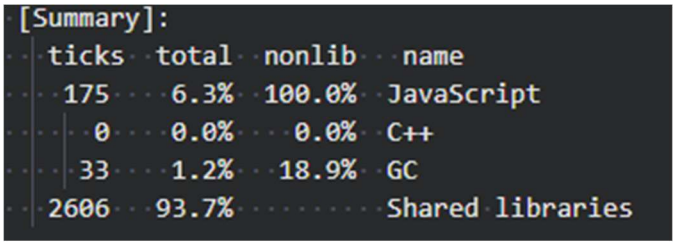
Resultados de las pruebas

Se realizaron pruebas sobre los endpoint:

1. /randoms
2. /info que utiliza un fork

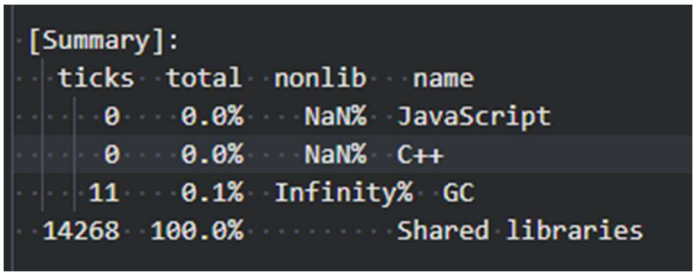
Prueba con profiler de Node

/randoms



```
[Summary]:
   ticks  total  nonlib   name
     175   6.3%  100.0%  JavaScript
       0   0.0%    0.0%  C++
      33   1.2%   18.9%  GC
    2606  93.7%          Shared libraries
```

/info



```
[Summary]:
   ticks  total  nonlib   name
       0   0.0%    NaN%  JavaScript
       0   0.0%    NaN%  C++
      11   0.1%  Infinity%  GC
   14268 100.0%          Shared libraries
```

Puede observarse que /info ocupo muchos mas ticks, a pesar de ser un fork process, pero puede deberse a que la carga fue mayor con Artillery.

Test de carga con Autocannon

```
Running 20s test @ http://localhost:3000/randoms
10 connections

| Stat    | 2.5%  | 50%   | 97.5% | 99%   | Avg   | Stdev | Max   |
| Latency | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  |

| Stat      | 1%  | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
| Req/Sec   | 0   | 0    | 0   | 0     | 0   | 0     | 0   |
| Bytes/Sec | 0 B | 0 B  | 0 B | 0 B   | 0 B | 0 B   | 0 B |

Req/Bytes counts sampled once per second.

40k requests in 20.04s, 0 B read
40k errors (0 timeouts)
Running 20s test @ http://localhost:3000/info
10 connections

| Stat    | 2.5%  | 50%   | 97.5% | 99%   | Avg   | Stdev | Max   |
| Latency | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  | 0 ms  |

| Stat      | 1%  | 2.5% | 50% | 97.5% | Avg | Stdev | Min |
| Req/Sec   | 0   | 0    | 0   | 0     | 0   | 0     | 0   |
| Bytes/Sec | 0 B | 0 B  | 0 B | 0 B   | 0 B | 0 B   | 0 B |

Req/Bytes counts sampled once per second.

40k requests in 20.09s, 0 B read
40k errors (0 timeouts)
```

En el siguiente gráfico generado por 0x, podemos observar que hay muchos picos y pocas mesetas, lo cual es positivo ya que refleja procesos no bloqueantes.