ANÁLISIS COMPLETO DE PERFORMANCE

- **Artillery**

Bloqueante

En el caso de la función bloqueante donde se agregó un console.log previamente a la respuesta del servidor se tuvieron 65 ticks para que se resolviera la ruta.

```
[Summary]:
  ticks  total  nonlib   name
     65   0.7%   97.0%  JavaScript
      0   0.0%    0.0%  C++
     44   0.5%   65.7%  GC
   8987  99.3%          Shared libraries
      2   0.0%          Unaccounted
```

No bloqueante

En el caso de la función no bloqueante donde no se agregó un console.log previamente a la respuesta del servidor se tuvieron 52 ticks para que se resolviera la ruta.

```
[Summary]:
  ticks  total  nonlib   name
     52   1.1%   98.1%  JavaScript
      0   0.0%    0.0%  C++
     32   0.7%   60.4%  GC
   4762  98.9%          Shared libraries
      1   0.0%          Unaccounted
```

- **Autocannon**

Bloqueante

En el caso de la función bloqueante donde se agregó un console.log previamente a la respuesta del servidor se tuvo un periodo de latencia promedio de 2688.53ms con un máximo de 5967ms.

Se hicieron 814 llamadas en 20.3 segundos

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	2027 ms	2451 ms	5162 ms	5358 ms	2688.53 ms	792.96 ms	5967 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	38	57	35.71	14.69	15
Bytes/Sec	0 B	0 B	24.1 kB	36.1 kB	22.6 kB	9.29 kB	9.49 kB

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

814 requests in 20.3s, 452 kB read
```

No bloqueante

En el caso de la función no bloqueante donde no se agregó un console.log previamente a la respuesta del servidor se tuvo un periodo de latencia promedio de 1668.57 ms con un máximo de 7480ms.

Se hicieron 1000 llamadas en 20.22 segundos

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	771 ms	1090 ms	4599 ms	5027 ms	1668.57 ms	1139.91 ms	7480 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	3	3	59	126	58.75	37.68	3
Bytes/Sec	1.89 kB	1.89 kB	37.3 kB	79.6 kB	37.1 kB	23.8 kB	1.89 kB

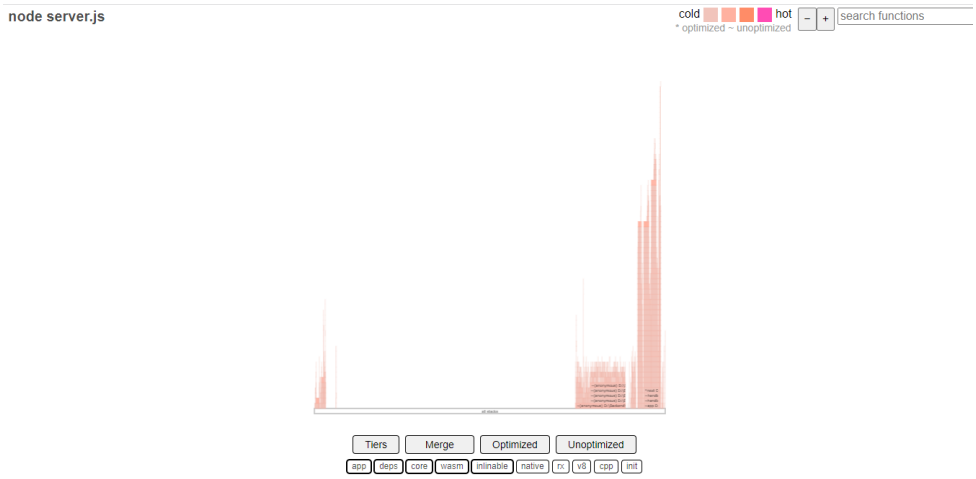
```
Req/Bytes counts sampled once per second.
# of samples: 20

1k requests in 20.22s, 742 kB read
```

- **Ox**

Bloqueante

En el caso de la función bloqueante donde se agregó un console.log previamente a la respuesta del servidor se puede ver que tomo más tiempo para que resolviera la función y además se puede ver que tuvo picos donde se muestra un color anaranjado que muestra que no fue tan eficiente y necesito más esfuerzo y tiempo para continuar.



No bloqueante

En el caso de la función no bloqueante donde no se agregó un console.log previamente a la respuesta del servidor se puede ver que tomo menos tiempo para que resolviera la función y además se puede ver que no tuvo picos donde se muestra un color anaranjado, sino todo es de color rosado claro, lo que demuestra que fue más eficiente.

Conclusión

- La función bloqueante fue menos eficiente que la no bloqueante en cada uno de los test. La bloqueante requiero de más ticks en el caso del test de artillery y requiero más milisegundos en el caso del test de autocannon.
- Se recomienda usar llamadas asíncronas o no bloqueantes porque esto permite que las llamadas siguientes sigan entrando, que no se bloquee la ejecución, que no se bloquee nuestro único hilo, porque si nuestro único hilo se bloquea nuestro servidor se va a parar hasta que ese hilo se desbloquee nuevamente.