

1) Pruebas en la ruta /api/info con artillery agregando o quitando el console.log de los datos brindados. Se utilizan los siguientes comandos para hacer dichas pruebas:

```
node --prof server.js
```

```
artillery quick --count 50 -n 20 "http://localhost:8080/api/info" > infowlog.txt
```

```
artillery quick --count 50 -n 20 "http://localhost:8080/api/info" > infowolog.txt
```

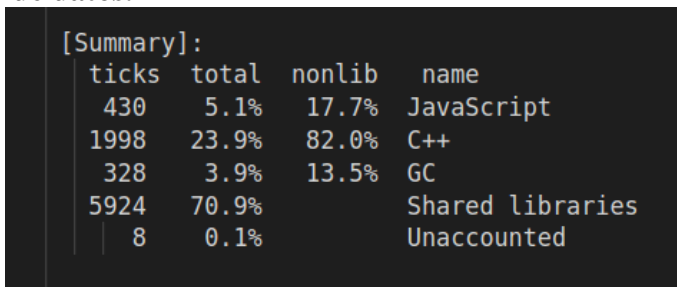
Luego los archivos log generados se traducen a archivos txt:

```
node --prof-process wlog.log > infowlog.txt
```

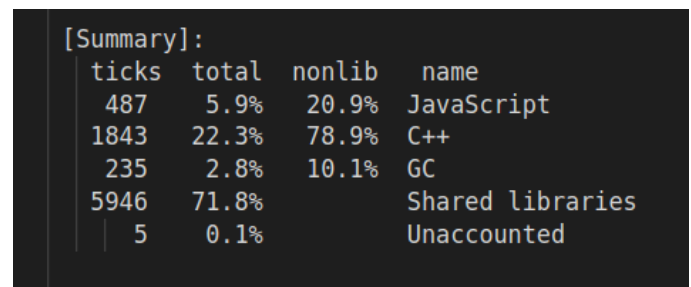
```
node --prof-process wolog.log > infowolog.txt
```

Analizando los archivos .txt generados podria decir que:

- Yendo al apartado "Summary" los ticks son menores en el archivo correspondiente a la prueba sin el console.log. La diferencia no es tan significativa pues no era una gran cantidad de datos.



[Summary]:			
ticks	total	nonlib	name
430	5.1%	17.7%	JavaScript
1998	23.9%	82.0%	C++
328	3.9%	13.5%	GC
5924	70.9%		Shared libraries
8	0.1%		Unaccounted



[Summary]:			
ticks	total	nonlib	name
487	5.9%	20.9%	JavaScript
1843	22.3%	78.9%	C++
235	2.8%	10.1%	GC
5946	71.8%		Shared libraries
5	0.1%		Unaccounted

(imagen 1 console.log, imagen 2 sin console log)

2) y 3)

Para realizar los tests de carga con Autocannon y 0x se utilizaron los siguientes comandos:

```
./node_modules/.bin/0x server.js (ya que no se instaló de forma global)
```

npm test (se modificó el package.json para que autocannon se inicie con el comando test)

Test de Autocannon por consola: Ruta info con console.log:

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	903 ms	1012 ms	4649 ms	5168 ms	1366.97 ms	886.89 ms	6177 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	3	3	79	119	71.95	36.15	3
Bytes/Sec	12.1 kB	12.1 kB	318 kB	480 kB	290 kB	146 kB	12.1 kB

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

```
2k requests in 20.05s, 5.8 MB read
```

Ruta info sin console.log

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

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	898 ms	1007 ms	3608 ms	4141 ms	1277.86 ms	663.5 ms	5166 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	4	4	95	105	76.75	32.21	4
Bytes/Sec	16.1 kB	16.1 kB	383 kB	423 kB	309 kB	130 kB	16.1 kB

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

```
2k requests in 20.05s, 6.19 MB read
```

Se observa menor latencia en la consulta sin el console.log, así como más peticiones resueltas por segundo.

En relación al modo inspector de nodejs se utiliza `node --inspect server.js`

Puede observarse el siguiente resultado de la consulta en la ruta info sin el console.log:

	})
	router.get('/info', (req, res) => {
1.6 ms	const information = { OS: process.platform,
0.2 ms	nodeversion: process.version,
1.4 ms	memoryusage: process.memoryUsage().rss,
5.9 ms	execpath: process.title, pid: process.pid,
1.2 ms	projfolder: process.cwd(), procnum: numCpus};
	// console.log(information); //comentar o descomentar para hacer los tests
7.9 ms	return res.render('pages/info.ejs', {info: process, cpus: numCpus})
	})

Y este resultado en la consulta de la ruta info con el console.log:

3	
4	router.get('/info', (req, res) => {
5 2.0 ms	const information = { OS: process.platform,
6	nodeversion: process.version,
7 1.4 ms	memoryusage: process.memoryUsage().rss,
8 6.7 ms	execpath: process.title, pid: process.pid,
9 1.0 ms	projfolder: process.cwd(), procnum: numCpus};
0	
1 4.2 ms	console.log(information); //comentar o descomentar para hacer los tests
2 16.4 ms	return res.render('pages/info.ejs', {info: process, cpus: numCpus})
3	})
4	
5	

Así,
se

puede ver que los tiempos de espera de la consulta al servidor sin el console log en la ruta /info son menores a los de la consulta con el console.log.

Por último los gráficos de flama generados con 0x pueden verse en las carpetas 0x console log y 0x sin console log respectivamente ya que no entraban en este documento. Las diferencias que se pueden observar son mínimas, siendo el proceso bloqueante el mismo en ambas pruebas (la ruta /info). Habría que pensar un proceso que sea mas bloqueante para ver diferencias notorias.

