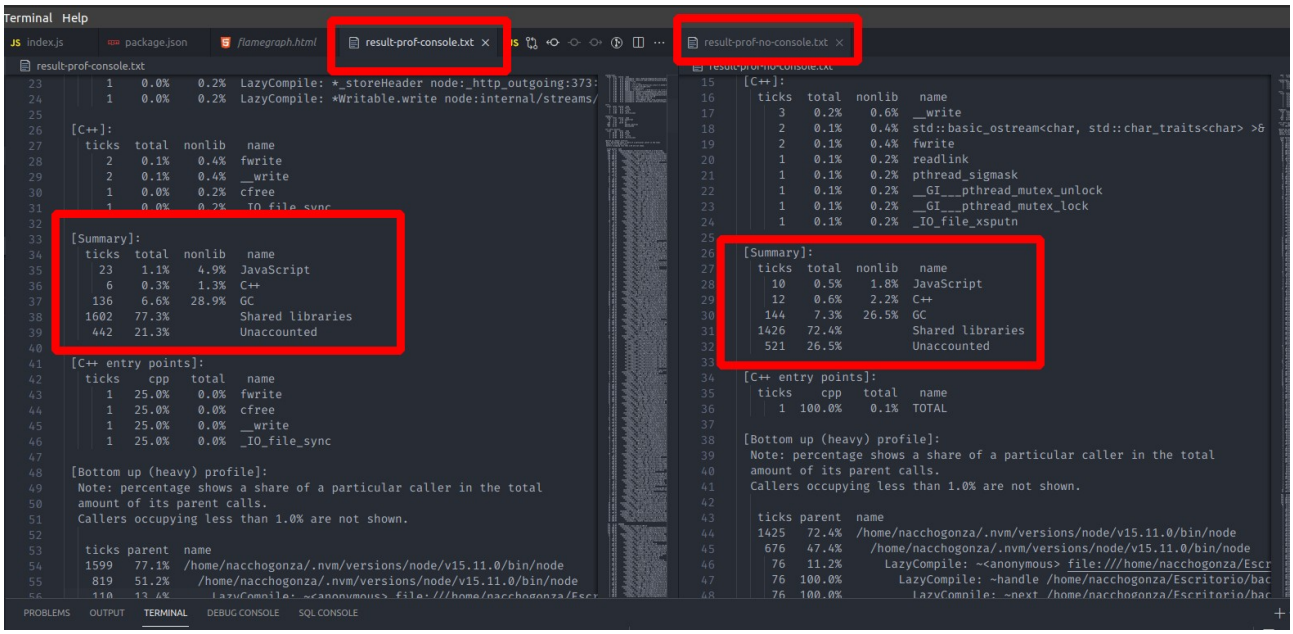
Profiling utilizando “node –prof” y artillery para realizar el test de carga.

(En el repositorio se encuentran los dos archivos correspondientes con la decodificación del .log completo)



En este caso podemos apreciar que el número de ticks es mayor en el caso de la ruta que cuenta con “console.log”, similar a lo que vimos en clase para el caso bloqueante vs. No bloqueante.

No me queda claro porque aparece una fila “Unaccounted” en este caso

Profiling con Autocannon + Inspect Google Chrome

- Resultado del test de carga con Autocannon

```
nacchogonza@nacchogonza-Lenovo-V330-14IKB:~/Escritorio/backend-mern-clase-32$ npm run autocannon

> entregable-22@1.0.0 autocannon
> node ./src/benchmark.js

Running all benchmarks in parallel...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	320 ms	375 ms	525 ms	537 ms	387.03 ms	63.9 ms	546 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	198	198	253	300	255	37.85	198
Bytes/Sec	136 kB	136 kB	174 kB	206 kB	175 kB	26 kB	136 kB

```
Req/Bytes counts sampled once per second.

5k requests in 19.07s, 3.5 MB read
Running 20s test @ http://localhost:8080/info-console
100 connections
```

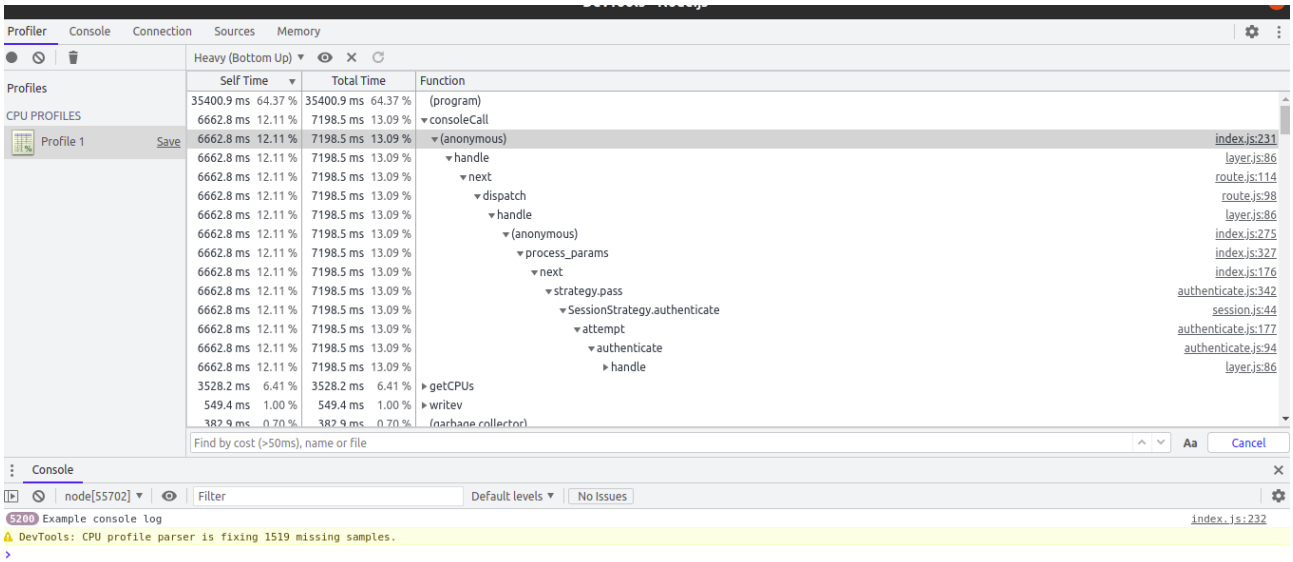
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	321 ms	372 ms	522 ms	531 ms	389.89 ms	58.73 ms	544 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	188	188	256	300	255	37.67	188
Bytes/Sec	129 kB	129 kB	176 kB	206 kB	175 kB	25.8 kB	129 kB

```
Req/Bytes counts sampled once per second.

5k requests in 19.09s, 3.5 MB read
nacchogonza@nacchogonza-Lenovo-V330-14IKB:~/Escritorio/backend-mern-clase
nacchogonza@nacchogonza-Lenovo-V330-14IKB:~/Escritorio/backend-mern-clase-32$
```

- Capturas del Inspector de Chrome: En primer lugar los procesos de la app (ordenados de mayor a menor por tiempo de duracion) y luego el detalle en el codigo de los tiempos de duración en los casos con y sin console.log



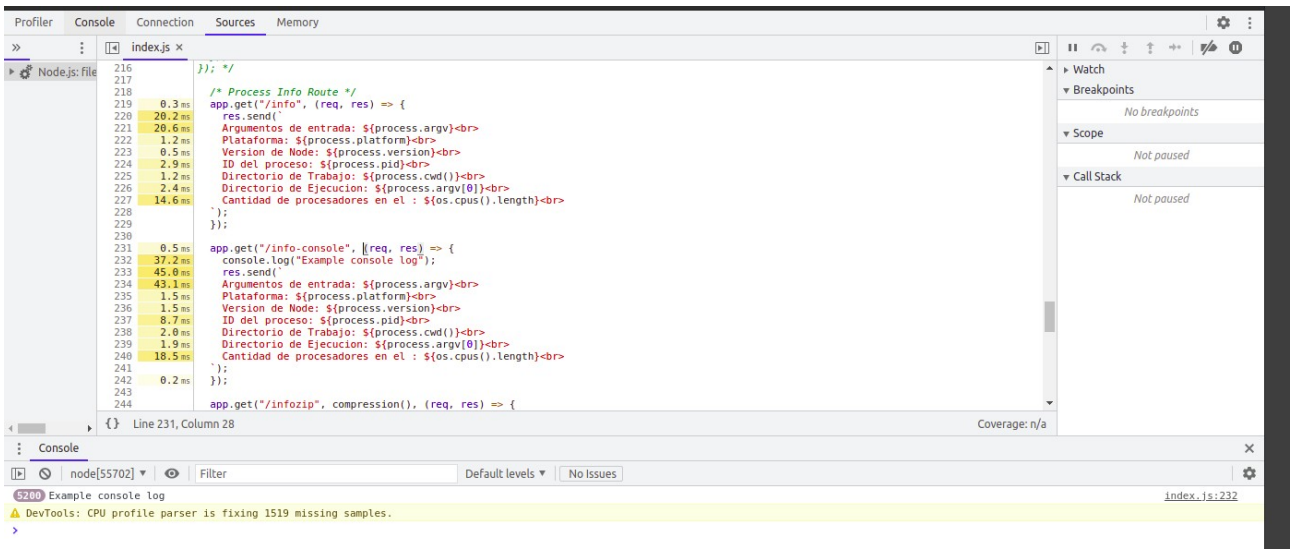
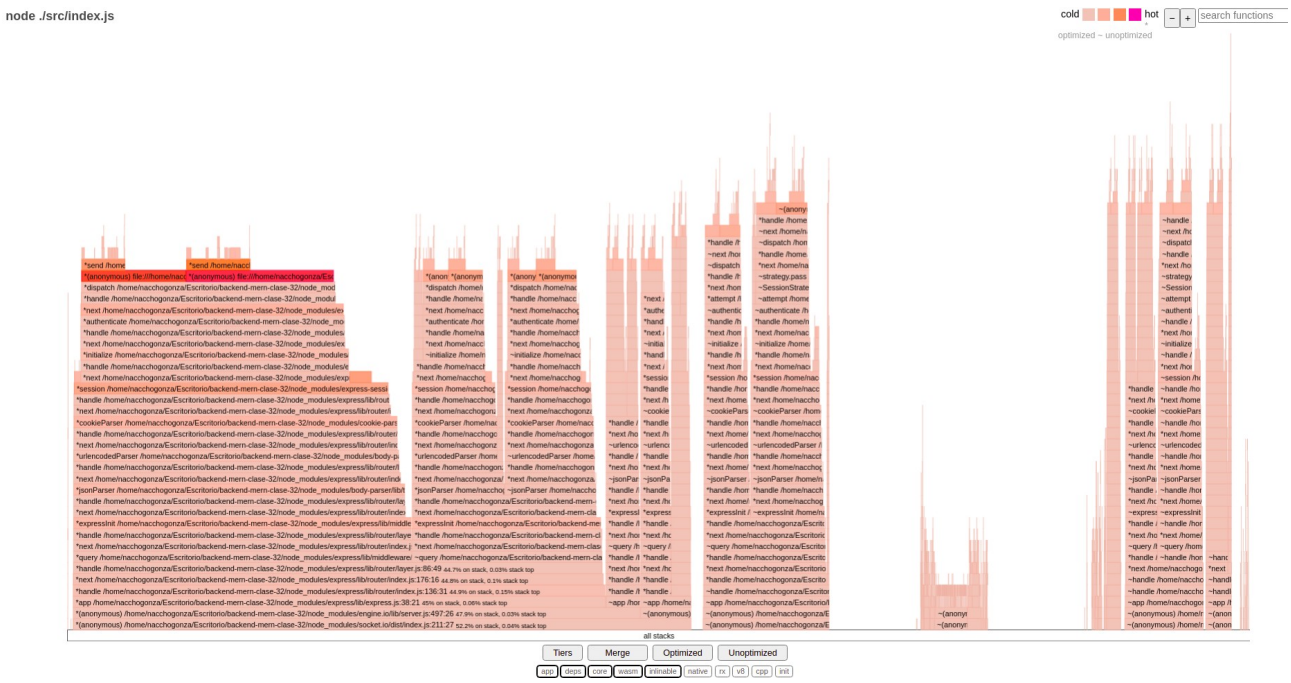
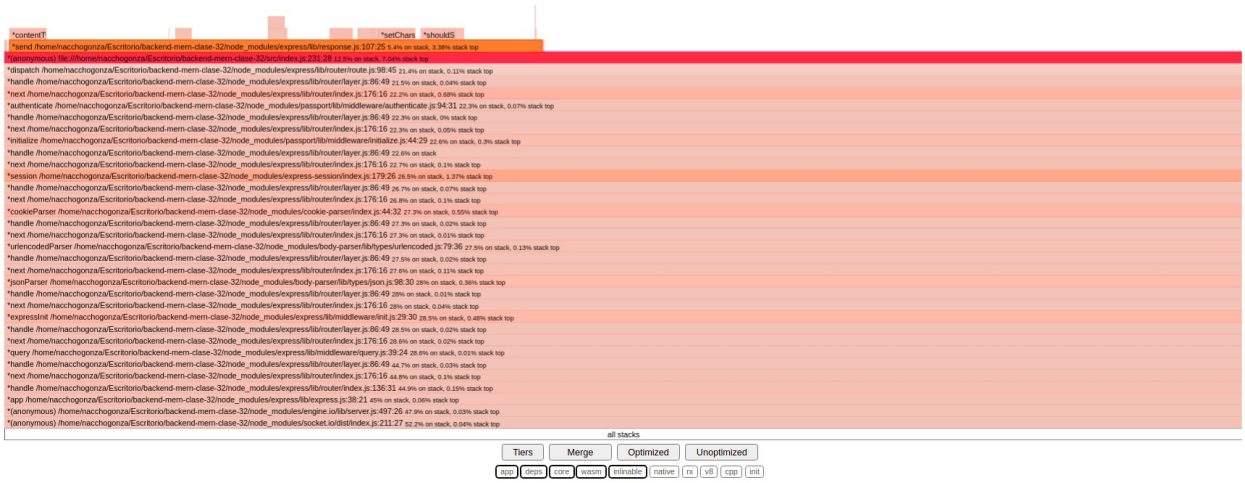

Profiler
Console
Connection
Sources
Memory
index.js
Node.js: file
}); */
/* Process Info Route */
app.get("/info", (req, res) => {
res.send(`
Argumentos de entrada: ${process.argv}
Plataforma: ${process.platform}
Version de Node: ${process.version}
ID del proceso: ${process.pid}
Directorio de Trabajo: ${process.cwd()}
Directorio de Ejecucion: ${process.argv[0]}
Cantidad de procesadores en el : ${os.cpus().length}
`);
});
app.get("/info-console", (req, res) => {
console.log("Example console log");
res.send(`
Argumentos de entrada: ${process.argv}
Plataforma: ${process.platform}
Version de Node: ${process.version}
ID del proceso: ${process.pid}
Directorio de Trabajo: ${process.cwd()}
Directorio de Ejecucion: ${process.argv[0]}
Cantidad de procesadores en el : ${os.cpus().length}
`);
});
app.get("/infozip", compression(), (req, res) => {
Line 231, Column 28
Coverage: n/a
Watch
Breakpoints
No breakpoints
Scope
Not paused
Call Stack
Not paused
Console
node[55702]
Filter
Default levels
No Issues
5200 Example console log
index.js:232
DevTools: CPU profile parser is fixing 1519 missing samples.

Diagrama de Flama con 0x

- Del lado izquierdo y mas duradero vemos el caso con el console.log, en el cual se marca como HOT el código a partir de la línea 231 del código (correspondiente al endpoint del /info-console)

- Debajo podemos ver en más detalle el proceso del caso del endpoint de console.log





A modo de conclusión, creo que en cada uno de los casos, tanto en las capturas como en los archivos de resultados de test de carga que quedaron en el repositorio, podemos observar como los tiempos de repuesta son mayores en el caso del endpoint con “console.log”, demostrando asi como cada pequeño proceso que forme parte de la respuesta que se da desde el servidor va a generar un mayor tiempo de respuesta para poder realizar la acción determinada

Por otra parte, creo que es muy buena la comparación visual que puede hacerse en el diagrama de flamas con 0x, que nos demuestra de forma muy simple el incremento de tiempo de un simple “console.log” y como en ese caso aparece el proceso calificado como Hot que aumenta asi los tiempos de respuesta