Profiling

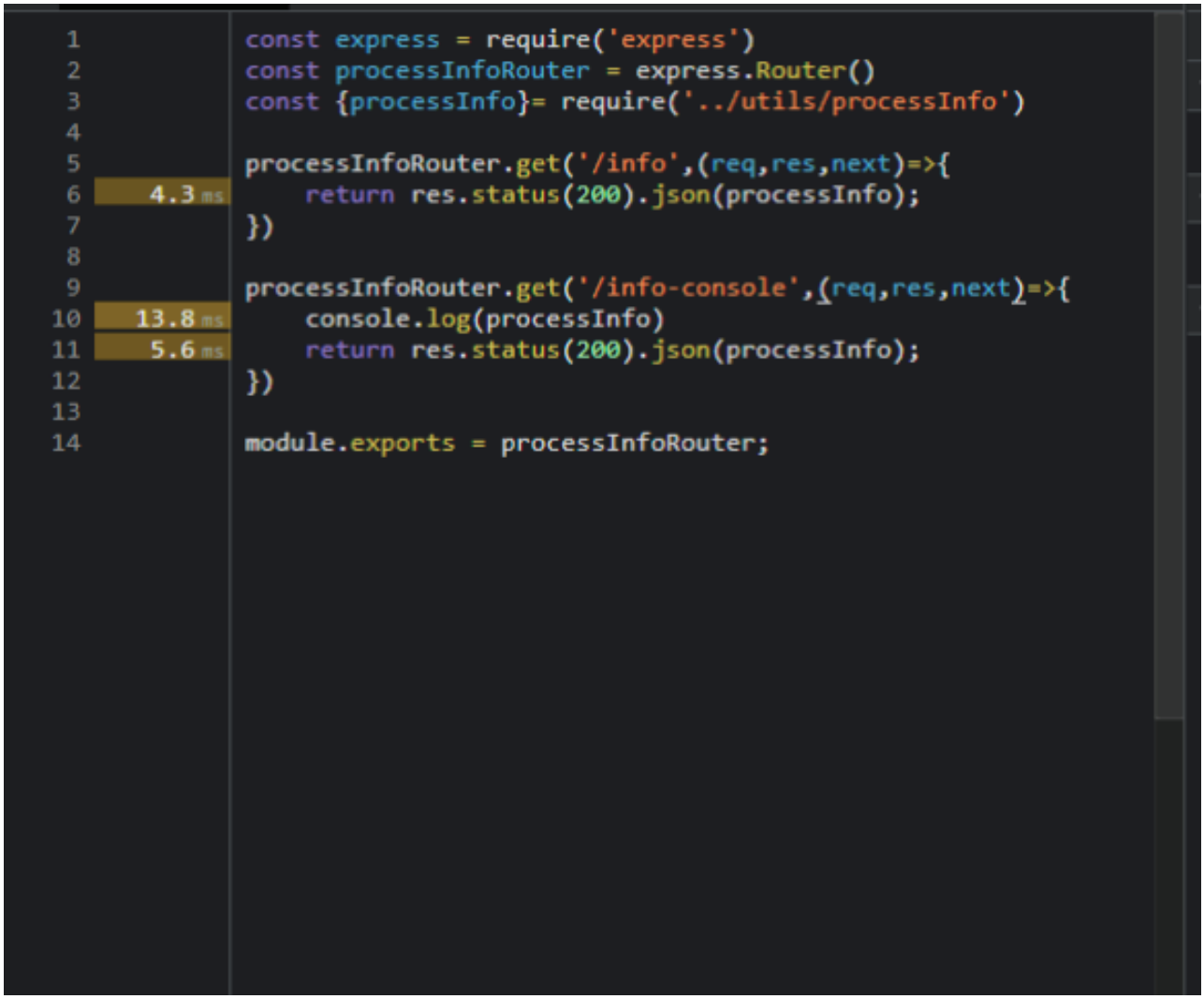
Test con Artillery y default NodeJs profiler

Con console.log

```
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
```

```
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 con Artillery y NodeJS inspect



```
1   const express = require('express')
2   const processInfoRouter = express.Router()
3   const {processInfo}= require('../utils/processInfo')
4
5   processInfoRouter.get('/info',(req,res,next)=>{
6       return res.status(200).json(processInfo);
7   })
8
9   processInfoRouter.get('/info-console',(req,res,next)=>{
10      console.log(processInfo)
11      return res.status(200).json(processInfo);
12  })
13
14  module.exports = processInfoRouter;
```

4.3 ms
13.8 ms
5.6 ms

Autocannon test

Con console.log en route

```
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
```

Sin console.log en route

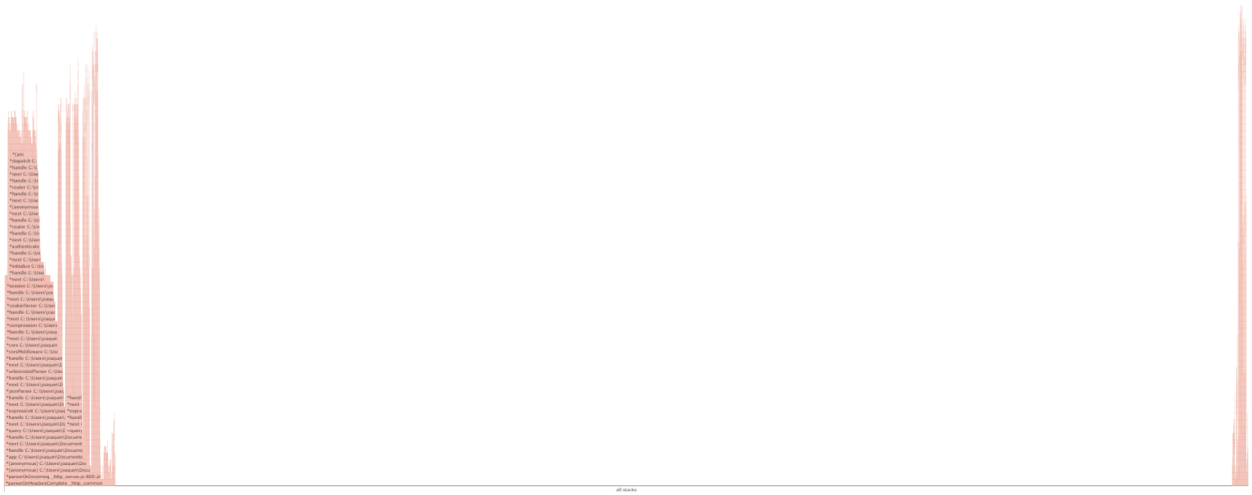
```
Running 20s test @ http://localhost:8080/info
100 connections
```

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 con Autocannon



Izquierda: console.log test a la ruta "/info-console"

- Derecha el test a la ruta "/info" sin console.log

Conclusión

En los test se nota que el console.log bloquea o ralentiza la ejecución de los procesos de la aplicación.

En el caso del ejemplo no es muy perceptible (13 milisegundos tarde), pero en procesos bloqueantes de mayor tamaño si podría ser un problema.