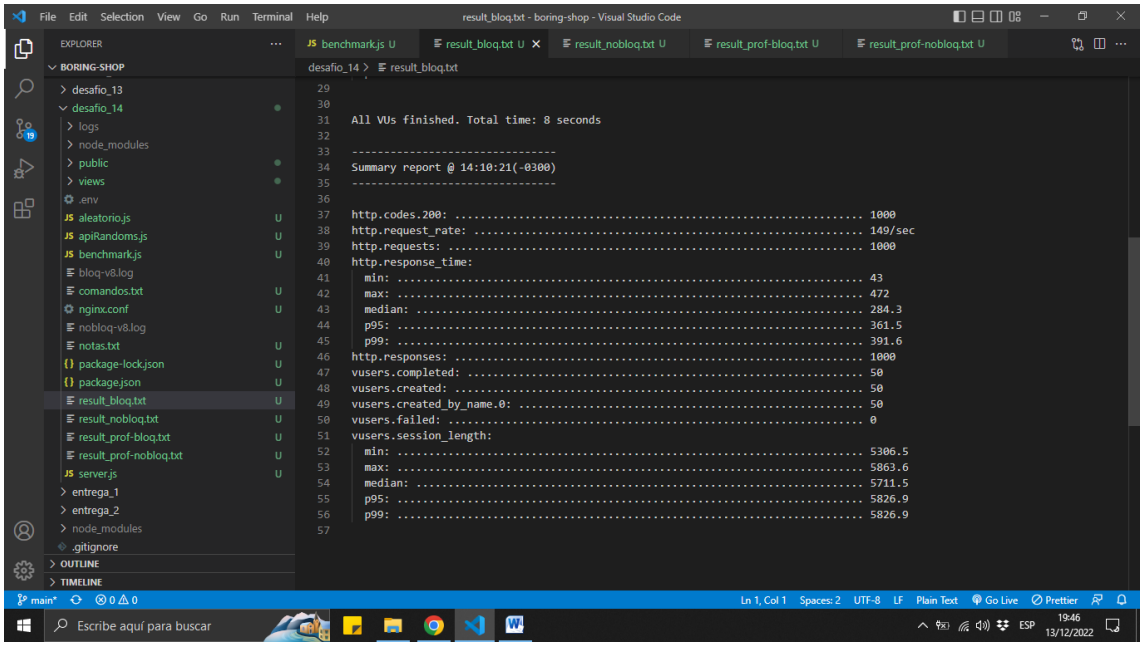
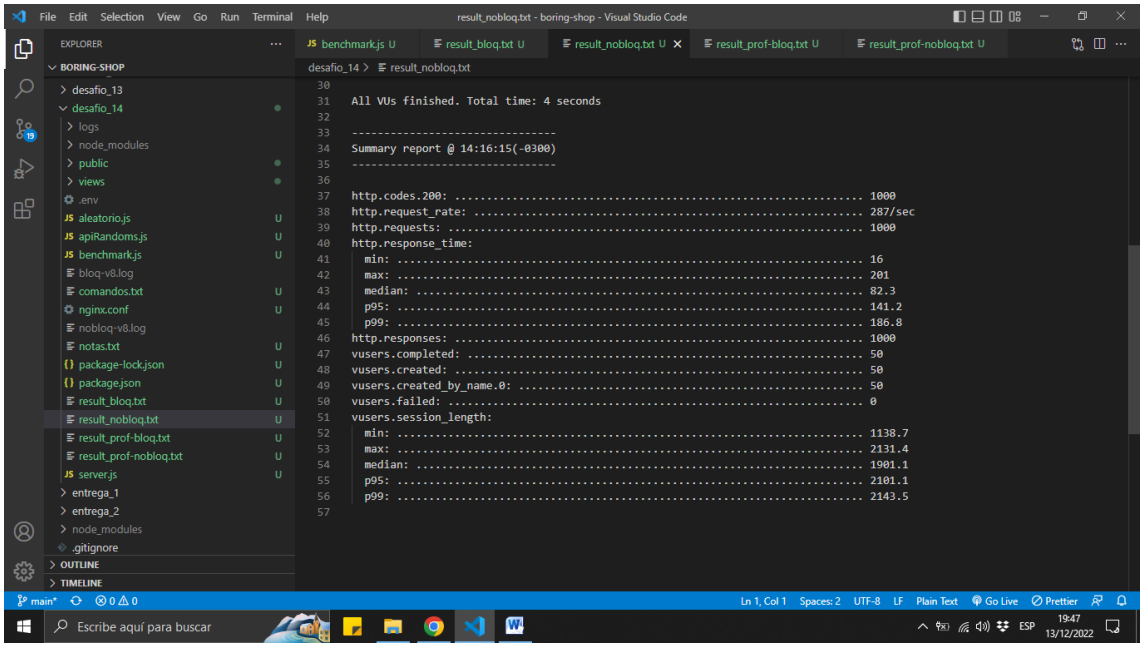
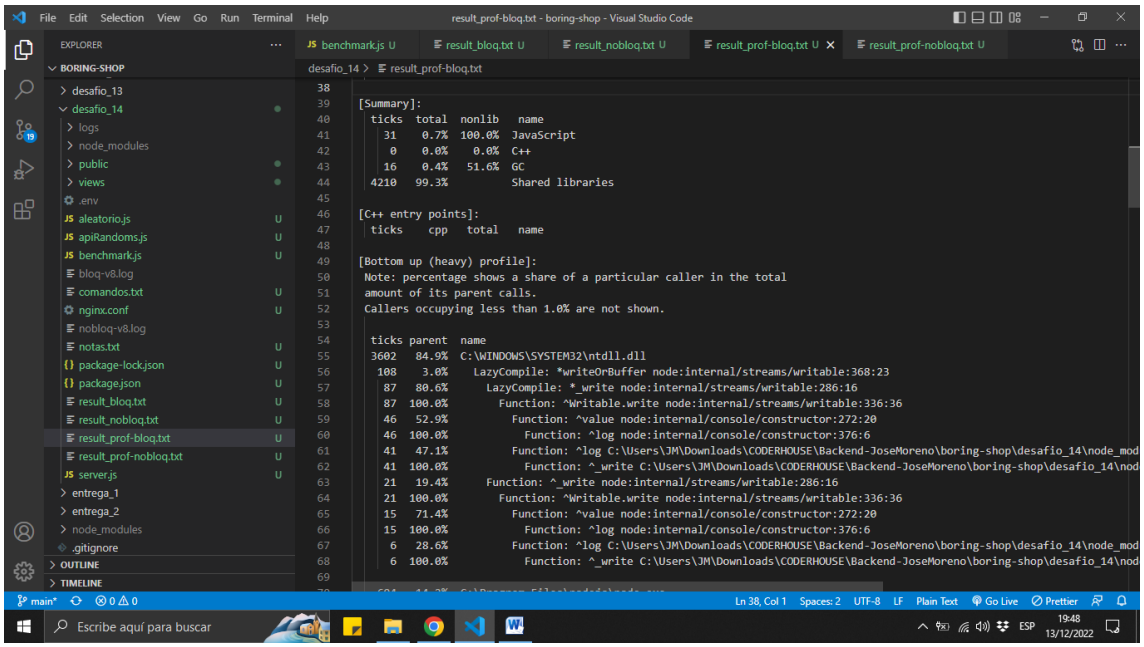
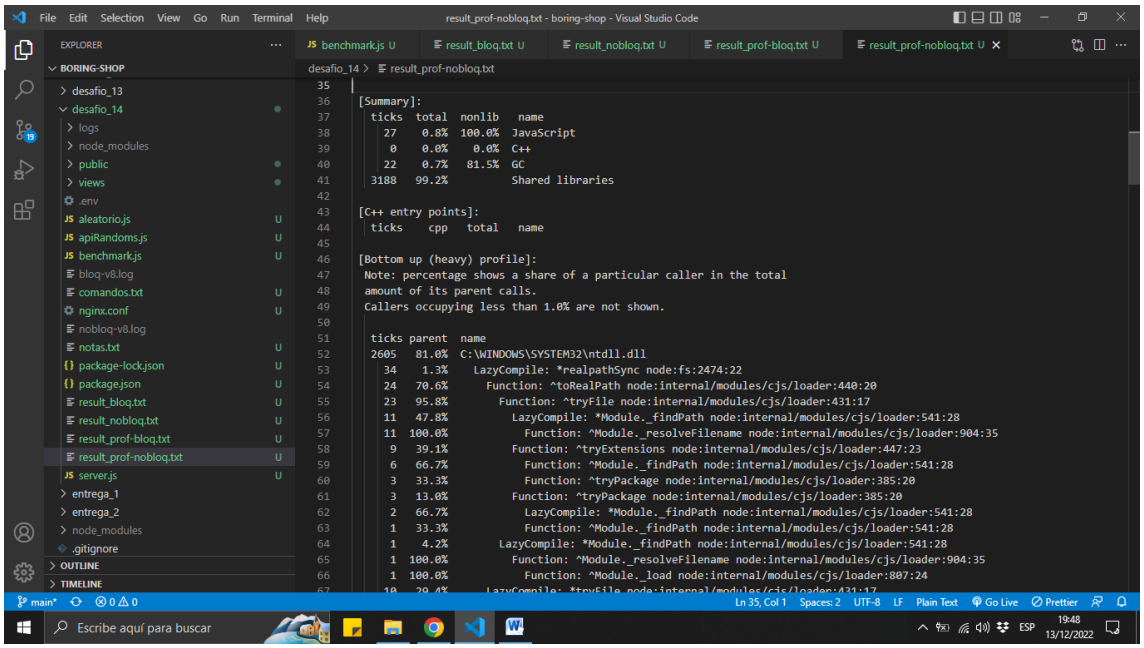
1) El perfilamiento del servidor, realizando el test con --prof de node.js. Analizar los resultados obtenidos luego de procesarlos con --prof-process.

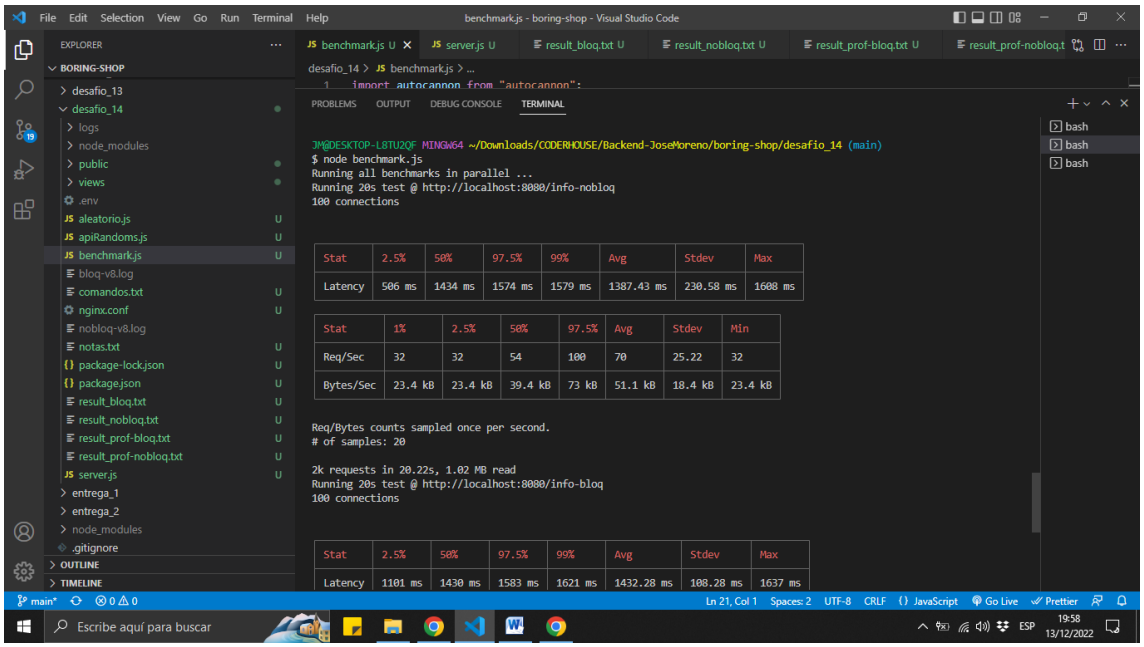


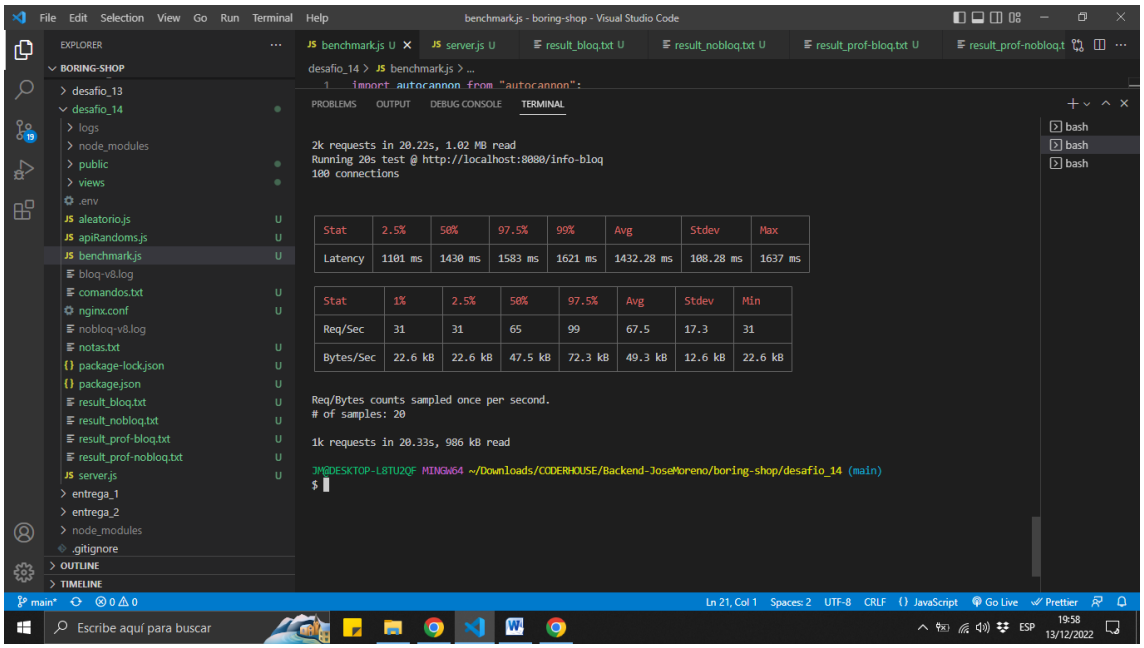


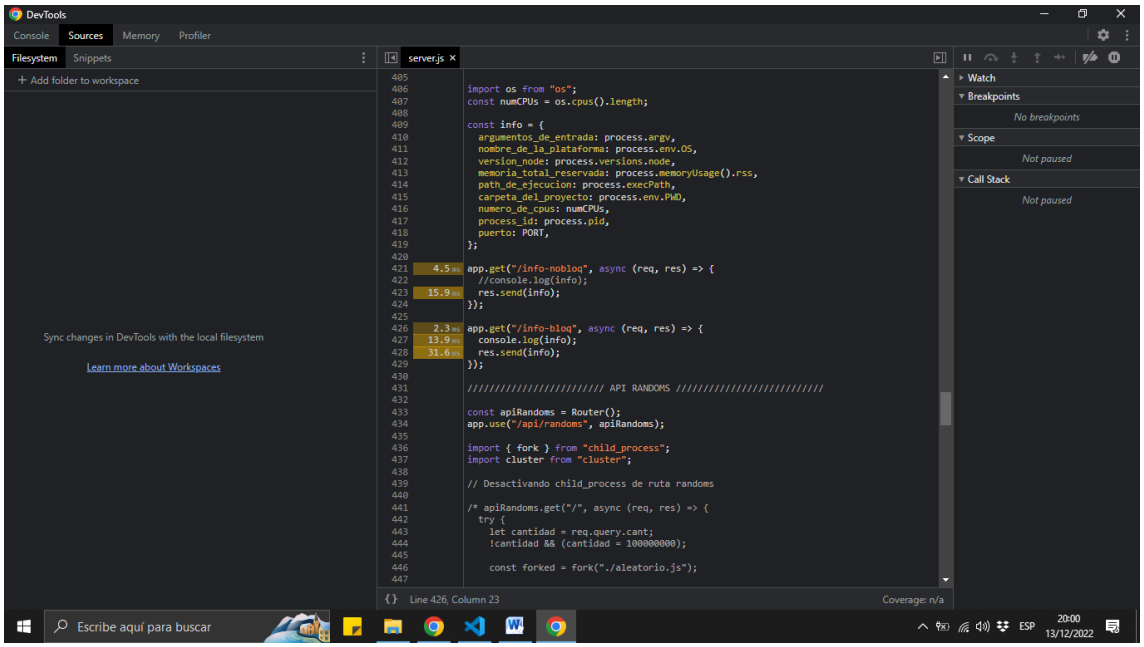




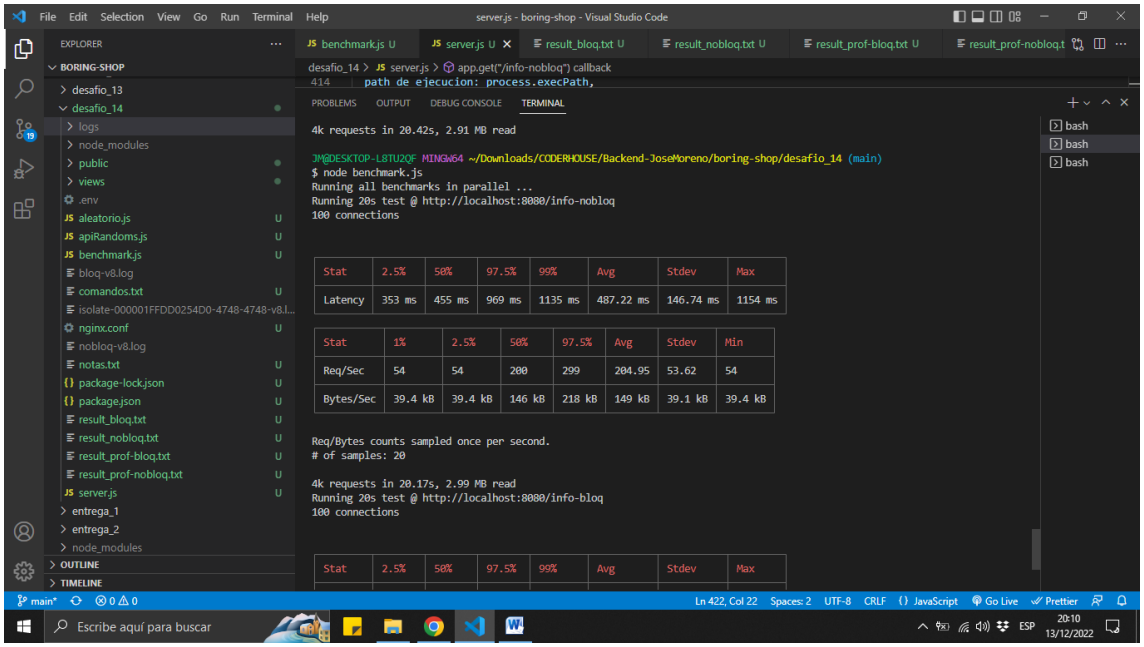
2) El perfilamiento del servidor con el modo inspector de node.js --inspect. Revisar el tiempo de los procesos menos performantes sobre el archivo fuente de inspección.

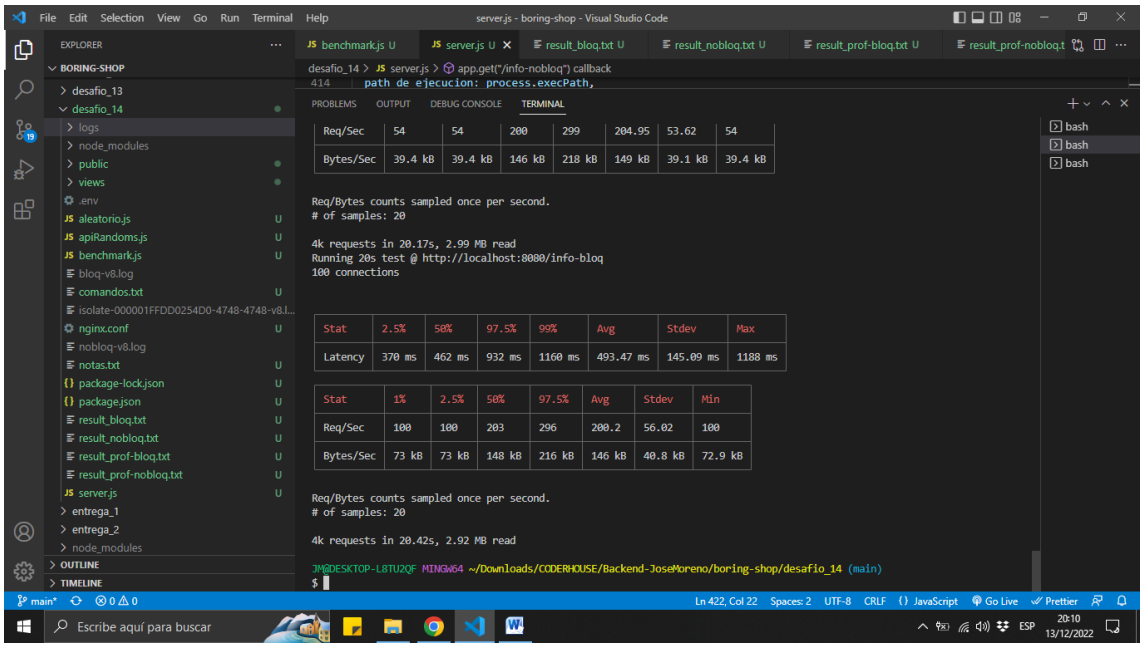

File Edit Selection View Go Run Terminal Help
benchmark.js - boring-shop - Visual Studio Code
EXPLORER
BORING-SHOP
desafio_13
desafio_14
logs
node_modules
public
views
.env
aleatorio.js
apiRandoms.js
benchmark.js
bloq-v8.log
comandos.txt
nginx.conf
nobloq-v8.log
notas.txt
package-lock.json
package.json
result_bloq.txt
result_nobloq.txt
result_prof-bloq.txt
result_prof-nobloq.txt
server.js
entrega_1
entrega_2
node_modules
.gitignore
OUTLINE
TIMELINE
benchmark.js U
server.js U
result_bloq.txt U
result_nobloq.txt U
result_prof-bloq.txt U
result_prof-nobloq.t
desafio_14 > benchmark.js > ...
1 import autocannon from "autocannon";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
bash
bash
JM@DESKTOP-L8TU2QF MINGW64 ~/Downloads/CODERHOUSE/Backend-JoseMoreno/boring-shop/desafio_14 (main)
$ node benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info-nobloq
100 connections
Stat 2.5% 50% 97.5% 99% Avg Stdev Max
Latency 506 ms 1434 ms 1574 ms 1579 ms 1387.43 ms 230.58 ms 1608 ms
Stat 1% 2.5% 50% 97.5% Avg Stdev Min
Req/Sec 32 32 54 100 70 25.22 32
Bytes/Sec 23.4 kB 23.4 kB 39.4 kB 73 kB 51.1 kB 18.4 kB 23.4 kB
Req/Bytes counts sampled once per second.
# of samples: 20
2k requests in 20.22s, 1.02 MB read
Running 20s test @ http://localhost:8080/info-bloq
100 connections
Stat 2.5% 50% 97.5% 99% Avg Stdev Max
Latency 1101 ms 1430 ms 1583 ms 1621 ms 1432.28 ms 108.28 ms 1637 ms
main* 0 0
Ln 21, Col 1 Spaces: 2 UTF-8 CRLF JavaScript Go Live Prettier
Escribe aquí para buscar
ESP 19:58 13/12/2022

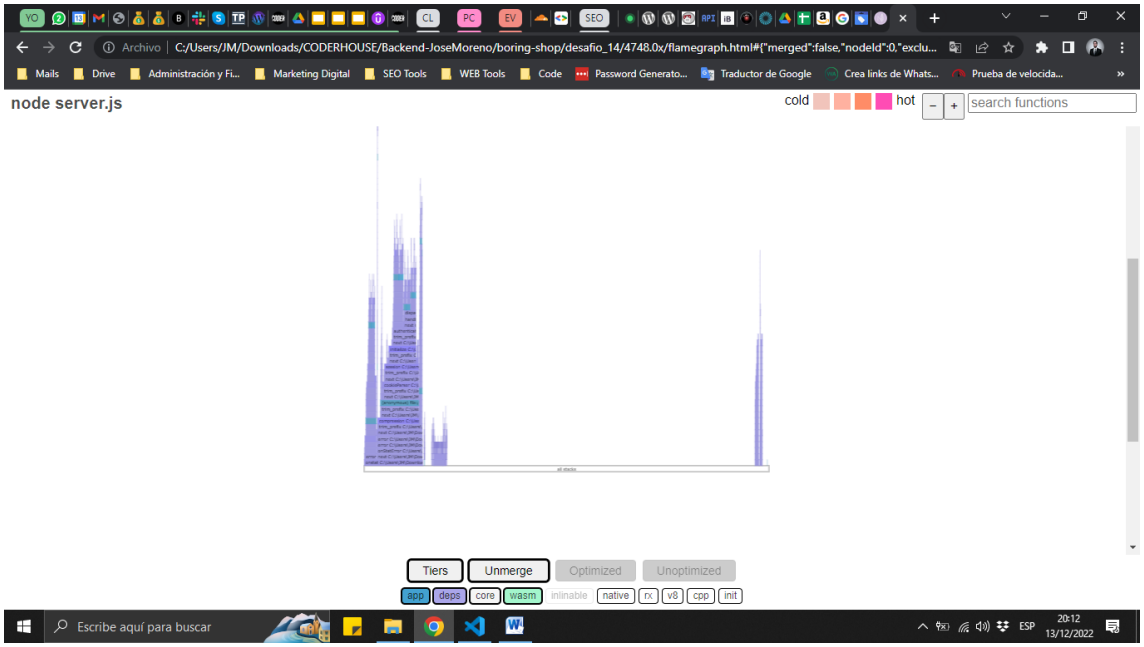

File Edit Selection View Go Run Terminal Help
benchmark.js - boring-shop - Visual Studio Code
EXPLORER
BORING-SHOP
desafio_13
desafio_14
logs
node_modules
public
views
.env
aleatorio.js
apiRandoms.js
benchmark.js
bloq-v8.log
comandos.txt
nginx.conf
nobloq-v8.log
notas.txt
package-lock.json
package.json
result_bloq.txt
result_nobloq.txt
result_prof-bloq.txt
result_prof-nobloq.txt
server.js
entrega_1
entrega_2
node_modules
.gitignore
OUTLINE
TIMELINE
benchmark.js U
server.js U
result_bloq.txt U
result_nobloq.txt U
result_prof-bloq.txt U
result_prof-nobloq.t
desafio_14 > benchmark.js > ...
1 import autocannon from "autocannon";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash
bash
bash
2k requests in 20.22s, 1.02 MB read
Running 20s test @ http://localhost:8080/info-bloq
100 connections
Stat 2.5% 50% 97.5% 99% Avg Stdev Max
Latency 1101 ms 1430 ms 1583 ms 1621 ms 1432.28 ms 108.28 ms 1637 ms
Stat 1% 2.5% 50% 97.5% Avg Stdev Min
Req/Sec 31 31 65 99 67.5 17.3 31
Bytes/Sec 22.6 kB 22.6 kB 47.5 kB 72.3 kB 49.3 kB 12.6 kB 22.6 kB
Req/Bytes counts sampled once per second.
# of samples: 20
1k requests in 20.33s, 986 kB read
JM@DESKTOP-L8TU2QF MINGW64 ~/Downloads/CODERHOUSE/Backend-JoseMoreno/boring-shop/desafio_14 (main)
$
main* 0 0
Ln 21, Col 1 Spaces: 2 UTF-8 CRLF JavaScript Go Live Prettier
Escribe aquí para buscar
ESP 19:58 13/12/2022



3) El diagrama de flama con 0x, emulando la carga con Autocannon con los mismos parámetros anteriores.







4) Conclusión: En todas las pruebas realizadas se confirma que los procesos bloqueantes son menos eficientes y veloces que los procesos no bloqueantes.