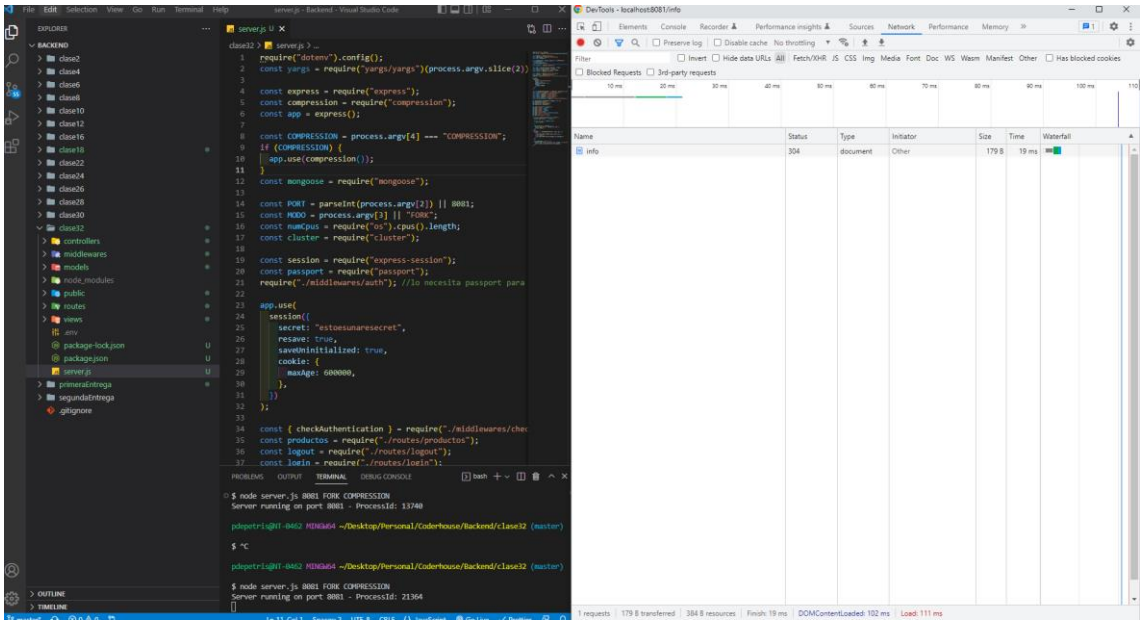
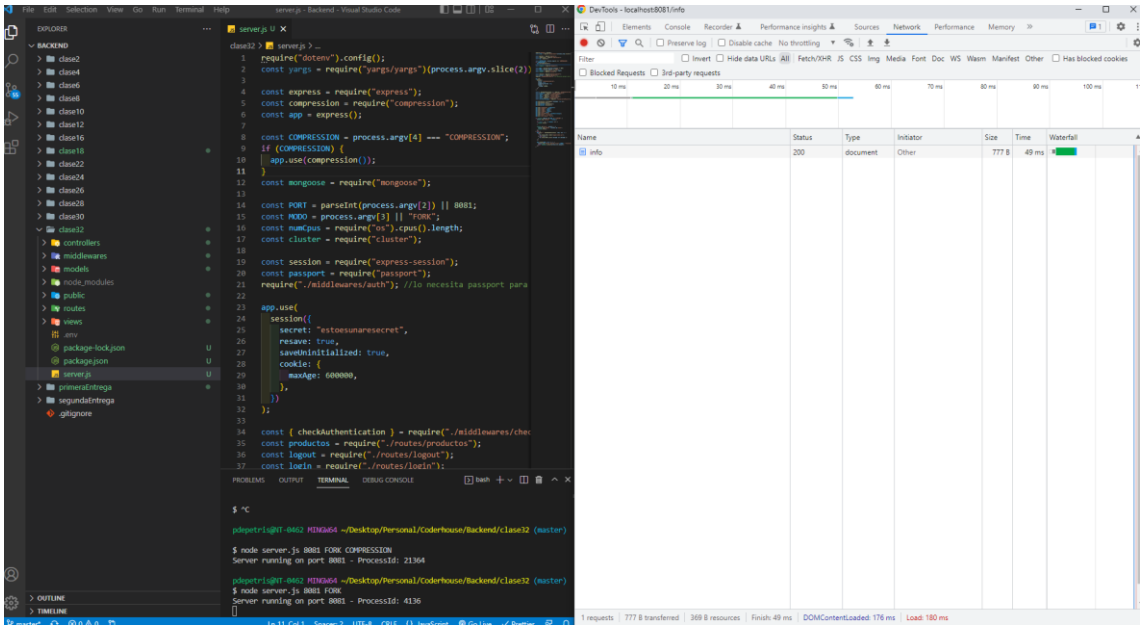
Prueba compression:





La versión comprimida tiene casi 600 bytes menos.

PERFILAMIENTO DE SERVIDOR CON NODE BUILT IN PROFILER

en /info sin console.log(info) NO BLOQUEANTE

$ artillery quick --count 50 -n 40 http://localhost:8081/info >result_nobloq.txt

tatistical profiling result from nobloq-v8.log, (8828 ticks, 0 unaccounted, 0 excluded).

Statistical profiling result from nobloq-v8.log, (8828 ticks, 0 unaccounted, 0 excluded).

[Shared libraries]:

ticks	total	nonlib	name
8372	94.8%		C:\WINDOWS\SYSTEM32\ntdll.dll
451	5.1%		C:\Program Files\nodejs\node.exe
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
1	0.0%	50.0%	Function: ^toNamespacedPath node:path:618:19
1	0.0%	50.0%	Function: ^Module._load node:internal/modules/cjs/loader:771:24

[C++]:

ticks	total	nonlib	name

[Summary]:

ticks	total	nonlib	name
2	0.0%	100.0%	JavaScript
0	0.0%	0.0%	C++
1	0.0%	50.0%	GC
8826	100.0%		Shared libraries

[C++ entry points]:

ticks	cpp	total	name

[Bottom up (heavy) profile]:

Note: percentage shows a share of a particular caller in the total

amount of its parent calls.

Callers occupying less than 1.0% are not shown.

ticks parent name

8372 94.8% C:\WINDOWS\SYSTEM32\ntdll.dll

451 5.1% C:\Program Files\nodejs\node.exe

424 94.0% C:\Program Files\nodejs\node.exe

242 57.1% Function: ^openSync node:fs:585:18

241 99.6% Function: ^readFileSync node:fs:459:22

221 91.7% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

221 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

13 5.4% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

12 92.3% Function: ^Module.load node:internal/modules/cjs/loader:992:33

1 7.7% LazyCompile: ~Module.load node:internal/modules/cjs/loader:992:33

6 2.5% LazyCompile: ~Module._extensions..json node:internal/modules/cjs/loader:1185:39

6 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

69 16.3% Function: ^read node:internal/modules/package_json_reader:16:14

67 97.1% Function: ^readPackage node:internal/modules/cjs/loader:301:21

59 88.1% Function: ^resolveExports node:internal/modules/cjs/loader:483:24

59 100.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

6 9.0% Function: ^readPackageScope node:internal/modules/cjs/loader:332:26

5 83.3% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

1 16.7% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 3.0% LazyCompile: ~resolveExports node:internal/modules/cjs/loader:483:24

2 100.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

2 2.9% LazyCompile: *readPackageScope node:internal/modules/cjs/loader:332:26

2 100.0% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

29 6.8% Function: ^compileFunction node:vm:316:25

28 96.6% Function: ^wrapSafe node:internal/modules/cjs/loader:1040:18

28 100.0% Function: ^Module._compile node:internal/modules/cjs/loader:1080:37

25 89.3% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

3 10.7% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

1 3.4% LazyCompile: ~wrapSafe node:internal/modules/cjs/loader:1040:18

1 100.0% LazyCompile: ~Module._compile node:internal/modules/cjs/loader:1080:37

1 100.0% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

28 6.6% Function: ^stat node:internal/modules/cjs/loader:151:14

20 71.4% Function: ^tryFile node:internal/modules/cjs/loader:395:17

18 90.0% Function: ^tryExtensions node:internal/modules/cjs/loader:411:23

16 88.9% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

2 11.1% Function: ^tryPackage node:internal/modules/cjs/loader:349:20

2 10.0% Function: ^tryPackage node:internal/modules/cjs/loader:349:20

1 50.0% LazyCompile: *Module._findPath node:internal/modules/cjs/loader:505:28

1 50.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

8 28.6% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

8 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:865:35

8 100.0% Function: ^Module._load node:internal/modules/cjs/loader:771:24

PERFILAMIENTO DE SERVIDOR CON NODE BUILT IN PROFILER

en /info con console.log(info) BLOQUEANTE

$ artillery quick --count 50 -n 40 http://localhost:8081/info >result_bloq.txt

Statistical profiling result from nobloq-v8.log, (9101 ticks, 0 unaccounted, 0 excluded).

[Shared libraries]:

ticks total nonlib name

8912 97.9% C:\WINDOWS\SYSTEM32\ntdll.dll

185 2.0% C:\Program Files\nodejs\node.exe

1 0.0% C:\WINDOWS\System32\KERNELBASE.dll

[JavaScript]:

ticks total nonlib name

1 0.0% 33.3% LazyCompile: *normalizeString node:path:66:25

1 0.0% 33.3% Function: ^measureRoundTripTime C:\Users\pdepetris\Desktop\Personal\Coderhouse\Backend\clase32\node_modules\mongodb\lib\sdam\monitor.js:287:30

1 0.0% 33.3% Function: ^assert node:internal/assert:11:16

[C++]:

ticks total nonlib name

[Summary]:

ticks total nonlib name

3 0.0% 100.0% JavaScript

0 0.0% 0.0% C++

4 0.0% 133.3% GC

9098 100.0% Shared libraries

[C++ entry points]:

ticks cpp total name

[Bottom up (heavy) profile]:

Note: percentage shows a share of a particular caller in the total

amount of its parent calls.

Callers occupying less than 1.0% are not shown.

ticks parent name

8912 97.9% C:\WINDOWS\SYSTEM32\ntdll.dll

185 2.0% C:\Program Files\nodejs\node.exe

145 78.4% C:\Program Files\nodejs\node.exe

35 24.1% Function: ^compileFunction node:vm:316:25

34 97.1% Function: ^wrapSafe node:internal/modules/cjs/loader:1040:18

34 100.0% Function: ^Module._compile node:internal/modules/cjs/loader:1080:37

32 94.1% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 5.9% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

1 2.9% LazyCompile: ~wrapSafe node:internal/modules/cjs/loader:1040:18

1 100.0% LazyCompile: ~Module._compile node:internal/modules/cjs/loader:1080:37

1 100.0% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

25 17.2% Function: ^openSync node:fs:585:18

25 100.0% Function: ^readFileSync node:fs:459:22

23 92.0% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

23 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

2 8.0% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

14 9.7% Function: ^read node:internal/modules/package_json_reader:16:14

14 100.0% Function: ^readPackage node:internal/modules/cjs/loader:301:21

10 71.4% Function: ^resolveExports node:internal/modules/cjs/loader:483:24

10 100.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

3 21.4% Function: ^readPackageScope node:internal/modules/cjs/loader:332:26

2 66.7% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

1 33.3% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

1 7.1% LazyCompile: *Module._findPath node:internal/modules/cjs/loader:505:28

1 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:865:35

12 8.3% Function: ^stat node:internal/modules/cjs/loader:151:14

7 58.3% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

7 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:865:35

7 100.0% Function: ^Module._load node:internal/modules/cjs/loader:771:24

5 41.7% Function: ^tryFile node:internal/modules/cjs/loader:395:17

5 100.0% Function: ^tryExtensions node:internal/modules/cjs/loader:411:23

5 100.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

11 7.6% Function: ^readSync node:fs:699:18

11 100.0% Function: ^tryReadSync node:fs:438:21

11 100.0% Function: ^readFileSync node:fs:459:22

9 81.8% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 18.2% LazyCompile: ~Module._extensions..js node:internal/modules/cjs/loader:1135:37

5 3.4% Function: ^compileForInternalLoader node:internal/bootstrap/loaders:316:27

5 100.0% Function: ^nativeModuleRequire node:internal/bootstrap/loaders:349:29

2 40.0% Function: ~<anonymous> node:http:1:1

2 100.0% Function: ^compileForInternalLoader node:internal/bootstrap/loaders:316:27

2 40.0% Function: ~<anonymous> node:crypto:1:1

2 100.0% Function: ^compileForInternalLoader node:internal/bootstrap/loaders:316:27

1 20.0% Function: ~<anonymous> node:_tls_common:1:1

1 100.0% Function: ^compileForInternalLoader node:internal/bootstrap/loaders:316:27

3 2.1% Function: ^readPackage node:internal/modules/cjs/loader:301:21

3 100.0% Function: ^resolveExports node:internal/modules/cjs/loader:483:24

3 100.0% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

2 66.7% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:865:35

1 33.3% LazyCompile: ~Module._resolveFilename node:internal/modules/cjs/loader:865:35

3 2.1% C:\Program Files\nodejs\node.exe

1 33.3% LazyCompile: ~resolvePackageTargetString node:internal/modules/esm/resolve:499:36

1 100.0% LazyCompile: ~resolvePackageTarget node:internal/modules/esm/resolve:567:30

1 100.0% LazyCompile: ~resolvePackageTarget node:internal/modules/esm/resolve:567:30

1 33.3% LazyCompile: ~module.exports.get_best_napi_build_version C:\Users\pdepetris\Desktop\Personal\Coderhouse\Backend\clase32\node_modules\@mapbox\node-pre-gyp\lib\util\napi.js:187:54

1 100.0% LazyCompile: ~module.exports.validate_package_json C:\Users\pdepetris\Desktop\Personal\Coderhouse\Backend\clase32\node_modules\@mapbox\node-pre-gyp\lib\util\napi.js:46:48

1 100.0% LazyCompile: ~validate_config C:\Users\pdepetris\Desktop\Personal\Coderhouse\Backend\clase32\node_modules\@mapbox\node-pre-gyp\lib\util\versioning.js:195:25

1 33.3% Function: ^Module._findPath node:internal/modules/cjs/loader:505:28

1 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:865:35

1 100.0% Function: ^Module._load node:internal/modules/cjs/loader:771:24

2 1.4% LazyCompile: ~compileForInternalLoader node:internal/bootstrap/loaders:316:27

2 100.0% Function: ^nativeModuleRequire node:internal/bootstrap/loaders:349:29

1 50.0% LazyCompile: ~initializeCJSLoader node:internal/bootstrap/pre_execution:521:29

1 100.0% LazyCompile: ~prepareMainThreadExecution node:internal/bootstrap/pre_execution:29:36

1 50.0% Function: ~<anonymous> node:internal/modules/esm/fetch_module:1:1

1 100.0% LazyCompile: ~compileForInternalLoader node:internal/bootstrap/loaders:316:27

2 1.4% Function: ^tryStatSync node:fs:413:21

2 100.0% Function: ^readFileSync node:fs:459:22

2 100.0% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

2 100.0% Function: ^Module.load node:internal/modules/cjs/loader:992:33

2 1.4% Function: ^realpathSync node:fs:2460:22

2 100.0% Function: ^toRealPath node:internal/modules/cjs/loader:404:20

2 100.0% Function: ^tryFile node:internal/modules/cjs/loader:395:17

1 50.0% LazyCompile: ~finalizeEsmResolution node:internal/modules/cjs/loader:962:31

1 50.0% Function: ^tryExtensions node:internal/modules/cjs/loader:411:23

2 1.4% Function: ^closeSync node:fs:535:19

2 100.0% Function: ^closeSync C:\Users\pdepetris\Desktop\Personal\Coderhouse\Backend\clase32\node_modules\graceful-fs\graceful-fs.js:72:24

2 100.0% Function: ^readFileSync node:fs:459:22

2 100.0% Function: ^Module._extensions..js node:internal/modules/cjs/loader:1135:37

El proceso no bloqueante se lleva menos ticks (8372 vs 8912)

ARTILLERY en /info sin console.log(info) NO BLOQUEANTE

$ artillery quick --count 50 -n 40 http://localhost:8081/info >result_nobloq.txt

Running scenarios...

Phase started: unnamed (index: 0, duration: 1s) 16:28:32(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 16:28:33(-0300)

All VUs finished. Total time: 10 seconds

Summary report @ 16:28:37(-0300)

---

errors.ERR_GOT_REQUEST_ERROR: 50

http.codes.200: 50

http.request_rate: 25/sec

http.requests: 50

http.response_time:

min: 1

max: 42

median: 2

p95: 4

p99: 30.3

http.responses: 50

vusers.created: 50

vusers.created_by_name.0: 50

vusers.failed: 50

ARTILLERY con console.log(info) BLOQUEANTE

$ artillery quick --count 50 -n 40 http://localhost:8081/info >result_bloq.txt

Running scenarios...

Phase started: unnamed (index: 0, duration: 1s) 16:24:32(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 16:24:33(-0300)

All VUs finished. Total time: 10 seconds

Summary report @ 16:24:37(-0300)

---

errors.ERR_GOT_REQUEST_ERROR: 50

http.codes.200: 50

http.request_rate: 25/sec

http.requests: 50

http.response_time:

min: 2

max: 62

median: 3

p95: 15

p99: 47.9

http.responses: 50

vusers.created: 50

vusers.created_by_name.0: 50

vusers.failed: 50

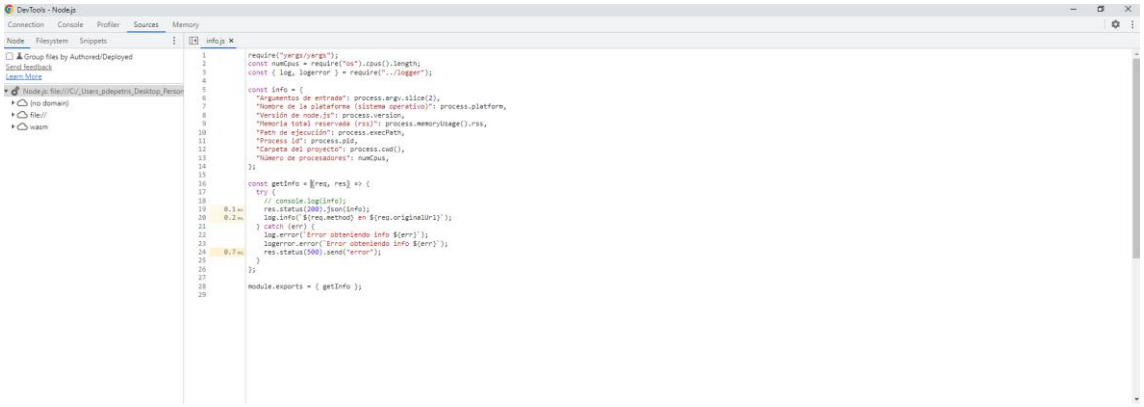
El tiempo de respuesta es superior cuando se interpone una función sincrónica/bloqueante como console.log(info) en la request a la ruta /info

---

MODO INSPECTOR DE NODE - PROCESO NO BLOQUEANTE (sin console.log(info)

node --inspect server.js 8081 FORK

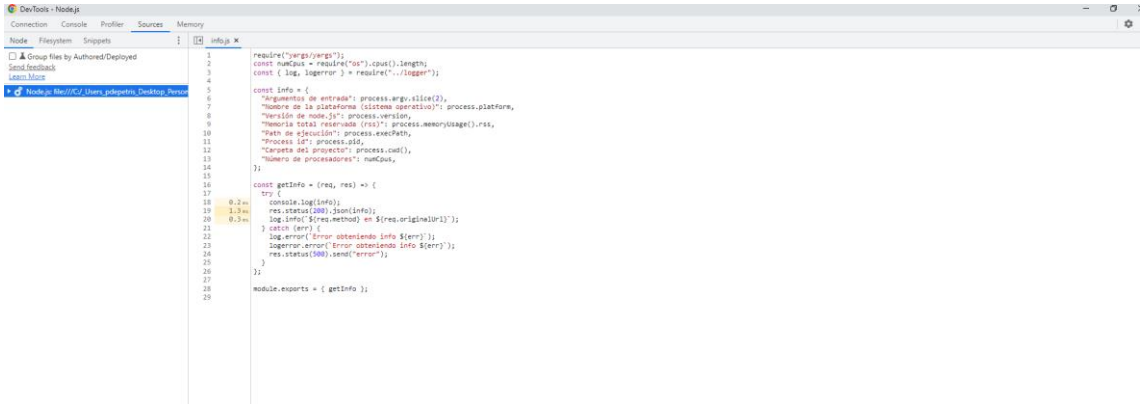
$ artillery quick --count 50 -n 40 http://localhost:8081/info



MODO INSPECTOR DE NODE - PROCESO BLOQUEANTE (con console.log(info)

node --inspect server.js 8081 FORK

$ artillery quick --count 50 -n 40 http://localhost:8081/info



Se observa un cuello de botella en el console.log(info)

AUTOCANNON Y 0X - PROCESO NO BLOQUEANTE (sin console.log(info)

Npm start

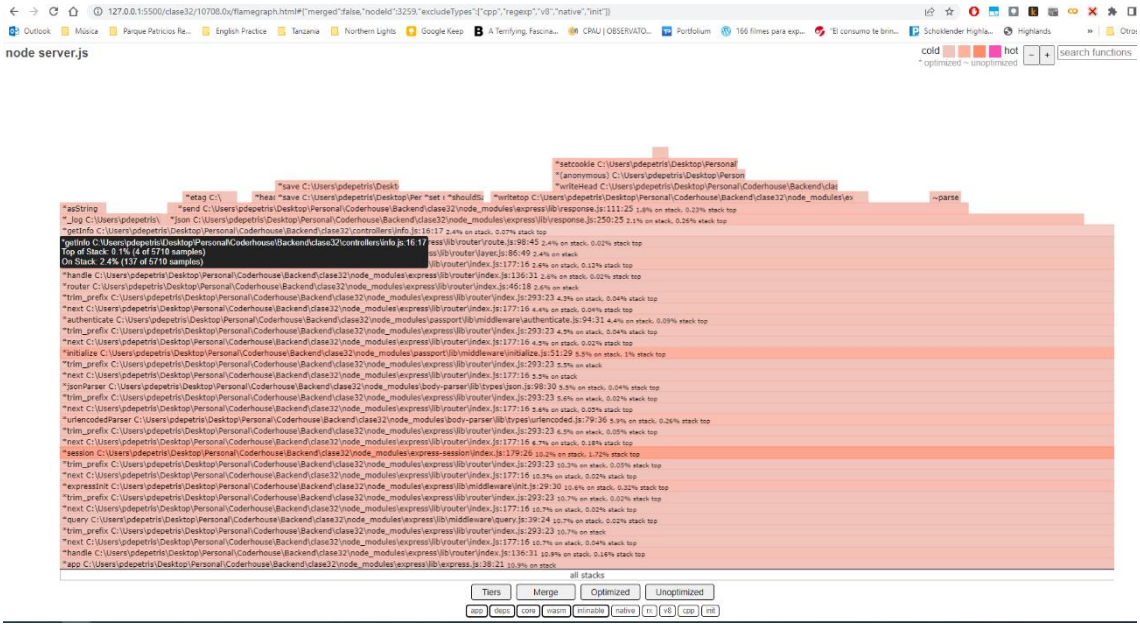
Npm test

```
$ npm test

> clase32@1.0.0 test
> node benchmark.js

Running autocannon in parallel....
Running 20s test @ http://localhost:8081/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	49 ms	62 ms	111 ms	122 ms	68.24 ms	17.73 ms	204 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	895	895	1500	1706	1455.05	223.89	895
Bytes/Sec	680 kB	680 kB	1.14 MB	1.3 MB	1.11 MB	170 kB	680 kB

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

29k requests in 20.08s, 22.1 MB read
```



AUTOCANNON Y 0X - PROCESO BLOQUEANTE (con console.log(info)

Npm start

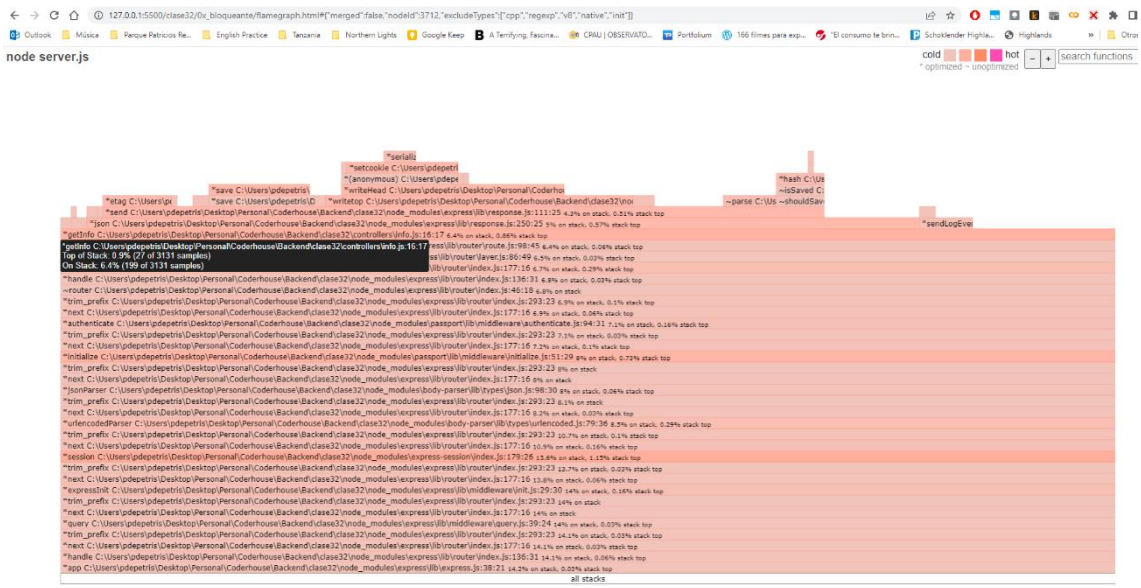
Npm test

```
$ npm test

> clase32@1.0.0 test
> node benchmark.js

Running autocannon in parallel....
Running 20s test @ http://localhost:8081/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	91 ms	132 ms	232 ms	246 ms	140.82 ms	34.64 ms	307 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	401	401	745	900	707.45	135.78	401
Bytes/Sec	304 kB	304 kB	565 kB	683 kB	537 kB	103 kB	304 kB

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

14k requests in 20.1s, 10.7 MB read

pdepetris@NT-0462 MINGW64 ~/Desktop/Personal/Coderhouse/Backend/clase32 (master)
$
```



Se observa en el proceso bloqueante que la latencia es mayor y procesa menos bytes y request por segundo. En el flamograph se observa que el proceso de getinfo tiene un porcentaje mayor del stack.

Todos los test permitieron corroborar que los procesos sincrónicos como el console.log entorpecen el servidor por lo que no es buena práctica utilizarlos en productivo.