Clase 32. Logs, profiling & debug - Parte 2(Desafio)

ANÁLISIS COMPLETO DE PERFORMANCE

Punto 1

Arranque servidor

```
node --prof dist/index.js 12345 315974313260683 1ceb7ea6bd853655fc646f96fb6941cf FORK
```

Con Console.log Activado

```
artillery quick --count 20 -n 50 http://localhost:12345/info > result_console.txt
```

```
All virtual users finished
Summary report @ 16:04:41(-0300) 2021-05-21
  Scenarios launched:  20
  Scenarios completed: 20
  Requests completed:  1000
  Mean response/sec: 130.72
  Response time (msec):
    min: 12
    max: 236
    median: 130.5
    p95: 175
    p99: 198
  Scenario counts:
    0: 20 (100%)
  Codes:
    200: 1000
```

```
node --prof-process ConConsola-v8.log > result_ConConsola.txt
```

```
[Summary]:
  ticks  total  nonlib   name
     51    1.0%   94.4%  JavaScript
      0    0.0%    0.0%  C++
     23    0.5%   42.6%  GC
   5027   98.9%          Shared libraries
      3    0.1%          Unaccounted
```

Con Console.log Desactivado

```
artillery quick --count 20 -n 50 http://localhost:12345/info > result_sconsole.txt
```

```
All virtual users finished
Summary report @ 16:21:58(-0300) 2021-05-21
  Scenarios launched:  20
  Scenarios completed: 20
  Requests completed:  1000
  Mean response/sec: 151.29
  Response time (msec):
    min: 11
    max: 206
    median: 114
    p95: 147.5
    p99: 166.5
  Scenario counts:
    0: 20 (100%)
  Codes:
    200: 1000
```

```
node --prof-process SinConsola-v8.log > result_SinConsola.txt
```

```
[Summary]:
  ticks  total  nonlib   name
     34   0.5%   91.9%  JavaScript
      0   0.0%    0.0%  C++
     26   0.4%   70.3%  GC
   6159  99.4%          Shared libraries
      3   0.0%          Unaccounted
```

AUTOCANNON

```
autocannon -d 20 -c 100 http://localhost:12345/info
```

Con Console.log Activado

```
C:\Users\feruyo\Google Drive\CoderHouse\BckEnd\DesafiosBackend\D-32>autocannon -d 20 -c 100 http://localhost:12345/info
Running 20s test @ http://localhost:12345/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	555 ms	626 ms	895 ms	944 ms	651.53 ms	81.21 ms	1190 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	93	93	164	188	152.7	26.64	93
Bytes/Sec	160 kB	160 kB	282 kB	323 kB	262 kB	45.8 kB	160 kB

```
Req/Bytes counts sampled once per second.

3k requests in 20.28s, 5.25 MB read
```

Con Console.log Desactivado

```
C:\Users\feruyo\Google Drive\CoderHouse\BckEnd\DesafiosBackend\D-32>autocannon -d 20 -c 100 http://localhost:12345/info
Running 20s test @ http://localhost:12345/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	346 ms	388 ms	566 ms	723 ms	401.77 ms	60.61 ms	757 ms

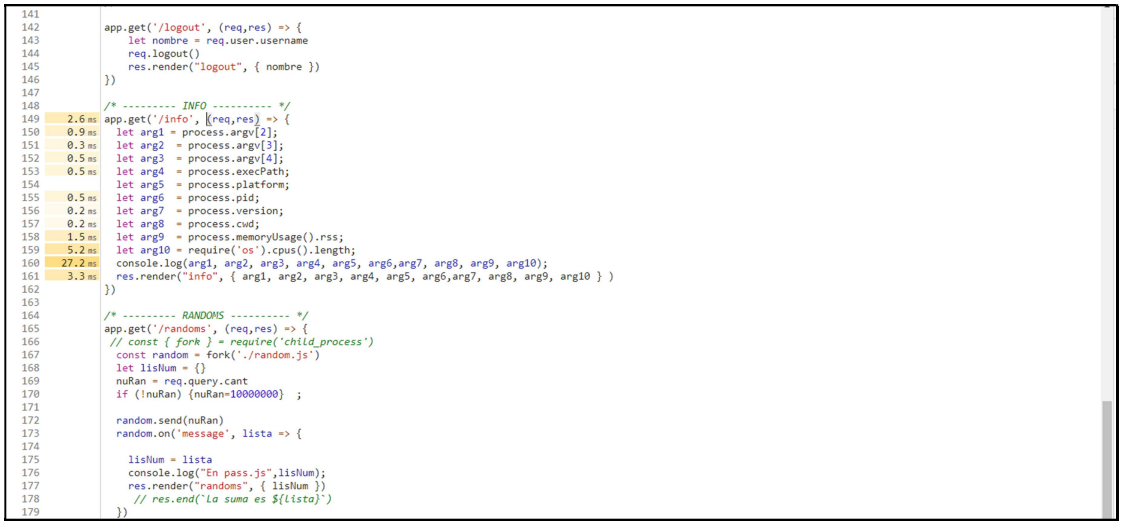
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	100	100	262	293	248.6	44.93	100
Bytes/Sec	172 kB	172 kB	450 kB	504 kB	427 kB	77.2 kB	172 kB

```
Req/Bytes counts sampled once per second.

5k requests in 20.21s, 8.54 MB read
```

Punto 2

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



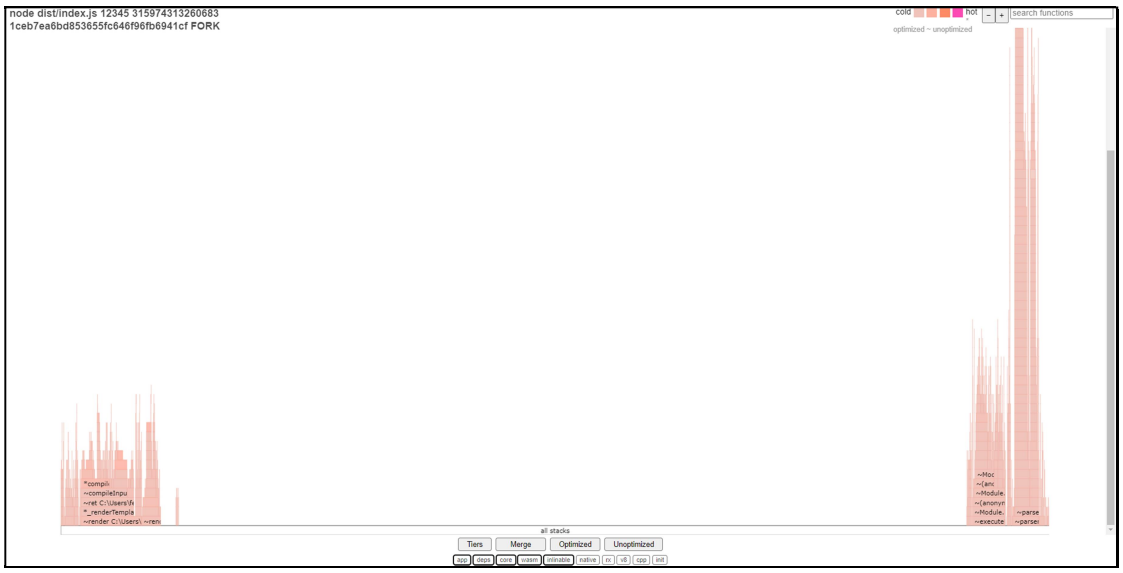
```
141
142         app.get('/logout', (req,res) => {
143             let nombre = req.user.username
144             req.logout()
145             res.render("logout", { nombre })
146         })
147
148         /* --------- INFO ---------- */
149    2.6 ms app.get('/info', (req,res) => {
150    0.9 ms   let arg1 = process.argv[2];
151    0.3 ms   let arg2  = process.argv[3];
152    0.5 ms   let arg3  = process.argv[4];
153    0.5 ms   let arg4  = process.execPath;
154             let arg5  = process.platform;
155    0.5 ms   let arg6  = process.pid;
156    0.2 ms   let arg7  = process.version;
157    0.2 ms   let arg8  = process.cwd;
158    1.5 ms   let arg9  = process.memoryUsage().rss;
159    5.2 ms   let arg10 = require('os').cpus().length;
160   27.2 ms   console.log(arg1, arg2, arg3, arg4, arg5, arg6,arg7, arg8, arg9, arg10);
161    3.3 ms   res.render("info", { arg1, arg2, arg3, arg4, arg5, arg6,arg7, arg8, arg9, arg10 } )
162         })
163
164         /* --------- RANDOMS ---------- */
165         app.get('/randoms', (req,res) => {
166          // const { fork } = require('child_process')
167           const random = fork('./random.js')
168           let lisNum = {}
169           nuRan = req.query.cant
170           if (!nuRan) {nuRan=10000000}  ;
171
172           random.send(nuRan)
173           random.on('message', lista => {
174
175             lisNum = lista
176             console.log("En pass.js",lisNum);
177             res.render("randoms", { lisNum })
178              // res.end(`La suma es ${lista}`)
179           })
```

.

Punto 3

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

```
artillery quick --count 20 -n 50 http://localhost:12345/info
```



Informe:

Utilizando el estresando del servidor en Node.js mediante las dependencias de Artillery y Autocannon podemos ver en el punto 1 que es apreciable el incremento de la media de respuestas (Mean response/sec) por segundo en el caso de no usar el console.log de la ruta “/info”.

Mean response/sec:

Con console.log	130.72
Sin console.log	151.29

Los milisegundos de latencia (Response time, median) es más alto en el caso de usar el console.log de la ruta “/info”.

Response time (msec):

Con console.log	130.5
Sin console.log	114

En el punto 2, observando el perfilamiento del servidor con el modo inspector de node.js –inspect, podemos ver que el tiempo que tarda la instrucción “console.log” de la ruta “/info” es por lejos el mayor tiempo empleado por las instrucciones del servidor(27,2 ms).

En el punto 3 se puede observar a la izquierda los procesos de render de “Handlebars”, que tienen una magnitud de tiempo empleado comparativa similar a los procesos generados por Artillery, a la izquerda.

Los procesos de Artillery son 50 conexiones concurrentes con 20 request por cada una y “Handlebars” era una sola conexión, lo que nos dice del gran tiempo que nos toma el renderizado de una página.

Aparte de eso, la curva formada por Artillery es bastante optima, ya que para serlo tiene que tener la forma de una aguja, alta pero de poca base(tiempo).