Pruebas en ruta /info con Loggers

```
Usuario@Martin MINGW64 /c/backend/desafio (loggers)
$ npm test

> 8@1.0.0 test
> node benchmark.js

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

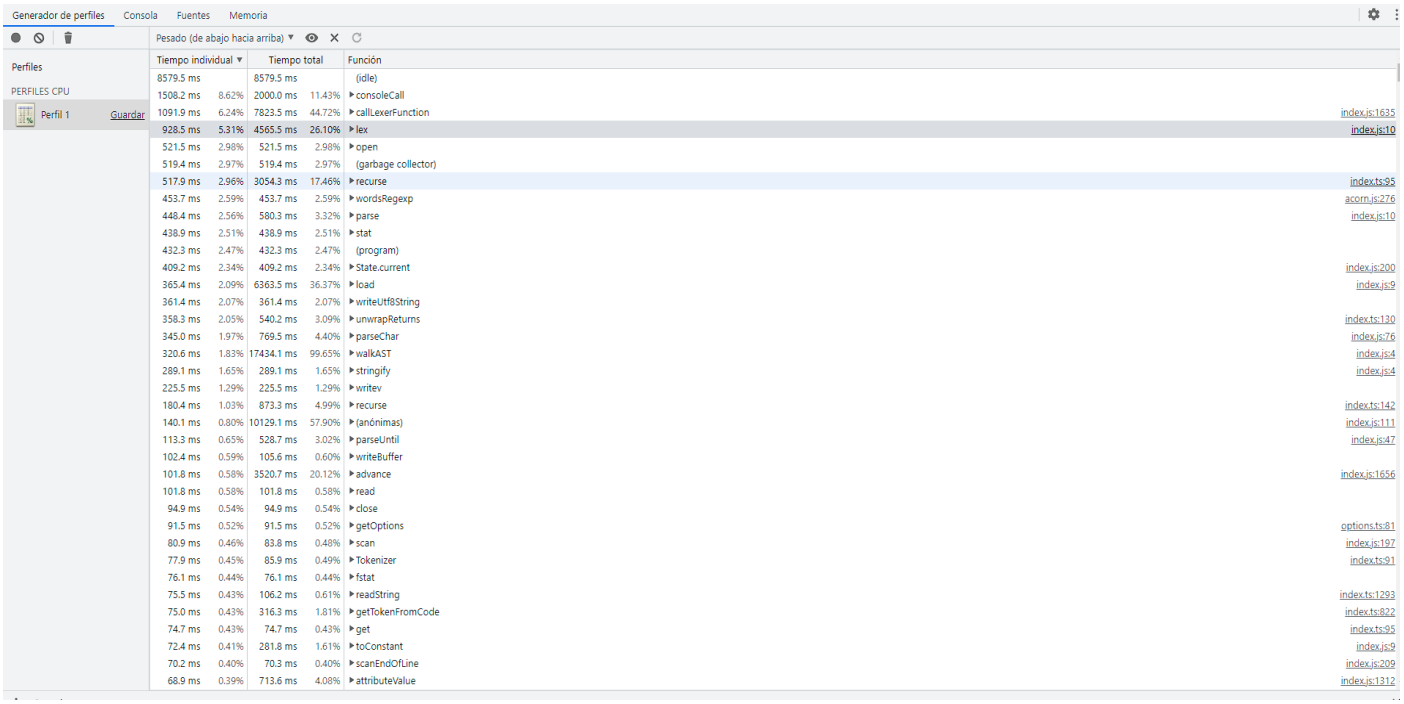
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	834 ms	1011 ms	2000 ms	2121 ms	1087.73 ms	290.13 ms	2752 ms

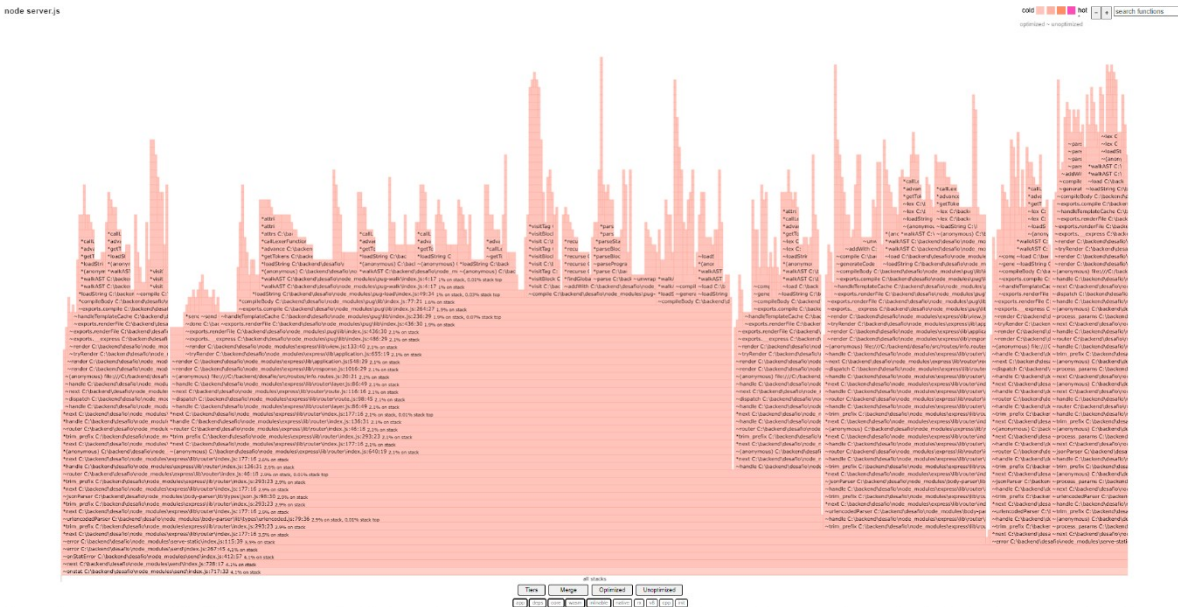
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	95	109	89.95	21.36	76
Bytes/Sec	0 B	0 B	130 kB	149 kB	123 kB	29.2 kB	104 kB

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

2k requests in 20.15s, 2.46 MB read

Usuario@Martin MINGW64 /c/backend/desafio (loggers)
$
```





Pruebas en ruta /info con Console.log

```
Usuario@Martin MINGW64 /c/backend/desafio (loggers)
$ npm test

> 8@1.0.0 test
> node benchmark.js

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

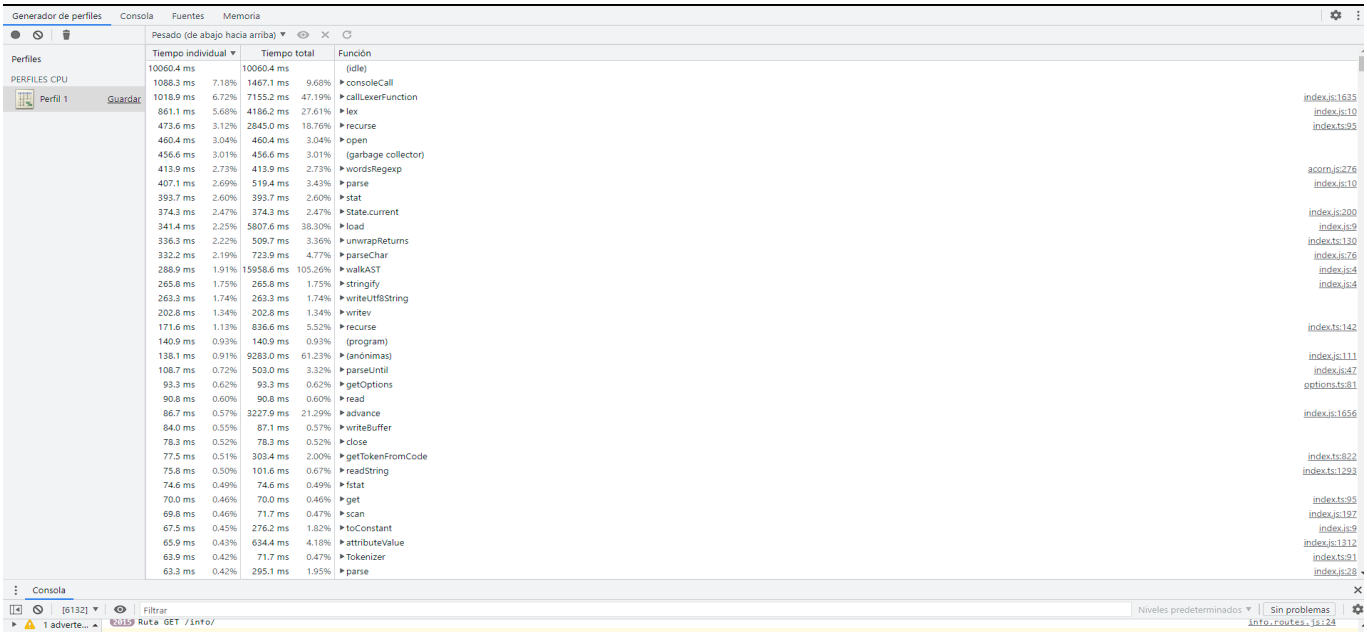
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	837 ms	1010 ms	1964 ms	2059 ms	1083.42 ms	278.25 ms	2733 ms

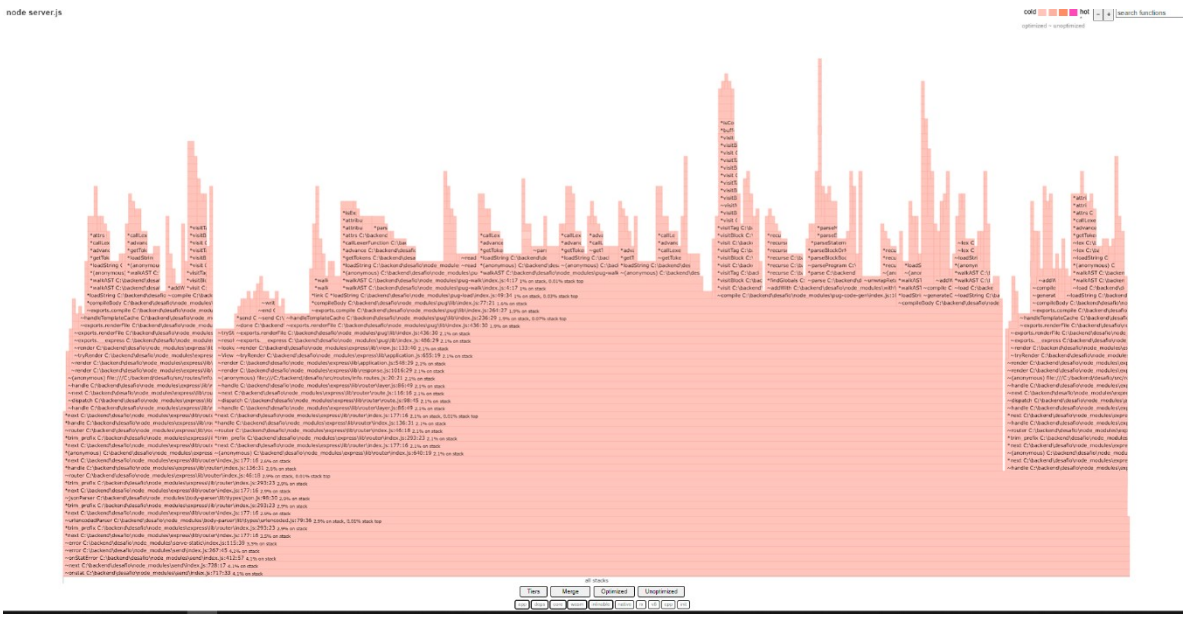
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1	1	95	104	89.85	20.73	1
Bytes/Sec	1.37 kB	1.37 kB	130 kB	142 kB	123 kB	28.4 kB	1.37 kB

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

2k requests in 20.17s, 2.46 MB read

Usuario@Martin MINGW64 /c/backend/desafio (loggers)
$
```


Generador de perfiles
Consola
Fuentes
Memoria
Pesado (de abajo hacia arriba)
Perfiles
PERFILES CPU
Perfil 1
Guardar
Tiempo individual
Tiempo total
Función
10060.4 ms 10060.4 ms (idle)
1088.3 ms 7.18% 1467.1 ms 9.68% consoleCall
1018.9 ms 6.72% 7155.2 ms 47.19% callLexerFunction index.js:1635
861.1 ms 5.68% 4186.2 ms 27.61% lex index.js:10
473.6 ms 3.12% 2845.0 ms 18.76% recurse index.ts:95
460.4 ms 3.04% 460.4 ms 3.04% open
456.6 ms 3.01% 456.6 ms 3.01% (garbage collector)
413.9 ms 2.73% 413.9 ms 2.73% wordsRegexp acorn.js:276
407.1 ms 2.69% 519.4 ms 3.43% parse index.js:10
393.7 ms 2.60% 393.7 ms 2.60% stat
374.3 ms 2.47% 374.3 ms 2.47% State.current index.js:200
341.4 ms 2.25% 5807.6 ms 38.30% load index.js:9
336.3 ms 2.22% 509.7 ms 3.36% unwrapReturns index.ts:130
332.2 ms 2.19% 723.9 ms 4.77% parseChar index.js:76
288.9 ms 1.91% 15958.6 ms 105.26% walkAST index.js:4
265.8 ms 1.75% 265.8 ms 1.75% stringify index.js:4
263.3 ms 1.74% 263.3 ms 1.74% writeUtf8String
202.8 ms 1.34% 202.8 ms 1.34% writev
171.6 ms 1.13% 836.6 ms 5.52% recurse index.ts:142
140.9 ms 0.93% 140.9 ms 0.93% (program)
138.1 ms 0.91% 9283.0 ms 61.23% (anónimas) index.js:111
108.7 ms 0.72% 503.0 ms 3.32% parseUntil index.js:47
93.3 ms 0.62% 93.3 ms 0.62% getOptions options.ts:81
90.8 ms 0.60% 90.8 ms 0.60% read
86.7 ms 0.57% 3227.9 ms 21.29% advance index.js:1656
84.0 ms 0.55% 87.1 ms 0.57% writeBuffer
78.3 ms 0.52% 78.3 ms 0.52% close
77.5 ms 0.51% 303.4 ms 2.00% getTokenFromCode index.ts:822
75.8 ms 0.50% 101.6 ms 0.67% readString index.ts:1293
74.6 ms 0.49% 74.6 ms 0.49% fstat
70.0 ms 0.46% 70.0 ms 0.46% get index.ts:95
69.8 ms 0.46% 71.7 ms 0.47% scan index.js:197
67.5 ms 0.45% 276.2 ms 1.82% toConstant index.js:9
65.9 ms 0.43% 634.4 ms 4.18% attributeValue index.js:1312
63.9 ms 0.42% 71.7 ms 0.47% Tokenizer index.ts:91
63.3 ms 0.42% 295.1 ms 1.95% parse index.js:28
Consola
Filtrar
Niveles predeterminados
Sin problemas
1 adverte...
Ruta GET /info/
info.routes.js:24


node server.js
cold
hot
search functions
Tiers
Merge
Optimized
Unoptimized

Conclusión: Aumenta mínimamente la velocidad de respuesta al utilizar loggers en reemplazo del console.log