

Análisis de performance con test bloqueante y no bloqueante con artillery:

1. Inicio el servidor en modo profiler: con **node --prof server.js**
2. ingreso al navegador para generar la petición
3. Realizó el test de carga con Artillery con línea de comando: **artillery quick -c 50 -n 20 "http://localhost:8080/infoNoBloq">artillery_NObloq.txt**
4. Cerramos la terminal y renombramos el archivo Isolate generado como **"no_bloq-v8.log"**
5. Iniciamos **node --prof server.js**
6. Ingreso al navegador para generar la petición
7. Realizó el test de carga con Artillery con línea de comando: **artillery quick -c 50 -n 20 "http://localhost:8080/infoBloq">artillery_bloq.txt**
8. Cerramos la terminal y renombramos el archivo Isolate generado como **"bloq-v8.log"**
9. Decodificamos los dos archivos .log que se crearon:
node.exe --prof-proces bloq-v8.log > result_prof-bloq.txt
node.exe --prof-proces no_bloq-v8.log > result_prof-no_bloq.txt

```
artillery_bloq.txt
DESAFIO14 > conclusionTest > artillery_bloq.txt
1 Phase started: unnamed (index: 0, duration: 1s) 02:27:06(-0300)
2
3 Phase completed: unnamed (index: 0, duration: 1s) 02:27:07(-0300)
4
5 -----
6 Metrics for period to: 02:27:10(-0300) (width: 2.787s)
7 -----
8
9 http.codes.200: .....
10 http.request_rate: .....
11 http.requests: .....
12 http.response time:
13   min: .....
14   max: .....
15   median: .....
16   p95: .....
17   p99: .....
18 http.responses: .....
19 users.completed: .....
20 users.created: .....
21 users.created_by_name.0: .....
22 users.failed: .....
23 users.session_length:
24   min: .....
25   max: .....
26   median: .....
27   p95: .....
28   p99: .....
29
30 All VUs finished. Total time: 3 seconds
31
32 Summary report @ 02:27:09(-0300)
33 -----
34 http.codes.200: .....
35
36 artillery_NObloq.txt
C:\> Users > can_D > OneDrive > Escritorio > BACKEND > DESAFIO14 > conclusionTest > artillery_NObloq.txt
1 Phase started: unnamed (index: 0, duration: 1s) 18:59:59(-0300)
2
3 Phase completed: unnamed (index: 0, duration: 1s) 19:00:00(-0300)
4
5 -----
6 Metrics for period to: 19:00:00(-0300) (width: 0.305s)
7 -----
8
9 http.codes.200: .....
10 http.request_rate: .....
11 http.requests: .....
12 http.response time:
13   min: .....
14   max: .....
15   median: .....
16   p95: .....
17   p99: .....
18 http.responses: .....
19 users.completed: .....
20 users.created: .....
21 users.created_by_name.0: .....
22 users.failed: .....
23 users.session_length:
24   min: .....
25   max: .....
26   median: .....
27   p95: .....
28   p99: .....
29
30 -----
31 Metrics for period to: 19:00:10(-0300) (width: 1.308s)
32 -----
33
34 http.codes.200: .....
35 http.request_rate: .....
36 http.requests: .....
```

-Conclusión: Cuanto menos sincronía más velocidad tiene el servidor.

Autocannon:

Emular 100 conexiones en un periodo de 20 segundos.

1. Configurar el package.json agregando el scripts: "test": "node benchmark.js", "start": "0x server.js"
2. En la consola **npm start**
3. Otra consola y ejecutamos **npm test**

```
Req/Bytes counts sampled once per second.
# of samples: 20
```

```
19k requests in 20.06s, 16 MB read
Running 20s test @ http://localhost:8080/infoNoBlog
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	60 ms	99 ms	180 ms	198 ms	104.46 ms	31.72 ms	331 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	554	554	981	1061	950	114.59	554
Bytes/Sec	464 kB	464 kB	821 kB	888 kB	795 kB	95.9 kB	464 kB

```
Req/Bytes counts sampled once per second.
# of samples: 20
```

```
19k requests in 20.09s, 15.9 MB read
```

```
Can_D@Candeladominguez MINGW64 ~/OneDrive/Esritorio/BACKEND/DESAFIOS/DESAFI014 (main)
```

```
node server.js
```

cold  hot
* optimized ~ unoptimized  search functions

