Informe

Temas: gzip, profiling, 0x, artillery, autocannon, inspect.

Introducción:
Los análisis se van a realizar en en dos rutas del proyecto, /info y /*api/*randmon.

En la ruta /*info se realizo análisis comparativo de la ruta con/*sin gzip(compress).
Ademas se realizaron dos análisis con profiling recibiendo solicitudes con artillery, agregando en un caso un console.log y en otro sin el console.log

En la ruta /*api/*randmon se realizaron dos análisis con 0x y con inspect, ambos recibiendo solicitudes con autocanon

Localización:
En en la carpeta **análisis** que esta en la raíz del proyecto, podemos encontrar 3 carpetas, el análisis de compres y una carpeta por cada ruta analizada, dentro de ellas estarán los reportes de todos los análisis

Análisis:

compress: se midió el peso de la respuesta de la ruta /info sin el middleware compres y luego con el midleware activado.

/info – sin compres: 2.5KB
/info – con compres: 1.4KB

/info: Detalle de los resultados de profiling.

-Con console.log:
Summary: 2611 99.6% Shared libraries

ticks	total	name
2158	82.3%	C:\Windows\SYSTEM32\ntdll.dll
452	17.2%	C:\Program Files\nodejs\node.exe
1	0.0%	C:\Windows\System32\KERNELBASE.dll

-Sin console.log:
Summary: 6612 99.9% Shared libraries

ticks	total	name
6281	94.9%	C:\Windows\SYSTEM32\ntdll.dll
330	5.0%	C:\Program Files\nodejs\node.exe
1	0.0%	C:\Windows\System32\KERNELBASE.dll

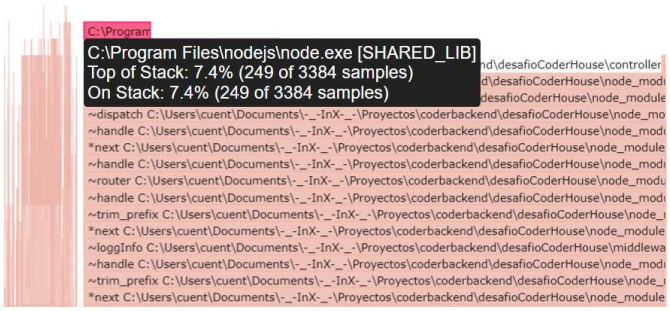
/api/randmon: Detalle de los resultados de profiling.

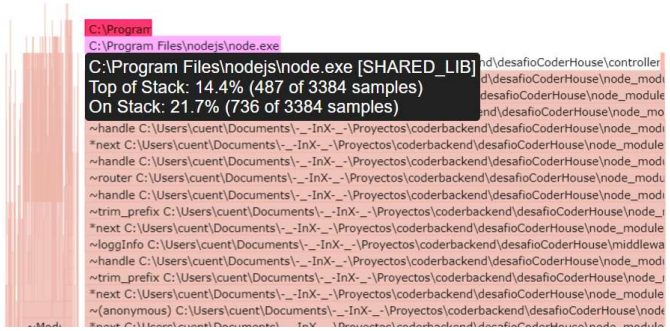
[Summary]:
ticks total name
5701 57.0% JavaScript
4307 43.0% Shared libraries

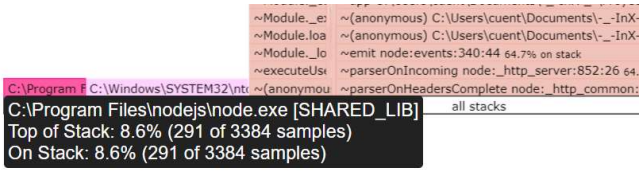
[JavaScript]:
ticks total name
5692 99.8% \controllers\random.controller.js:3:17

[Shared libraries]:
ticks total name
3642 36.4% C:\Program Files\nodejs\node.exe
660 6.6% C:\Windows\SYSTEM32\ntdll.dll

/api/randmon: Detalle de los resultados de 0x.







/api/randmon: Resultados de autocannon.

```
> Desafio Coder@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:5000/api/randoms
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1401 ms	4353 ms	8699 ms	8699 ms	4836.5 ms	2745.23 ms	8699 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	0	1	0.3	0.46	1
Bytes/Sec	0 B	0 B	0 B	12.7 kB	3.79 kB	5.8 kB	12.6 kB

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

301 requests in 20.22s, 75.9 kB read
195 errors (195 timeouts)
```

/api/randmon: Resultados de inspect.

Self Time	Total Time	File
5,04ms	2.041.155,09ms	next
3,65ms	437.340,57ms	handle
0,72ms	291.502,17ms	loggInfo
0,00ms	145.840,61ms	query
0,00ms	145.837,73ms	expressInit
0,33ms	145.816,96ms	session
0,22ms	145.814,84ms	initialize
0,00ms	145.812,89ms	strategy.pass
0,00ms	145.810,53ms	jsonParser
0,00ms	145.809,83ms	urlencodedParser
0,00ms	145.806,89ms	compression
0,12ms	145.762,08ms	(anonymous)
401,50ms	2.041.102,17ms	handle
0,78ms	1.895.380,08ms	process_params
1,82ms	1.895.379,30ms	(anonymous)
2,47ms	1.749.649,38ms	trim_prefix
4,64ms	437.345,56ms	handle
1,15ms	291.565,38ms	loggInfo
1,15ms	291.565,38ms	handle
0,45ms	291.499,83ms	router
0,00ms	156.163,00ms	(root)
0,99ms	145.914,82ms	emit
0,99ms	145.859,62ms	parserOnHeadersComplete
1,52ms	145.856,17ms	parserOnIncoming
0,66ms	145.848,83ms	(anonymous)
0,44ms	145.848,16ms	(anonymous)
0,00ms	145.847,62ms	app
1,33ms	145.847,51ms	handle
0,23ms	145.841,81ms	query
1,43ms	145.840,16ms	expressInit
3,20ms	145.836,62ms	session
0,78ms	145.815,63ms	initialize
0,11ms	145.813,92ms	authenticate
0,81ms	145.813,81ms	attempt
0,00ms	145.812,89ms	SessionStrategy.authenticate
0,00ms	145.812,89ms	strategy.pass
1,12ms	145.812,22ms	jsonParser
0,12ms	145.810,06ms	urlencodedParser

Conclusión:

/info:

En el caso de la ruta info podemos observar que hay muchos mas cantidad de golpes(ticks) en las pruebas sin console.log, aproximadamente 3 veces que con console.log.

Ademas en ambos casos vemos que los ticks, el 99,8% corresponde a librerías, a su vez casi el 100% de los ticks de la librerías corresponde a \SYSTEM32\ntdll.dll y en segundo lugar esta \ nodejs\node.exe

/api/randmon:

En el caso del **profiling**, en esta ruta el total de ticks están divididos de manera pareja entre JavaScript y Librerías.

En el caso de JS el 99,8% de los ticks son a: \controllers\random.controller.js.
Y en el caso de las librerías, las dos mas notables son \nodejs\node.exe y \SYSTEM32\ntdll.dll

El análisis de **0x**, muestra que en la pila hay una demora critica en la librería node.exe

Por ultimo en el resultado de muestra **inspect**, se puede ver en los consumos de tiempo que express testa muy presente, y también se observa que los logs(realizado con winston), están presentes, no representan un gran consume de tiempo pero tiene un consumo de tiempo a tener en cuenta