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
          2.0%
                        C:\Program Files\nodejs\node.exe
          0.1%
                        C:\Windows\System32\KERNELBASE.dll
     - 4
          0.1%
                       C:\Windows\System32\KERNEL32.DLL
[JavaScript]:
  ticks total nonlib
   1 -- 0.0% 100.0% LazyCompile: *resolve path.js:130:10
[C++]:
  ticks total nonlib
[Summary]:
  ticks total nonlib
                        name
                       JavaScript
          0.0% 100.0%
    - 0 -
          0.0%
                0.0%
                        C++
          0.1% 300.0%
                        GC
  3919 100.0%
                        Shared libraries
```

Sin console.log

```
Statistical profiling result from isolate-000001B15E79CEB0-25608-v8.log, (4379 ticks, 0 unaccounted, 0 excluded).
[Shared libraries]:
  ticks total nonlib
                        name
                       C:\Windows\SYSTEM32\ntdll.dll
                       C:\Program Files\nodejs\node.exe
   73
        1.7%
[JavaScript]:
  ticks total nonlib
                       - name
         0.0%
                - 50.0% LazyCompile: *resolve path.js:130:10
                50.0% LazyCompile: *normalizeString path.js:52:25
          0.0%
[C++]:
  ticks total nonlib
[Summary]:
  ticks total nonlib
                       - name
    2 0.0% 100.0% JavaScript
        0.0%
     - 0 -
                0.0% C++
         0.1% 200.0% GC
  4377 100.0%
                       Shared libraries
```

Test con Artillery y NodeJS inspect

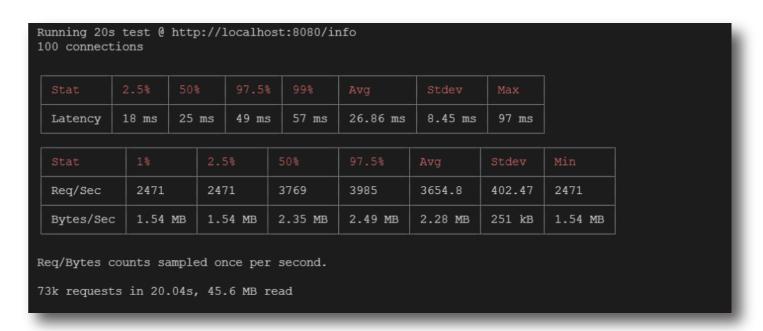
```
const express = require('express')
             const processInfoRouter = express.Router()
             const {processInfo}= require('../utils/processInfo')
            processInfoRouter.get('/info',(req,res,next)=>{
      4.3 ms
 6
                 return res.status(200).json(processInfo);
             })
            processInfoRouter.get('/info-console',(req,res,next)=>{
     13.8 ms
                 console.log(processInfo)
10
      5.6 ms
                 return res.status(200).json(processInfo);
11
12
             })
             module.exports = processInfoRouter;
```

Autocannon test

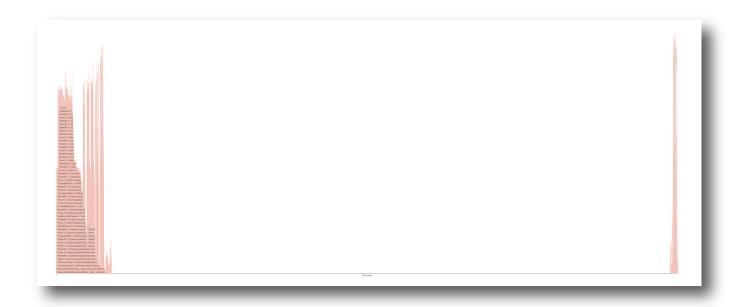
Con console.log en route

tat	2.5% 50		0%		97.5%		99%		7 g	Stdev		Max	
Latency 35 ms		116 ms		195 ms		222 ms		114.46 ms		36.15 ms		323 ms	
Stat	tat 1%		2.5%				97.5%		Avg	Stdev	Min		
Req/Sec	462	462			904		967		868.5	108.98	4	462	
Bytes/Sec	289 1	cВ	289 1	сВ	564 k	ß	604 k	сВ	542 kB	68 kB	2	88 kB	

Sin console.log en route



Flame test con Autocannon



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