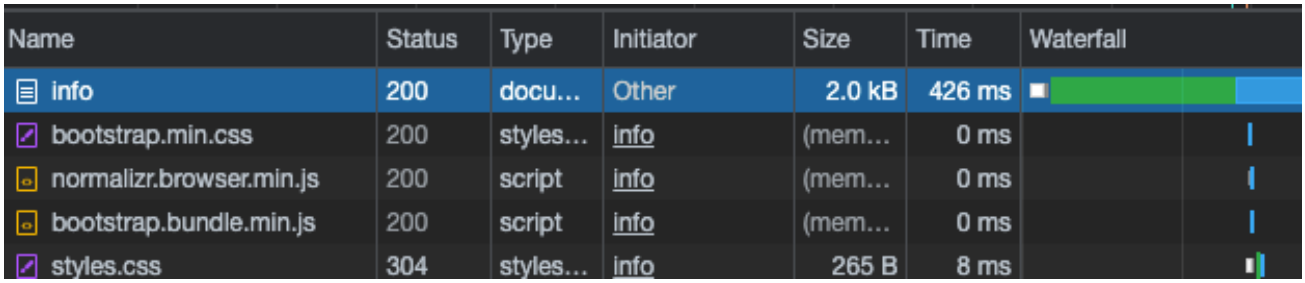
Rodrigo Cabrera

Informe de gzip y profiling

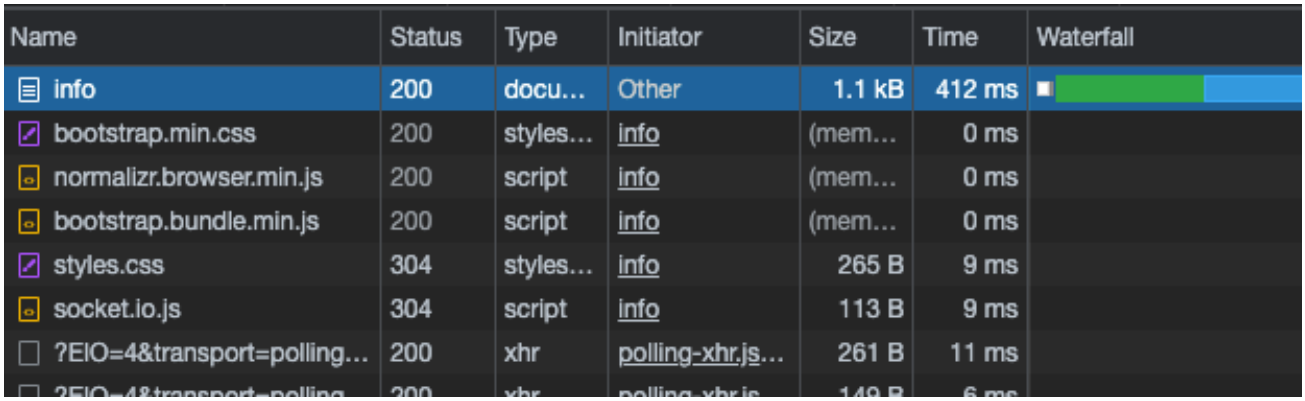
1. Resultados de la compresión con GZIP

SIN GZIP



Name	Status	Type	Initiator	Size	Time	Waterfall
info	200	docu...	Other	2.0 kB	426 ms	
bootstrap.min.css	200	styles...	info	(mem...	0 ms	
normalizr.browser.min.js	200	script	info	(mem...	0 ms	
bootstrap.bundle.min.js	200	script	info	(mem...	0 ms	
styles.css	304	styles...	info	265 B	8 ms	

CON GZIP



Name	Status	Type	Initiator	Size	Time	Waterfall
info	200	docu...	Other	1.1 kB	412 ms	
bootstrap.min.css	200	styles...	info	(mem...	0 ms	
normalizr.browser.min.js	200	script	info	(mem...	0 ms	
bootstrap.bundle.min.js	200	script	info	(mem...	0 ms	
styles.css	304	styles...	info	265 B	9 ms	
socket.io.js	304	script	info	113 B	9 ms	
?EIO=4&transport=polling...	200	xhr	polling-xhr.js...	261 B	11 ms	

Compresión: 0.9 *kB*

2. Profiling con node y Artillery

CON CONSOLE.LOG()

```
--------------------------------
Summary report @ 15:17:58(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 14/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 7
  max: .............................................................................. 4550
  median: ........................................................................... 214.9
  p95: .............................................................................. 1022.7
  p99: .............................................................................. 2725
```

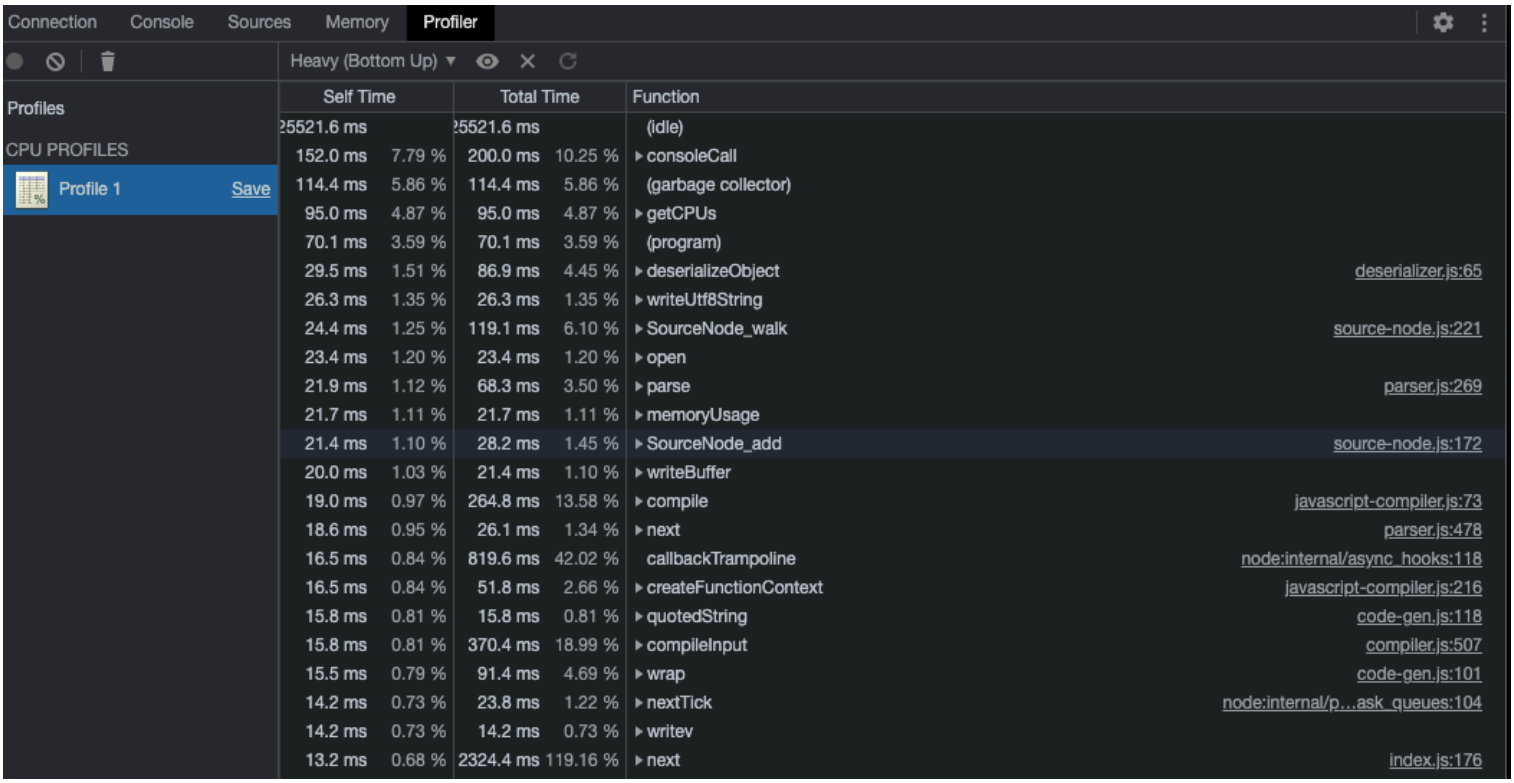
SIN CONSOLE.LOG()

```
-------------------------------
Summary report @ 15:22:02(-0300)
-------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 14/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 5
  max: .............................................................................. 4631
  median: ........................................................................... 219.2
  p95: .............................................................................. 1130.2
  p99: .............................................................................. 2836.2
http.responses: ..................................................................... 1000
```

- Llama la atención que el promedio de tiempo de respuesta sea más alto para los requests que no utilizaron console.log

3. Profiling con node inspect y chrome



- Tiene sentido que el proceso que más haya demandado al servidor sea la impresión por consola de cada request (por parte de log4js)

4. Impresión por consola de Autocannon

```
> loggers@1.0.0 test
> node ac_test.js

Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	10 ms	6470 ms	9625 ms	9633 ms	4646.99 ms	3638.72 ms	9636 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	4	46	7.75	10.72	1
Bytes/Sec	0 B	0 B	8.97 kB	103 kB	17.4 kB	24.1 kB	2.24 kB

```
Req/Bytes counts sampled once per second.
# of samples: 20

310 requests in 20.13s, 348 kB read
55 errors (55 timeouts)
```

El informe de 0x está disponible en la carpeta 0x_profiling ubicada en el root del proyecto