"/info" - Bytes devueltos:

Con compresión

Uso de memoria	92270592

Sin compresión

Uso de memoria	138641408

ARTILLERY CON/SIN CONSOLE LOG

artillery quick --count 50 -n 20 http://localhost/info > result_clg.txt
artillery quick --count 50 -n 20 http://localhost/info > result_sin-clg.txt

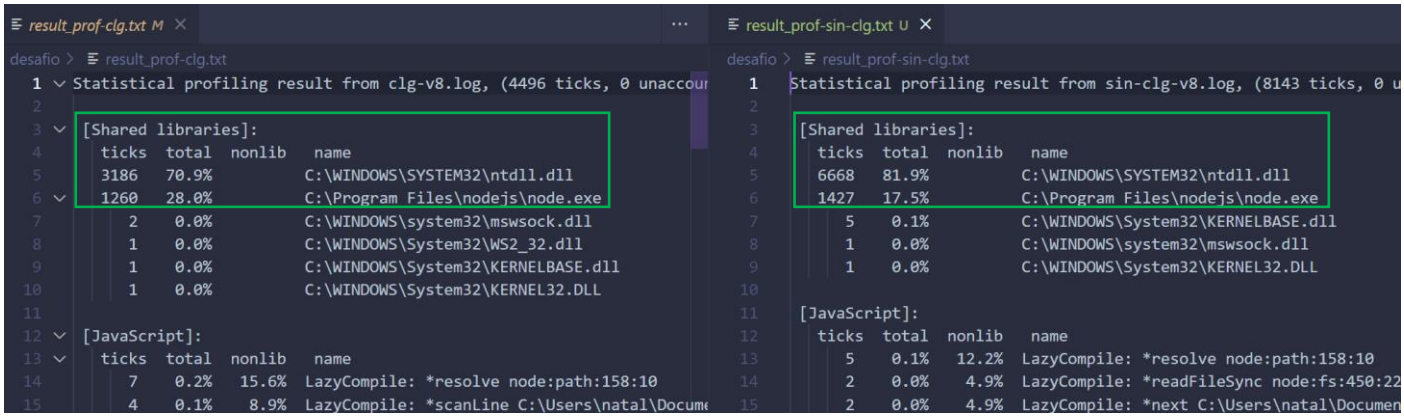
```
desafio > routes > JS infoRouter.js > ...
const parseArgs = require("minimist");
const os = require("os");

const infoRouter = (req, res) => {
  try {
    const args = parseArgs(process.argv.slice(2));
    const info = {
      argumentos: JSON.stringify(args),
      directorioActual: process.cwd(),
      idProceso: process.pid,
      vNode: process.version,
      rutaEjecutable: process.execPath,
      sistemaOperativo: process.platform,
      memoria: JSON.stringify(process.memoryUsage().rss, null, 2),
      processNum: os.cpus().length,
    };
    console.log(info);
    res.render("pages/info", info);
  } catch (error) {
    res.render(error.message);
  }
};

module.exports = infoRouter
```

```
desafio > routes > JS infoRouter.js > [ø] <unknown>
const parseArgs = require("minimist");
const os = require("os");

const infoRouter = (req, res) => {
  try {
    const args = parseArgs(process.argv.slice(2));
    const info = {
      argumentos: JSON.stringify(args),
      directorioActual: process.cwd(),
      idProceso: process.pid,
      vNode: process.version,
      rutaEjecutable: process.execPath,
      sistemaOperativo: process.platform,
      memoria: JSON.stringify(process.memoryUsage().rss, null, 2),
      processNum: os.cpus().length,
    };
    // console.log(info);
    res.render("pages/info", info);
  } catch (error) {
    res.render(error.message);
  }
};

module.exports = infoRouter
```

PROFILING CON/SIN CONSOLE LOG

node --prof server.js 8080 FORK

node.exe --prof-process clg-v8.log > result_prof-clg.txt
node.exe --prof-process sin-clg-v8.log > result_prof-sin-clg.txt

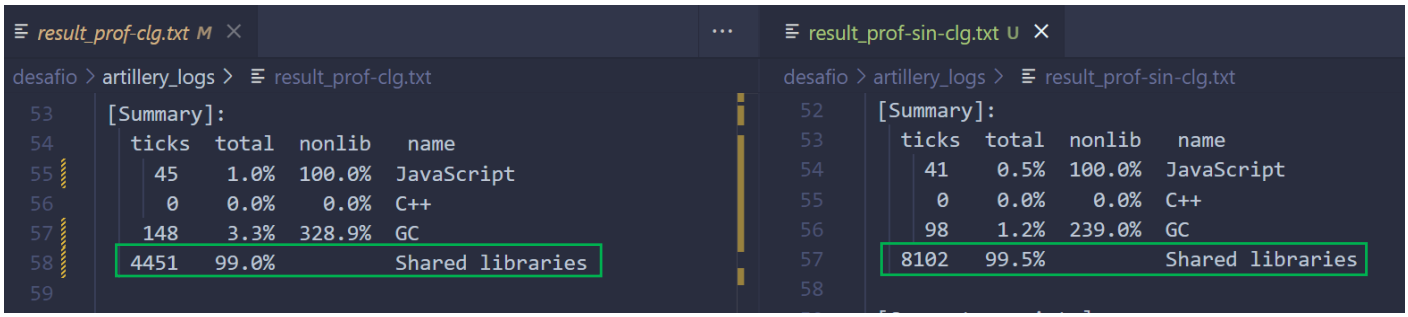


result_prof-clg.txt M

```
desafio > result_prof-clg.txt
1  Statistical profiling result from clg-v8.log, (4496 ticks, 0 unaccour

   [Shared libraries]:
     ticks  total  nonlib   name
      3186   70.9%          C:\WINDOWS\SYSTEM32\ntdll.dll
      1260   28.0%          C:\Program Files\nodejs\node.exe
         2    0.0%          C:\WINDOWS\system32\mswsock.dll
         1    0.0%          C:\WINDOWS\System32\WS2_32.dll
         1    0.0%          C:\WINDOWS\System32\KERNELBASE.dll
         1    0.0%          C:\WINDOWS\System32\KERNEL32.DLL

   [JavaScript]:
     ticks  total  nonlib   name
         7    0.2%   15.6%  LazyCompile: *resolve node:path:158:10
         4    0.1%    8.9%  LazyCompile: *scanLine C:\Users\natal\Docume
```

result_prof-sin-clg.txt U

```
desafio > result_prof-sin-clg.txt
1  Statistical profiling result from sin-clg-v8.log, (8143 ticks, 0 u

   [Shared libraries]:
     ticks  total  nonlib   name
      6668   81.9%          C:\WINDOWS\SYSTEM32\ntdll.dll
      1427   17.5%          C:\Program Files\nodejs\node.exe
         5    0.1%          C:\WINDOWS\System32\KERNELBASE.dll
         1    0.0%          C:\WINDOWS\system32\mswsock.dll
         1    0.0%          C:\WINDOWS\System32\KERNEL32.DLL

   [JavaScript]:
     ticks  total  nonlib   name
         5    0.1%   12.2%  LazyCompile: *resolve node:path:158:10
         2    0.0%    4.9%  LazyCompile: *readFileSync node:fs:450:22
         2    0.0%    4.9%  LazyCompile: *next C:\Users\natal\Documen
```



result_prof-clg.txt M

```
desafio > artillery_logs > result_prof-clg.txt
   [Summary]:
     ticks  total  nonlib   name
        45   1.0%  100.0%  JavaScript
         0   0.0%    0.0%  C++
       148   3.3%  328.9%  GC
      4451  99.0%          Shared libraries
```

result_prof-sin-clg.txt U

```
desafio > artillery_logs > result_prof-sin-clg.txt
   [Summary]:
     ticks  total  nonlib   name
        41   0.5%  100.0%  JavaScript
         0   0.0%    0.0%  C++
        98   1.2%  239.0%  GC
      8102  99.5%          Shared libraries
```

Análisis: *Se observa que tiene mayor cantidad de Ticks sin console.log (por lo visto en clase debería ser menor que con el console.log)*

AUTOCANNON INSPECT CON CONSOLE LOG vs AUTOCANNON INSPECT SIN CONSOLE LOG

Sources Memory

infoRouter.js ×

```
1   const parseArgs = require("minimist");
2   const os = require("os");
3
4   const infoRouter = (req, res) => {
5     try {
6  11.4 ms    const args = parseArgs(process.argv.slice(2));
7   1.7 ms    const info = {
8  13.6 ms      argumentos: JSON.stringify(args),
9   1.2 ms      directorioActual: process.cwd(),
10            idProceso: process.pid,
11  0.2 ms      vNode: process.version,
12  0.3 ms      rutaEjecutable: process.execPath,
13            sistemaOperativo: process.platform,
14  5.1 ms      memoria: JSON.stringify(process.memoryUsage().rss, null, 2),
15  5.6 ms            processNum: os.cpus().length,
16          };
17 11.4 ms        console.log(info);
18 20.0 ms    res.render("pages/info", info);
19    } catch (error) {
20      res.render(error.message);
21    }
22  };
23
24  module.exports = infoRouter
```

Sources Memory

infoRouter.js ×

```
1   const parseArgs = require("minimist");
2   const os = require("os");
3
4   const infoRouter = (req, res) => {
5     try {
6   8.7 ms    const args = parseArgs(process.argv.slice(2));
7   0.5 ms    const info = {
8   8.8 ms      argumentos: JSON.stringify(args),
9   0.2 ms      directorioActual: process.cwd(),
10  0.3 ms      idProceso: process.pid,
11  0.2 ms      vNode: process.version,
12            rutaEjecutable: process.execPath,
13            sistemaOperativo: process.platform,
14  4.8 ms      memoria: JSON.stringify(process.memoryUsage().rss, null, 2),
15  5.6 ms            processNum: os.cpus().length,
16          };
17            // console.log(info);
18 22.8 ms    res.render("pages/info", info);
19    } catch (error) {
20      res.render(error.message);
21    }
22  };
23
24  module.exports = infoRouter
```

Análisis: *Se observa que sin console.log el Render demora 2ms más (por lo visto en clase debería ser menor que con el console.log)*

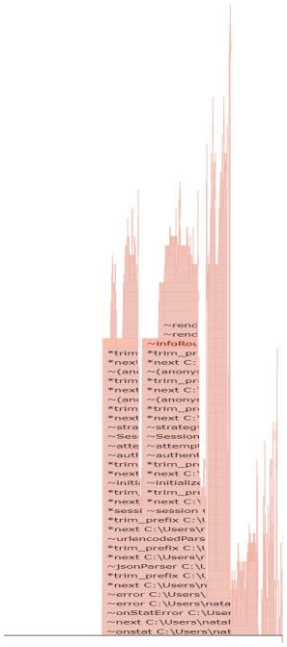
AUTOCANNON CON CONSOLE LOG

```
natal@NUNE-DESP018 MINGW64 ~/Documents/GitHub/Backend_MERN/desafio (loadbalancer)
$ npm test

> backend_mern@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	480 ms	901 ms	2761 ms	2830 ms	984.84 ms	473.5 ms	3123 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	6	6	100	155	95.35	40.91	6
Bytes/Sec	11.4 kB	11.4 kB	191 kB	295 kB	182 kB	77.9 kB	11.4 kB

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

2k requests in 20.31s, 3.63 MB read
```



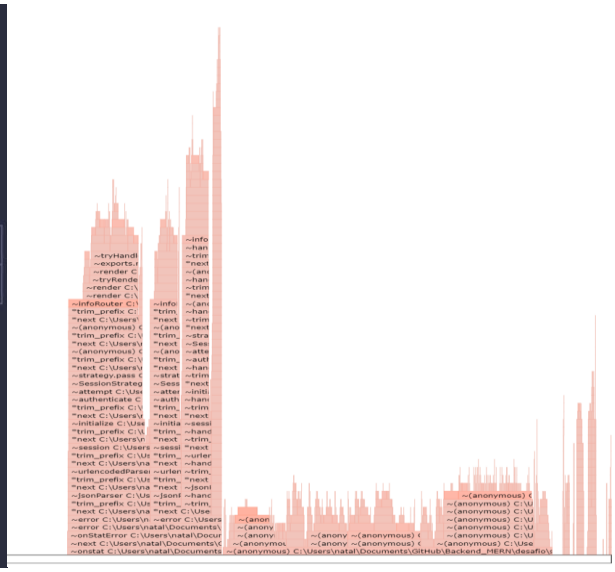
AUTOCANNON SIN CONSOLE LOG

```
natal@NUNE-DESP018 MINGW64 ~/Documents/GitHub/Backend_MERN/desafio (loadbalancer)
$ npm test

> backend_mern@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	14747 ms	16961 ms	19171 ms	19234 ms	16875.75 ms	1263.12 ms	19266 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	0	276	63.4	100.06	143
Bytes/Sec	0 B	0 B	0 B	526 kB	121 kB	191 kB	272 kB

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

28k requests in 20.33s, 2.42 MB read
27k errors (0 timeouts)
```



Análisis: *Se observa que tiene mayor Latencia sin console.log pero la cantidad de Requests es mayor.*