	8529	258	660		
Home	15000	11845	3155	21%	89.82	1	267	2171	2922	5263	8529	1051	1286		
Login	15000	13792	1208	8%	89.82	0	41	311	2380	3180	5413	473	869		
request_2	13792	13792	0	0%	82.587	0	10	51	231	487	1299	49	100		
Logged	15000	14571	429	3%	89.82	1	48	170	1716	2553	4785	249	564		
Logged Redirect 1	14571	14571	0	0%	87.251	0	10	79	325	607	2040	73	168		
NuevaIncidenciaForm	15000	14845	155	1%	89.82	1	27	147	671	2227	3539	161	377		
NuevaInci...direct 1	14845	14845	0	0%	88.892	0	9	76	387	690	2860	80	180		
NuevaInci...iaCreada	15000	15000	0	0%	89.82	0	10	109	378	1343	3557	100	226		
NuevaInci...direct 1	15000	15000	0	0%	89.82	0	7	65	301	613	2030	65	166		

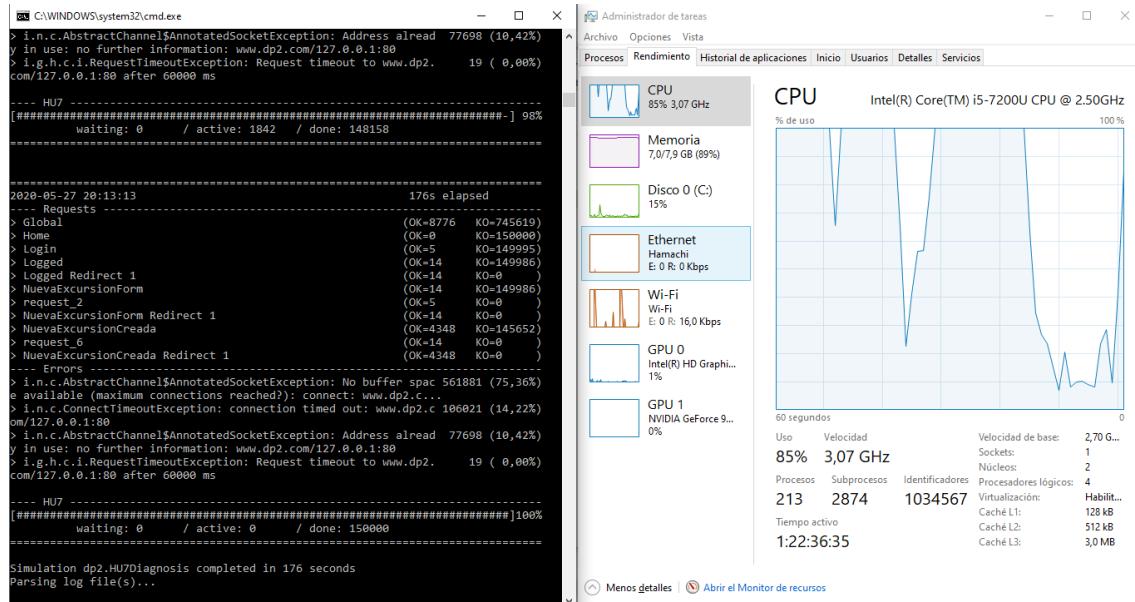
Test de Rendimiento de la HU7

Analizando la Historia de Usuario 7 con Gatling obtenemos los siguientes resultados:

1) Con 150000 usuarios concurrentes durante 10 segundos:

```
setUp(
    HU7 scn.inject(rampUsers(150000) during (10 seconds))
).protocols(httpProtocol)
```

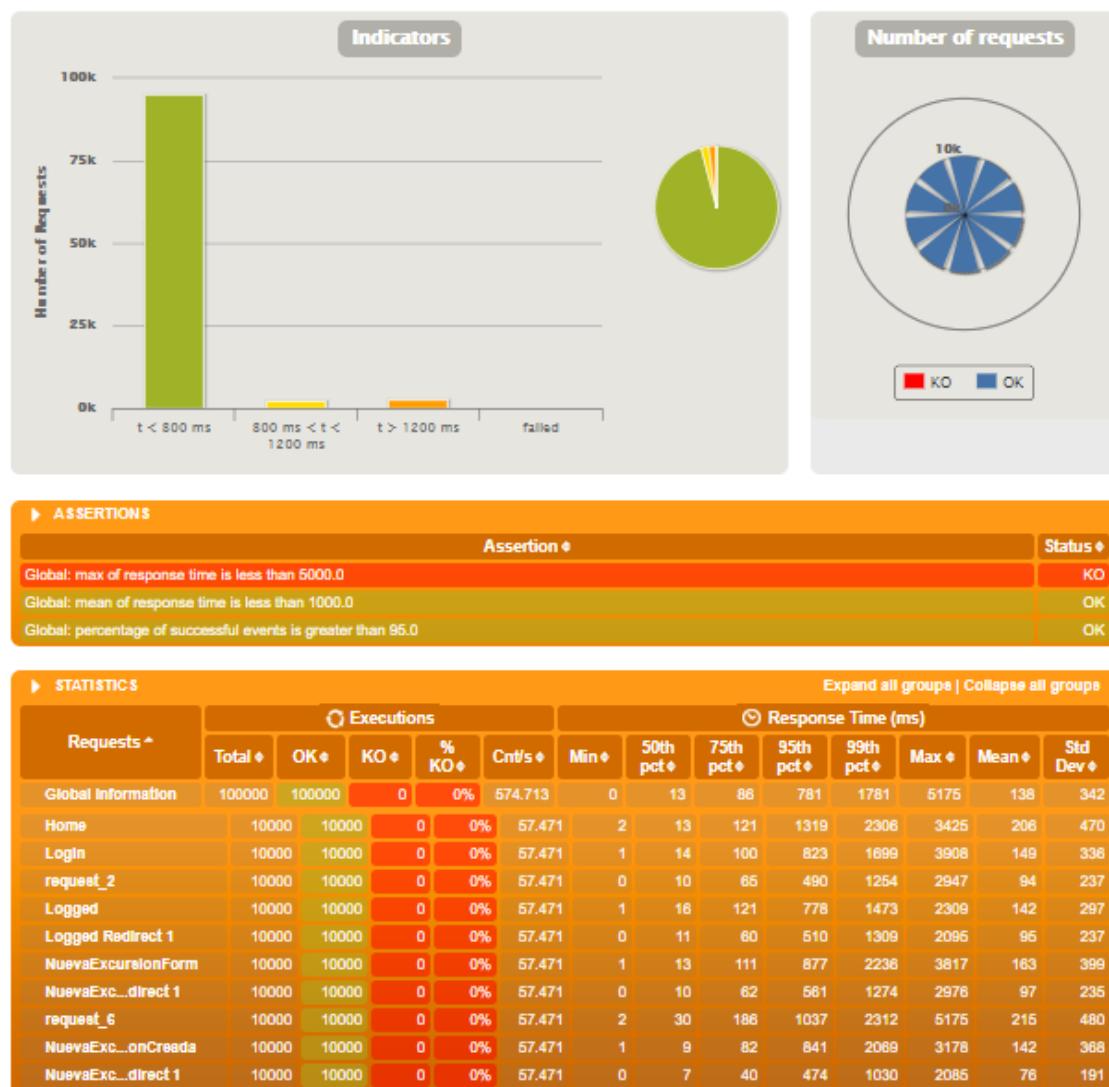
Vemos que el sistema no funciona correctamente, aparecen demasiadas peticiones KO:



2) Probamos con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, uno medio de 1 segundo y un porcentaje de éxito de 95%:

```

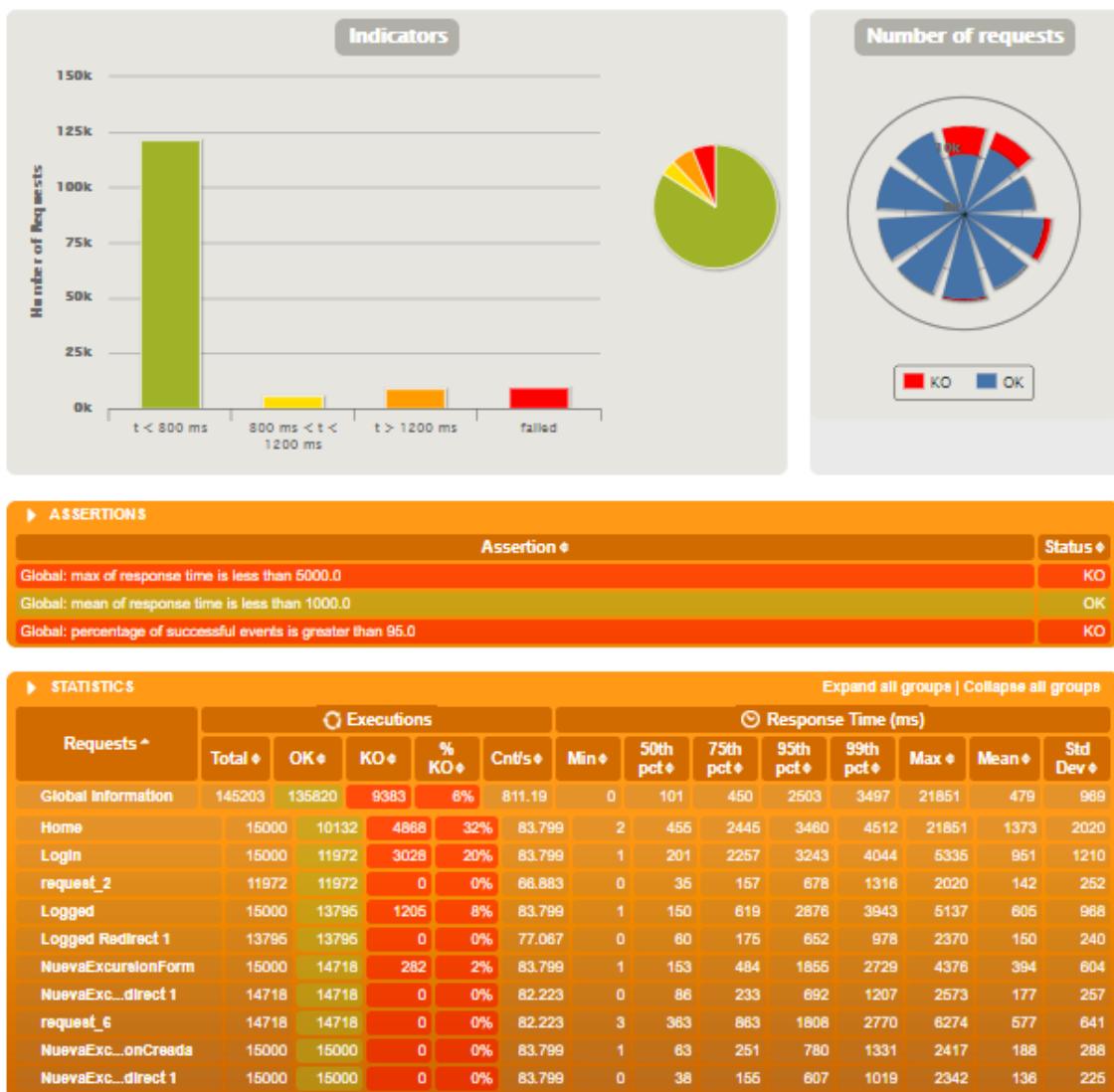
setUp(
    HU7scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
    
```



Observamos que todas las peticiones son OK y, en su gran mayoría, con un buen tiempo de respuesta.

Intentamos aumentar un poco el número de usuarios para averiguar el máximo número de concurrentes, con 15000 usuarios y mismos ajustes de respuesta:

```
setUp(
    HU7scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que ya hay un número notable de peticiones KO, y el tiempo de respuesta también ha aumentado ligeramente, así que concluimos que el máximo número de usuarios concurrentes es 10000.

Test de Rendimiento de la HU8

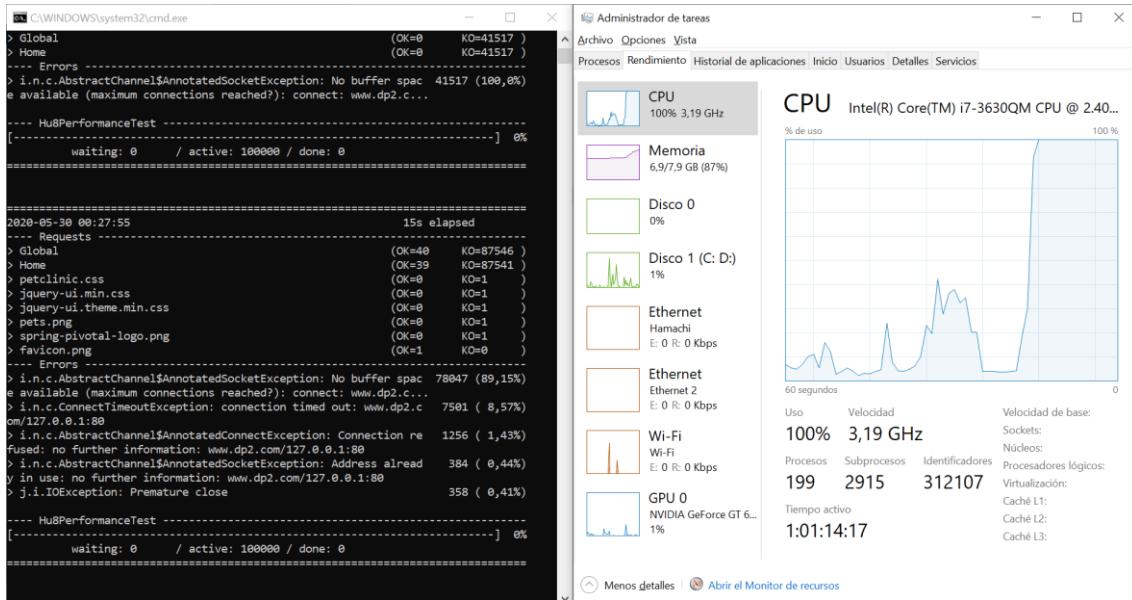
Analizando la Historia de Usuario 8 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp scn.inject(rampUsers(100000) during (10 seconds))
    .protocols(httpProtocol)
}
```

El sistema no funciona correctamente:



```
Simulation dp2PerformanceTests.Hu8PerformanceTest completed in 392 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

=====
---- Global Information -----
> request count                                856762 (OK=110720 KO=746042)
> min response time                            0 (OK=0      KO=0      )
> max response time                            61995 (OK=60587 KO=61995 )
> mean response time                           4048 (OK=4740   KO=3945   )
> std deviation                               5500 (OK=8554   KO=4879   )
> response time 50th percentile                2136 (OK=37     KO=2295   )
> response time 75th percentile                5171 (OK=6209   KO=5136   )
> response time 95th percentile                15655 (OK=24220  KO=14735  )
> response time 99th percentile                24004 (OK=34234  KO=20242  )
> mean requests/sec                          2180.056 (OK=281.73 KO=1898.326)

---- Response Time Distribution -----
> t < 800 ms                                  71274 ( 8%)
> 800 ms < t < 1200 ms                      1857 ( 0%)
> t > 1200 ms                                37589 ( 4%)
> failed                                     746042 ( 87%)
```

2)

Ahora probemos con 100 s y el mismo número de usuarios

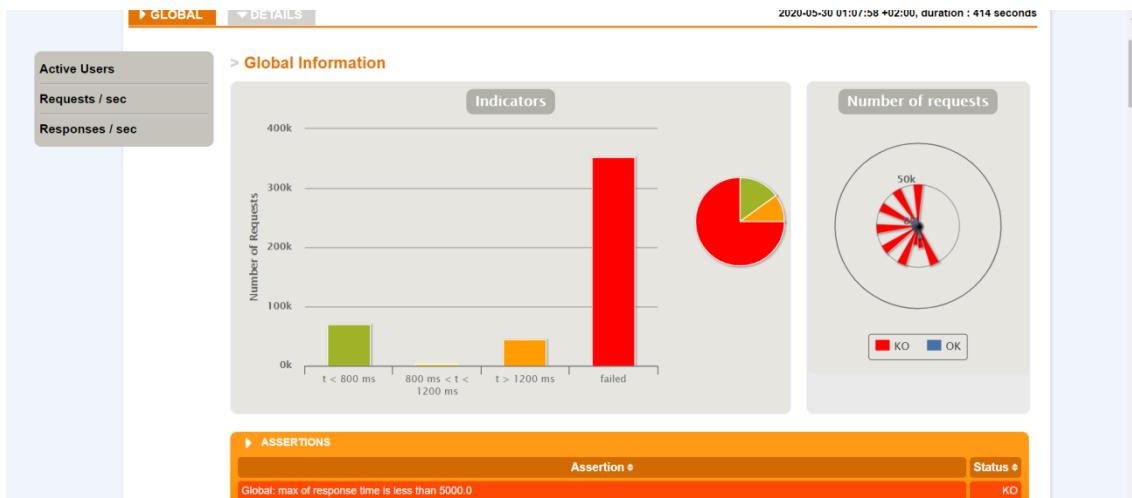


El sistema sigue sin funcionar correctamente.

3)

Veamos con 50000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

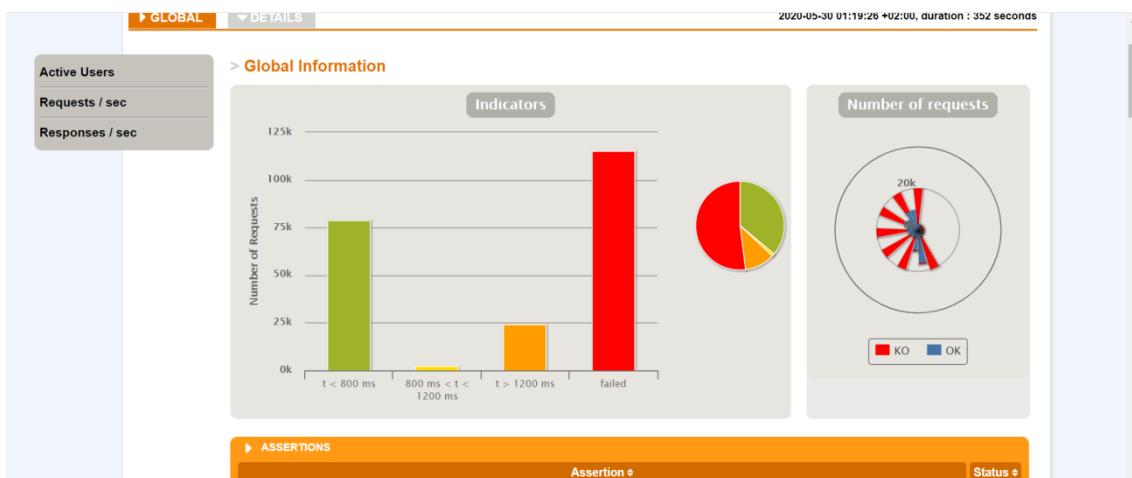
```
setUp scn.inject(rampUsers(100000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95),
```



Sigue habiendo muchos errores, aunque no tantos como antes.

4)

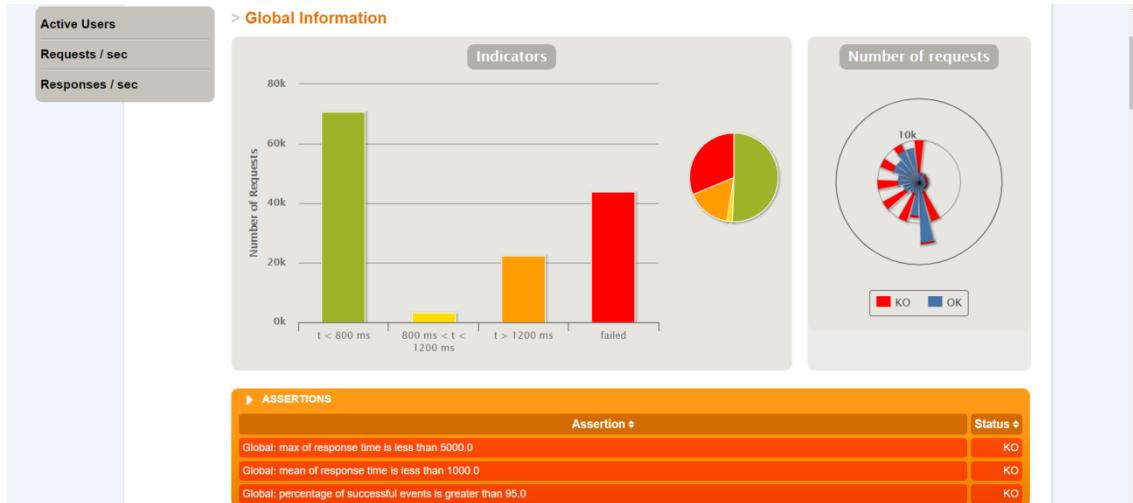
Probemos con 20000 usuarios.



Sigue habiendo un 50% de KOs aprox.

5)

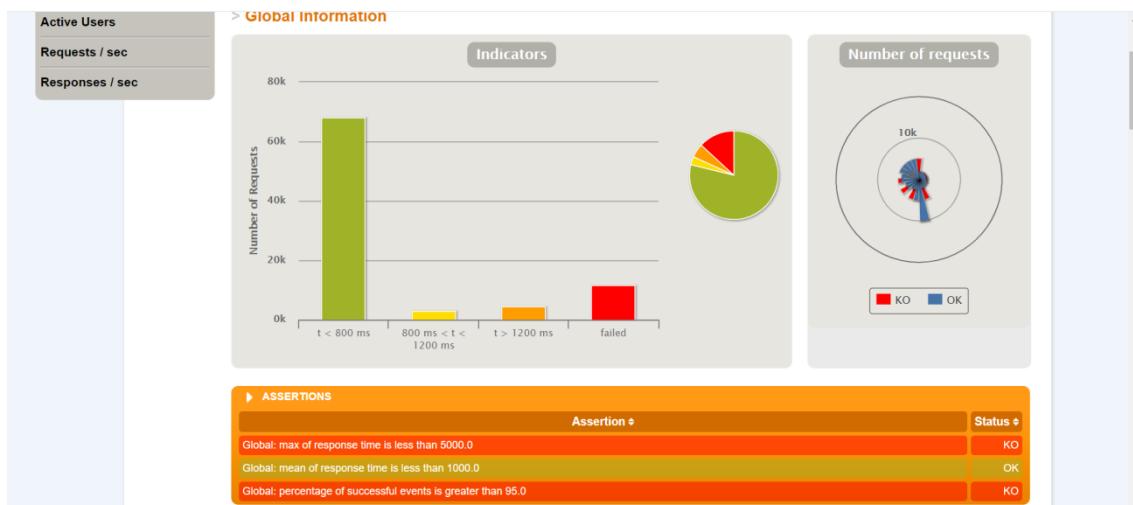
Probemos con 10000 usuarios.



Sigue sin cumplir los requisitos.

6)

Probemos con 5000 usuarios.



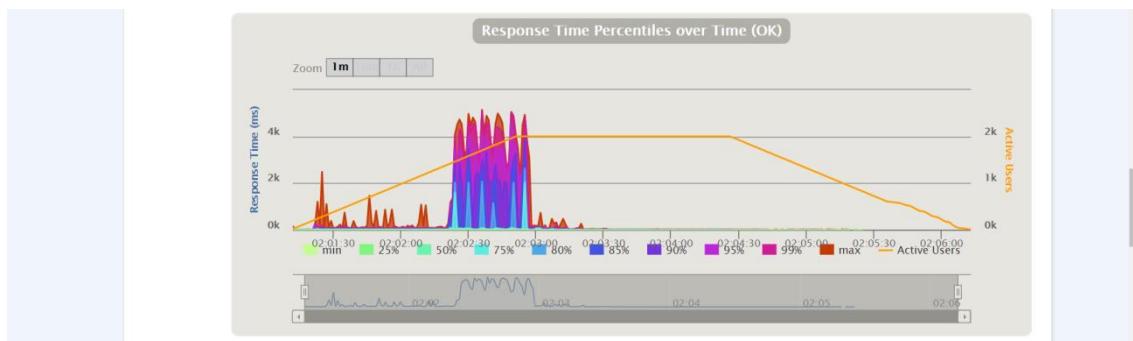
Casi conseguimos el objetivo, pero hace falta bajar más el número de usuarios.

7)

Probemos con 2000 usuarios.



Podemos ver que no cumple el tiempo max de respuesta de 5 s:



Pero al ser superado por tan poco se da por buena la prueba.

Test de Rendimiento de la HU9

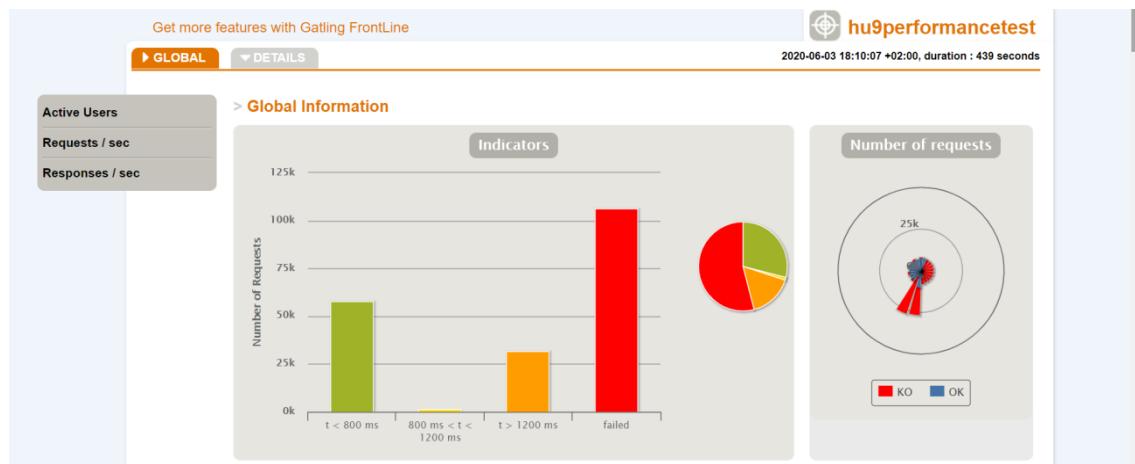
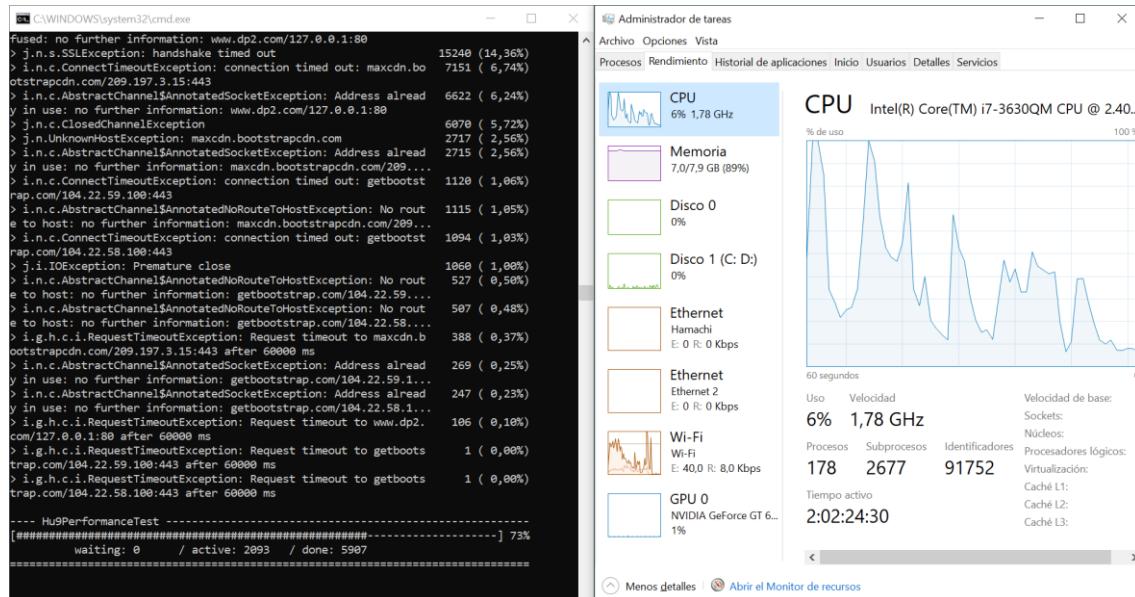
Analizando la Historia de Usuario 9 con Gatling obtenemos los siguientes resultados:

1)

Con 8000 usuarios concurrentes durante 10 segundos:

```
setUp scn.inject(rampUsers(8000) during (10 seconds))
      .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

Intentemos con 3000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%



Como vemos, no cumple el tiempo de respuesta

3)

Probemos con 1000 usuarios



Funciona correctamente

Test de Rendimiento de la HU10

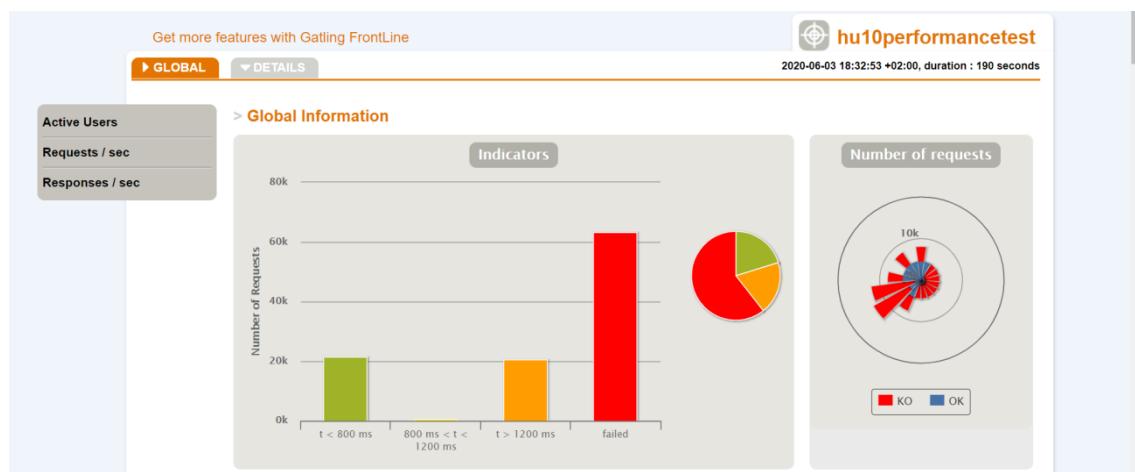
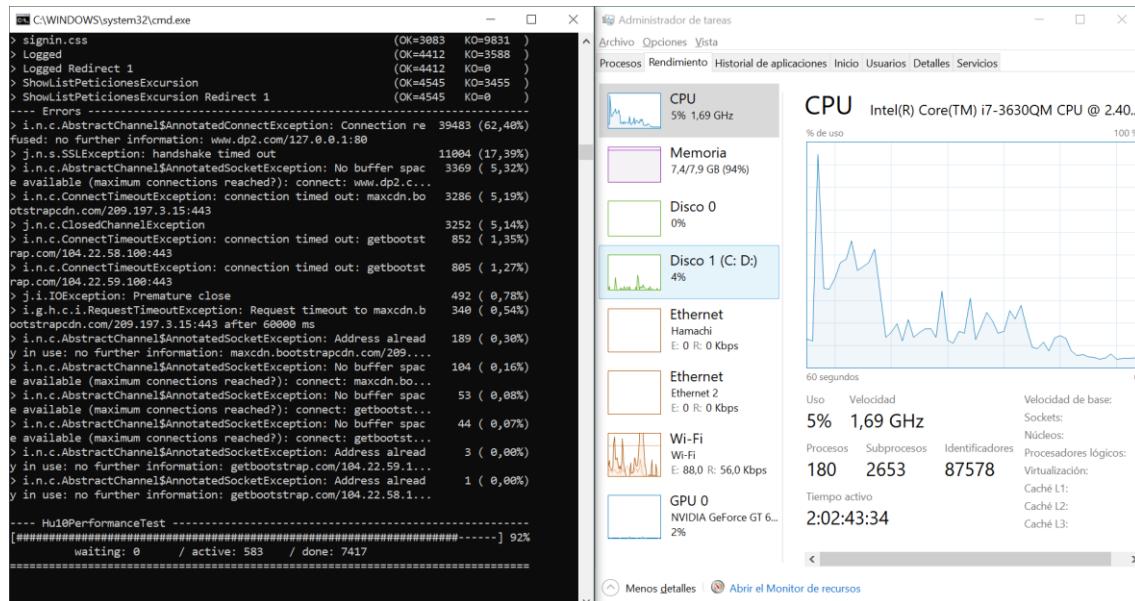
Analizando la Historia de Usuario 10 con Gatling obtenemos los siguientes resultados:

1)

Con 8000 usuarios concurrentes durante 10 segundos:

```
setUp scn.inject(rampUsers(8000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

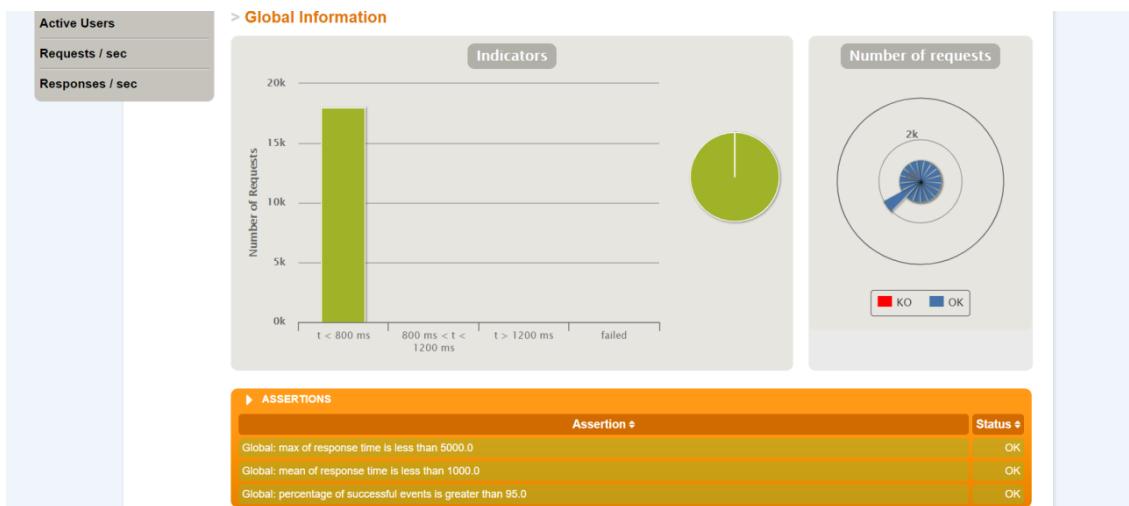
Ahora probaremos con 3000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp scn.inject(rampUsers(3000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95))
```



En su mayor parte el sistema funciona correctamente pero no cumple el tiempo máximo de respuesta.

3) Lo intentaremos con 1000 usuarios



El sistema funciona correctamente.

Test de Rendimiento de la HU11

Analizando la Historia de Usuario 11 con Gatling obtenemos los siguientes resultados:

1) Con 120000 usuarios concurrentes durante 10 segundos:

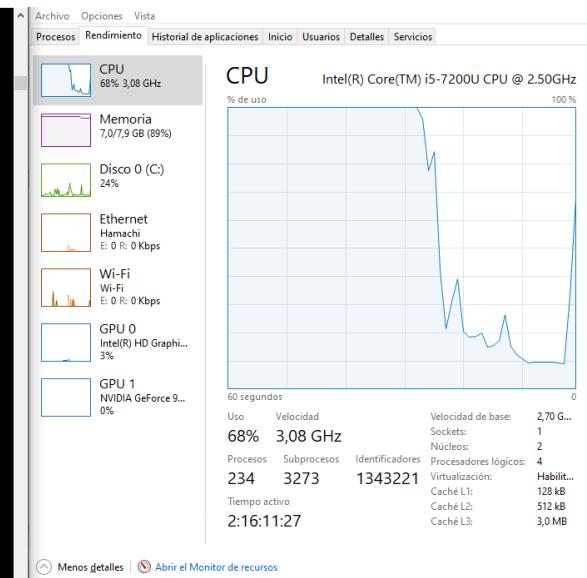
```
setUp(HU11 scn.inject(rampUsers(120000) during (10 seconds))
).protocols(httpProtocol)
```

Vemos que el sistema no funciona correctamente, aparecen demasiadas peticiones KO:

```
> i.n.c.AbstractChannel$AnnotatedSocketException: Address already 48150 ( 8,27%)
y in use: no further information: www.dp2.com/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedConnectException: Connection re 428 ( 0,07%)
Fused: no further information: www.dp2.com/127.0.0.1:80

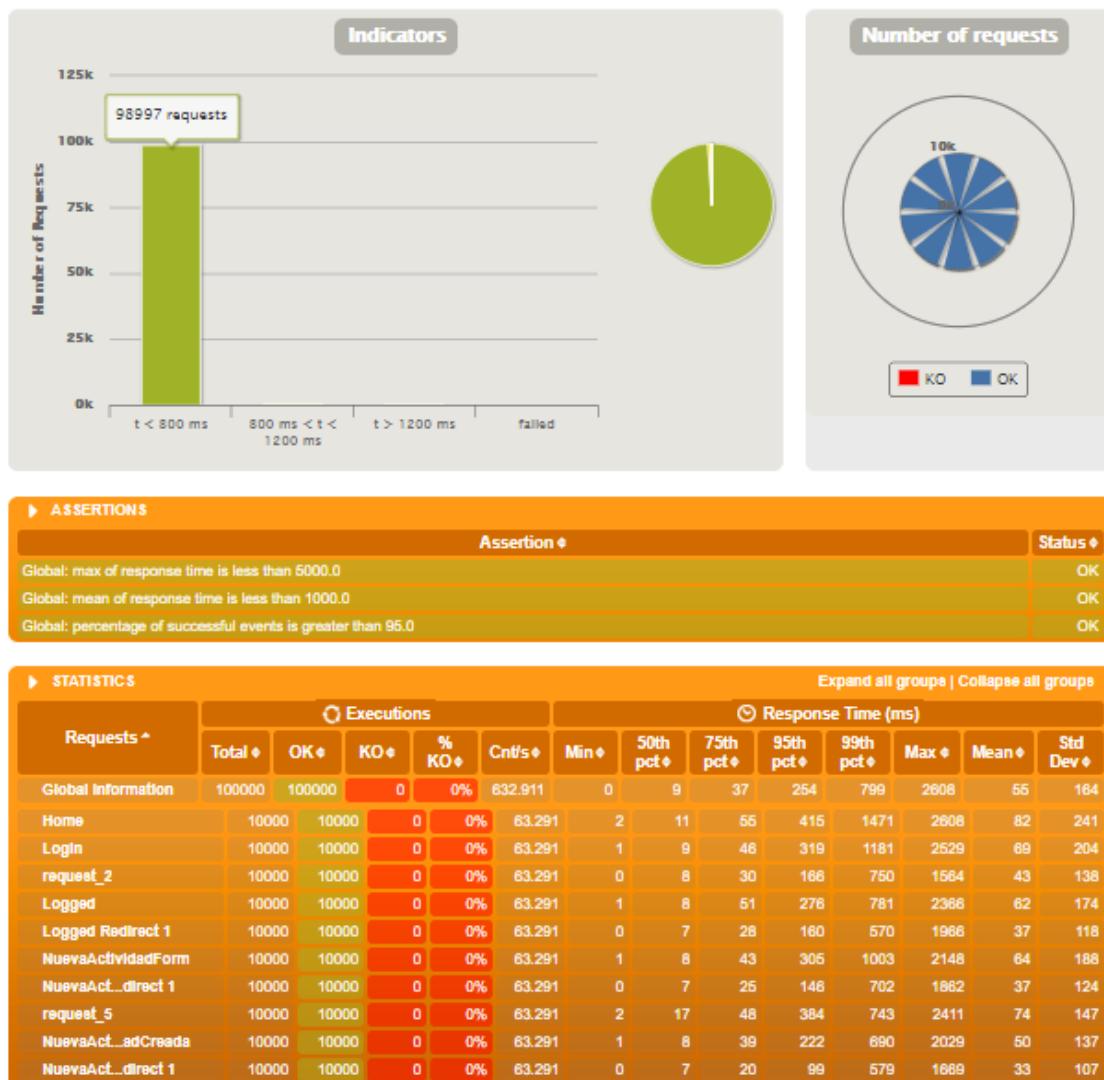
---- HU11 -----
[#########################################-----] 99%
waiting: 0      / active: 6      / done: 11994
-----
2020-05-28 13:48:06          184s elapsed
-- Requests --
> inicial          (OK=49974 KO=592110)
> Home             (OK=8   KO=11992)
> Login            (OK=276 KO=119724)
> request_2        (OK=276 KO=0   )
> Logged           (OK=5011 KO=114989)
> Logged Redirect 1 (OK=5011 KO=0   )
> NuevaActividadForm (OK=5218 KO=114782)
> NuevaActividadForm Redirect 1 (OK=5218 KO=0   )
> NuevaActividadCreada (OK=7369 KO=112631)
> request_5        (OK=5218 KO=0   )
> NuevaActividadCreada Redirect 1 (OK=7369 KO=0   )
---- Errors --
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer spac 461238 (79,23%)
e available (maximum connections reached?): connect: www.dp2.c...
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.c 72302 (12,42%)
om/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedSocketException: Address already 48150 ( 8,27%)
y in use: no further information: www.dp2.com/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedConnectException: Connection re 428 ( 0,07%)
Fused: no further information: www.dp2.com/127.0.0.1:80

---- HU11 -----
[#########################################-----]100%
waiting: 0      / active: 0      / done: 120000
-----
Simulation dp2.HU11Diagnosis completed in 184 seconds
Parsing log file(s)...
```



2) Probamos con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, uno medio de 1 segundo y un porcentaje de éxito de 95%:

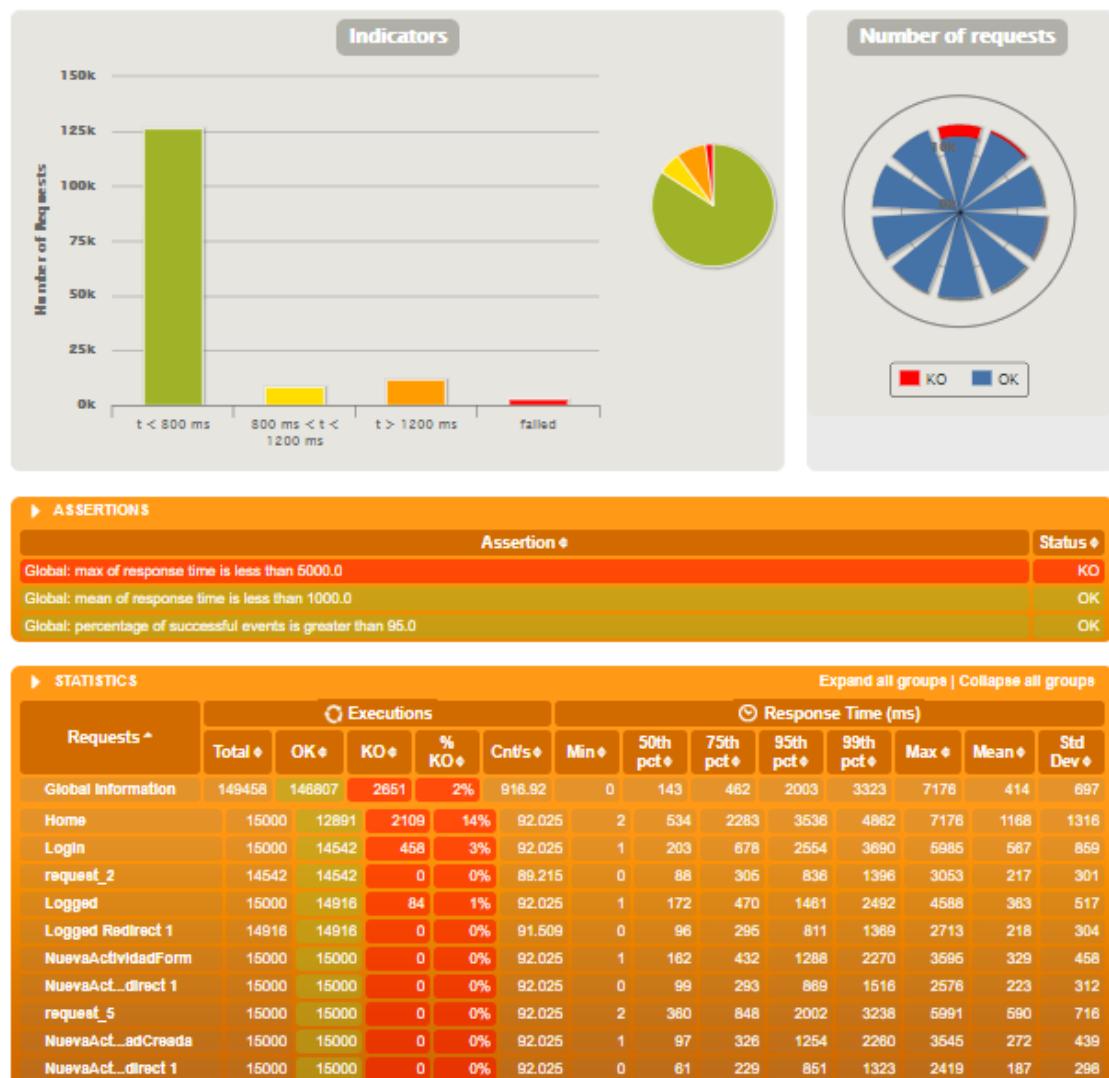
```
setUp(HU11scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Observamos que todas las peticiones son OK y todas con un muy buen tiempo de respuesta.

Intentamos aumentar un poco el número de usuarios para averiguar el máximo número de concurrentes, con 15000 usuarios y mismos ajustes de respuesta:

```
setUp(HU11scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que ya empiezan a aparecer algunas peticiones KO, y el tiempo de respuesta también se ha multiplicado notablemente, así que concluimos que el máximo número de usuarios concurrentes es aproximadamente 10000.

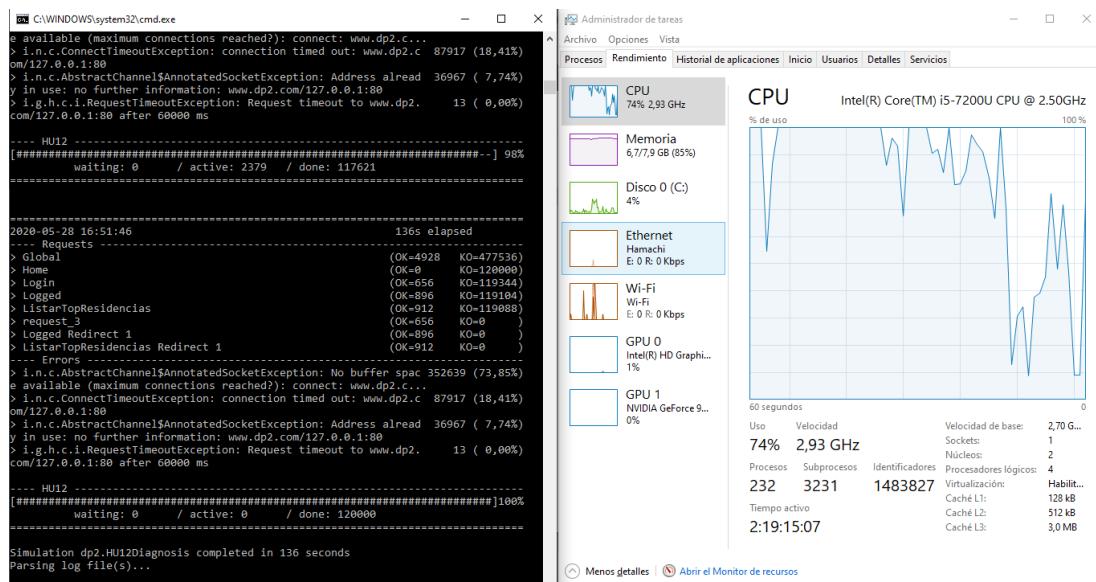
Test de Rendimiento de la HU12

Analizando la Historia de Usuario 12 con Gatling obtenemos los siguientes resultados:

1) Con 120000 usuarios concurrentes durante 10 segundos:

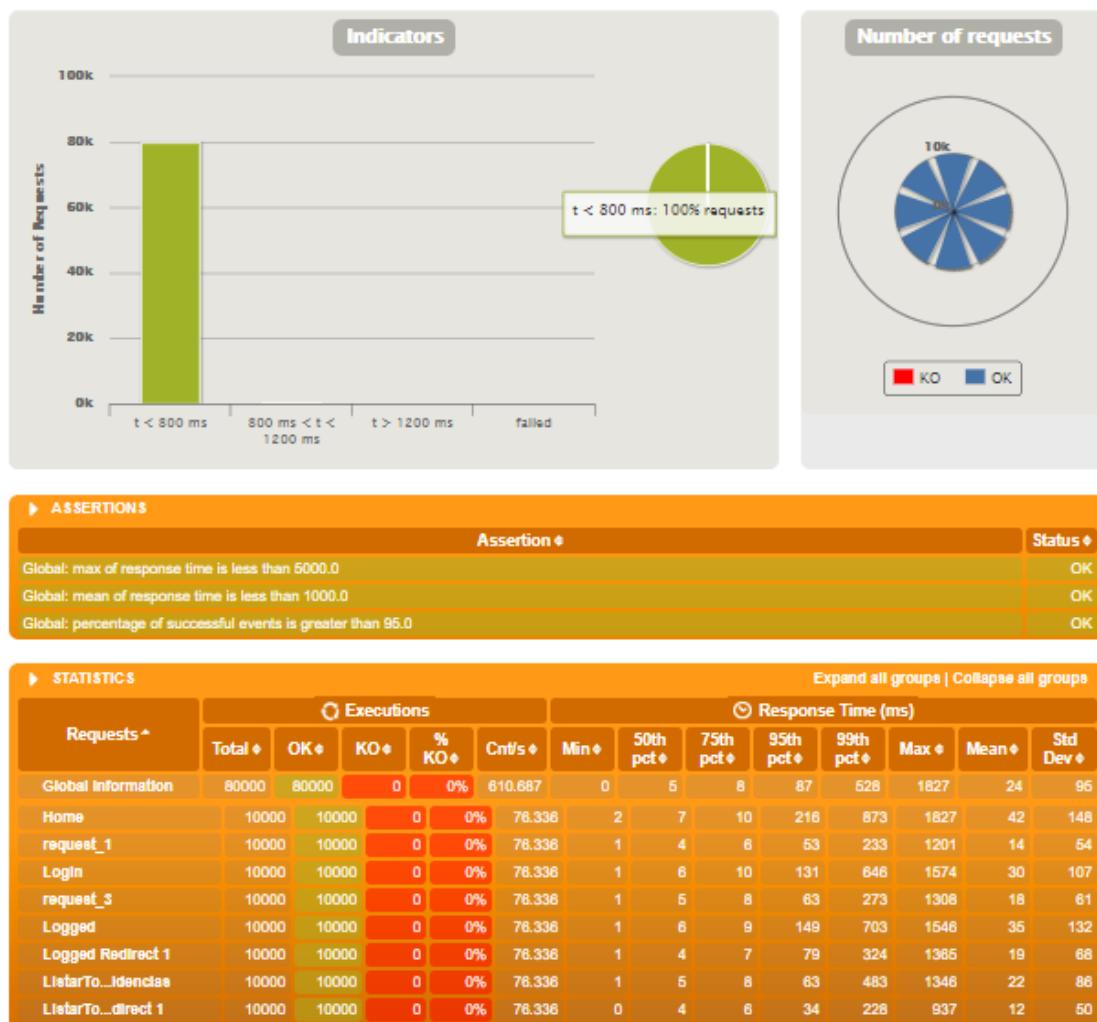
```
setUp(HU12scn.inject(rampUsers(120000) during (10 seconds))
).protocols(httpProtocol)
```

Vemos que el sistema no funciona correctamente, aparecen demasiadas peticiones KO:



2) Probamos con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, uno medio de 1 segundo y un porcentaje de éxito de 95%:

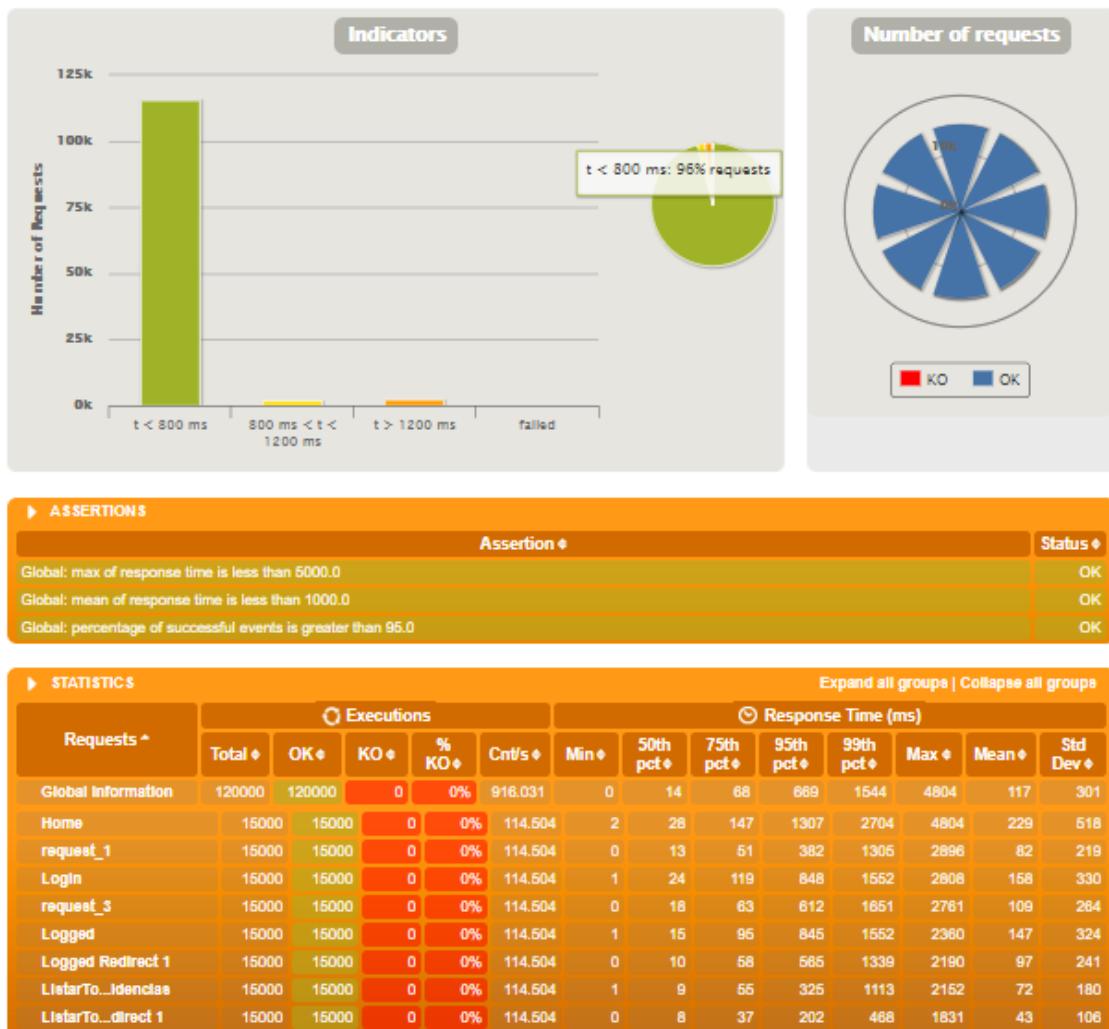
```
setUp(HU12scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Observamos que todas las peticiones son OK y todas con un muy buen tiempo de respuesta.

Intentamos aumentar un poco el número de usuarios para averiguar el máximo número de concurrentes, con 15000 usuarios y mismos ajustes de respuesta:

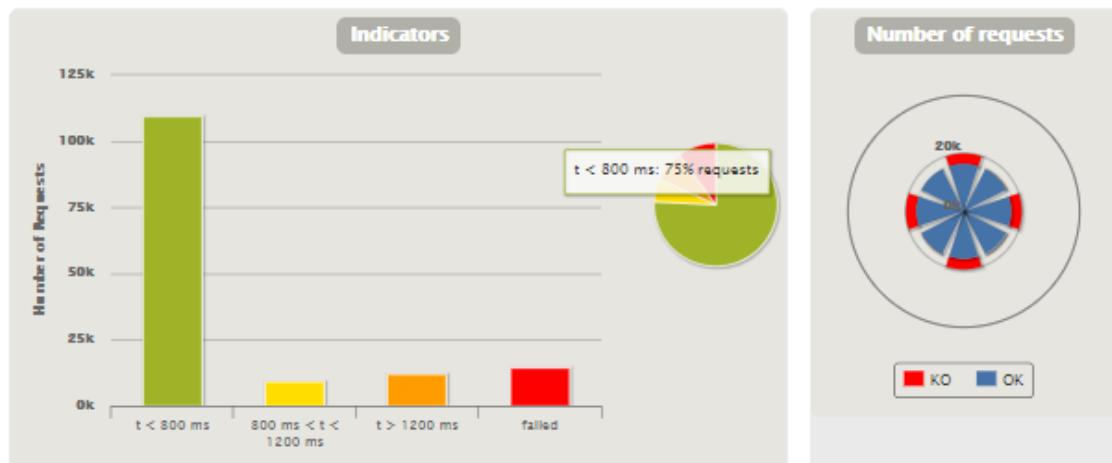
```
setUp(HU12scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que todas las peticiones siguen siendo OK y que, y que el tiempo de respuesta ha aumentado, pero siguen siendo muy buenos resultados.

Seguimos aumentando un poco el número de usuarios para averiguar el máximo, esta vez con 20000 usuarios y mismos ajustes de respuesta:

```
setUp(HU12scn.inject(rampUsers(20000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



▶ ASSERTIONS		Assertion ↗		Status ↗									
Global: max of response time is less than 5000.0				KO									
Global: mean of response time is less than 1000.0				OK									
Global: percentage of successful events is greater than 95.0				KO									
▶ STATISTICS				Expand all groups Collapse all groups									
Requests ↗		🕒 Executions		🕒 Response Time (ms)									
Total ↗	OK ↗	KO ↗	% KO ↗	Cnt/s ↗	Min ↗	50th pct ↗	75th pct ↗	95th pct ↗	99th pct ↗	Max ↗	Mean ↗	Std Dev ↗	
Global Information	145329	130658	14671	10%	1078.511	0	190	546	1727	3622	6355	442	664
Home	20000	16331	3669	18%	148.148	2	298	869	2220	3850	5355	622	801
request_1	16331	16331	0	0%	120.97	0	93	291	852	1476	2840	220	312
Login	20000	16332	3668	18%	148.148	1	286	719	1998	3788	5001	560	740
request_3	16332	16332	0	0%	120.978	0	169	428	1127	1947	4323	319	440
Logged	20000	16333	3667	18%	148.148	1	252	687	1815	3572	4923	510	677
Logged Redirect 1	16333	16333	0	0%	120.985	0	167	517	1679	3888	5084	433	710
ListarTo_Idencias	20000	16333	3667	18%	148.148	1	159	506	1728	3530	4524	417	652
ListarTo_direct 1	16333	16333	0	0%	120.985	0	115	390	1585	3853	5099	378	678

Podemos ver que ya empiezan a aparecer algunas peticiones KO, y el tiempo de respuesta también se ha multiplicado notablemente, así que concluimos que el máximo número de usuarios concurrentes en esta historia de usuario es aproximadamente 15000.

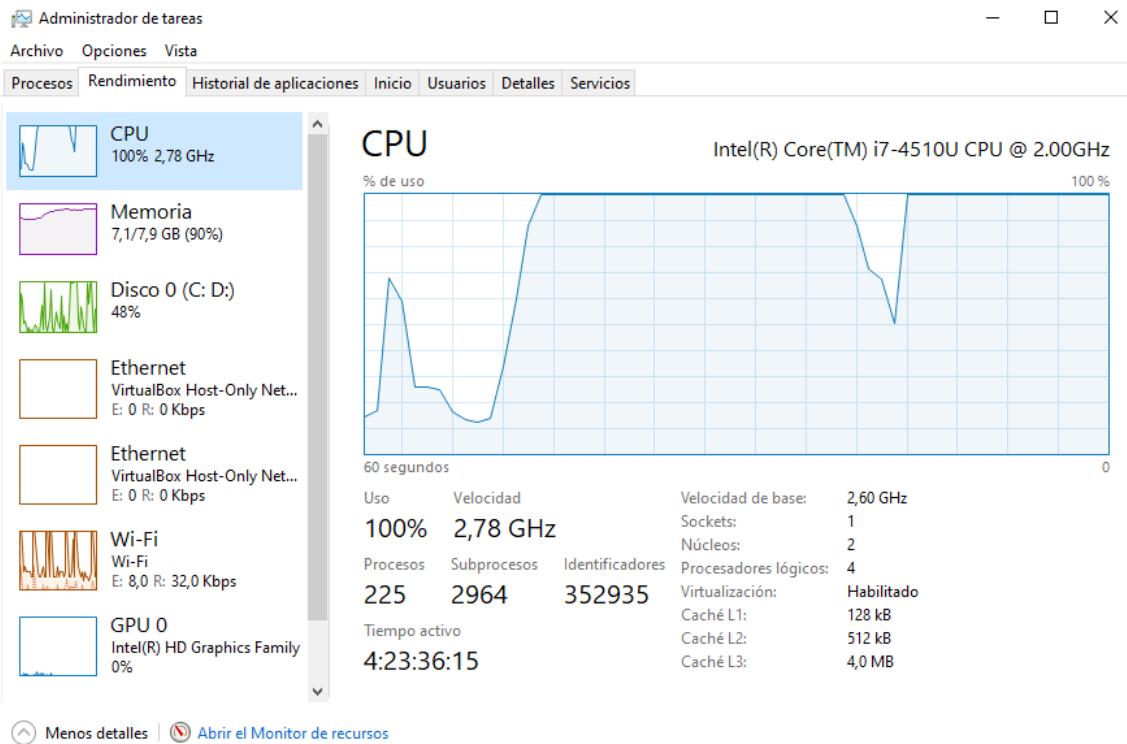
Test de Rendimiento de la HU13

Analizando la Historia de Usuario 13 con Gatling obtenemos los siguientes resultados:

- 1) Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(HU13 scn.inject(rampUsers(100000) during (10 seconds))
).protocols(httpProtocol)
```

El sistema no funciona correctamente:



```
C:\WINDOWS\system32\cmd.exe
---- Requests -----
> Global                                     (OK=7765  KO=495687)
> Home                                       (OK=861   KO=99139 )
> Login                                      (OK=860   KO=99140 )
> Logged                                     (OK=860   KO=99140 )
> NuevoFeedbackForm                         (OK=861   KO=99139 )
> NuevoFeedbackCreado                      (OK=871   KO=99129 )
> NuevoFeedbackForm Redirect 1              (OK=861   KO=0     )
> request_2                                  (OK=860   KO=0     )
> NuevoFeedbackCreado Redirect 1            (OK=871   KO=0     )
> Logged Redirect 1                         (OK=860   KO=0     )
---- Errors -----
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer space available (maximum connections reached?): connect: www.dp2.c... (69,08%)
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.c 84318 (17,01%) com/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedSocketException: Address already in use: no further information: www.dp2.com/127.0.0.1:80
> i.g.h.c.i.RequestTimeoutException: Request timeout to www.dp2. 20 ( 0,00%) com/127.0.0.1:80 after 60000 ms

---- HU13 -----
[########################################]100%
      waiting: 0      / active: 0      / done: 100000
=====
Simulation dp2.HU13 completed in 149 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
=====
```

- 2) En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU13scn.inject(rampUsers(5000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente.

> **Global Information**



> **ASSERTIONS**

Assertion	Status
Global: max of response time is less than 5000.0	OK
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

> **STATISTICS**

Requests	Executions												Response Time (ms)						
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev						
Global Information	45000	45000	0	0%	326.087	0	1	3	5	14	429	3	12						
Home	5000	5000	0	0%	36.232	2	4	5	12	261	429	9	34						
Login	5000	5000	0	0%	36.232	0	2	2	6	16	94	2	4						
request_2	5000	5000	0	0%	36.232	0	1	2	4	11	48	2	3						
Logged	5000	5000	0	0%	36.232	0	2	3	5	17	82	3	4						
Logged Redirect 1	5000	5000	0	0%	36.232	0	1	2	3	6	50	1	2						
NuevoFeedbackForm	5000	5000	0	0%	36.232	0	2	2	5	10	102	2	3						
NuevoFee...direct 1	5000	5000	0	0%	36.232	0	1	2	3	5	44	1	2						
NuevoFeedbackCreado	5000	5000	0	0%	36.232	0	2	2	4	10	68	2	3						
NuevoFee...direct 1	5000	5000	0	0%	36.232	0	1	1	3	5	45	1	1						

- 3) Probamos a estresar más, aumentando 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU13scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema sigue funcionando correctamente.

> Global Information



- 4) Seguimos estresando aumentando otros 5 mil usuarios con los mismos parámetros que antes:

```
setUp(HU13scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema, una vez más, funciona correctamente.

> Global Information



> ASSERTIONS

Assertion	Status
Global: max of response time is less than 5000.0	OK
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

> STATISTICS

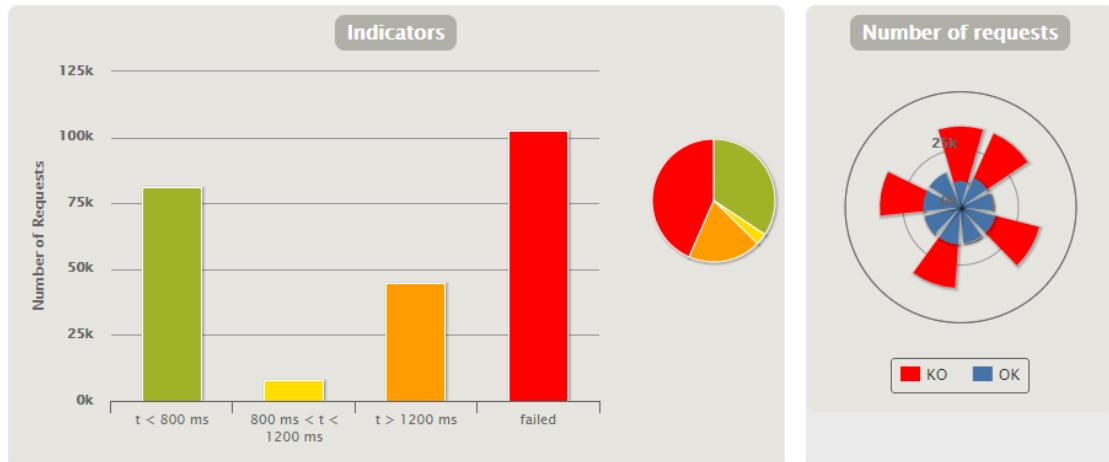
Requests	Executions					Response Time (ms)								
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev	
Global Information	135000	135000	0	0%	978.261	0	5	10	58	226	2065	16	52	
Home	15000	15000	0	0%	108.696	1	8	16	117	355	1168	28	87	
Login	15000	15000	0	0%	108.696	0	5	13	74	222	1168	19	62	
request_2	15000	15000	0	0%	108.696	0	5	10	38	161	1076	12	34	
Logged	15000	15000	0	0%	108.696	0	5	12	69	224	2065	19	63	
Logged Redirect 1	15000	15000	0	0%	108.696	0	4	8	36	189	968	11	32	
NuevoFeedbackForm	15000	15000	0	0%	108.696	0	5	13	89	220	633	19	45	
NuevoFee...direct 1	15000	15000	0	0%	108.696	0	4	8	41	154	440	11	30	
NuevoFeedbackCreado	15000	15000	0	0%	108.696	0	5	9	73	240	672	18	48	
NuevoFee...direct 1	15000	15000	0	0%	108.696	0	4	7	34	166	633	10	29	

5) Probamos a estresar el sistema con 35 mil usuarios esta vez:

```
setUp(HU13scn.inject(rampUsers(35000) during (100 seconds))
    .protocols(httpProtocol).assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95)
    )
```

Esta vez como vemos, el sistema ha reaccionado más estresado pues se nota un alto número de peticiones KOs. Por tanto, estableceremos que con este número de usuarios el sistema alcanza su pico de peticiones en nuestro equipo.

> Global Information



▶ ASSERTIONS
Assertion ↴
Status ↴

Global: max of response time is less than 5000.0	KO
Global: mean of response time is less than 1000.0	KO
Global: percentage of successful events is greater than 95.0	KO

▶ STATISTICS
Expand all groups | Collapse all groups

Requests ▲	🕒 Executions					⌚ Response Time (ms)								
	Total ↴	OK ↴	KO ↴	% KO ↴	Cnt/s ↴	Min ↴	50th pct ↴	75th pct ↴	95th pct ↴	99th pct ↴	Max ↴	Mean ↴	Std Dev ↴	
Global Information	236390	133931	102459	43%	1432.667	0	2198	5720	10829	12822	18672	3328	3606	
Home	35000	11151	23849	68%	212.121	2	2314	3665	10748	13518	18637	3087	3269	
Login	35000	13794	21206	61%	212.121	1	2482	7574	10426	13260	18239	4017	3830	
request_2	13794	13794	0	0%	83.6	0	161	482	3099	12364	14304	716	1786	
Logged	35000	15370	19630	56%	212.121	1	3199	7022	10738	12783	18672	4080	3626	
Logged Redirect 1	15370	15370	0	0%	93.152	0	235	1024	3402	11965	14298	937	1779	
NuevoFeedbackForm	35000	16065	18935	54%	212.121	2	3379	6629	9765	12505	18666	4145	3428	
NuevoFee...direct 1	16065	16065	0	0%	97.364	0	519	1703	3500	7895	14275	1174	1680	
NuevoFeedbackCreado	35000	16161	18839	54%	212.121	2	5263	9510	11361	12738	14949	5394	4124	
NuevoFee...direct 1	16161	16161	0	0%	97.945	0	775	1566	3251	6019	14206	1133	1481	

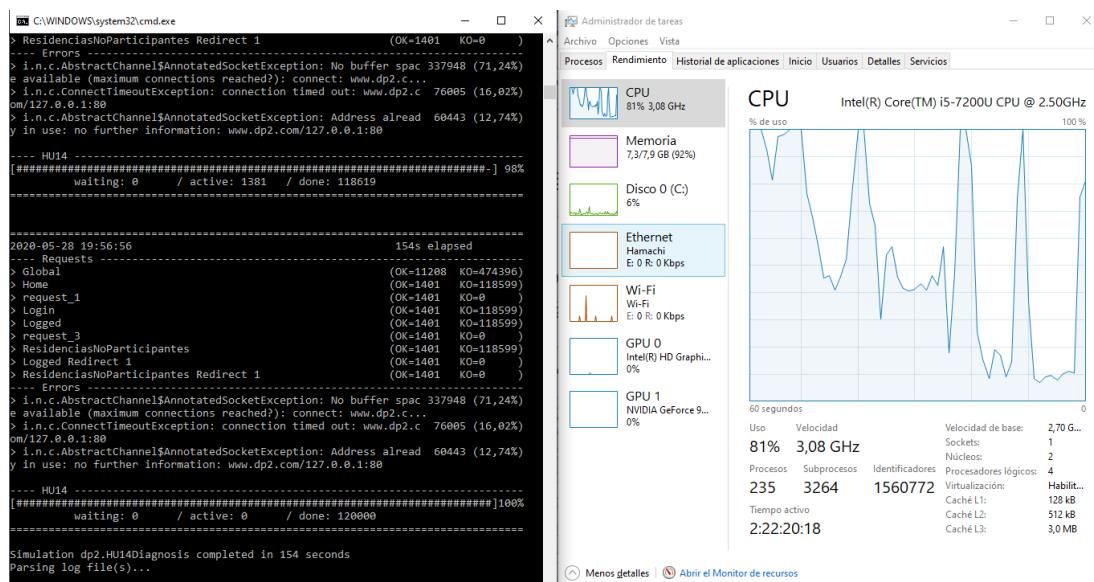
Test de Rendimiento de la HU14

Analizando la Historia de Usuario 14 con Gatling obtenemos los siguientes resultados:

1) Con 120000 usuarios concurrentes durante 10 segundos:

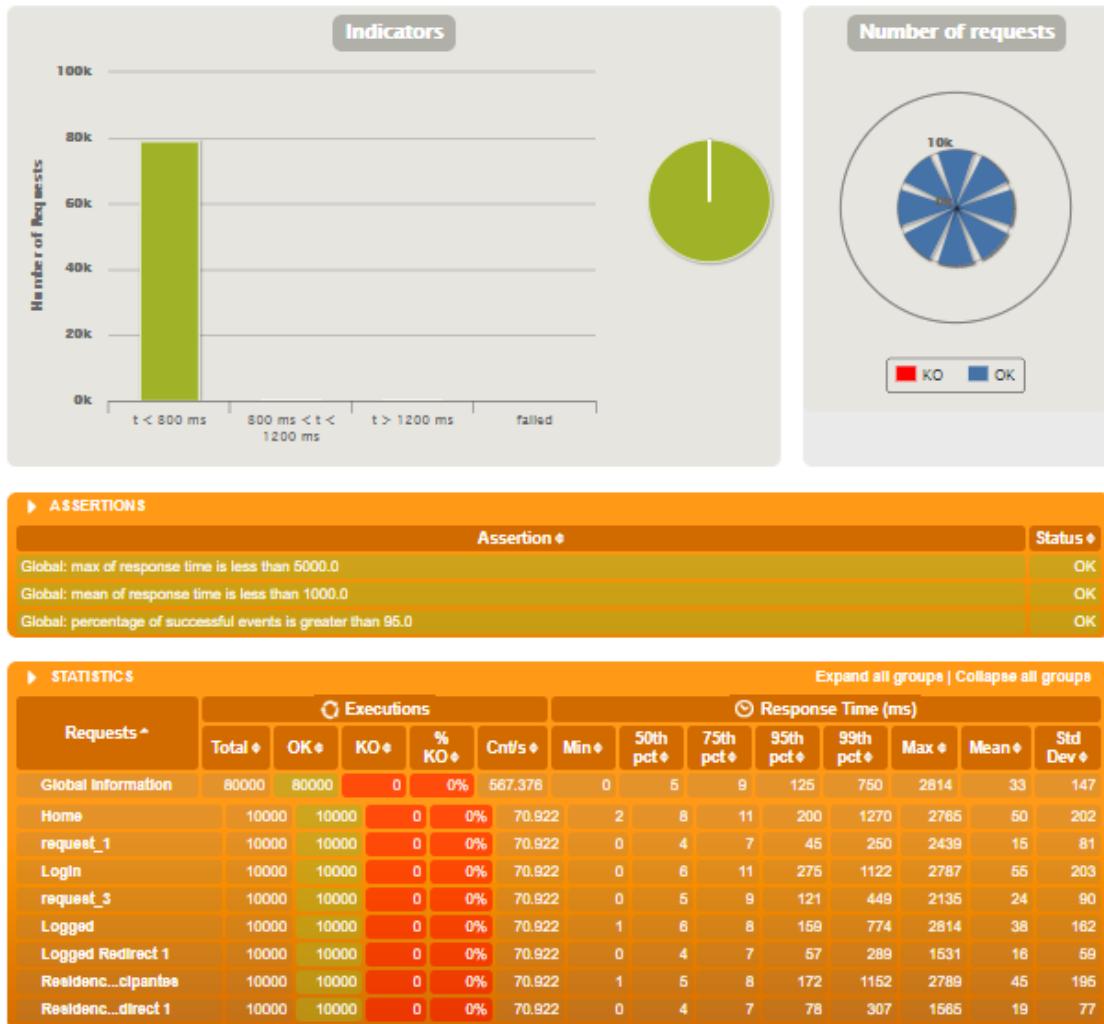
```
setUp(HU14scn.inject(rampUsers(120000) during (10 seconds))
).protocols(httpProtocol)
```

Vemos que el sistema no funciona correctamente, aparecen demasiadas peticiones KO:



2) Probamos con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, uno medio de 1 segundo y un porcentaje de éxito de 95%:

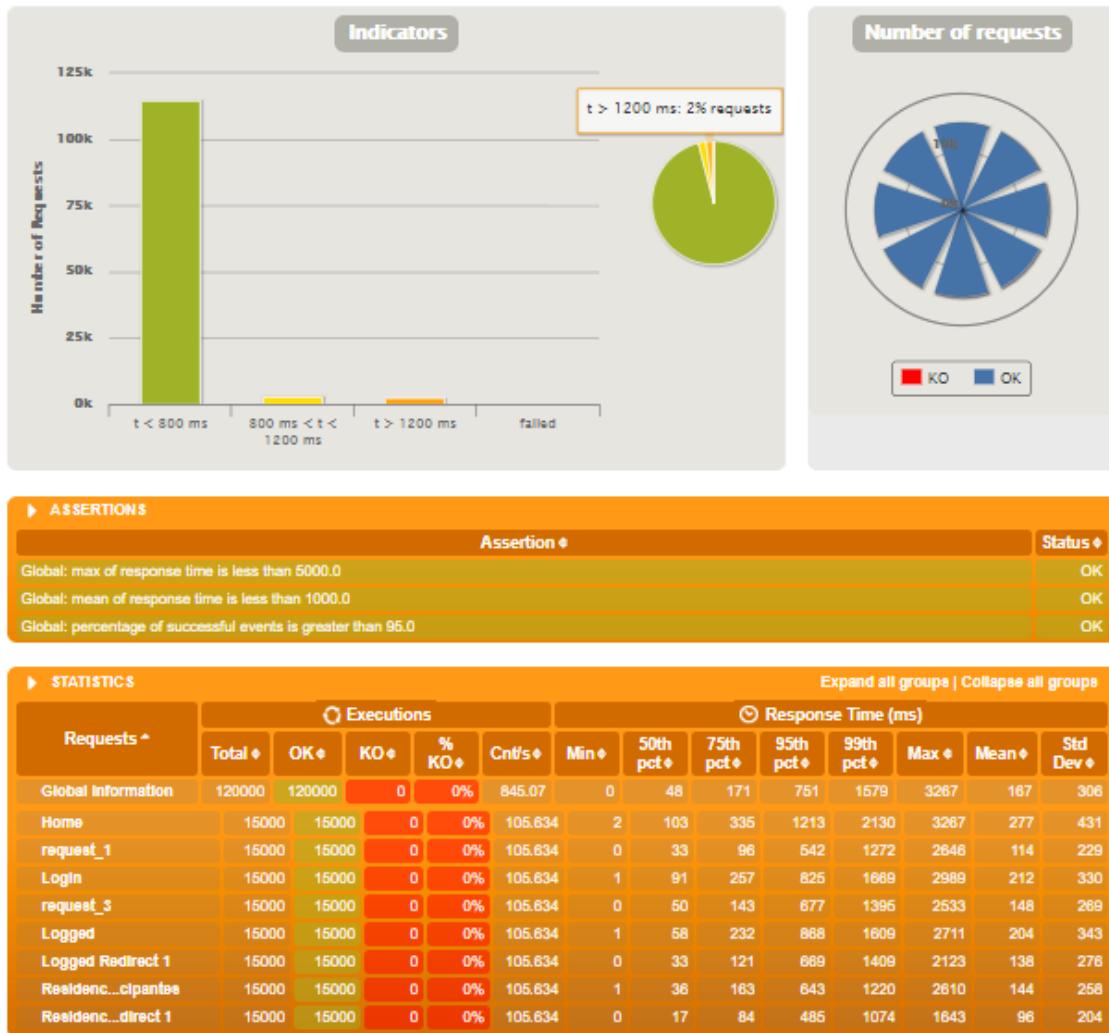
```
setUp(HU14scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Observamos que todas las peticiones son OK y todas con un excelente tiempo de respuesta.

Intentamos aumentar un poco el número de usuarios para averiguar el máximo número de concurrentes, con 15000 usuarios y mismos ajustes de respuesta:

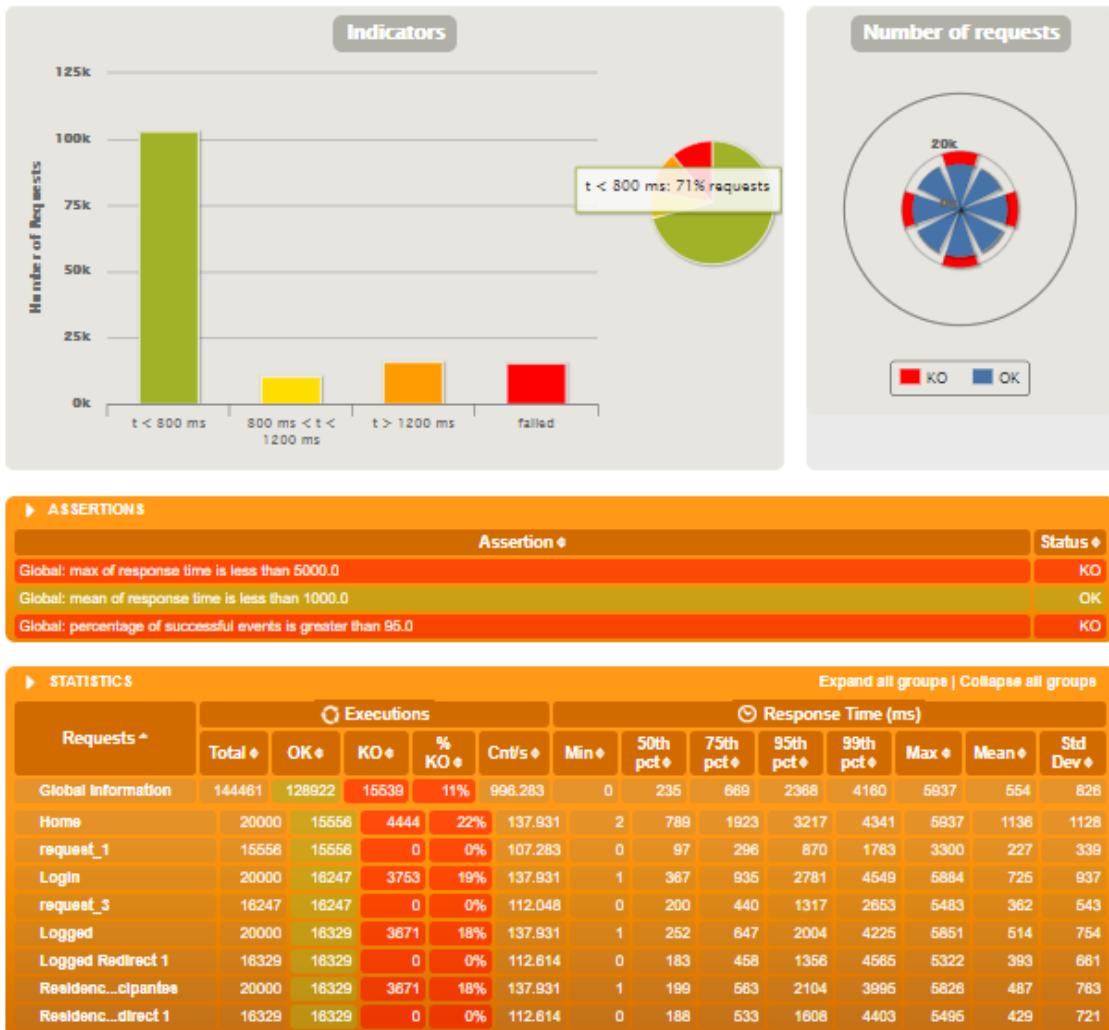
```
setUp(HU14scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que todas las peticiones siguen siendo OK y que, y que el tiempo de respuesta ha aumentado, pero siguen siendo aún buenos resultados.

Seguimos aumentando un poco el número de usuarios para averiguar el máximo, esta vez con 20000 usuarios y mismos ajustes de respuesta:

```
setUp(HU14scn.inject(rampUsers(20000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que ya empiezan a aparecer algunas peticiones KO, y el tiempo de respuesta también se ha multiplicado notablemente, así que concluimos que el máximo número de usuarios concurrentes en esta historia de usuario es aproximadamente 15000.

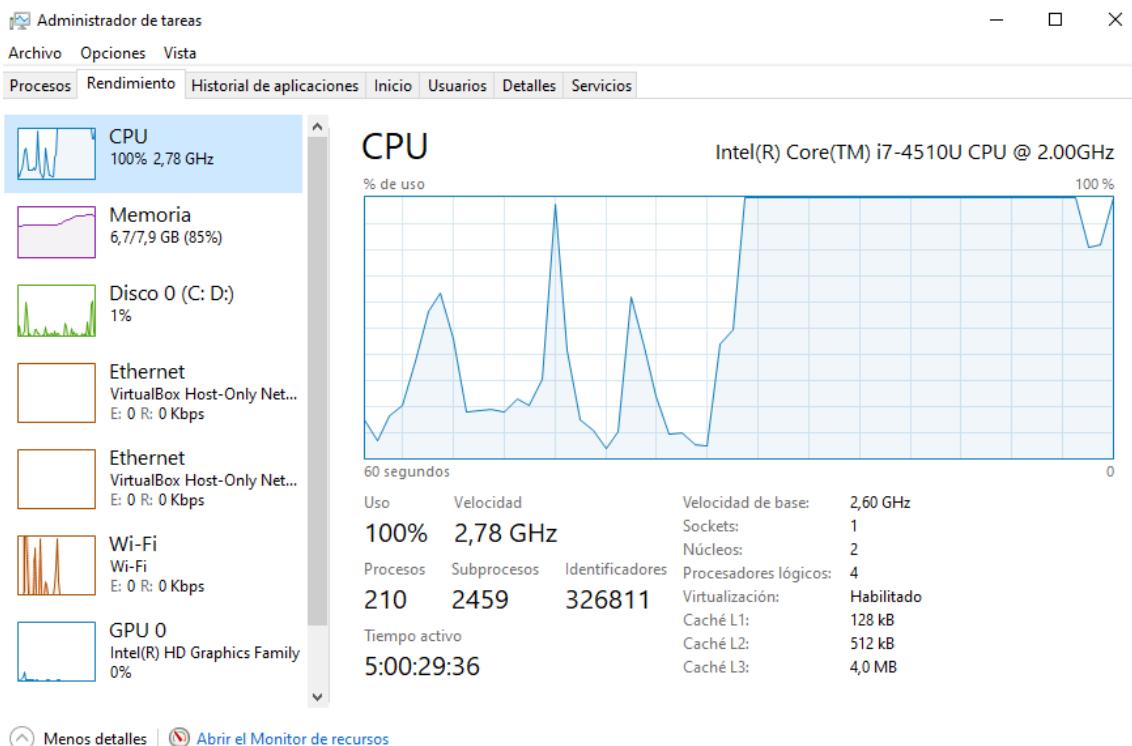
Test de Rendimiento de la HU15

Analizando la Historia de Usuario 15 con Gatling obtenemos los siguientes resultados:

- 1) Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(HU15 scn.inject(rampUsers(100000) during (10 seconds))
).protocols(httpProtocol)
```

El sistema no funciona correctamente:



```
---- Requests ----
> Global                                     (OK=0      KO=500000)
> Home                                       (OK=0      KO=100000)
> Login                                      (OK=0      KO=100000)
> Logged                                     (OK=0      KO=100000)
> EditarFeedbackForm                         (OK=0      KO=100000)
> FeedbackEditado                           (OK=0      KO=100000)
---- Errors --
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer space available (maximum connections reached?): connect: www.dp2.c...
> i.n.c.AbstractChannel$AnnotatedConnectException: Connection refused: no further information: www.dp2.com/127.0.0.1:80
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.com/127.0.0.1:80   90096 (18,02%)
                                          59267 (11,85%)
```

```
---- HU15 ----
[########################################] 100%
    waiting: 0      / active: 0      / done: 100000
-----
Simulation dp2.HU15 completed in 86 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
```

2)

En cambio, con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU15scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente:

> Global Information



► ASSERTIONS

Assertion	Status
Global: max of response time is less than 5000.0	KO
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

► STATISTICS

Requests	Executions					Response Time (ms)								
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev	
Global Information	90000	90000	0	0%	743.802	0	2	5	709	2116	9705	127	538	
Home	10000	10000	0	0%	82.645	1	5	187	3737	7439	9705	516	1386	
Login	10000	10000	0	0%	82.645	0	3	9	653	1178	3287	103	263	
request_2	10000	10000	0	0%	82.645	0	2	4	517	1019	2781	71	235	
Logged	10000	10000	0	0%	82.645	0	3	7	663	1113	3295	107	252	
Logged Redirect 1	10000	10000	0	0%	82.645	0	1	3	587	1031	2778	71	242	
EditarFeedbackForm	10000	10000	0	0%	82.645	0	3	5	906	2064	2576	130	378	
EditarFe...direct 1	10000	10000	0	0%	82.645	0	1	3	642	1297	2592	72	267	
FeedbackEditado	10000	10000	0	0%	82.645	0	2	4	324	1127	1609	47	176	
Feedback...direct 1	10000	10000	0	0%	82.645	0	1	2	140	876	1332	27	137	

3) Probamos a estresar un poco más el sistema aumentando el número de usuarios a 35000:

```
setUp(HU15scn.inject(rampUsers(35000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

Vemos que no responde correctamente pues hay varias peticiones hechas al servidor que terminan en KO, luego estimamos que el número de usuarios aquí establece el cuello de botella en el sistema.

> Global Information



Test de Rendimiento de la HU16

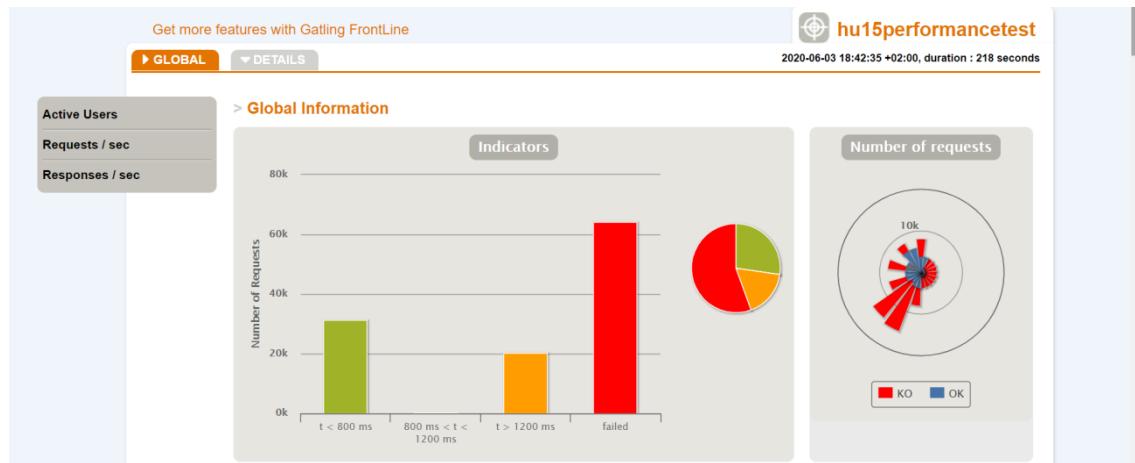
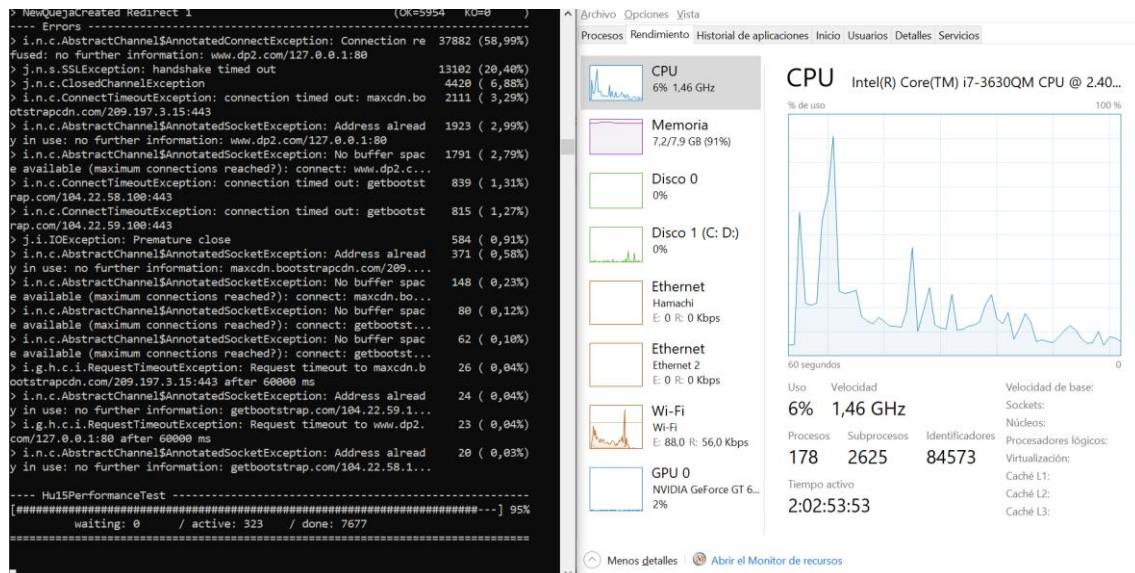
Analizando la Historia de Usuario 16 con Gatling obtenemos los siguientes resultados:

1)

Con 8000 usuarios concurrentes durante 10 segundos:

```
setUp scn.inject(rampUsers(8000) during (10 seconds))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2) Ahora probaremos con 3000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp scn.inject(rampUsers(3000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95))
```



Hay pocos errores, pero no cumple con los requisitos.

3) Lo intentaremos con 1000 usuarios:



El sistema funciona perfectamente.

Test de Rendimiento de la HU17

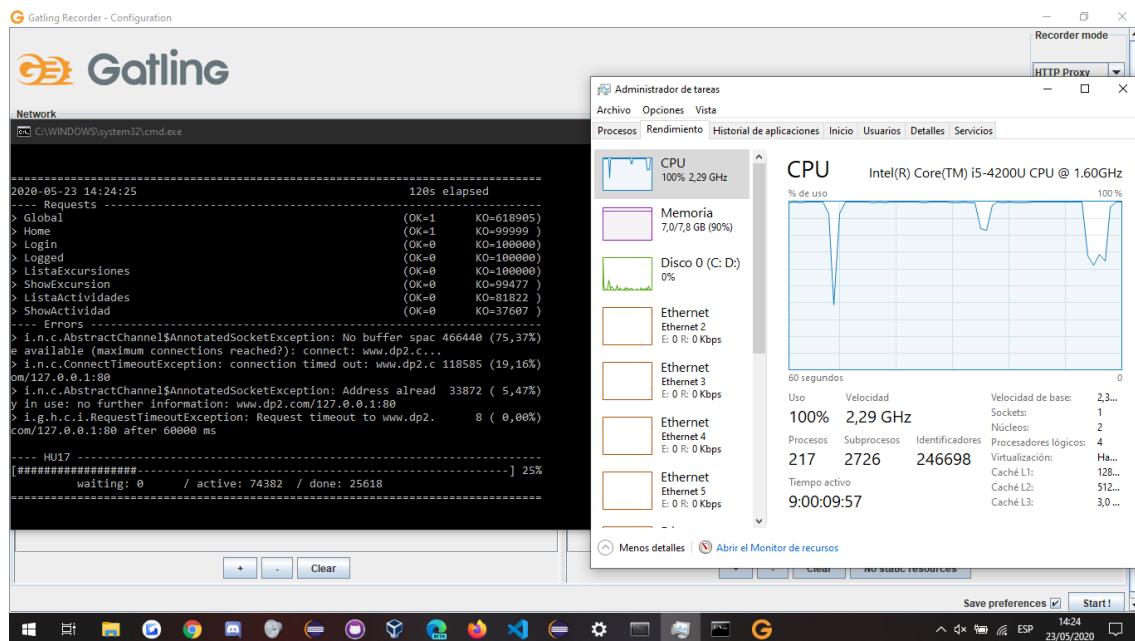
Analizando la Historia de Usuario 17 con Gatling obtenemos los siguientes resultados:

1)

Con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(
    scn.inject(rampUsers(100000) during (10 seconds))
).protocols([httpProtocol])
```

El sistema no funciona correctamente:



2)

En cambio, con 5000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(
    scn.inject(rampUsers(5000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente:

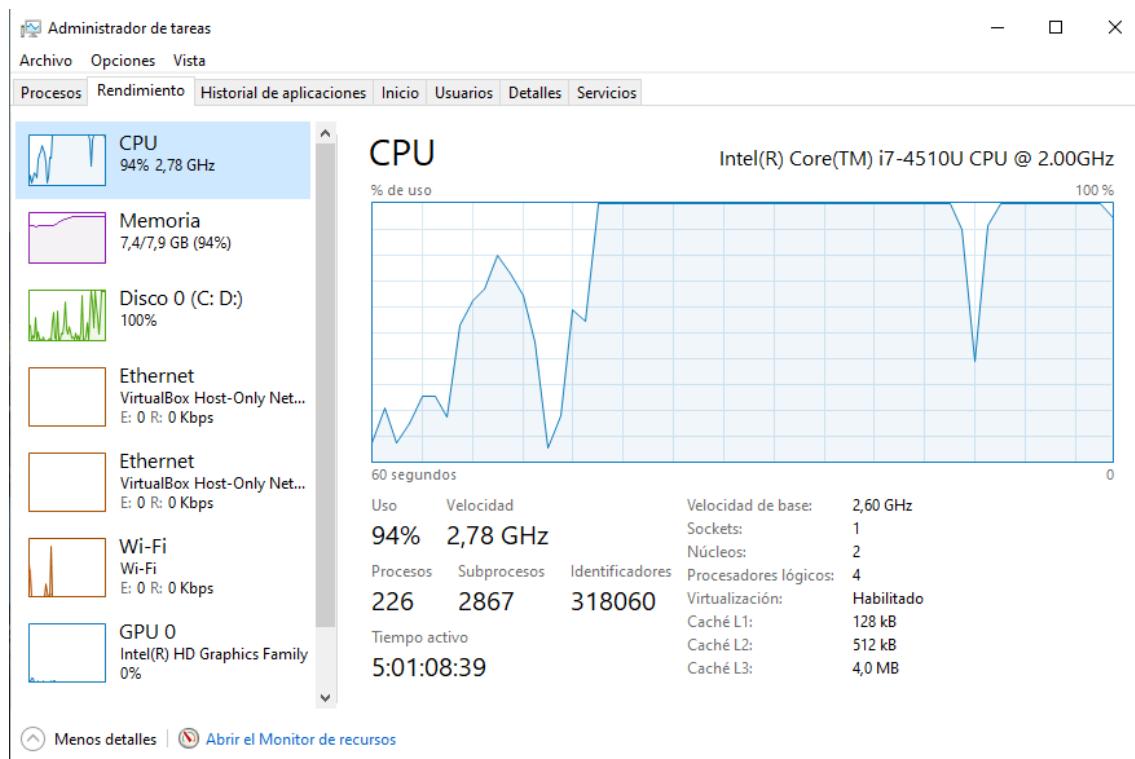


Test de Rendimiento de la HU18

- 1) Comenzamos como siempre probando con 100000 usuarios concurrentes durante 10 segundos:

```
setUp(HU18scn.inject(rampUsers(100000) during (10 seconds))
).protocols(httpProtocol)
```

El sistema no funciona correctamente:



```
C:\WINDOWS\system32\cmd.exe
---- Requests -----
> Global                                (OK=5003   KO=397496)
> Home                                   (OK=5      KO=99995  )
> Login                                  (OK=833    KO=99167  )
> request_2                             (OK=833    KO=0     )
> Logged                                 (OK=833    KO=99167  )
> ListarRatioResidencias                (OK=833    KO=99167  )
> Logged Redirect 1                      (OK=833    KO=0     )
> ListarRatioResidencias Redirect 1       (OK=833    KO=0     )
---- Errors -----
> i.n.c.AbstractChannel$AnnotatedSocketException: No buffer spac 277805 (69,89%)
e available (maximum connections reached?): connect: www.dp2.c...
> i.n.c.ConnectTimeoutException: connection timed out: www.dp2.c 61759 (15,54%)
om/127.0.0.1:80
> i.n.c.AbstractChannel$AnnotatedSocketException: Address alread 57932 (14,57%)
y in use: no further information: www.dp2.com/127.0.0.1:80

---- HU18 -----
[########################################################################]100%
      waiting: 0      / active: 0      / done: 100000
=====
Simulation dp2.HU18 completed in 108 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...
```

- 2) En cambio, con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp(HU18scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

El sistema funciona correctamente:

> **Global Information**



> **ASSERTIONS**

Assertion	Status
Global: max of response time is less than 5000.0	OK
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK

> **STATISTICS**

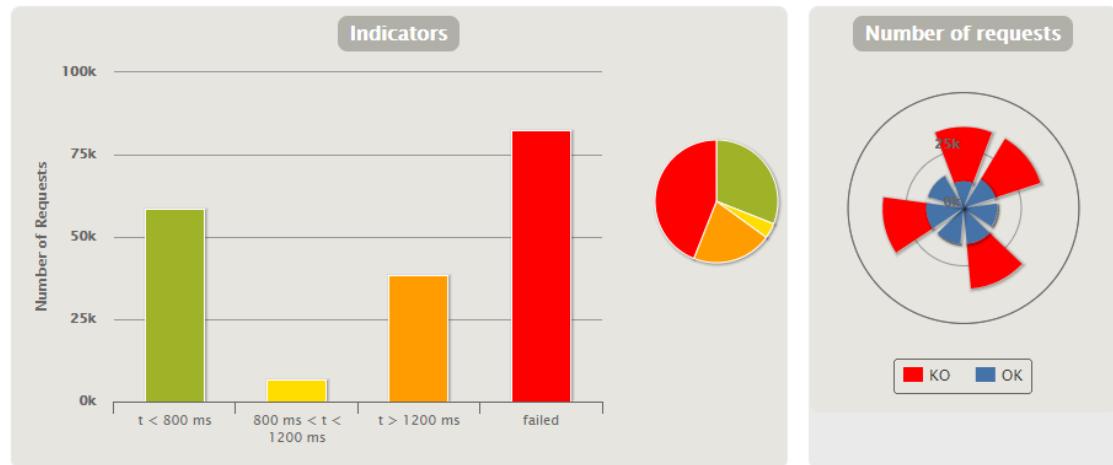
Requests	Executions										Response Time (ms)						
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev				
Global Information	70000	70000	0	0%	500	0	3	4	8	22	331	4	8				
Home	10000	10000	0	0%	71.429	1	5	7	12	35	304	7	11				
Login	10000	10000	0	0%	71.429	0	3	4	10	29	180	4	9				
request_2	10000	10000	0	0%	71.429	0	2	4	7	17	143	3	3				
Logged	10000	10000	0	0%	71.429	0	3	4	8	27	331	4	9				
Logged Redirect 1	10000	10000	0	0%	71.429	0	2	3	6	13	169	3	5				
ListarRa...idencias	10000	10000	0	0%	71.429	0	3	4	9	22	174	4	7				
ListarRa...direct 1	10000	10000	0	0%	71.429	0	2	3	6	13	133	2	4				

- 3) Probamos a estresar un poco más el sistema aumentando el número de usuarios a 35 mil:

```
setUp(HU18scn.inject(rampUsers(35000) during (100 seconds))
).protocols(httpProtocol).assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```

Vemos que no responde correctamente pues hay varias peticiones hechas al servidor que terminan en KO, luego estimamos que el número de usuarios aquí establece el cuello de botella en el sistema.

> Global Information



> ASSERTIONS

Assertion ↴	Status ↴
Global: max of response time is less than 5000.0	KO
Global: mean of response time is less than 1000.0	KO
Global: percentage of successful events is greater than 95.0	KO

> STATISTICS

Requests ↴	⌚ Executions					⌚ Response Time (ms)							
	Total ↴	OK ↴	KO ↴	% KO ↴	Cnt/s ↴	Min ↴	50th pct ↴	75th pct ↴	95th pct ↴	99th pct ↴	Max ↴	Mean ↴	Std Dev ↴
Global Information	186158	103812	82346	44%	1142.074	0	2239	4676	9709	14105	27457	3068	3389
Home	35000	11496	23504	67%	214.724	2	2320	3776	10778	16372	27252	3312	3623
Login	35000	14470	20530	59%	214.724	1	2373	5938	10462	12662	27457	3615	3592
request_2	14470	14470	0	0%	88.773	0	183	400	2546	8559	22541	576	1453
Logged	35000	15562	19438	56%	214.724	1	3082	5152	9574	14582	22797	3673	3247
Logged Redirect 1	15562	15562	0	0%	95.472	0	586	1981	6057	22339	22863	1688	3091
ListarRa...dencias	35000	16126	18874	54%	214.724	0	3548	6084	10145	12478	27396	4032	3293
ListarRa...direct 1	16126	16126	0	0%	98.933	0	1014	1884	6055	10404	22542	1516	2264

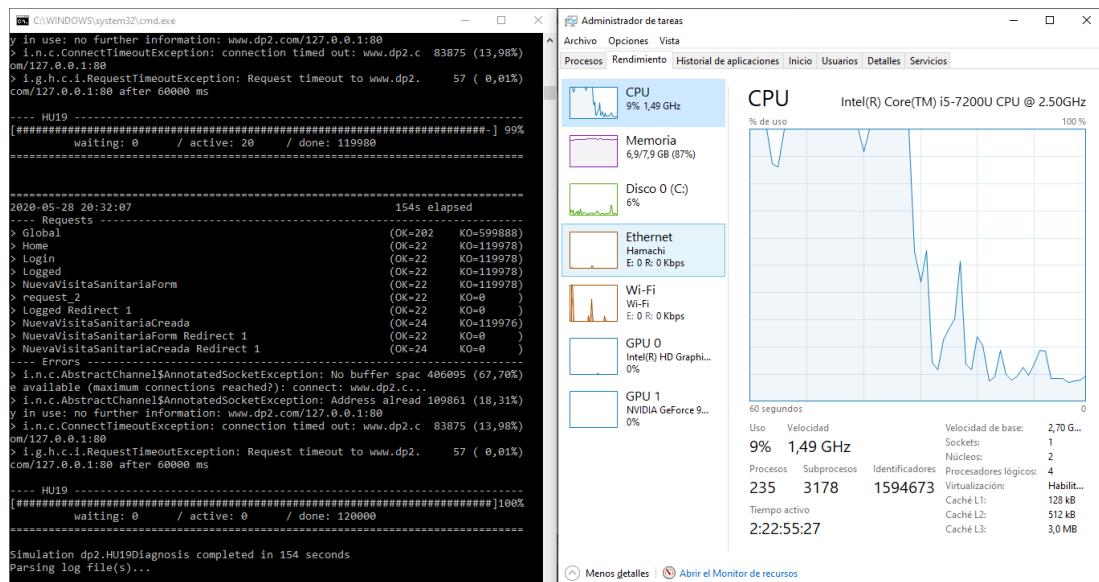
Test de Rendimiento de la HU19

Analizando la Historia de Usuario 19 con Gatling obtenemos los siguientes resultados:

1) Con 120000 usuarios concurrentes durante 10 segundos:

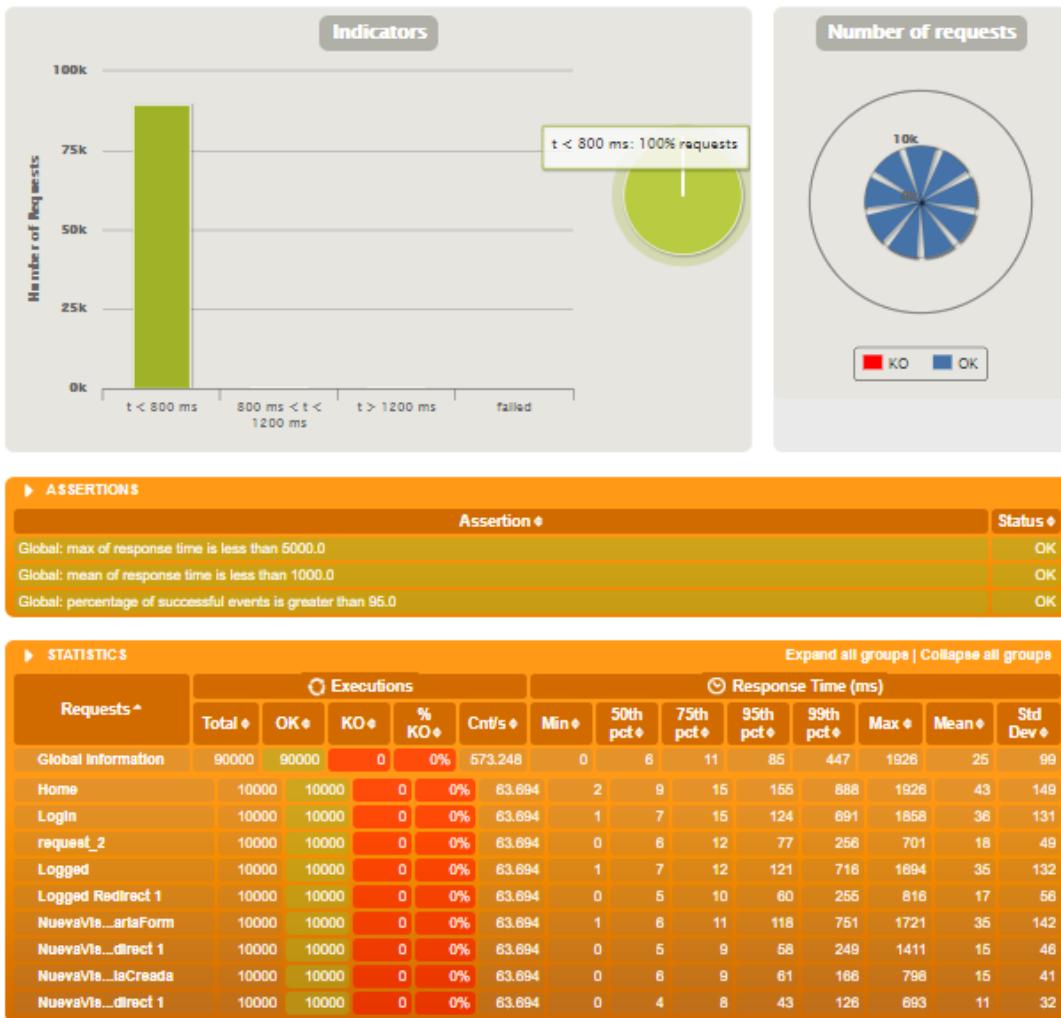
```
setUp(HU19scn.inject(rampUsers(120000) during (10 seconds))
).protocols(httpProtocol)
```

Vemos que el sistema no funciona correctamente, aparecen demasiadas peticiones KO:



2) Probamos con 10000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, uno medio de 1 segundo y un porcentaje de éxito de 95%:

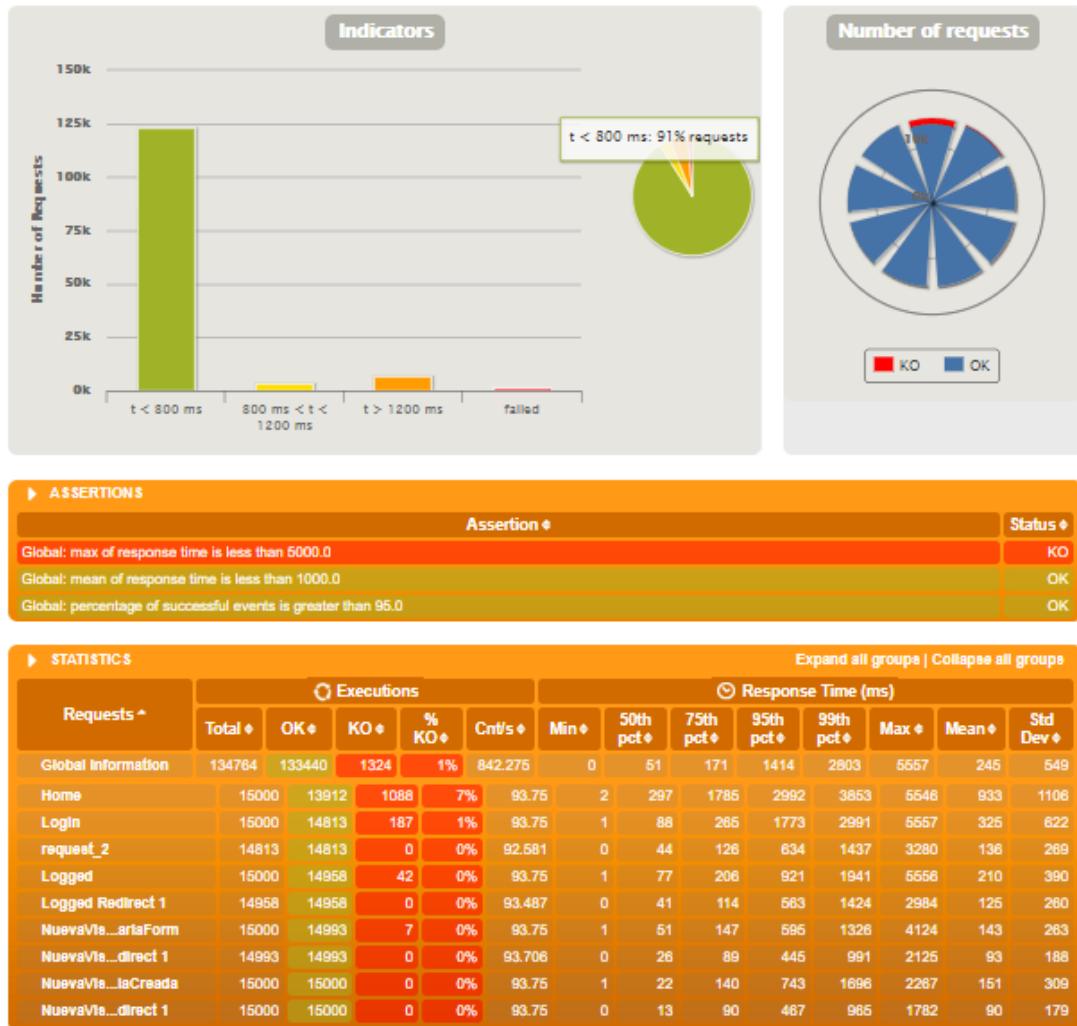
```
setUp(HU19scn.inject(rampUsers(10000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Observamos que todas las peticiones son OK y todas con un excelente tiempo de respuesta.

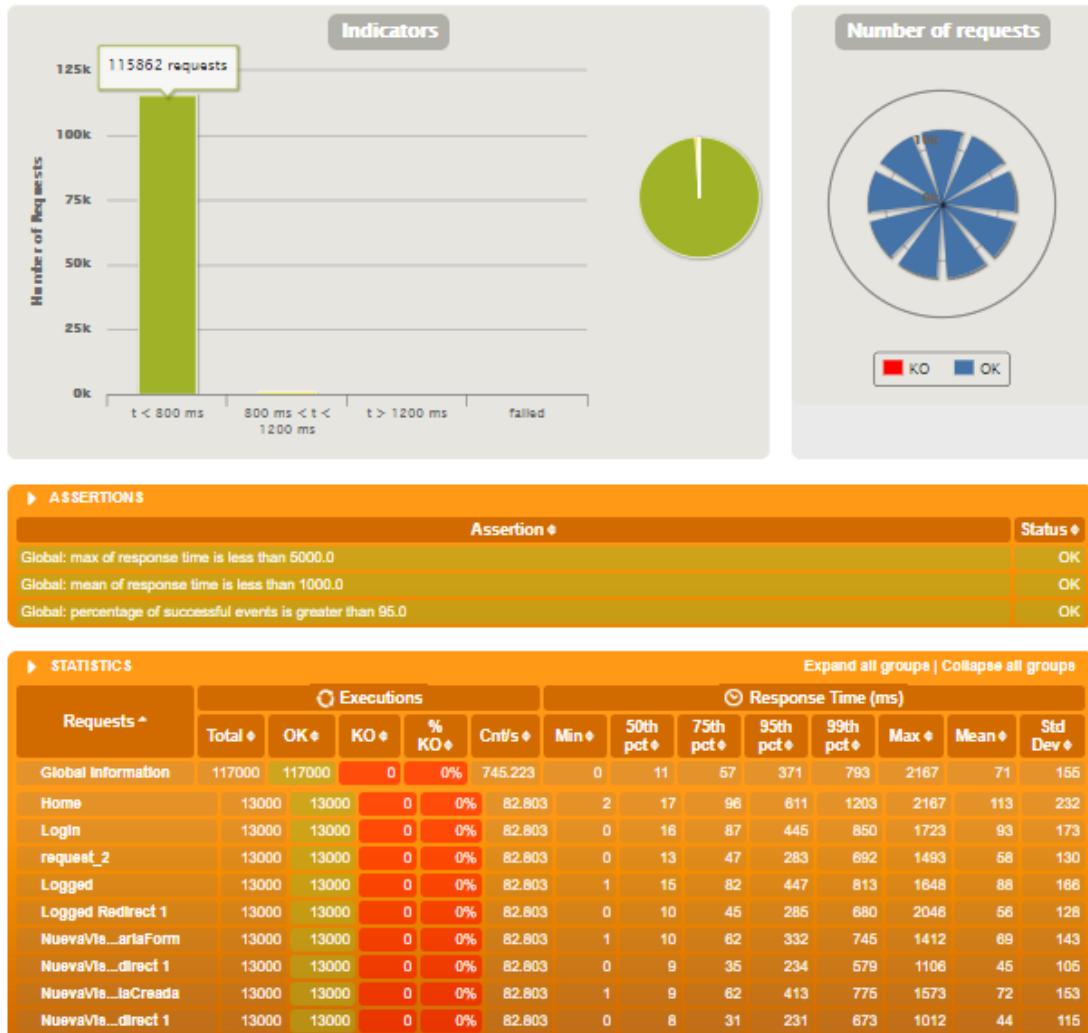
Intentamos aumentar un poco el número de usuarios para averiguar el máximo número de concurrentes, con 15000 usuarios y mismos ajustes de respuesta:

```
setUp(HU19scn.inject(rampUsers(15000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que ya han aparecido algunas peticiones KO, pero en un número muy reducido. Por esto, vamos a intentar ajustar un poco más el número de usuarios concurrentes, esta vez con 13k.

```
setUp(HU19scn.inject(rampUsers(13000) during (100 seconds))
).protocols(httpProtocol)
.assertions(
    global.responseTime.max.lt(5000),
    global.responseTime.mean.lt(1000),
    global.successfulRequests.percent.gt(95)
)
```



Podemos ver que este número sigue dando un excelente resultado, así que podemos decir que el máximo número de usuarios concurrentes para esta historia de usuario es de, aproximadamente, 13000 usuarios.

Test de Rendimiento de la HU20

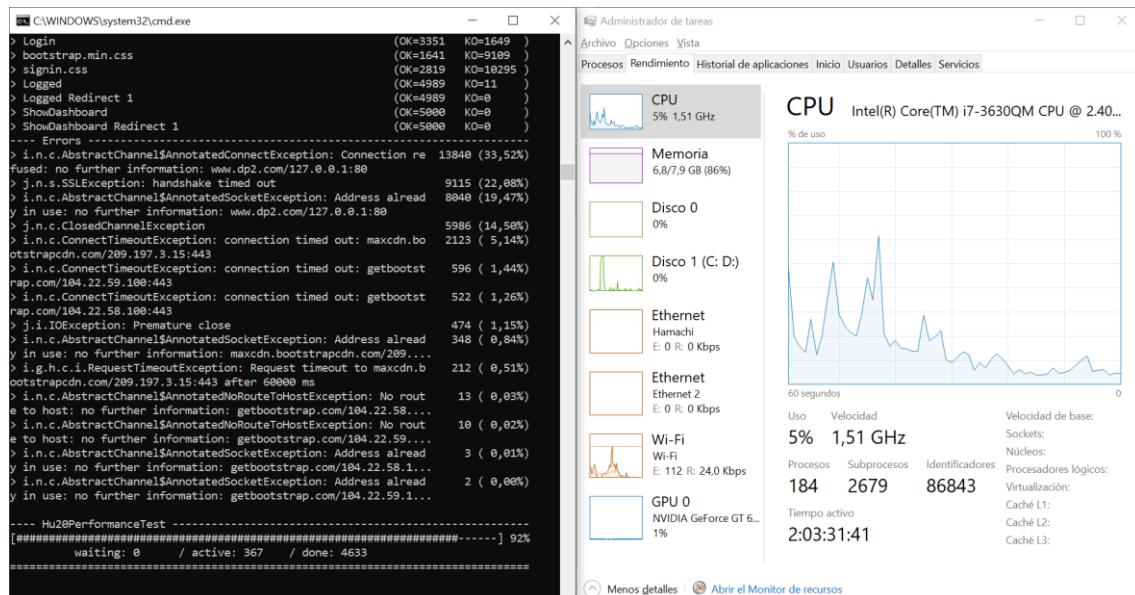
Analizando la Historia de Usuario 20 con Gatling obtenemos los siguientes resultados:

1)

Con 5000 usuarios concurrentes durante 10 segundos:

```
setUp(scn.inject(rampUsers(5000) during (10 seconds)))
    .protocols(httpProtocol)
```

El sistema no funciona correctamente:



2)

Ahora probaremos con 3000 usuarios concurrentes durante 100 segundos, con tiempo de respuesta máximo de 5 segundos, medio de 1 segundo, y porcentaje de éxito de 95%:

```
setUp scn.inject(rampUsers(3000) during (100 seconds))
    .protocols(httpProtocol)
    .assertions(
        global.responseTime.max.lt(5000),
        global.responseTime.mean.lt(1000),
        global.successfulRequests.percent.gt(95))
```



El sistema ha obtenido pocos errores, pero no cumple el objetivo.

3) Lo intentaremos con 1000 usuarios:



El sistema funciona perfectamente.