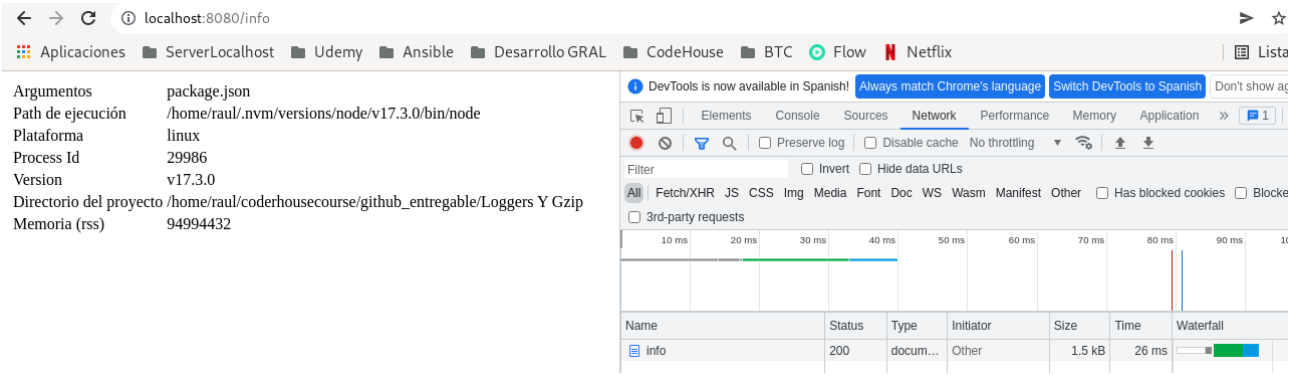
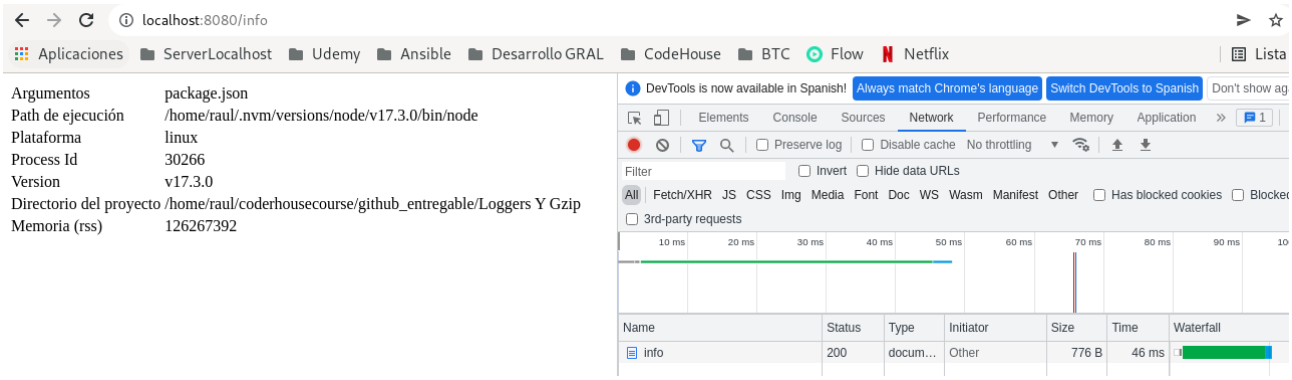
Informe Logger y Gzip

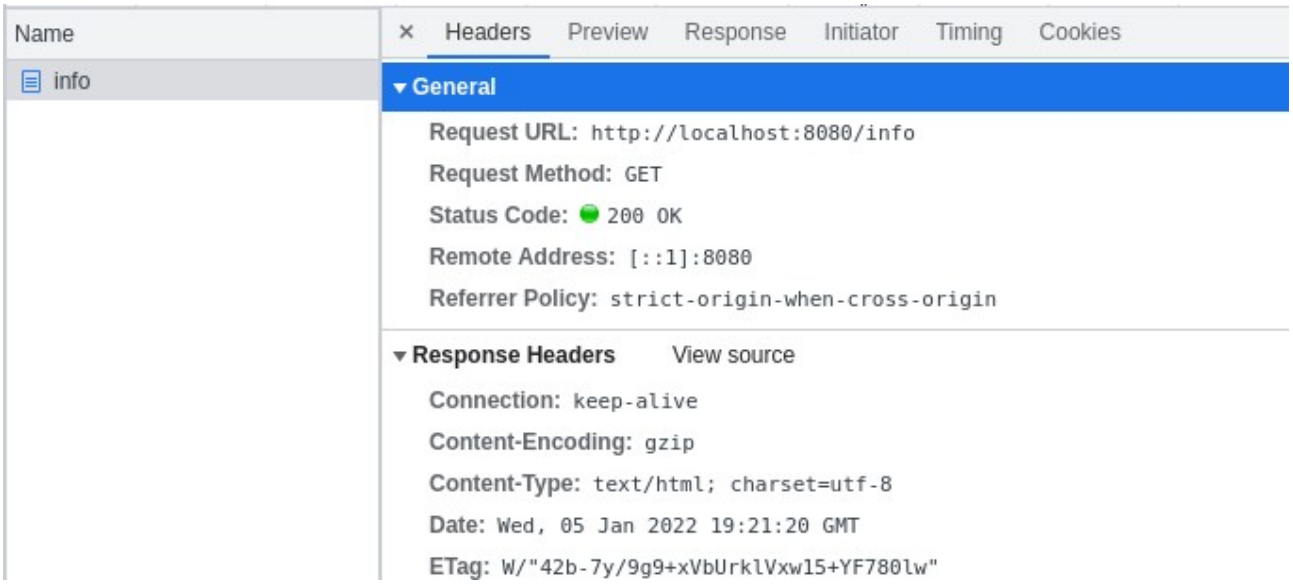
Compresion con Gzip

Sin Gzip



con Gzip





Logger

Para log se utilizo winston, el codigo del logger esta en /utils/logger.js
Los archivos de logs de warn como de error se guardan en /logs

Analisis

Con el comando node --prof server.js se generó **isolate-0x6371090-74757-v8.log**
Con el comando node --prof-process **isolate-0x6371090-74757-v8.log** > **processed.txt**
Se envio al archivo processed.txt el resultado de la ejecusion del comando antes mencionando

Dentro del apartado [JavaScript] la mayor cantidad de ticks la tiene el proceso random.js

[JavaScript]:
ticks total nonlib name
1583 21.9% 66.9% LazyCompile: *<anonymous>
file:///home/raul/coderhousecourse/github_entregable/Loggers%20Y%20Gzip/routes/
random.js:10:23

Se ejecutó:

artillery quick --count 50 -n 20 http://localhost:8080/info > result_info_con_console.txt

En /info sin console.log el comando
artillery quick --count 50 -n 20 http://localhost:8080/info > result_info_sin_console.txt

Analizando los archivos result_info_con_console.txt y result_info_sin_console.txt el valor de response es mayor al usar console.log

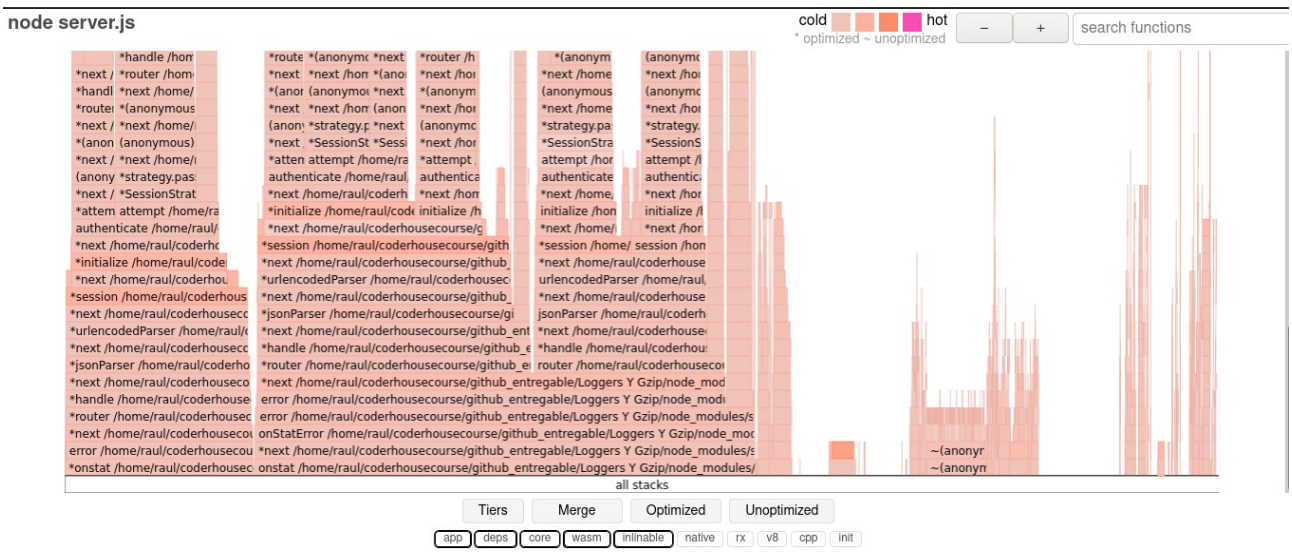
Se uso benchmark.js para la ejecución de autocannon y se modificó package.json con los scripts a ejecutar
Comandos: test y start
"test": "node benchmark.js",
"start": "0x server.js"

npm test

```
raul@debian:~/coderhousecourse/github_entregable/Loggers Y Gzip$ npm test

> clase-16@1.0.0 test
> node benchmark.js

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

┌─────────┬──────┬──────┬──────┬──────┬──────────┬──────────┬──────┐
│ Stat    │ 2.5% │ 50%  │ 97.5%│ 99%  │ Avg      │ Stdev    │ Max  │
├─────────┼──────┼──────┼──────┼──────┼──────────┼──────────┼──────┤
│ Latency │318 ms│479 ms│802 ms│806 ms│475.66 ms │116.14 ms │888 ms│
└─────────┴──────┴──────┴──────┴──────┴──────────┴──────────┴──────┘
┌─────────┬──────┬──────┬──────┬──────┬──────┬──────┬──────┐
│ Stat    │ 1%   │ 2.5% │ 50%  │ 97.5%│ Avg  │ Stdev│ Min  │
├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┤
│ Req/Sec │ 100  │ 100  │ 200  │ 298  │210.05│54.76 │ 100  │
├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┤
│ Bytes/Sec│42.7 kB│42.7 kB│85.4 kB│127 kB│89.6 kB│23.4 kB│42.7 kB│
└─────────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┘

Req/Bytes counts sampled once per second.

4k requests in 20.08s, 1.79 MB read
Running 20s test @ http://localhost:8080/info
100 connections

┌─────────┬──────┬──────┬──────┬──────┬──────────┬──────────┬──────┐
│ Stat    │ 2.5% │ 50%  │ 97.5%│ 99%  │ Avg      │ Stdev    │ Max  │
├─────────┼──────┼──────┼──────┼──────┼──────────┼──────────┼──────┤
│ Latency │317 ms│488 ms│800 ms│804 ms│481.85 ms │119.81 ms │895 ms│
└─────────┴──────┴──────┴──────┴──────┴──────────┴──────────┴──────┘
┌─────────┬──────┬──────┬──────┬──────┬──────┬──────┬──────┐
│ Stat    │ 1%   │ 2.5% │ 50%  │ 97.5%│ Avg  │ Stdev│ Min  │
├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┤
│ Req/Sec │ 100  │ 100  │ 200  │ 300  │205.5 │56.21 │ 100  │
├─────────┼──────┼──────┼──────┼──────┼──────┼──────┼──────┤
│ Bytes/Sec│150 kB│150 kB│300 kB│450 kB│308 kB│84.3 kB│150 kB│
└─────────┴──────┴──────┴──────┴──────┴──────┴──────┴──────┘

Req/Bytes counts sampled once per second.

4k requests in 20.11s, 6.16 MB read
```

npm start

El resultado de la ejecucion se encuentra en la carpeta 6328.0x resultado de “0x server.js”



Sobre el perfilamiento del server **modo inspector de node.js**
se obtuvo el archivo **CPU-20220123T133321.cpuprofile** en el cual se ve que el archivo menos performante es **random.js**