

Desafio 14

PROF

[Summary]:

ticks	total	nonlib	name
4	0.0%	100.0%	JavaScript
0	0.0%	0.0%	C++
9	0.1%	225.0%	GC
9324	100.0%		Shared libraries

Podemos ver que lo que hace más trabajo son las librerías de node, ya que lo que hicimos es algo muy ligero, no tiene iteraciones ni es un proceso bloqueante.

Artillery

Al ser una ruta con un proceso muy ligero podemos ver que la respuesta media de /info es de 18ms

```
1 Phase started: unnamed (index: 0, duration: 1s) 15:38:29(-0300)
2
3 Phase completed: unnamed (index: 0, duration: 1s) 15:38:30(-0300)
4
5 -----
6 Metrics for period to: 15:38:40(-0300) (width: 2.050s)
7 -----
8
9 http.codes.200: ..... 1000
10 http.request_rate: ..... 486/sec
11 http.requests: ..... 1000
12 http.response_time:
13   min: ..... 1
14   max: ..... 70
15   median: ..... 18
16   p95: ..... 34.8
17   p99: ..... 46.1
18 http.responses: ..... 1000
19 vusers.completed: ..... 20
20 vusers.created: ..... 19
21 vusers.created_by_name.0: ..... 19
22 vusers.failed: ..... 0
23 vusers.session_length:
24   min: ..... 977.2
25   max: ..... 1485.5
26   median: ..... 1326.4
27   p95: ..... 1465.9
28   p99: ..... 1465.9
29
30
31
```

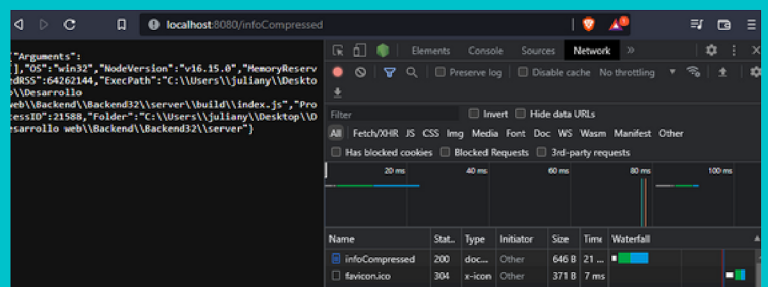
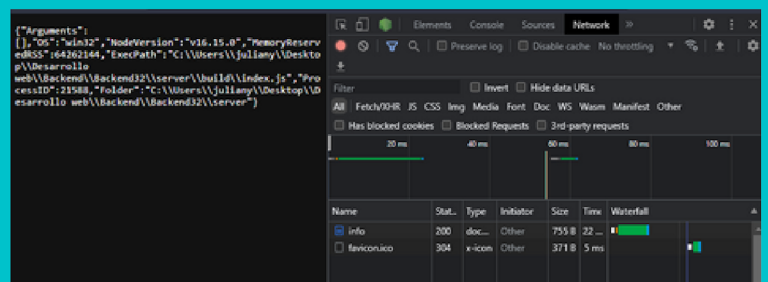
Compression

Acá podemos ver que la ruta con gzip tiene menor tamaño y tardo ligeramente menos, 1ms, que el que no está comprimido.

```
// Desafios
const info = {
  Arguments: config.arguments,
  OS: config.os,
  NodeVersion: config.NodeVersion,
  MemoryReservedRSS: config.MemoryReservedRSS,
  ExecPath: config.ExecPath,
  ProcessID: config.ProcessID,
  Folder: config.Folder,
}

app.get('/info', (req, res) => {
  res.send(info)
})

app.get('/infoCompressed', compression(), (req, res) => {
  res.send(info)
})
```



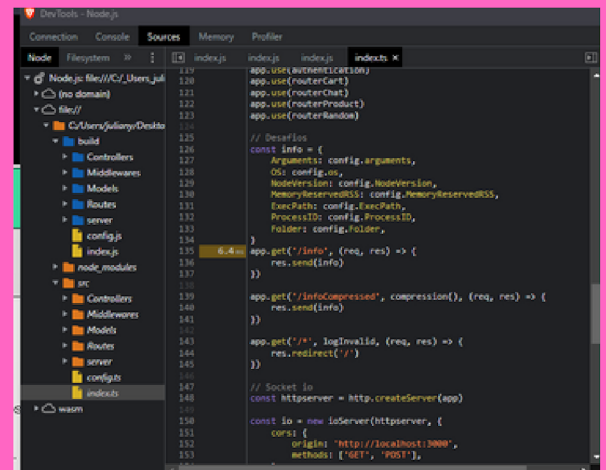
Logging

Cree la ruta /api/random-error para que genere un error y lo muestre en la consola. Así también, puse un middleware para que detecte las rutas inexistentes.

```
info: Worker 21820 connected to Mongo {"timestamp":"2022-06-30T14:21:57.898Z"}
info: Listening from 8080 - http://localhost:8080 {"timestamp":"2022-06-30T14:21:59.026Z"}
info: Worker 10956 start on port 8080 (Fork) {"timestamp":"2022-06-30T14:21:59.028Z"}
info: Route: /api/randoms-error, Method: GET {"timestamp":"2022-06-30T14:21:59.782Z"}
error: Error with Api Products: Error: Error random {"timestamp":"2022-06-30T14:21:59.784Z"}
error: Error with Api Products: Error: Error random {"timestamp":"2022-06-30T14:21:59.784Z"}
error: Error with Api Products: Error: Error random {"timestamp":"2022-06-30T14:21:59.784Z"}
info: Route: /api/randoms-errorasdasd, Method: GET {"timestamp":"2022-06-30T14:22:03.633Z"}
warn: Route: /api/randoms-errorasdasd, method: GET. Invalid rute {"timestamp":"2022-06-30T14:22:03.635Z"}
warn: Route: /api/randoms-errorasdasd, method: GET. Invalid rute {"timestamp":"2022-06-30T14:22:03.635Z"}
[]
```

Inspect

Al tener el código en TypeScript, realmente no se donde tengo que buscar en especifico. Vi los archivos en JS y no me mostraban el tiempo de ejecución pero los archivos de TS si. Podemos ver que la ruta /info tardo 6.4ms en enviar la información.



Autocannon

A medida que va haciendo los request, la latencia sube. En cuanto a los bytes tiene un promedio de 875kB/s.

100 connections

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	59 ms	74 ms	142 ms	155 ms	87.36 ms	94.62 ms	1497 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	1300	1400	1160	322.82	800
Bytes/Sec	0 B	0 B	981 kB	1.06 MB	875 kB	244 kB	604 kB

Req/Bytes counts sampled once per second.
of samples: 20
23k requests in 20.54s, 17.5 MB read

0x

Me sorprende que lo que se ve parezca que tenga procesos bloqueantes, pero en realidad no. No se si puede interferir algo TS, pero realmente no entiendo mucho esta grafica. La mayoría de las cosas que hay en la muestra son librerías de NodeJS



Conclusión

Tener pruebas de rendimiento en nuestra aplicación nos ayuda bastante para poder mejorar nuestro código y visualizar los posibles errores.

Obviamente, hay que mejorar varios factores, principalmente aprender a leer bien los graficos e identificar los problemas. Ya es cuestión mía para ponerme a investigar más sobre el tema.