# Análisis de rendimiento

## Código

- ### Con console.log()

```
const getInfo = (req, res)=>{
    logger.info(`${req.method} request from ${req.originalUrl} route`)
    console.log({
        entryArgs : JSON.stringify(args),
        platform: process.platform,
        nodeVersion: process.version,
        memory: process.memoryUsage().rss,
        path: process.execPath,
        processId: process.pid,
        dir: process.cwd()
    })
    res.render('info', {
        entryArgs : JSON.stringify(args),
        platform: process.platform,
        nodeVersion: process.version,
        memory: process.memoryUsage().rss,
        path: process.execPath,
        processId: process.pid,
        dir: process.cwd()
    })
}
```

- ### Sin console.log()

```
const getInfo = (req, res)=>{
    logger.info(`${req.method} request from ${req.originalUrl} route`)
    // console.log({
    //     entryArgs : JSON.stringify(args),
    //     platform: process.platform,
    //     nodeVersion: process.version,
    //     memory: process.memoryUsage().rss,
    //     path: process.execPath,
    //     processId: process.pid,
    //     dir: process.cwd()
    // })
    res.render('info', {
        entryArgs : JSON.stringify(args),
        platform: process.platform,
        nodeVersion: process.version,
        memory: process.memoryUsage().rss,
        path: process.execPath,
        processId: process.pid,
        dir: process.cwd()
    })
}
```

## Gzip.

- ### Sin compression

| Nombre | Estado | Tipo | Inicia... | Tamaño | Hora | Cascada |
|---|---|---|---|---|---|---|
| info | 200 | doc... | Otro | 1.5 kB | 24 ms | |

podemos observar que sin compression el documento pesa 1.5 kB

- ### Con compression

| Nombre | Estado | Tipo | Inicia... | Tamaño | Hora | Cascada |
|---|---|---|---|---|---|---|
| info | 200 | doc... | Otro | 1.0 kB | 79 ms | |

utilizando compression podemos observar que el peso se reduce a 1.0 kB

# Resultado de Artillery.

- ## Con console.log().

```
--------------------------------
Summary report @ 18:24:41(-0300)
--------------------------------

http.codes.200: ..................................................... 1000
http.request_rate: .................................................. 965/sec
http.requests: ...................................................... 1000
http.response_time:
  min: .............................................................. 1
  max: .............................................................. 53
  median: ........................................................... 8.9
  p95: .............................................................. 21.1
  p99: .............................................................. 37
http.responses: ..................................................... 1000
vusers.completed: ................................................... 50
vusers.created: ..................................................... 50
vusers.created_by_name.0: ........................................... 50
vusers.failed: ...................................................... 0
vusers.session_length:
  min: .............................................................. 49.9
  max: .............................................................. 328.8
  median: ........................................................... 214.9
  p95: .............................................................. 308
  p99: .............................................................. 320.6
```

- ## Sin console.log()

```
--------------------------------
Summary report @ 18:33:21(-0300)
--------------------------------

http.codes.200: ..................................................... 1000
http.request_rate: .................................................. 1000/sec
http.requests: ...................................................... 1000
http.response_time:
  min: .............................................................. 0
  max: .............................................................. 46
  median: ........................................................... 4
  p95: .............................................................. 8.9
  p99: .............................................................. 21.1
http.responses: ..................................................... 1000
vusers.completed: ................................................... 50
vusers.created: ..................................................... 50
vusers.created_by_name.0: ........................................... 50
vusers.failed: ...................................................... 0
vusers.session_length:
  min: .............................................................. 22.6
  max: .............................................................. 202.7
  median: ........................................................... 89.1
  p95: .............................................................. 159.2
  p99: .............................................................. 186.8
```

## - - prof

- Con console.log()

```
[Summary]:
  ticks  total  nonlib   name
    7    0.7%    0.8%  JavaScript
   859   91.7%   97.7%  C++
    16    1.7%    1.8%  GC
    58    6.2%          Shared libraries
    13    1.4%          Unaccounted
```

- Sin console.log()

```
[Summary]:
  ticks  total  nonlib   name
    11    1.1%    1.2%  JavaScript
   902   92.7%   98.3%  C++
    15    1.5%    1.6%  GC
    55    5.7%          Shared libraries
     5    0.5%          Unaccounted
```

# Informe con - - inspect

- Con console.log()

```
const getInfo = (req, res) => {
    const cpus = os.cpus();
    //res.setHeader('Content-Type', 'application/json');
    console.log({
        "Input Args": args.port,
        "Operating System": process.platform,
        "Node Version": process.version,
        "Memory Usage": process.memoryUsage().rss,
        "ExecPath": process.execPath,
        "Process ID (PID)": process.pid,
        "Actual Folder ": process.cwd(),
        "Total Cores ": cpus.length,
    })
    res.end(JSON.stringify({
        "Input Args": args.port,
        "Operating System": process.platform,
        "Node Version": process.version,
        "Memory Usage": process.memoryUsage().rss,
        "ExecPath": process.execPath,
        "Process ID (PID)": process.pid,
        "Actual Folder ": process.cwd(),
        "Total Cores ": cpus.length,
    }, null, 2))
}
```

- Sin console.log()

```
const getInfo = (req, res) => {
    const cpus = os.cpus();
    //res.setHeader('Content-Type', 'application/json');
    /*     console.log({
            "Input Args": args.port,
            "Operating System": process.platform,
            "Node Version": process.version,
            "Memory Usage": process.memoryUsage().rss,
            "ExecPath": process.execPath,
            "Process ID (PID)": process.pid,
            "Actual Folder ": process.cwd(),
            "Total Cores ": cpus.length,
        }) */
    res.end(JSON.stringify({
        "Input Args": args.port,
        "Operating System": process.platform,
        "Node Version": process.version,
        "Memory Usage": process.memoryUsage().rss,
        "ExecPath": process.execPath,
        "Process ID (PID)": process.pid,
        "Actual Folder ": process.cwd(),
        "Total Cores ": cpus.length,
    }, null, 2))
}
```

# Flame Graph 0x con Autocannon

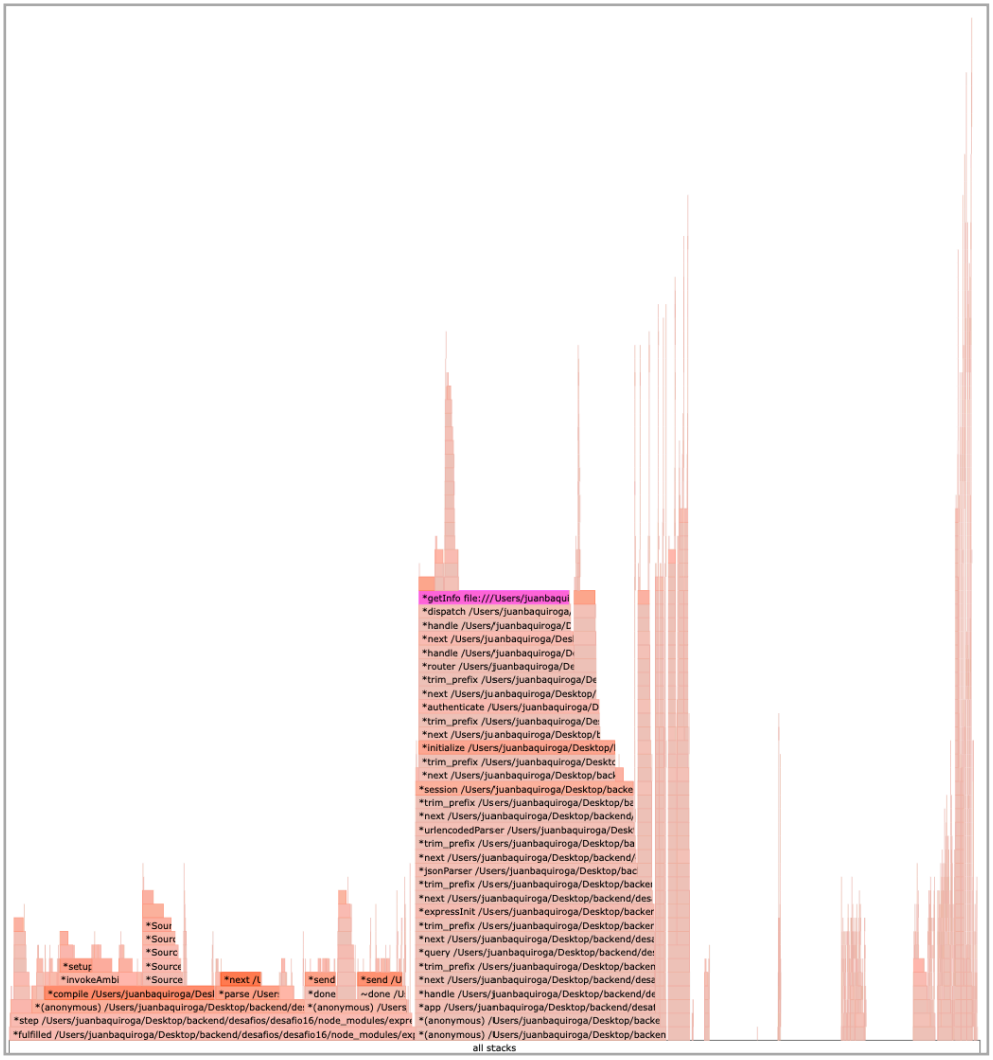
- ## Con console.log()

```
Running 20s test @ http://localhost:8081/info
100 connections
```

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 48 ms | 53 ms | 79 ms | 86 ms | 55.7 ms | 10.03 ms | 195 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 1139 | 1139 | 1805 | 1896 | 1777.65 | 166.4 | 1139 |
| Bytes/Sec | 1.74 MB | 1.74 MB | 2.75 MB | 2.89 MB | 2.71 MB | 253 kB | 1.73 MB |

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



# Flame Graph 0x con Autocannon

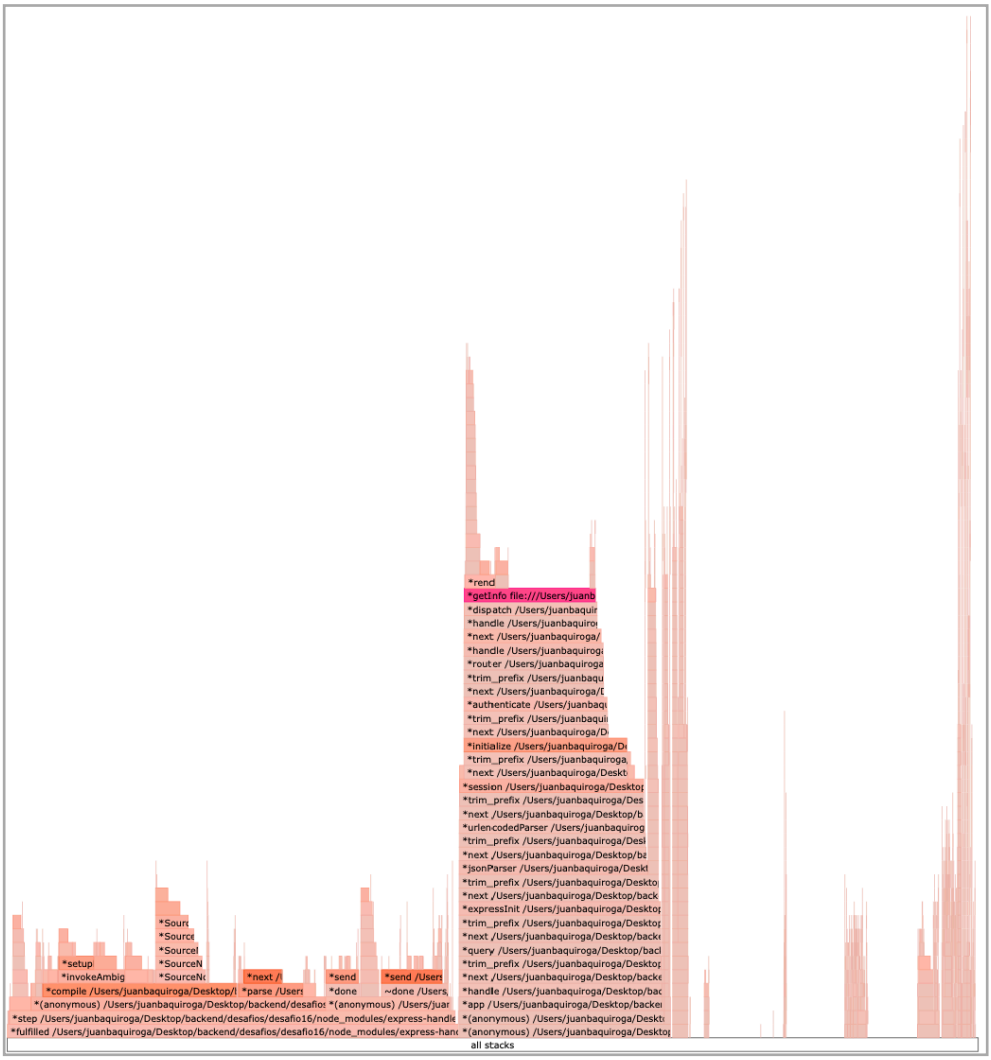
- **Sin console.log()**

```
Running 20s test @ http://localhost:8081/info
100 connections
```

| Stat | 2.5% | 50% | 97.5% | 99% | Avg | Stdev | Max |
|---|---|---|---|---|---|---|---|
| Latency | 44 ms | 46 ms | 70 ms | 76 ms | 48.7 ms | 8.75 ms | 188 ms |

| Stat | 1% | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
|---|---|---|---|---|---|---|---|
| Req/Sec | 1233 | 1233 | 2099 | 2147 | 2030.95 | 199.43 | 1233 |
| Bytes/Sec | 1.88 MB | 1.88 MB | 3.2 MB | 3.27 MB | 3.09 MB | 304 kB | 1.88 MB |

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



# Conclusion personal

Mi conclusión es que el uso excesivo de console.log puede tener un impacto significativo en la eficiencia y la velocidad de una aplicación en entornos de producción. Recomendaría evitar su uso en producción y limitar su uso para fines de desarrollo y pruebas. En lugar de console.log, se deben utilizar herramientas de registro y seguimiento de errores dedicadas para recopilar información de depuración y diagnóstico en tiempo real. De esta manera, se puede garantizar una mejor eficiencia y velocidad en la aplicación en entornos de producción.