COMANDO PARA EL ANÁLISIS DE RENDIMIENTO

(en la consola de powerSheel)
node –prof server.js
(en la console de Git Bash)
artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_bloq.txt
—----------
(en la consola de powerSheel)
node –prof server.js
(en la console de Git Bash)
artillery quick --count 20 -n 50 "http://localhost:8080/info/logger" > result_nobloq.txt
—----------
DECODIFICAR ARCHIVOS LOG
node --prof-process bloq-v8.log > result_prof_bloq.txt
node --prof-process nobloq-v8.log > result_prof_nobloq.txt

RESULTADO DEL ARCHIVO NO BLOQUEANTE

```
[Summary]:
  ticks  total  nonlib   name
     12    0.3%  100.0%  JavaScript
      0    0.0%    0.0%  C++
     13    0.3%  108.3%  GC
   4471   99.7%          Shared libraries
```

RESULTADO DEL ARCHIVO BLOQUEANTE

```
[Summary]:
  ticks  total  nonlib   name
      5    0.1%  100.0%  JavaScript
      0    0.0%    0.0%  C++
     16    0.3%  320.0%  GC
   5916   99.9%          Shared libraries
```

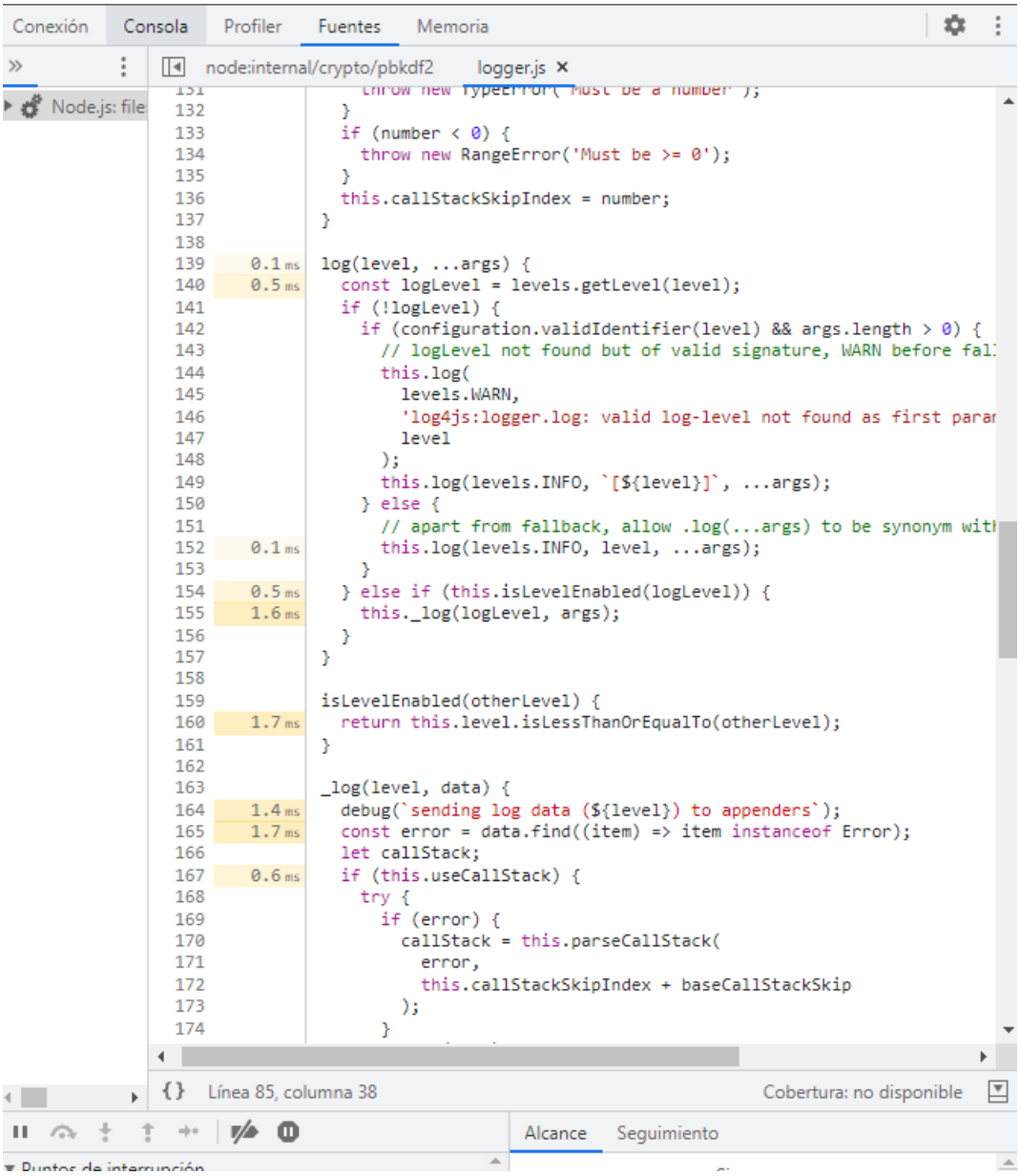
AHORA UTILIZAREMOS INSPECT

node --inspect profiling.js

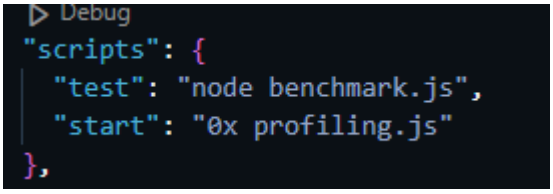
LUEGO ABRIMOS CON CHROME LA SIGUIENTE DIRECCIÓN

chrome://inspect

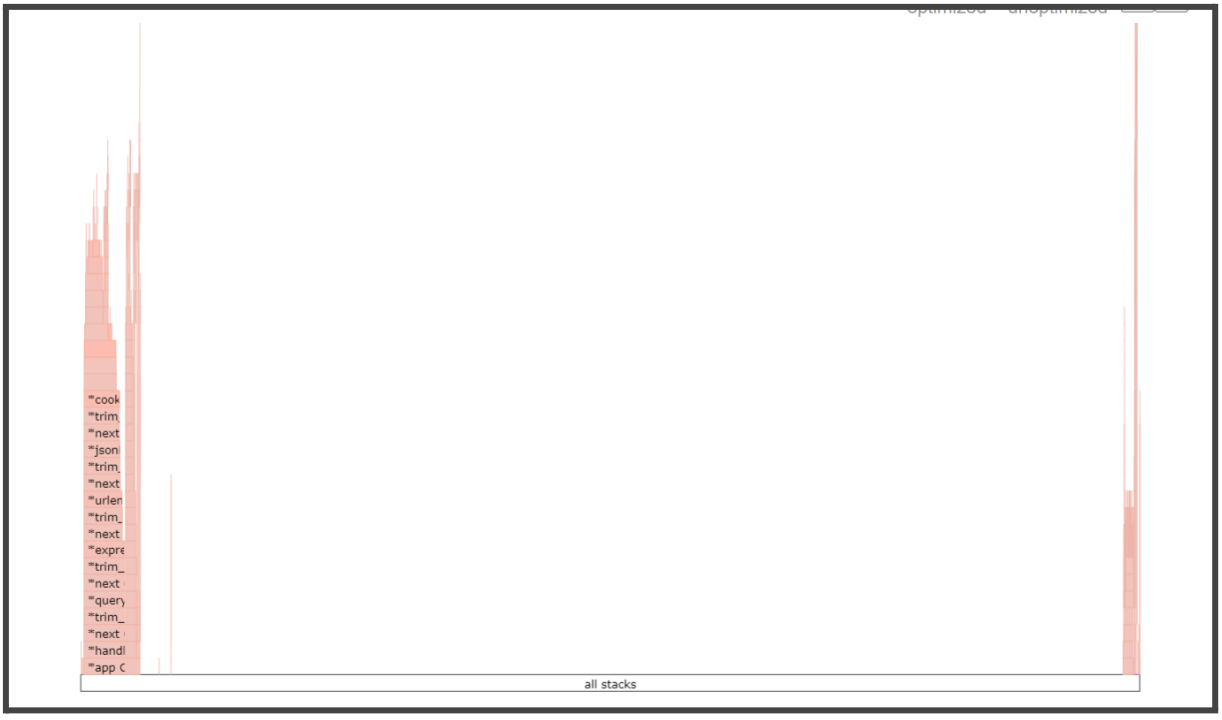
artillery quick --count 20 -n 50 "http://localhost:8080/info" > result_bloq.txt
artillery quick --count 20 -n 50 "http://localhost:8080/info/logger" > result_nobloq.txt



MODIFICAMOS EL PACKAGE.JSON



Y EJECUTAMOS LOS SIGUIENTES COMANDOS
(powershell)
npm start
(git bash)
npm test



En conclusión el uso de los logger al ser sincrónicos (no bloqueantes nos ayuda en el rendimiento de nuestra aplicación) si bien la diferencia no es muy grande la cantidad de procesos es menor (eso no sabemos por --prof-process) y tambien sabemos que el tiempo de ejecución es menor (gracias a lo que vemos en el diagrama flama con 0x)