Pruebas realizadas:

1) Perfilamiento de servidor con –prof de node:
   - Alta de servidor:
   ```
   node --prof index.js
   ```
   - Consultas Artillery:
   ```
   artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_sinlog.txt
   artillery quick --count 20 -n 50 "http://localhost:8080/info/con-log" > result_conlog.txt
   ```
   - Conversión de .log a .txt:
   ```
   node --prof-process result_sinlog-v8.log > result_prof-sinlog.txt
   ```
   ```
   node --prof-process result_conlog-v8.log > result_prof-conlog.txt
   ```
   - Resultados:
   - Sin logueo:
   ```
   [Summary]:
     ticks  total  nonlib   name
       30    1.0%  100.0%  JavaScript
        0    0.0%    0.0%  C++
       22    0.7%   73.3%  GC
     3047   99.0%          Shared libraries
   ```
   - Con logueo:
   ```
   [Summary]:
     ticks  total  nonlib   name
       23    0.9%  100.0%  JavaScript
        0    0.0%    0.0%  C++
       15    0.6%   65.2%  GC
     2591   99.1%          Shared libraries
   ```

Conclusión: El srv hace mas ticks sin devolver console logs.

Pruebas con autocannon:

- Ejecución:
```
npm test
```
- Resultados:
  - Sin Logueo:

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	419 ms	1549 ms	1696 ms	1703 ms	1441.62 ms	313.96 ms	1710 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	80	100	65.91	34.64	18
Bytes/Sec	0 B	0 B	36.2 kB	45.2 kB	29.8 kB	15.7 kB	8.14 kB

  - Con Logueo:

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	914 ms	1511 ms	1870 ms	1878 ms	1491.25 ms	222.5 ms	1910 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	22	22	69	99	65	19.63	22
Bytes/Sec	9.95 kB	9.95 kB	31.2 kB	44.8 kB	29.4 kB	8.87 kB	9.94 kB

- Conclusión: En este Test, por poca diferencia pero tiene menos latencia las respuestas sin el console.log.

2) Perfilamiento de servidor con –inspect de node y consola de chrome:
   - Alta de servidor: `node --inspect index.js`
   - Resultados en consola de chrome:

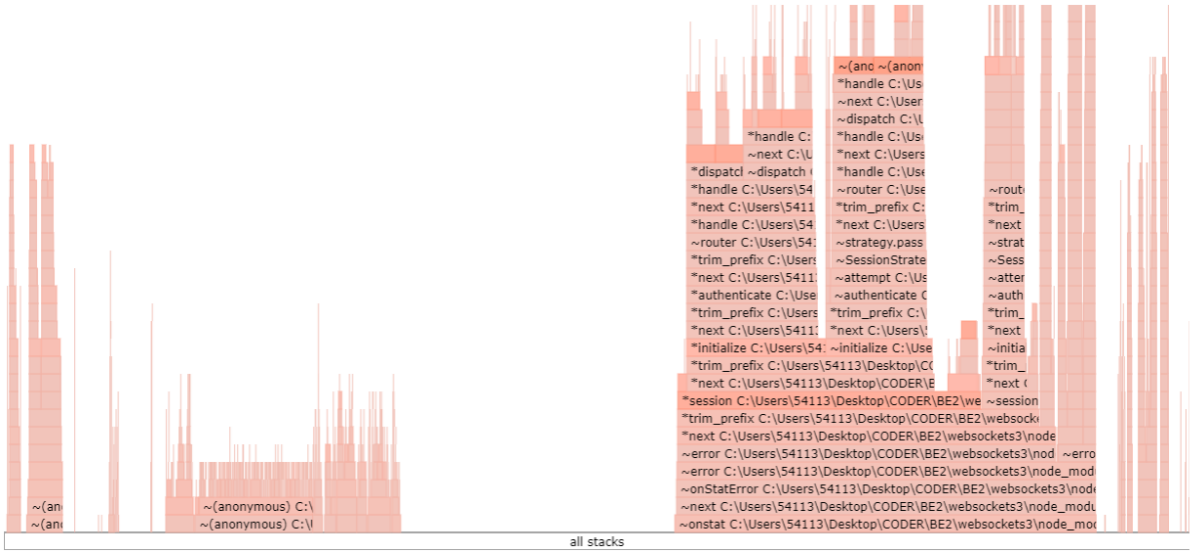
```
        routerInfo.get('/', (req, res) => {
 9.7 ms     logger.info(`Peticion ${req.method} en ruta: ${req.baseUrl}`);
 1.8 ms     const info = {
 2.6 ms         args: process.argv.slice(2),
 0.2 ms         path: process.argv[1],
 0.8 ms         so: process.platform,
 0.5 ms         pid: process.pid,
 0.3 ms         node: process.version,
 0.6 ms         folder: process.cwd(),
 2.1 ms         memoryRss: process.memoryUsage().rss,
 9.1 ms         cpus: cpus().length
            };
40.6 ms     res.send(info);
        });

        routerInfo.get('/con-log', (req, res) => {
 7.7 ms     logger.info(`Peticion ${req.method} en ruta: ${req.baseUrl}`);
 2.7 ms     const info = {
 2.7 ms         args: process.argv.slice(2),
 1.2 ms         path: process.argv[1],
 0.8 ms         so: process.platform,
 0.6 ms         pid: process.pid,
 0.3 ms         node: process.version,
 0.5 ms         folder: process.cwd(),
 1.1 ms         memoryRss: process.memoryUsage().rss,
 2.1 ms         cpus: cpus().length
            };
19.9 ms     console.log(info);
51.3 ms     res.send(info)
        });
```

Conclusión: Los procesos que más demoran son las respuestas del servidor a la peticion (res.send), y en el caso de la respuesta con el console log, también agrega bastante tiempo de espera.

3) Diagrama de Flama:
   - Iniciamos el servidor con el comando `npm start`
   - Ejecutamos el test con: `npm test`

- Cerramos el servidor y obtenemos los resultados:



PD: Todos los Archivos resultados estan en el repo de git.