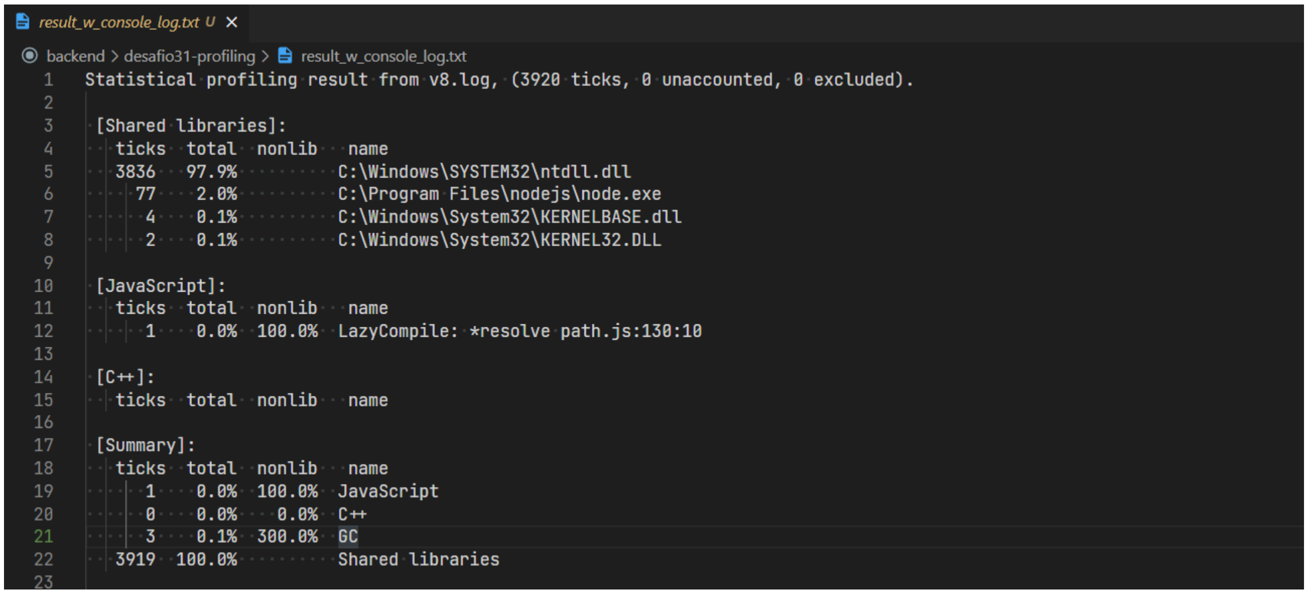
Profiling

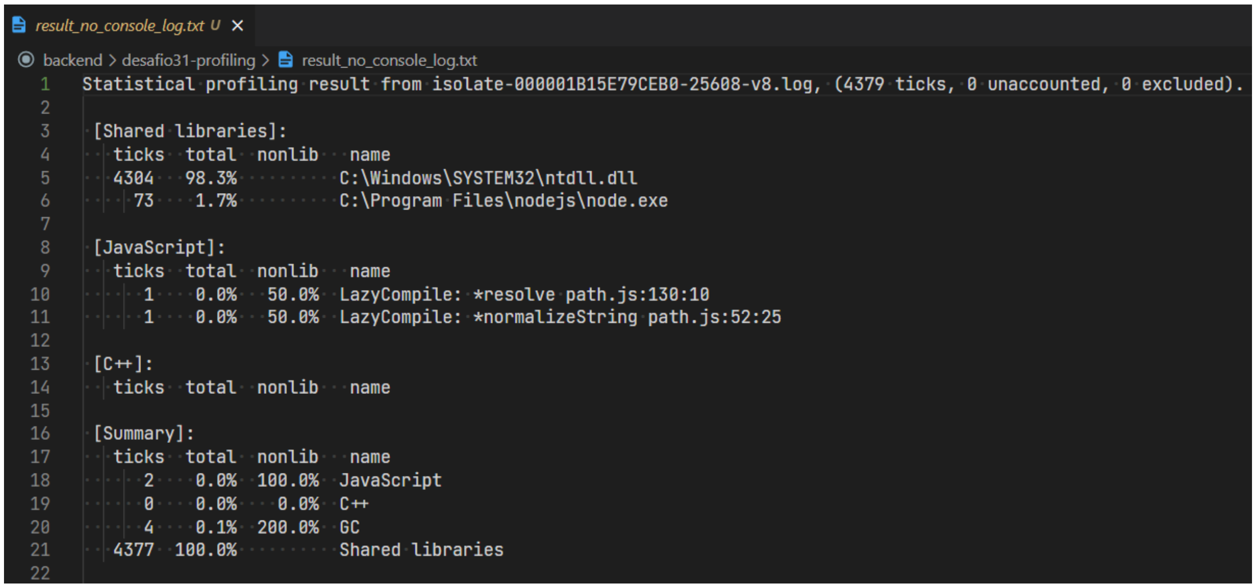
Tests with Artillery and default NodeJS profiler

With console.log


result_w_console_log.txt U
backend > desafio31-profiling > result_w_console_log.txt

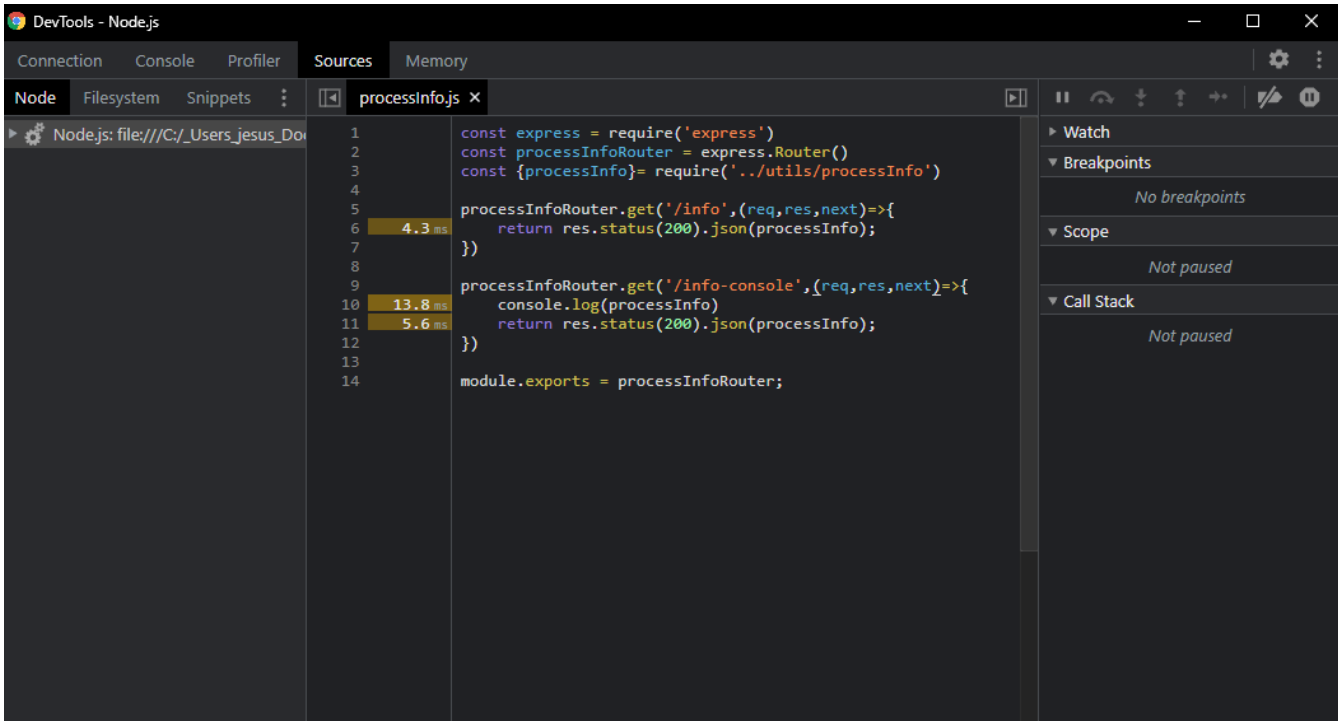
```
Statistical profiling result from v8.log, (3920 ticks, 0 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
   3836   97.9%          C:\Windows\SYSTEM32\ntdll.dll
     77    2.0%          C:\Program Files\nodejs\node.exe
      4    0.1%          C:\Windows\System32\KERNELBASE.dll
      2    0.1%          C:\Windows\System32\KERNEL32.DLL

 [JavaScript]:
   ticks  total  nonlib   name
      1    0.0%  100.0%  LazyCompile: *resolve path.js:130:10

 [C++]:
   ticks  total  nonlib   name

 [Summary]:
   ticks  total  nonlib   name
      1    0.0%  100.0%  JavaScript
      0    0.0%    0.0%  C++
      3    0.1%  300.0%  GC
   3919  100.0%          Shared libraries
```

Without console.log


result_no_console_log.txt U
backend > desafio31-profiling > result_no_console_log.txt

```
Statistical profiling result from isolate-000001B15E79CEB0-25608-v8.log, (4379 ticks, 0 unaccounted, 0 excluded).

 [Shared libraries]:
   ticks  total  nonlib   name
   4304   98.3%          C:\Windows\SYSTEM32\ntdll.dll
     73    1.7%          C:\Program Files\nodejs\node.exe

 [JavaScript]:
   ticks  total  nonlib   name
      1    0.0%   50.0%  LazyCompile: *resolve path.js:130:10
      1    0.0%   50.0%  LazyCompile: *normalizeString path.js:52:25

 [C++]:
   ticks  total  nonlib   name

 [Summary]:
   ticks  total  nonlib   name
      2    0.0%  100.0%  JavaScript
      0    0.0%    0.0%  C++
      4    0.1%  200.0%  GC
   4377  100.0%          Shared libraries
```

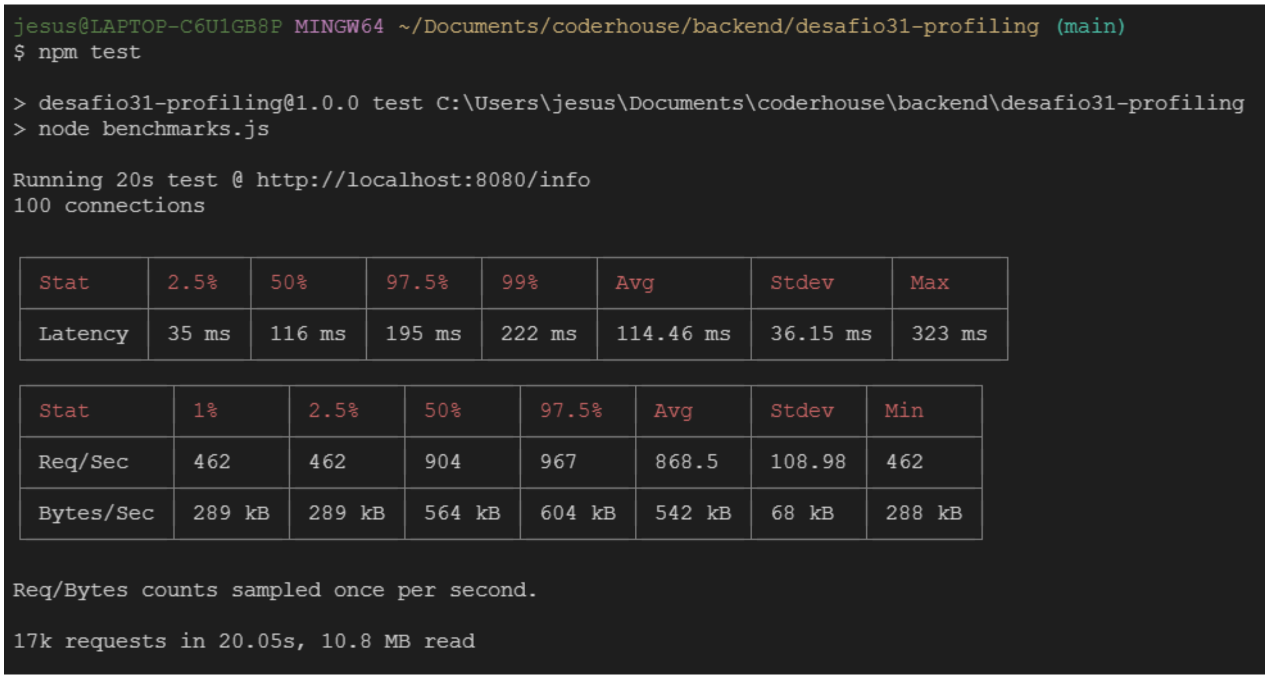
Test with Artillery and NodeJS inspect



```
const express = require('express')
const processInfoRouter = express.Router()
const {processInfo}= require('../utils/processInfo')

processInfoRouter.get('/info',(req,res,next)=>{
    return res.status(200).json(processInfo);
})

processInfoRouter.get('/info-console',(req,res,next)=>{
    console.log(processInfo)
    return res.status(200).json(processInfo);
})

module.exports = processInfoRouter;
```

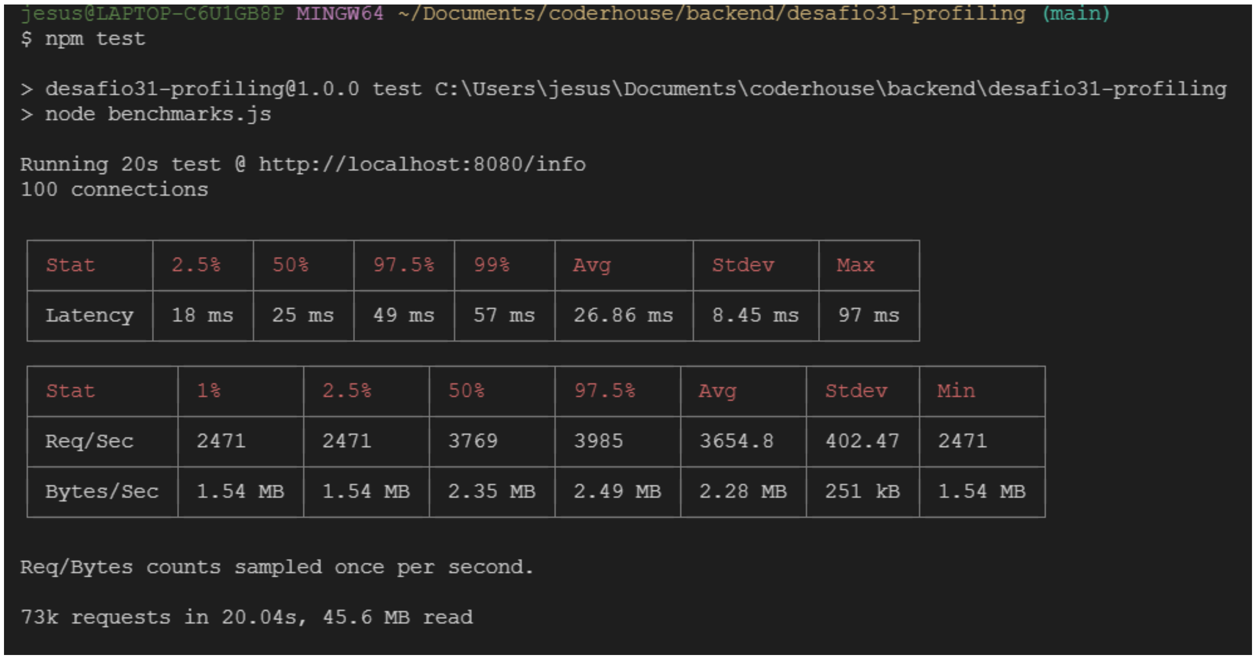
Autocanon tests

With console.log on route



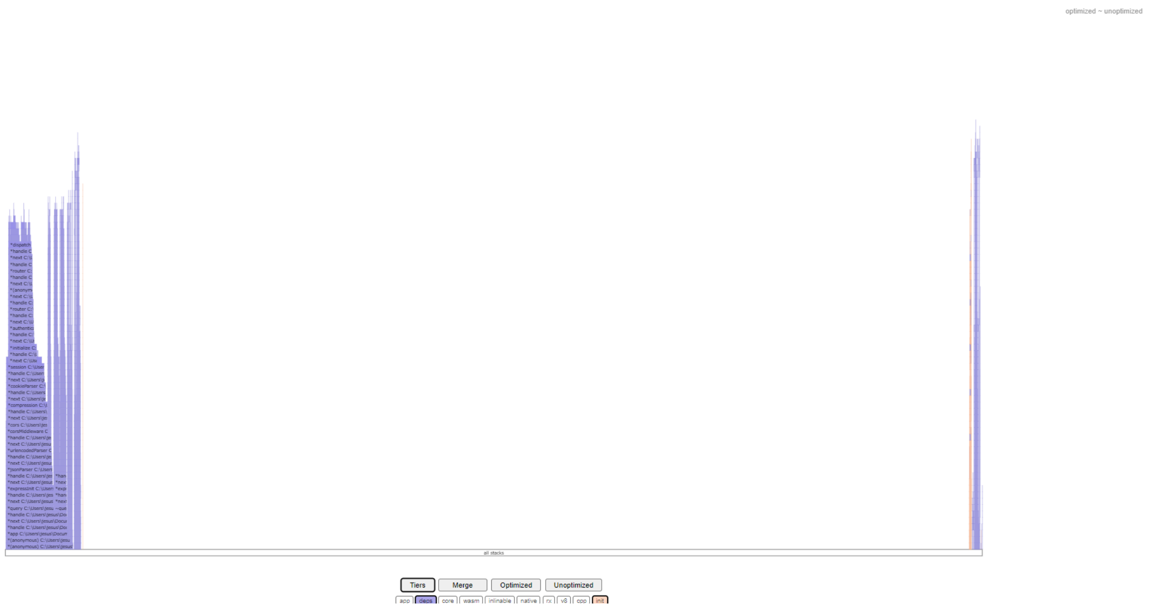
```
jesus@LAPTOP-C6U1GB8P MINGW64 ~/Documents/coderhouse/backend/desafio31-profiling (main)
$ npm test

> desafio31-profiling@1.0.0 test C:\Users\jesus\Documents\coderhouse\backend\desafio31-profiling
> node benchmarks.js

Running 20s test @ http://localhost:8080/info
100 connections

┌─────────┬───────┬────────┬────────┬────────┬───────────┬──────────┬────────┐
│ Stat    │ 2.5%  │ 50%    │ 97.5%  │ 99%    │ Avg       │ Stdev    │ Max    │
├─────────┼───────┼────────┼────────┼────────┼───────────┼──────────┼────────┤
│ Latency │ 35 ms │ 116 ms │ 195 ms │ 222 ms │ 114.46 ms │ 36.15 ms │ 323 ms │
└─────────┴───────┴────────┴────────┴────────┴───────────┴──────────┴────────┘
┌───────────┬────────┬────────┬────────┬────────┬────────┬────────┬────────┐
│ Stat      │ 1%     │ 2.5%   │ 50%    │ 97.5%  │ Avg    │ Stdev  │ Min    │
├───────────┼────────┼────────┼────────┼────────┼────────┼────────┼────────┤
│ Req/Sec   │ 462    │ 462    │ 904    │ 967    │ 868.5  │ 108.98 │ 462    │
├───────────┼────────┼────────┼────────┼────────┼────────┼────────┼────────┤
│ Bytes/Sec │ 289 kB │ 289 kB │ 564 kB │ 604 kB │ 542 kB │ 68 kB  │ 288 kB │
└───────────┴────────┴────────┴────────┴────────┴────────┴────────┴────────┘

Req/Bytes counts sampled once per second.

17k requests in 20.05s, 10.8 MB read
```

Without console.log on route



```
jesus@LAPTOP-C6U1GB8P MINGW64 ~/Documents/coderhouse/backend/desafio31-profiling (main)
$ npm test

> desafio31-profiling@1.0.0 test C:\Users\jesus\Documents\coderhouse\backend\desafio31-profiling
> node benchmarks.js

Running 20s test @ http://localhost:8080/info
100 connections

┌─────────┬───────┬───────┬───────┬───────┬──────────┬─────────┬───────┐
│ Stat    │ 2.5%  │ 50%   │ 97.5% │ 99%   │ Avg      │ Stdev   │ Max   │
├─────────┼───────┼───────┼───────┼───────┼──────────┼─────────┼───────┤
│ Latency │ 18 ms │ 25 ms │ 49 ms │ 57 ms │ 26.86 ms │ 8.45 ms │ 97 ms │
└─────────┴───────┴───────┴───────┴───────┴──────────┴─────────┴───────┘
┌───────────┬─────────┬─────────┬─────────┬─────────┬─────────┬────────┬─────────┐
│ Stat      │ 1%      │ 2.5%    │ 50%     │ 97.5%   │ Avg     │ Stdev  │ Min     │
├───────────┼─────────┼─────────┼─────────┼─────────┼─────────┼────────┼─────────┤
│ Req/Sec   │ 2471    │ 2471    │ 3769    │ 3985    │ 3654.8  │ 402.47 │ 2471    │
├───────────┼─────────┼─────────┼─────────┼─────────┼─────────┼────────┼─────────┤
│ Bytes/Sec │ 1.54 MB │ 1.54 MB │ 2.35 MB │ 2.49 MB │ 2.28 MB │ 251 kB │ 1.54 MB │
└───────────┴─────────┴─────────┴─────────┴─────────┴─────────┴────────┴─────────┘

Req/Bytes counts sampled once per second.

73k requests in 20.04s, 45.6 MB read
```

Flame test with Autocannon



To the left, the console.log test to the route “/info-console”
To the right the test to the route “/info” with no console.log

Conclusion

De los test se denota que el console.log ciertamente bloquea o ralentiza la ejecución de los procesos de la app.

En este caso no es tan perceptible (13 milisegundos tarde ese console.log), pero en procesos bloqueantes de mayor tamaño si podría ser un problema.