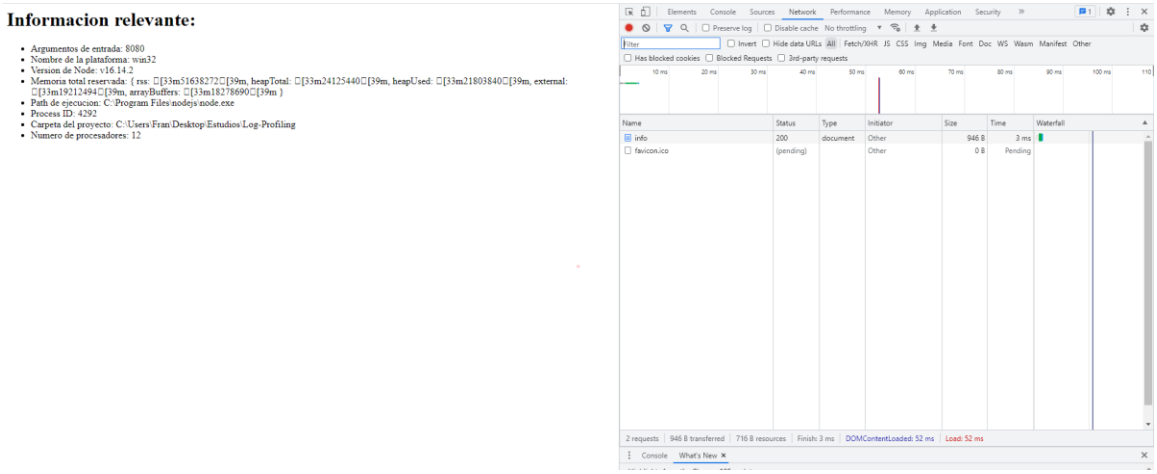
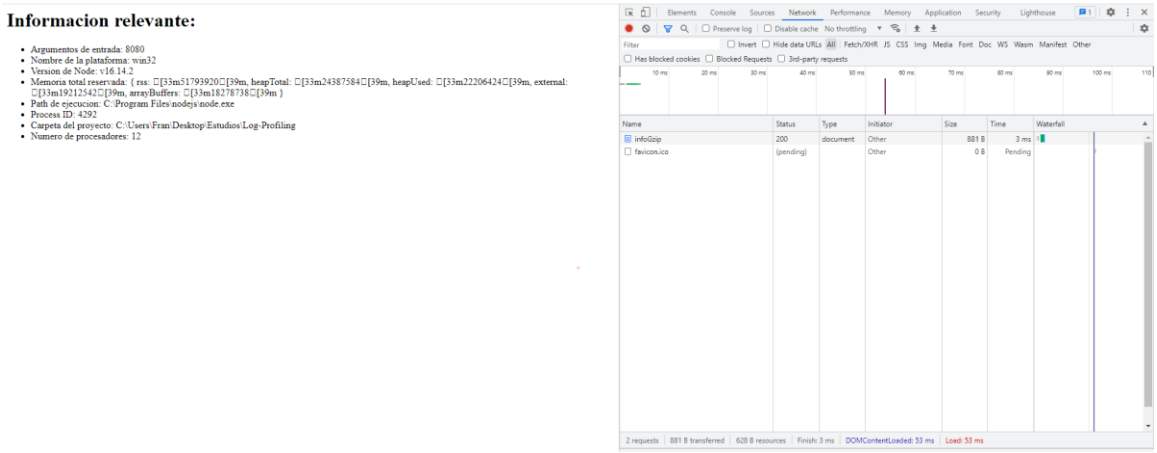
Desafio entregable N°32: Loggers, gzip y análisis de performance

Mejorando el rendimiento con Gzip:

Sin Gzip da un resultado de 946 B.



Con Gzip, el resultado mejora siendo 881 B.



Los comandos que se ejecutan en consola son:

node --prof index.js <por default 8080>

artillery quick --count=20 --num=50 http:localhost:8080/info > result_nobloq.txt

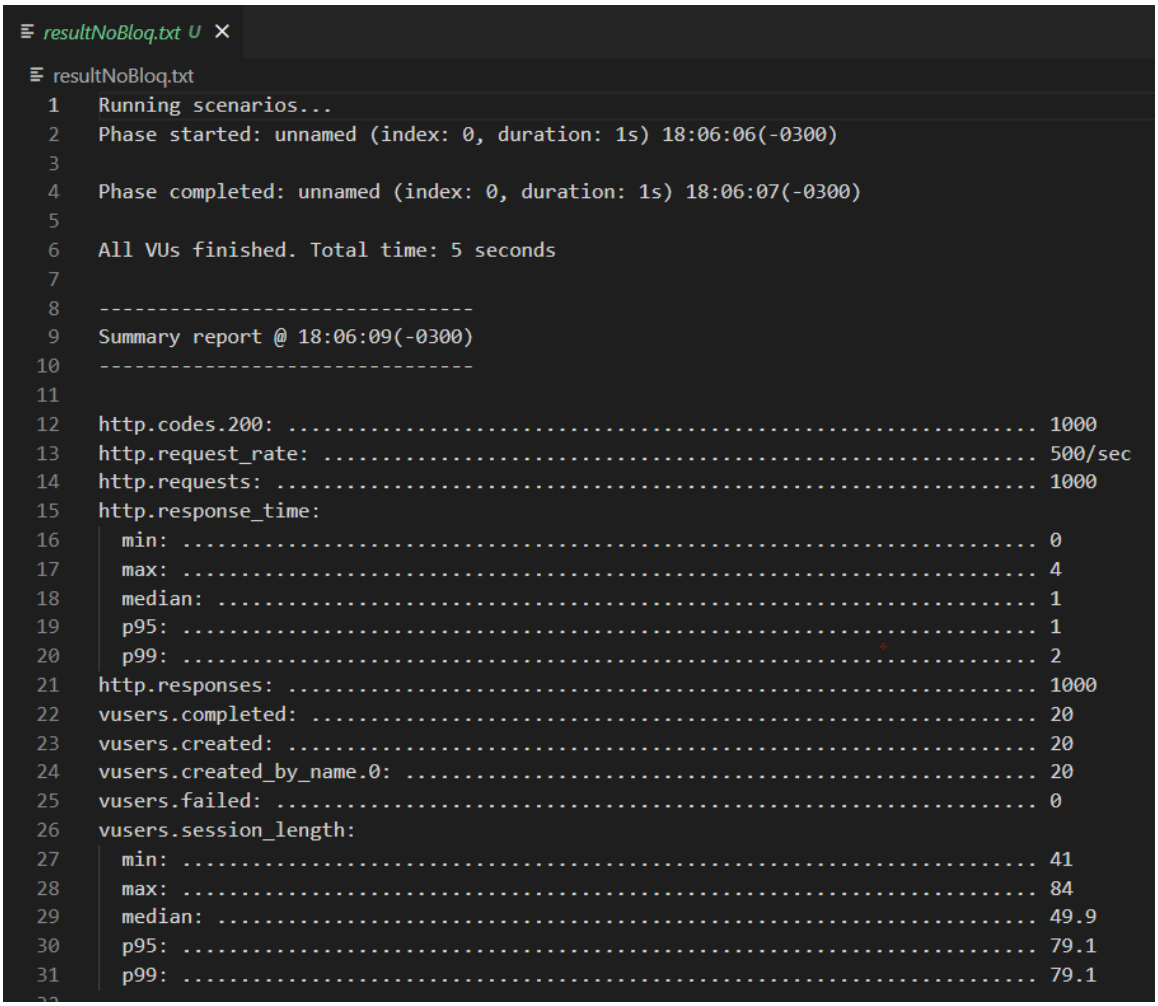
artillery quick --count=20 --num=50 http:localhost:8080/info-bloq > result_bloq.txt

node --prof-process bloq-v8.log > result_prof_bloq.txt

node --prof-process bloq-v8.log > result_prof_NoBloq.txt

Al comparar ambos procesos, el bloqueante y el no bloqueante podemos ver que el no bloqueante es mucho mas rapido que el bloqueante:

Proceso no bloqueante:



resultNoBloq.txt U

resultNoBloq.txt

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 18:06:06(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 18:06:07(-0300)

All VUs finished. Total time: 5 seconds

--------------------------------
Summary report @ 18:06:09(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 500/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 0
  max: .............................................................................. 4
  median: ........................................................................... 1
  p95: .............................................................................. 1
  p99: .............................................................................. 2
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 20
vusers.created: ..................................................................... 20
vusers.created_by_name.0: ........................................................... 20
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 41
  max: .............................................................................. 84
  median: ........................................................................... 49.9
  p95: .............................................................................. 79.1
  p99: .............................................................................. 79.1
```

```
138          app.get("/info", (req, res) => {
139              try {
140      3.7 ms      res.send(`
141                      <h1>Informacion relevante: </h1>
142                      <ul>
143      0.1 ms              <li>Argumentos de entrada: ${port}</li>
144      0.2 ms              <li>Nombre de la plataforma: ${process.platform}</li>
145                          <li>Version de Node: ${process.version}</li>
146      0.6 ms              <li>Memoria total reservada: ${util.inspect(process.memoryUsage(),{
147                              showHidden: false,
148                              depth: null,
149                              colors: true
150                          })}</li>
151      0.1 ms              <li>Path de ejecucion: ${process.execPath}</li>
152                          <li>Process ID: ${process.pid}</li>
153      0.1 ms              <li>Carpeta del proyecto: ${process.cwd()}</li>
154      0.1 ms              <li>Numero de procesadores: ${os.cpus().length}</li>
155                      </ul>`)
156              } catch (error) {
157                  logger.error('Error al buscar la informacion del sistema: ', error)
158              }
159          })
160
```

Proceso bloqueante:

resultBloq.txt U

resultBloq.txt

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 18:17:02(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 18:17:03(-0300)

All VUs finished. Total time: 5 seconds

--------------------------------
Summary report @ 18:17:05(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 268/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 2
  max: .............................................................................. 40
  median: ........................................................................... 19.9
  p95: .............................................................................. 32.1
  p99: .............................................................................. 34.8
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 20
vusers.created: ..................................................................... 20
vusers.created_by_name.0: ........................................................... 20
vusers.failed: ...................................................................... 0
vusers.session_length:
  min: .............................................................................. 522.3
  max: .............................................................................. 1182.4
  median: ........................................................................... 1085.9
  p95: .............................................................................. 1176.4
  p99: .............................................................................. 1176.4

```

```
161          app.get("/info-blog", (req, res) => {
162              try {
163                  const info = `
164                  <h1>Informacion relevante: </h1>
165                  <ul>
166      0.1 ms          <li>Argumentos de entrada: ${port}</li>
167      0.3 ms          <li>Nombre de la plataforma: ${process.platform}</li>
168      0.2 ms          <li>Version de Node: ${process.version}</li>
169      3.6 ms          <li>Memoria total reservada: ${util.inspect(process.memoryUsage(),{
170                          showHidden: false,
171                          depth: null,
172                          colors: true
173                      })}</li>
174      0.1 ms          <li>Path de ejecucion: ${process.execPath}</li>
175      0.2 ms          <li>Process ID: ${process.pid}</li>
176      0.1 ms          <li>Carpeta del proyecto: ${process.cwd()}</li>
177      1.7 ms          <li>Numero de procesadores: ${os.cpus().length}</li>
178                  </ul>`
179      1.9 ms      console.log(info)
180     13.8 ms      res.send(info)
181              } catch (error) {
182                  logger.error('Error al buscar la informacion del sistema: ', error)
183              }
184          })
185
```

Autocannon:

Para realizar test con autocannon y tener el diagrama con 0x debemos hacer una modificacion en el archivo package.json :

```
"scripts": {
  "test": "node benchmark.js",
  "start": "0x index.js"
},
```

Luego se debe prender el servidor en una terminal ejecutando "npm start", a continuacion en otra terminal ejecutar la linea de comando "npm test".

Los resultados del proceso no bloqueante son muchos mejores, observando una mayor cantidad de respuestas por segundo y una mejor capacidad de procesamiento.

Resultado de los procesos. La primer tabla corresponde al proceso No Bloqueante, y la segunda tabla al proceso Bloqueante.

Resultados:

```
> nginx@1.0.0 test
> node 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	79 ms	91 ms	130 ms	142 ms	93.74 ms	12.74 ms	158 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	700	700	1100	1179	1060	106.51	700
Bytes/Sec	663 kB	663 kB	1.04 MB	1.12 MB	1 MB	101 kB	663 kB

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

21k requests in 20.05s, 20.1 MB read
Running 20s test @ http://localhost:8080/info-bloq
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	79 ms	91 ms	128 ms	148 ms	93.9 ms	13.68 ms	185 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	697	697	1091	1182	1058.91	114.38	697
Bytes/Sec	629 kB	629 kB	986 kB	1.07 MB	957 kB	104 kB	629 kB

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

21k requests in 20.04s, 19.1 MB read
```

Una vez que damos por finalizado el proceso en consola, se crea automaticamente una carpeta la cual contiene un grafico de flama que se detalla a continuacion:

En conclusion y analizando los resultados obtenidos por las pruebas, encontramos que los procesos No Bloqueantes tienen un tiempo de respuesta favorable en la ejecucion de procesos en comparacion a los procesos Bloqueantes.

Tambien al utilizar la herramiento gzip vemos una diferencia de tamaño en la ejecucion de la aplicación.