Reporte de Rendimiento

1. Profiling con Node.js —prof / Artillery

a) Summary del Proceso Bloqueante *(Con el console.log())*

```
[Summary]:
   ticks  total  nonlib   name
     42    0.0%   97.7%  JavaScript
      0    0.0%    0.0%  C++
     14    0.0%   32.6%  GC
  133778  100.0%          Shared libraries
      1    0.0%          Unaccounted
```

```
--------------------------------
Summary report @ 22:44:50(-0300)
--------------------------------

http.codes.200: ..................................................................... 1000
http.request_rate: .................................................................. 43/sec
http.requests: ...................................................................... 1000
http.response_time:
  min: .............................................................................. 71
  max: .............................................................................. 1018
  median: ........................................................................... 528.6
  p95: .............................................................................. 658.6
  p99: .............................................................................. 713.5
http.responses: ..................................................................... 1000
vusers.completed: ................................................................... 50
```

```
vusers.created: ..................................................... 50
vusers.created_by_name.0: ........................................... 50
vusers.failed: ...................................................... 0
vusers.session_length:
  min: ............................................................. 10280.3
  max: ............................................................. 10937.4
  median: .......................................................... 10617.5
  p95: ............................................................. 10832
  p99: ............................................................. 10832
```

b) Summary del Proceso NO Bloqueante *(Sin el console.log())*

```
[Summary]:
   ticks  total  nonlib   name
     18    0.3%   90.0%  JavaScript
      0    0.0%    0.0%  C++
     13    0.2%   65.0%  GC
   6026   99.7%          Shared libraries
      2    0.0%          Unaccounted
```

```
-------------------------------
Summary report @ 22:37:36(-0300)
-------------------------------

http.codes.200: ..................................................... 1000
http.request_rate: .................................................. 107/sec
http.requests: ...................................................... 1000
http.response_time:
  min: ............................................................. 29
  max: ............................................................. 513
  median: .......................................................... 175.9
  p95: ............................................................. 340.4
  p99: ............................................................. 368.8
http.responses: ..................................................... 1000
vusers.completed: ................................................... 50
vusers.created: ..................................................... 50
vusers.created_by_name.0: ........................................... 50
vusers.failed: ...................................................... 0
vusers.session_length:
  min: ............................................................. 3780.5
  max: ............................................................. 4229.1
  median: .......................................................... 4065.2
  p95: ............................................................. 4147.4
  p99: ............................................................. 4231.1
```

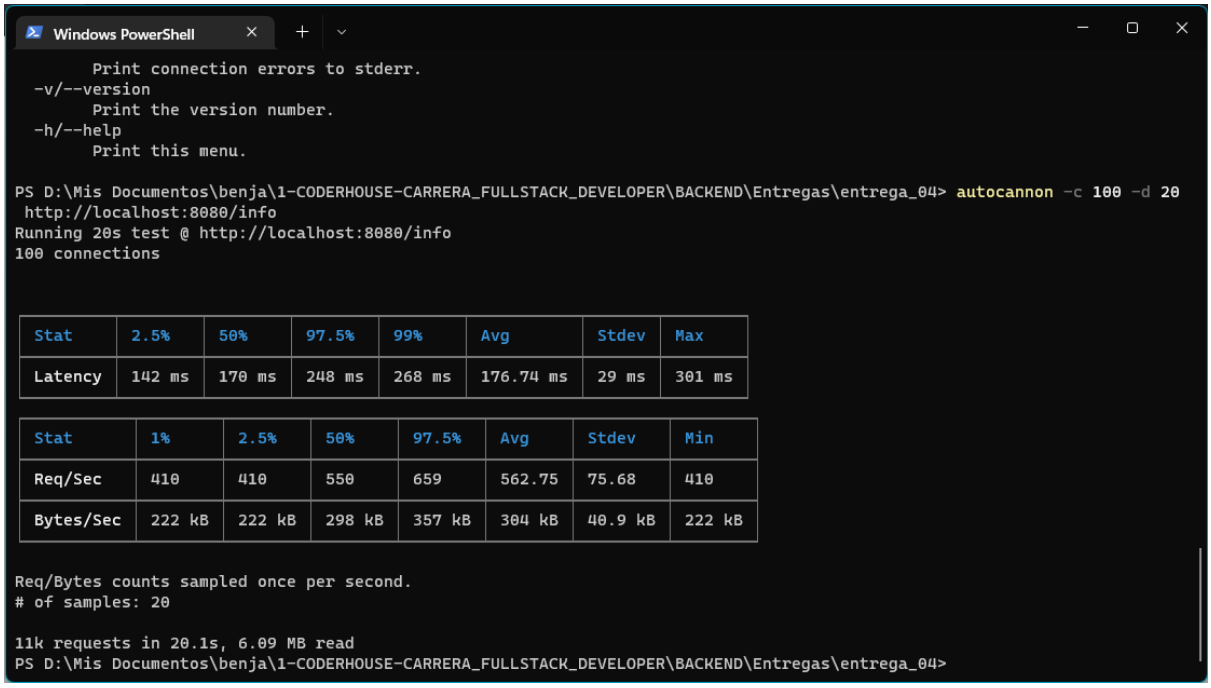
c) Conclusiones

- Como se puede observar, las repeticiones de eventos en el event loop, y por tanto mayor consumo de recursos y tiempo, es prácticamente el doble la de la ejecución del proceso síncrono bloqueante (`console.log()`) respecto de la del NO bloqueante.
- Como surge de los reportes de artillery, prácticamente se duplican la cantidad de peticiones http cuando no se incluyen procesos bloqueantes como el `console.log()`. Tanto las medias de tiempo como los percentiles 99 son claramente superiores.

2. Profiling con Node.js —prof / Autocannon

a) Summary del Proceso Bloqueante *(Con el console.log())*

```
[Summary]:
   ticks  total  nonlib   name
      76    0.6%   98.7%  JavaScript
       0    0.0%    0.0%  C++
      42    0.3%   54.5%  GC
   12145   99.4%          Shared libraries
       1    0.0%          Unaccounted
```

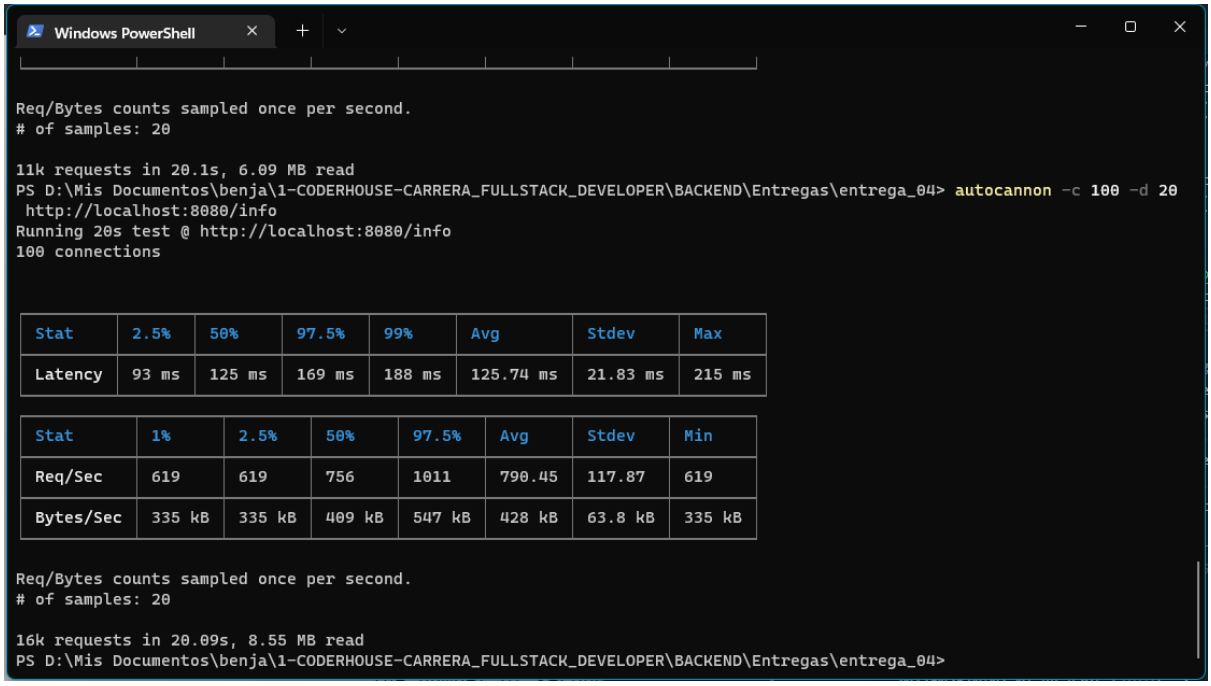


autocannon-bloq.png

Image: Autocannon results blocking process

b) Summary del Proceso NO Bloqueante *(Sin el console.log())*

```
[Summary]:
   ticks  total  nonlib   name
     99    2.4%   97.1%  JavaScript
      0    0.0%    0.0%  C++
     46    1.1%   45.1%  GC
   3960   97.5%          Shared libraries
      3    0.1%          Unaccounted
```



autocannon-non-bloq.png

Image: Autocannon results non-blocking process

c) Conclusiones

- Como se puede observar, las repeticiones de eventos en el event loop, y por tanto mayor consumo de recursos y tiempo, es prácticamente el doble la de la ejecución del proceso síncrono bloqueante (`console.log()`) respecto de la del NO bloqueante.
- Como se aprecia de los resultados de autocannon, el promedio de peticiones aumenta en los casos en donde no se encuentra presente el `console.log()` en la ruta, y el percentil 99% prácticamente duplica al del proceso bloqueante.

3. Profiling con Node.js —inspector/ Chrome DEV Tools

a) Summary del Proceso Bloqueante *(Con el console.log())*

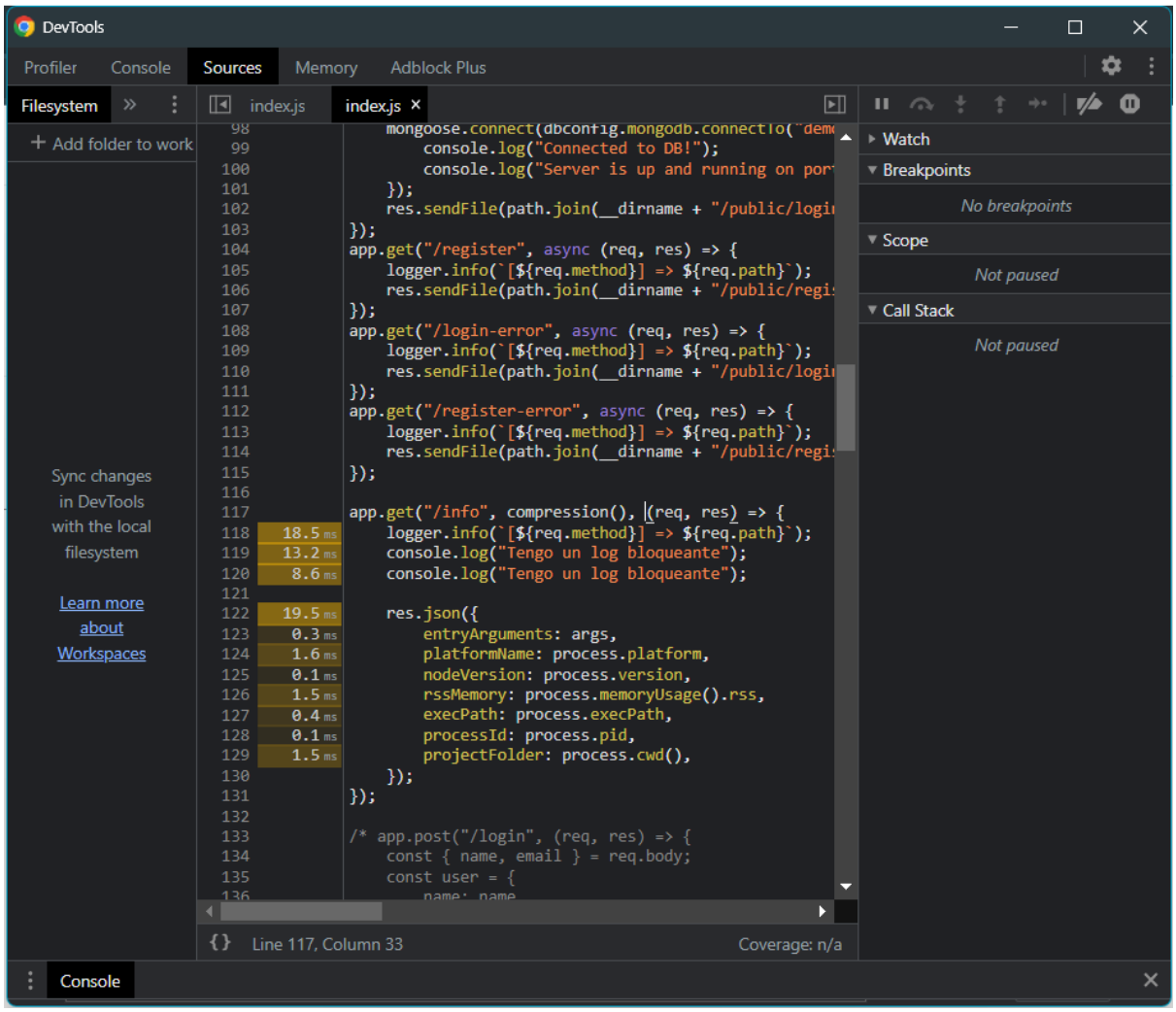

bloq.cpuprofile

Filter functions or files, or start a query()

Self Time	Total Time	File	
101.020,91ms	101.020,91ms	(idle)	
4.876,35ms	5.981,84ms	consoleCall	
1.192,20ms	1.192,20ms	writeUtf8String	
293,44ms	293,44ms	(program)	
154,71ms	154,71ms	writev	
154,55ms	154,55ms	(garbage collector)	
134,02ms	8.943,96ms	authenticate	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/passport/lib/middleware/authenticate.js:94
105,60ms	188,22ms	nextTick	node:internal/process/task_queues:104
105,41ms	105,41ms	getColorDepth	node:internal/tty:106
103,02ms	9.514,16ms	session	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:179
96,35ms	276,03ms	hash	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:596
79,66ms	8.689,84ms	compression	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/compression/index.js:59
78,52ms	78,52ms	Hash	
78,40ms	158,05ms	Hash	node:internal/crypto/hash:58
76,11ms	133,87ms	writeHead	node:_http_server:269
74,75ms	37.972,43ms	next	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/index.js:176
40,52ms	8.776,39ms	strategy.pass	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/passport/lib/middleware/authenticate.js:345
8,14ms	9.553,84ms	error	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/serve-static/index.js:115
7,57ms	9.024,67ms	session	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:179
4,90ms	8.963,16ms	initialize	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/passport/lib/middleware/initialize.js:51
4,41ms	325,27ms	expressInit	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/middleware/init.js:29
3,99ms	276,43ms	jsonParser	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/body-parser/lib/types/json.js:98
2,46ms	418,45ms	handle	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/index.js:136
1,01ms	382,18ms	query	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/middleware/query.js:39
0,86ms	251,17ms	urlencodedParser	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/body-parser/lib/types/urlencoded.js:79
0,44ms	0,44ms	emit	node:events:470
0,44ms	0,44ms	handle	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/layer.js:86
73,61ms	8.580,19ms	(anonymous)	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/index.js:117
73,16ms	1.447,74ms	send	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:107
71,02ms	55.099,53ms	handle	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/layer.js:86
63,65ms	205,42ms	deserializeObject	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/bson/lib/parser/deserializer.js:65
59,57ms	1.655,76ms	json	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:239

Quokka Port : 5500 Spell

bloq.cpuprofile



console-logs-times-inspect.png

b) Summary del Proceso NO Bloqueante *(Sin el console.log())*


non-bloq.cpuprofile
Filter functions or files, or start a query()

Self Time	Total Time	File	
160.674,16ms	160.674,16ms	(idle)	
401,07ms	401,07ms	writeUtf8String	
279,91ms	279,91ms	(program)	
141,40ms	141,40ms	(garbage collector)	
134,67ms	134,67ms	writev	
126,75ms	2.490,39ms	authenticate	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/passport/lib/middleware/authenticate.js:94
81,91ms	244,68ms	hash	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:596
81,33ms	2.987,59ms	session	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:179
78,44ms	144,28ms	nextTick	node:internal/process/task_queues:104
77,38ms	11.950,13ms	next	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/index.js:176
77,24ms	2.256,12ms	compression	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/compression/index.js:59
72,68ms	142,86ms	Hash	node:internal/crypto/hash:58
65,65ms	65,65ms	Hash	
61,50ms	107,28ms	writeHead	node:_http_server:269
53,72ms	16.223,53ms	handle	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/router/layer.js:86
48,80ms	1.190,48ms	send	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:107
46,88ms	4.689,93ms	emit	node:events:470
46,85ms	201,29ms	store.generate	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express-session/index.js:158
46,51ms	203,37ms	resOnFinish	node:_http_server:788
45,20ms	192,72ms	deserializeObject	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/bson/lib/parser/deserializer.js:65
44,07ms	44,07ms	asyncTaskScheduled	
43,87ms	1.333,90ms	json	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:239
43,20ms	2.159,41ms	(anonymous)	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/index.js:117
38,91ms	342,29ms	expressInit	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/middleware/init.js:29
38,85ms	545,33ms	processTicksAndRejections	node:internal/process/task_queues:68
36,96ms	41,66ms	parse	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/content-type/index.js:106
36,74ms	67,82ms	emitHook	node:internal/async_hooks:224
36,34ms	36,34ms	setWriteHeadHeaders	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/on-headers/index.js:105
35,47ms	35,47ms	stringify	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:1123
35,34ms	382,85ms	end	node:_http_outgoing:833
33,53ms	2.543,72ms	initialize	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/passport/lib/middleware/initialize.js:51
32,84ms	80,58ms	header	d:/Mis Documentos/benja/1-CODERHOUSE-CARRERA_FULLSTACK_DEVELOPER/BACKEND/Entregas/entrega_04/node_modules/express/lib/response.js:759

Quokka Port : 5500 Spell

non-bloq.cpuprofile

c) Conclusiones

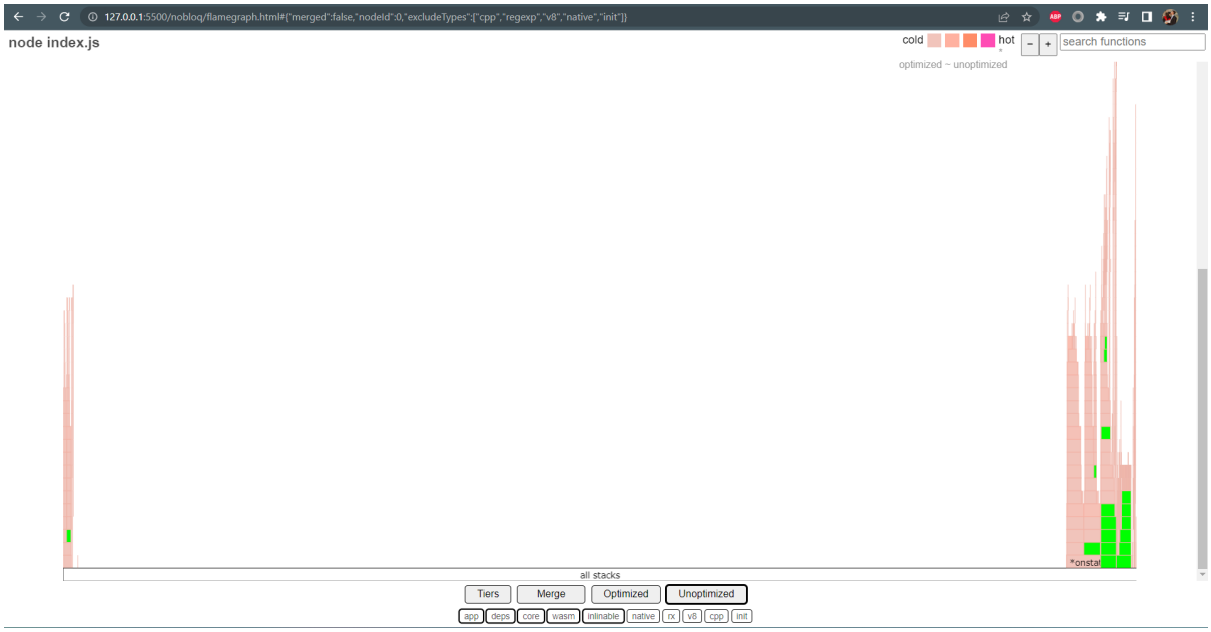
- Surge de las imágenes que se adjuntan, que claramente la función de imprimir por consola de manera síncrona, se posiciona como una de las que más consumen tiempo, ubicándose segunda en el informe del rendimiento, cuando el proceso es bloqueante.
- Tiene tiempos aproximados de entre 10s y 20s la utilización del método log del objeto console. Demorando prácticamente lo mismo que la response, lo cual resulta excesivo para un simple debugger.

4. Diagrama de Flama 0x

a) Summary del Proceso Bloqueante *(Con el console.log())*

bloq-0x.png

b) Summary del Proceso NO Bloqueante *(Sin el console.log())*



non-bloq-0x.png

c) Conclusiones

- Surge de las imágenes que se adjuntan, la diferencia no solo en la longitud de las velas, lo cual indica el mayor tiempo insumido en el test de los procesos bloqueantes, sino también remarcados en color verde, la mayor cantidad de procesos mal optimizados en el caso del supuesto en donde se incluye el `console.log()`.

5. Conclusiones finales

⚠️ Resulta evidente a todas luces, que a lo largo de las diferentes pruebas a las que hemos sometido a nuestro servidor, que el hecho de incorporar un sencillo `console.log()` , el cual resulta ser un método sincróno del objeto global console, perjudica el rendimiento del servidor de manera evidente e incluso exagerada.

💡 En función de ello, creemos conveniente evitar la impresión de mensajes en consola, con más razón si nos encontramos en producción, y emplear en cambio loggers que guarden éstos en archivos que se almacenen en el directorio del servidor, para poder analizarlos de conformidad.

✅ El descuido de olvidarnos al menos un `console.log()` puede llevar a que nuestro servidor funcione a mitad de su rendimiento, tal como surge de las estadísticas extraídas.