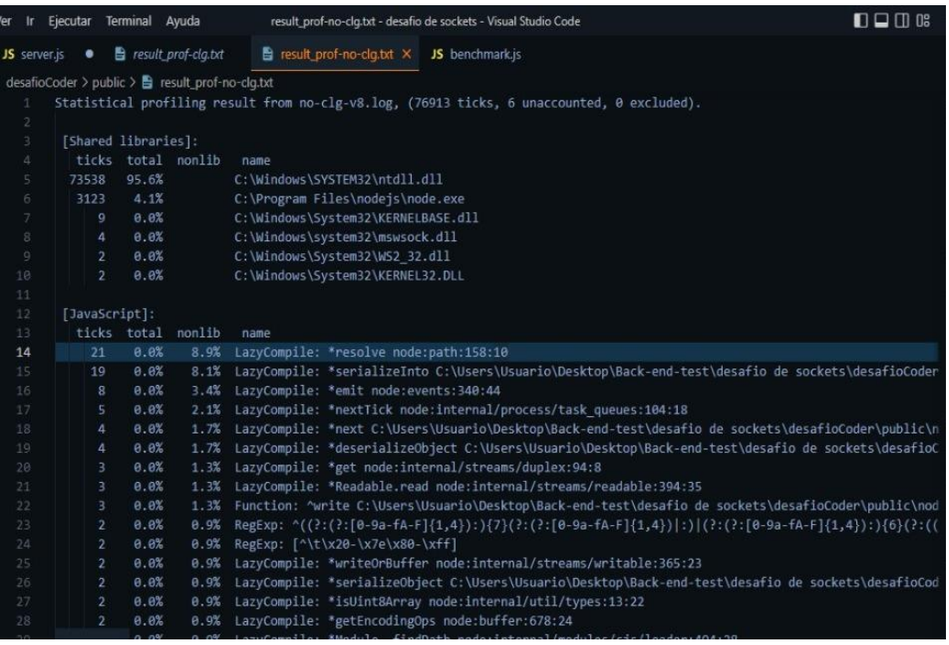
Informe de desafio de logs , y profilling

Tomas Juarez

sobre la ruta '/info', en modo fork, agregando ó extrayendo un console.log se hace un perfilamiento del servidor con --prof de node.js

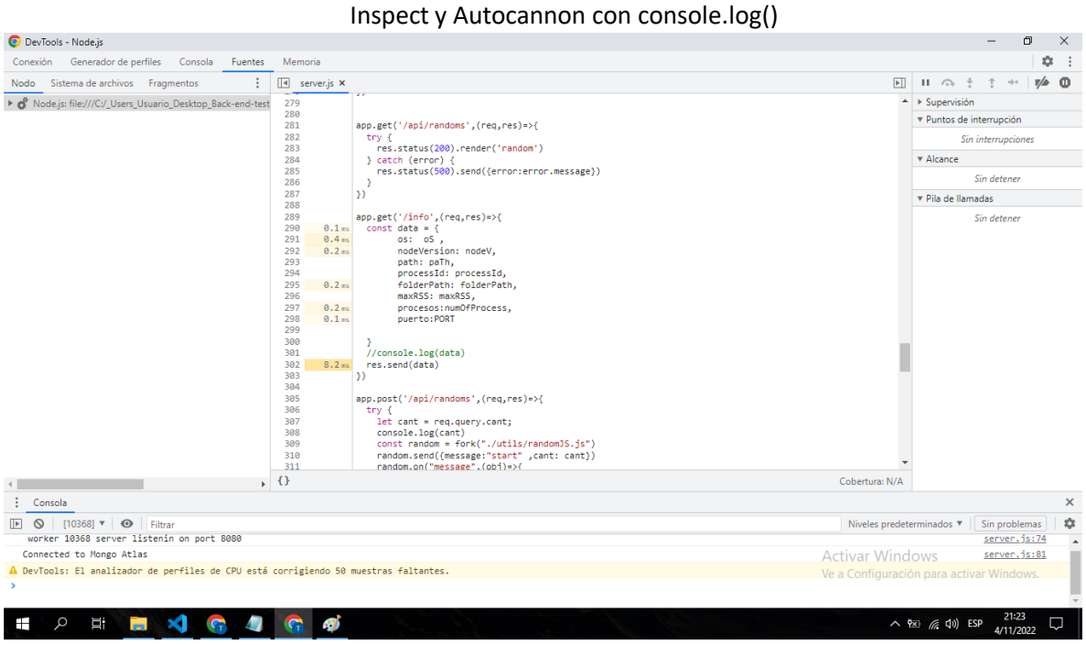
y procesando los datos con --prof-process

```
286     }
287   })
288
289   app.get('/info',(req,res)=>{
290     const data = {
291           os:  oS ,
292           nodeVersion: nodeV,
293           path: paTh,
294           processId: processId,
295           folderPath: folderPath,
296           maxRSS: maxRSS,
297           procesos:numOfProcess,
298           puerto:PORT
299
300     }
301     console.log(data)
302     res.send(data)
303   })
```

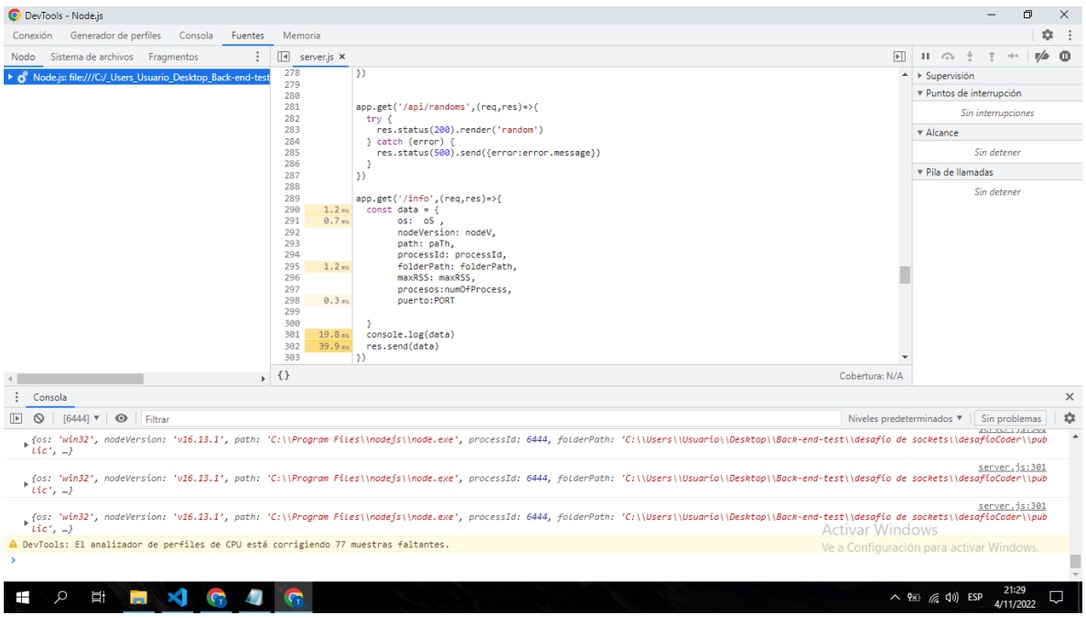


Como se puede ver en los resultados de los test (con console.log() en la imagen izquierda y sin console.log en la imagen de la derecha), en el test sin console.log() la cantidad de ticks disminuyo considerablemente .

Inspect y Autocannon con console.log()



Inspect y autocannon sin console.log()



Conclusión:

En el caso de inspect y autocannon vemos también que en el caso en el que el console.log() esta comentado se reduce el tiempo de renderizado unos 31,7 ms , en conclusión el uso de console.log() genera un perdida considerable de performance ya que consume demasiados recursos.