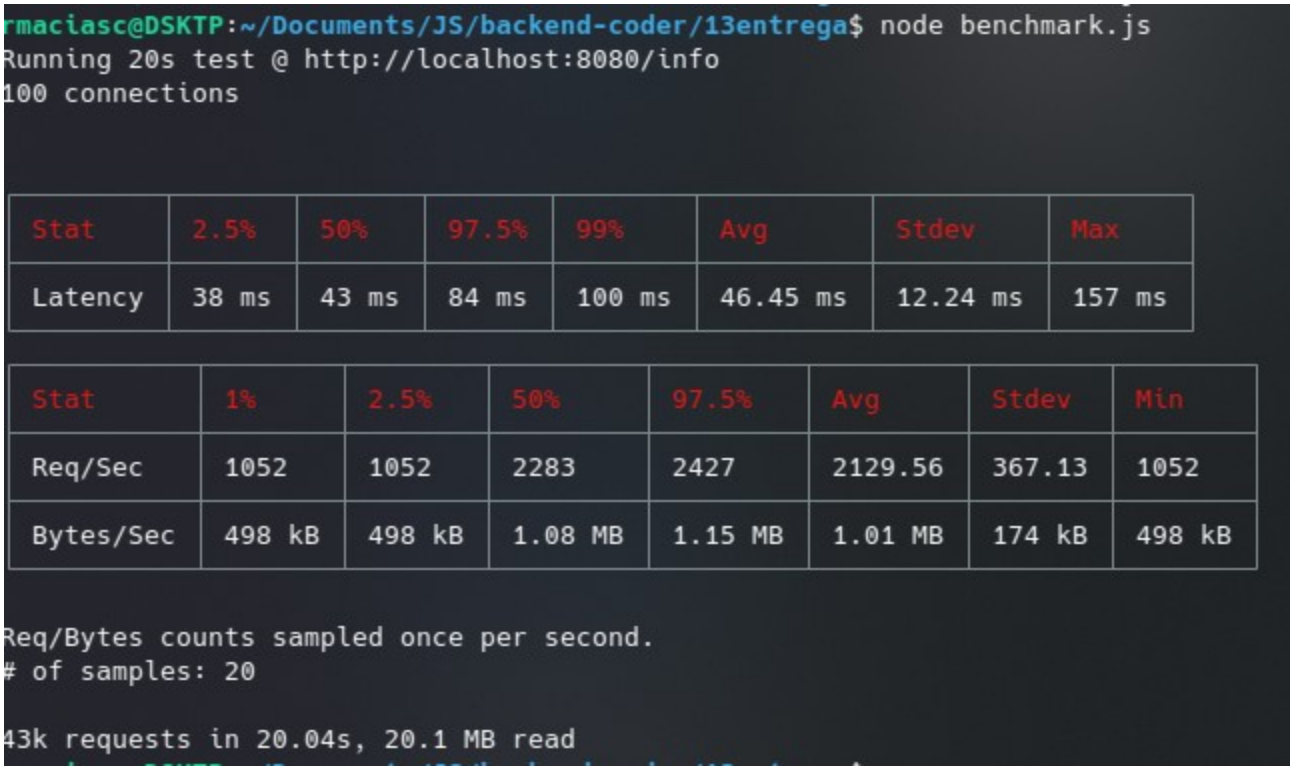
# Informe de pruebas realizadas

## Autocannon



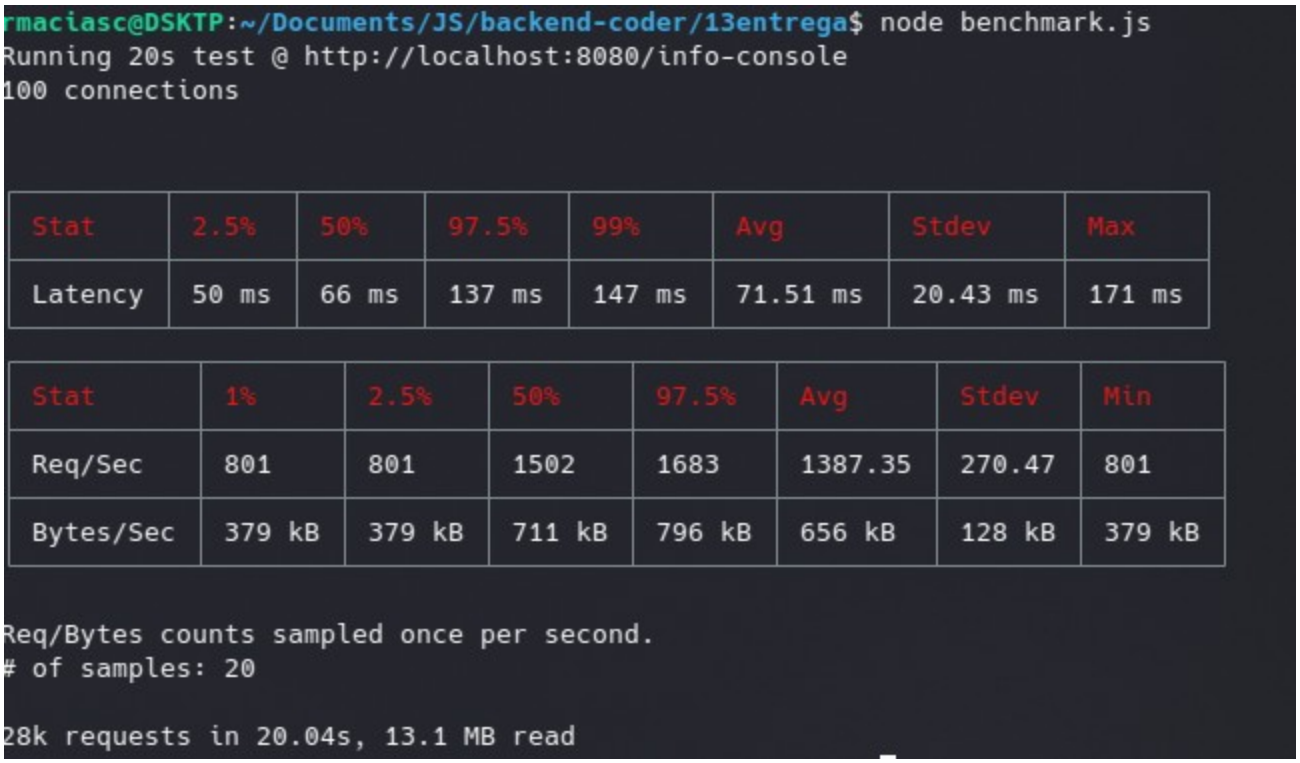
```
rmaciasc@DSKTP:~/Documents/JS/backend-coder/13entrega$ node benchmark.js
Running 20s test @ http://localhost:8080/info
100 connections
```

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 38 ms | 43 ms | 84 ms | 100 ms | 46.45 ms | 12.24 ms | 157 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 1052 | 1052 | 2283 | 2427 | 2129.56 | 367.13 | 1052 |
| Bytes/Sec | 498 kB | 498 kB | 1.08 MB | 1.15 MB | 1.01 MB | 174 kB | 498 kB |

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

43k requests in 20.04s, 20.1 MB read
```



```
rmaciasc@DSKTP:~/Documents/JS/backend-coder/13entrega$ node benchmark.js
Running 20s test @ http://localhost:8080/info-console
100 connections
```

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 50 ms | 66 ms | 137 ms | 147 ms | 71.51 ms | 20.43 ms | 171 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 801 | 801 | 1502 | 1683 | 1387.35 | 270.47 | 801 |
| Bytes/Sec | 379 kB | 379 kB | 711 kB | 796 kB | 656 kB | 128 kB | 379 kB |

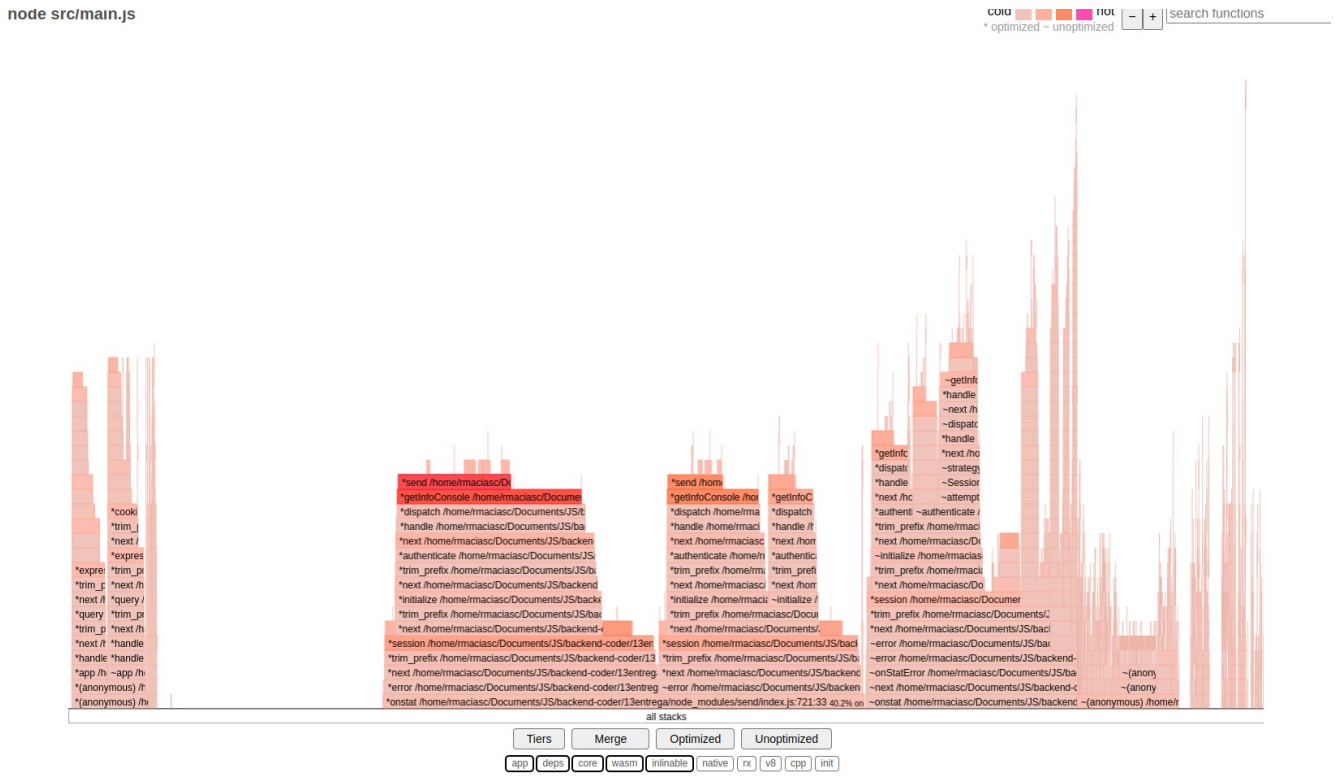
```
Req/Bytes counts sampled once per second.
# of samples: 20

28k requests in 20.04s, 13.1 MB read
```

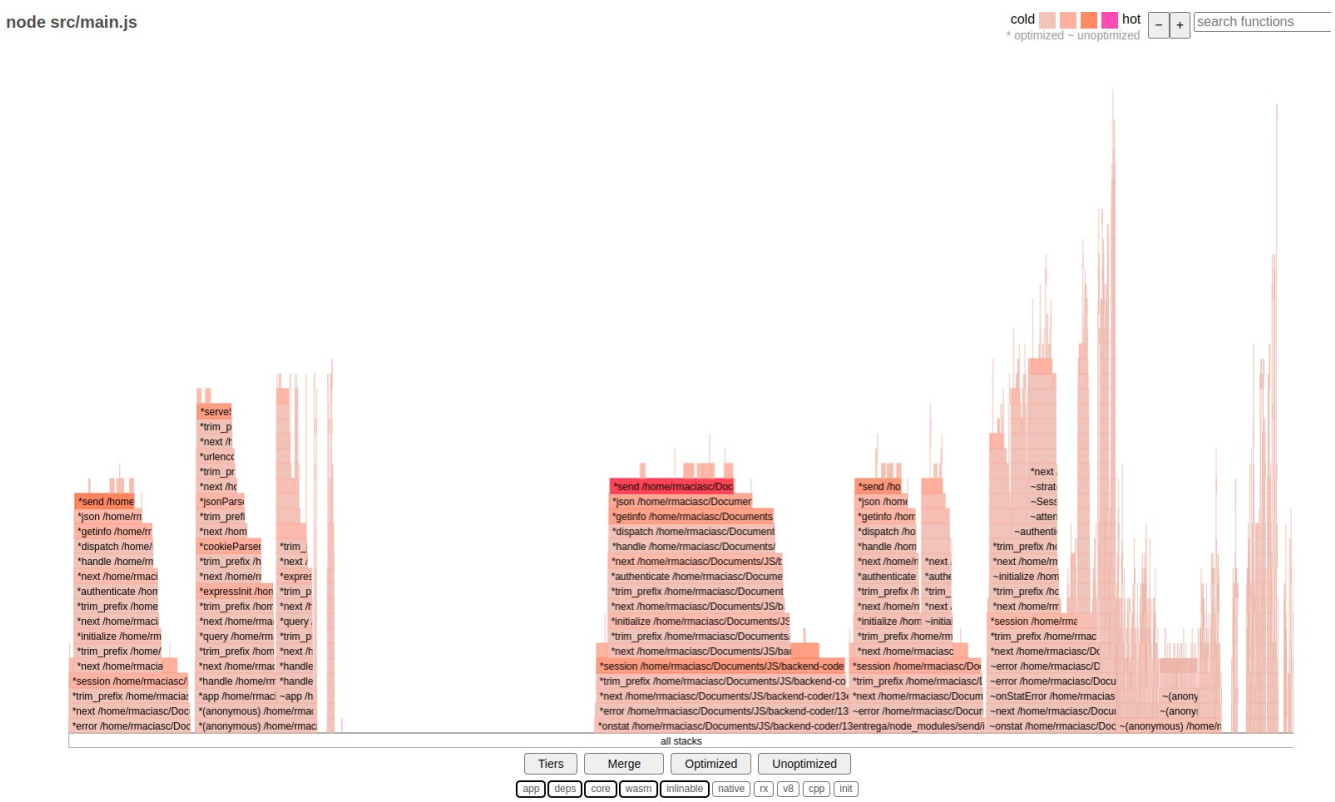
En la prueba de console log se aprecia que la latencia es más alta que aquella sin console.

# Diagrama de flama

Con console.log:




Sin console.log:




Se aprecian menos picos en la versión con console.

# Node Inspect

| | | | | |
|---|---|---|---|---|
| 14.5 ms | 0.42 % | 1083.0 ms | 31.11 % | ▸ getInfoConsole |
| 14.5 ms | 0.42 % | 14.5 ms | 0.42 % | ▸ asyncTaskScheduled |
| 14.0 ms | 0.40 % | 148.7 ms | 4.27 % | ▸ writeHead |
| 13.5 ms | 0.39 % | 33.9 ms | 0.97 % | ▸ _writeRaw |
| 13.4 ms | 0.38 % | 125.3 ms | 3.60 % | ▸ pipe |
| 13.2 ms | 0.38 % | 532.6 ms | 15.30 % | ▸ end |
| 13.1 ms | 0.38 % | 14.0 ms | 0.40 % | ▸ originalurl |
| 12.7 ms | 0.37 % | 13.4 ms | 0.38 % | ▸ getHeader |
| 12.4 ms | 0.36 % | 8885.5 ms | 255.23 % | ▸ (anonymous) |
| 12.4 ms | 0.36 % | 12.4 ms | 0.36 % | ▸ getColorDepth |
| 12.3 ms | 0.35 % | 17.3 ms | 0.50 % | ▸ (anonymous) |
| 12.1 ms | 0.35 % | 68.1 ms | 1.96 % | ▸ randomFillSync |
| 11.9 ms | 0.34 % | 2360.1 ms | 67.79 % | callbackTrampoline |
| 11.8 ms | 0.34 % | 227.9 ms | 6.55 % | ▸ Socket._writeGeneric |
| 11.8 ms | 0.34 % | 11.8 ms | 0.34 % | ▸ _addListener |
| 11.6 ms | 0.33 % | 29.4 ms | 0.84 % | ▸ endReadableNT |
| 11.5 ms | 0.33 % | 13.1 ms | 0.38 % | ▸ checkInvalidHeaderChar |
| 11.5 ms | 0.33 % | 82.3 ms | 2.36 % | ▸ formatProperty |
| 11.0 ms | 0.32 % | 15.5 ms | 0.44 % | ▸ FSReqCallback |
| 10.8 ms | 0.31 % | 396.0 ms | 11.38 % | ▸ parserOnIncoming |
| 10.8 ms | 0.31 % | 62.6 ms | 1.80 % | ▸ setCharset |
| 10.6 ms | 0.31 % | 594.3 ms | 17.07 % | ▸ getinfo |

El proceso con console tarda el doble (1083ms) que el proceso sin console (594.3ms).

```
67
68                  const getinfo = (req, res) => {
69      0.3 ms        const info = {
70                      'argumentos entrada': args,
71      0.3 ms          'path de ejecución': process.execPath,
72                      'nombre de la plataforma': process.platform,
73                      'process id': process.pid,
74                      'node version': process.version,
75      0.2 ms          'project folder': process.cwd,
76      0.6 ms          'rss memory': process.memoryUsage().rss,
77                      '# de CPUs': numCpu,
78                    };
79                    // console.log(info);
80      9.2 ms        res.json(info);
81                  };
82
83                  const getInfoConsole = (req, res) => {
84      0.8 ms        const info = {
85                      'argumentos entrada': args,
86      0.8 ms          'path de ejecución': process.execPath,
87                      'nombre de la plataforma': process.platform,
88                      'process id': process.pid,
89                      'node version': process.version,
90                      'project folder': process.cwd,
91      0.3 ms          'rss memory': process.memoryUsage().rss,
92                      '# de CPUs': numCpu,
93                    };
94      1.8 ms        console.log(info);
95     10.8 ms        res.json(info);
96                  };
97
```

De la misma manera se puede apreciar que el console.log agrega 1.8ms en cada request.

Conclusión:

Según las pruebas analizadas en el presente documento y en commands.txt dentro del proyecto, se considera que el dejar los console.log en modo producción representaría un aumento de recursos para el servidor, resultando en tiempos de espera más largos para los clientes.