PROFILING
este test de carga se hizo sin console.log y

1-modo profiling de la ruta /info , sin console.log
comando: node --prof server.js

2- test de carga en la ruta /info
comando: artillery quick --count 50 -n 20 "http://localhost:8080/info" > result_console.txt

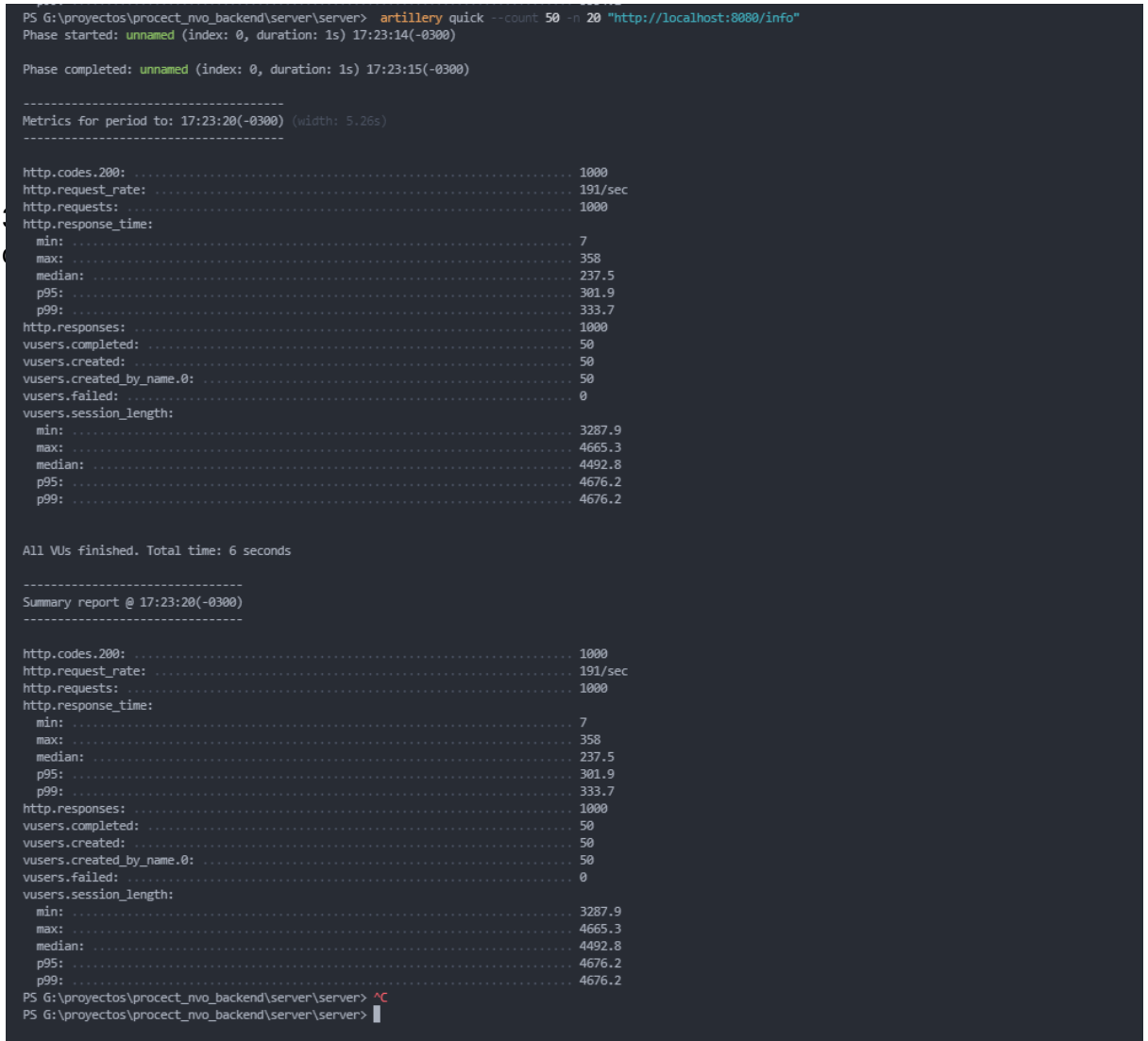
3-Con esto procesamos los profiling que generamos
comando: node --prof-process infoRut-v8.log > prof-infoRut.txt

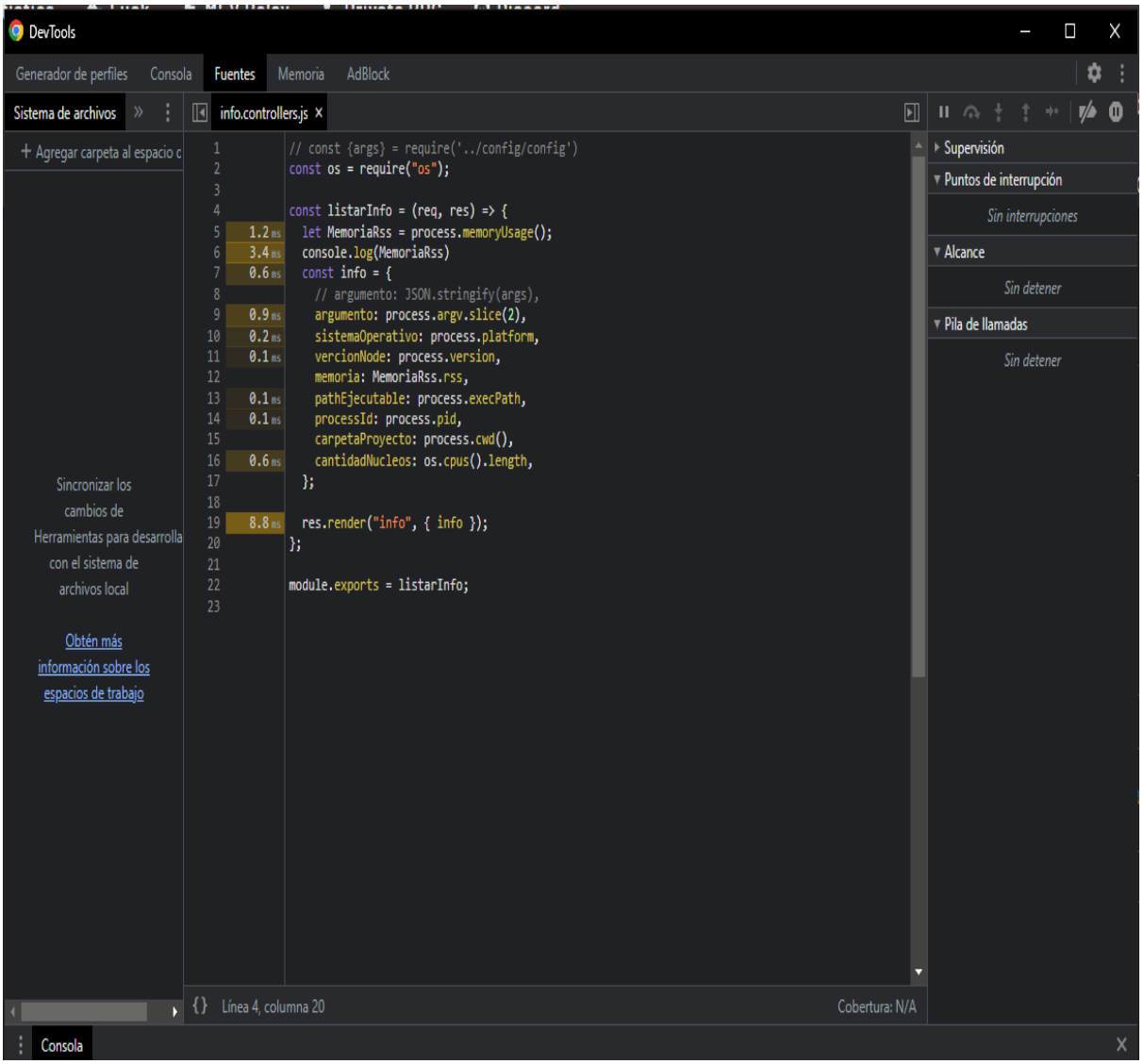
INSPECT
este test de carga se hizo con console.log

1-iniciamos con el comando
node --inspect index.js

2- test de carga en la ruta /info
comando: artillery quick --count 50 -n 20 "http://localhost:8080/info"

3-Una vez le damos stop a este inspect(DevTolss). Podemos ver los diferentes procesos o diferentes funciones que se ejecutan(se ve el tiempo que llevo cada una de las funciones)

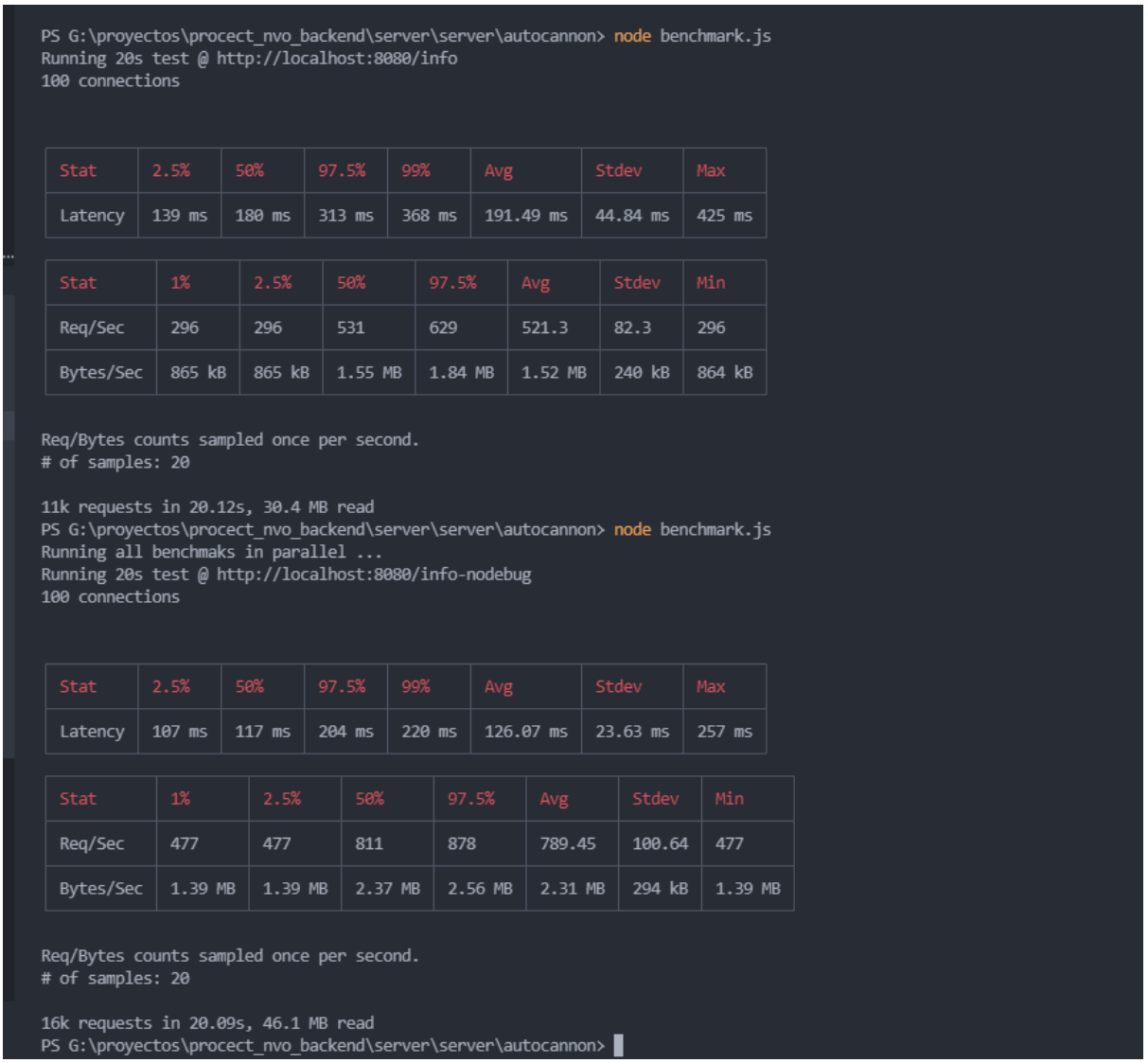




Autocannon

1-levantamos nuestro servidor desde la terminal con la herramienta 0x para generar nuestros gráficos de flama
comando:0x server.js

2- ejecutar los comandos (node benchmark.js,este contiene peticiones a la ruta /info con console.log y la ruta /info-nodebug sin console.log)(aca tenemos nuestro script de autocannon para probar nuestro servidor)
comando:node benchmark.js



```
PS G:\proyectos\procect_nvo_backend\server\server\autocannon> node benchmark.js
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	139 ms	180 ms	313 ms	368 ms	191.49 ms	44.84 ms	425 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	296	296	531	629	521.3	82.3	296
Bytes/Sec	865 kB	865 kB	1.55 MB	1.84 MB	1.52 MB	240 kB	864 kB

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

11k requests in 20.12s, 30.4 MB read
PS G:\proyectos\procect_nvo_backend\server\server\autocannon> node benchmark.js
Running all benchmaks in parallel ...
Running 20s test @ http://localhost:8080/info-nodebug
100 connections
```

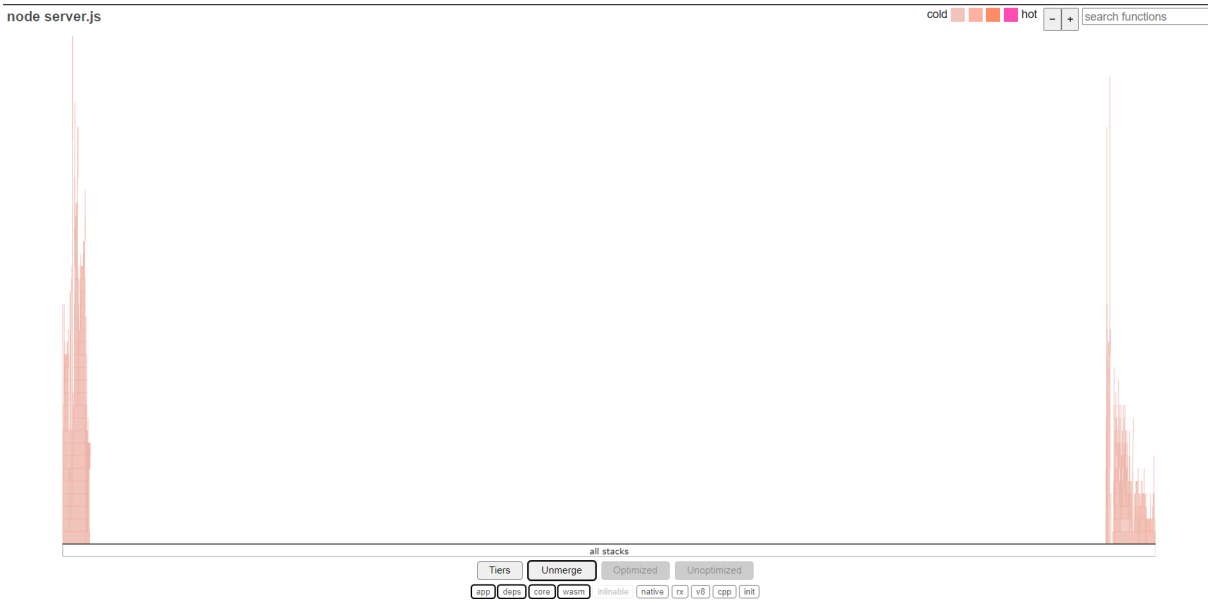
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	107 ms	117 ms	204 ms	220 ms	126.07 ms	23.63 ms	257 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	477	477	811	878	789.45	100.64	477
Bytes/Sec	1.39 MB	1.39 MB	2.37 MB	2.56 MB	2.31 MB	294 kB	1.39 MB

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

16k requests in 20.09s, 46.1 MB read
PS G:\proyectos\procect_nvo_backend\server\server\autocannon>
```

Por último cerramos el comando 0x index.js. Generando los diagramas de flama y demás archivos



conclusión:los procesos bloqueantes en este caso el console.log . Demora la ejecución de la aplicación por ende es menos eficiente nuestro servidor