

- 1) El perfilamiento del servidor, realizando el test con --prof de node.js. Analizar los resultados obtenidos luego de procesarlos con --prof-process.

The screenshot shows the Visual Studio Code interface with the Explorer view on the left displaying the file structure of a project named 'BORING-SHOP'. The main editor area shows the file 'result_bloq.txt' with the following content:

```
29
30
31 All VUs finished. Total time: 8 seconds
32
33 -----
34 Summary report @ 14:10:21(-0300)
35
36 -----
37 http.codes.200: ..... 1000
38 http.request_rate: ..... 149/sec
39 http.requests: ..... 1000
40 http.response_time:
41   min: ..... 43
42   max: ..... 472
43   median: ..... 284.3
44   p95: ..... 361.5
45   p99: ..... 391.6
46 http.responses: ..... 1000
47 vusers.completed: ..... 50
48 vusers.created: ..... 50
49 vusers.created_by_name.0: ..... 50
50 vusers.failed: ..... 0
51 vusers.session_length:
52   min: ..... 5306.5
53   max: ..... 5863.6
54   median: ..... 5711.5
55   p95: ..... 5826.9
56   p99: ..... 5826.9
57
```

The screenshot shows the Visual Studio Code interface with the Explorer view on the left displaying the file structure of a project named 'BORING-SHOP'. The main editor area shows the file 'result_nobloq.txt' with the following content:

```
30
31 All VUs finished. Total time: 4 seconds
32
33 -----
34 Summary report @ 14:16:15(-0300)
35
36 -----
37 http.codes.200: ..... 1000
38 http.request_rate: ..... 287/sec
39 http.requests: ..... 1000
40 http.response_time:
41   min: ..... 16
42   max: ..... 201
43   median: ..... 82.3
44   p95: ..... 141.2
45   p99: ..... 186.8
46 http.responses: ..... 1000
47 vusers.completed: ..... 50
48 vusers.created: ..... 50
49 vusers.created_by_name.0: ..... 50
50 vusers.failed: ..... 0
51 vusers.session_length:
52   min: ..... 1138.7
53   max: ..... 2131.4
54   median: ..... 1901.1
55   p95: ..... 2101.1
56   p99: ..... 2143.5
57
```

result_prof-bloq.txt - boring-shop - Visual Studio Code

desafio_14 > result_prof-bloq.txt

```

[Summary]:
ticks total nonlib name
31 0.7% 100.0% JavaScript
0 0.0% 0.0% C++
16 0.4% 51.6% GC
4210 99.3% Shared libraries

[C++ entry points]:
ticks cpp total name

[Bottom up (heavy) profile]:
Note: percentage shows a share of a particular caller in the total
amount of its parent calls.
Callers occupying less than 1.0% are not shown.

ticks parent name
3602 84.9% C:\WINDOWS\SYSTEM32\ntdll.dll
108 3.0% LazyCompile: *write node:internal/streams/writable:368:23
87 80.6% LazyCompile: *write node:internal/streams/writable:286:16
87 100.0% Function: ^Writable.write node:internal/streams/writable:336:36
46 52.9% Function: ^value node:internal/console/constructor:272:20
46 100.0% Function: ^log node:internal/console/constructor:376:6
41 47.1% Function: ^log C:\Users\JM\Downloads\CODERHOUSE\Backend-JoseMoreno\boring-shop\desafio_14\node_mod
41 100.0% Function: ^write C:\Users\JM\Downloads\CODERHOUSE\Backend-JoseMoreno\boring-shop\desafio_14\nod
21 19.4% Function: ^write node:internal/streams/writable:286:16
21 100.0% Function: ^Writable.write node:internal/streams/writable:336:36
15 71.4% Function: ^value node:internal/console/constructor:272:20
15 100.0% Function: ^log node:internal/console/constructor:376:6
6 28.6% Function: ^log C:\Users\JM\Downloads\CODERHOUSE\Backend-JoseMoreno\boring-shop\desafio_14\node_mod
6 100.0% Function: ^write C:\Users\JM\Downloads\CODERHOUSE\Backend-JoseMoreno\boring-shop\desafio_14\nod

```

result_prof-nobloq.txt - boring-shop - Visual Studio Code

desafio_14 > result_prof-nobloq.txt

```

[Summary]:
ticks total nonlib name
27 0.8% 100.0% JavaScript
0 0.0% 0.0% C++
22 0.7% 81.5% GC
3188 99.2% Shared libraries

[C++ entry points]:
ticks cpp total name

[Bottom up (heavy) profile]:
Note: percentage shows a share of a particular caller in the total
amount of its parent calls.
Callers occupying less than 1.0% are not shown.

ticks parent name
2605 81.0% C:\WINDOWS\SYSTEM32\ntdll.dll
34 1.3% LazyCompile: *realpathSync node:fs:2474:22
24 70.6% Function: ^toRealPath node:internal/modules/cjs/loader:440:20
23 95.8% Function: ^tryFile node:internal/modules/cjs/loader:431:17
11 47.8% LazyCompile: *Module._findPath node:internal/modules/cjs/loader:541:28
11 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:904:35
9 39.1% Function: ^tryExtensions node:internal/modules/cjs/loader:447:23
6 66.7% Function: ^Module._findPath node:internal/modules/cjs/loader:541:28
3 33.3% Function: ^tryPackage node:internal/modules/cjs/loader:385:20
3 13.0% Function: ^tryPackage node:internal/modules/cjs/loader:385:20
2 66.7% LazyCompile: *Module._findPath node:internal/modules/cjs/loader:541:28
1 33.3% Function: ^Module._findPath node:internal/modules/cjs/loader:541:28
1 4.2% LazyCompile: *Module._findPath node:internal/modules/cjs/loader:541:28
1 100.0% Function: ^Module._resolveFilename node:internal/modules/cjs/loader:904:35
1 100.0% Function: ^Module._load node:internal/modules/cjs/loader:807:24
10 70.4% LazyCompile: ^tryFile node:internal/modules/cjs/loader:431:17

```

- 2) El 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.

Visual Studio Code interface showing the execution of benchmarks in the 'boring-shop' project. The Explorer pane on the left shows the project structure, including files like 'desafio_13', 'desafio_14', 'logs', 'node_modules', 'public', 'views', '.env', 'aleatorio.js', 'apiRandoms.js', 'benchmark.js', 'bloq-v8.log', 'comandos.txt', 'nginx.conf', 'nobloq-v8.log', 'notas.txt', 'package-lock.json', 'package.json', 'result_bloq.txt', 'result_nobloq.txt', 'result_prof-bloq.txt', 'result_prof-nobloq.txt', 'server.js', 'entrega_1', 'entrega_2', 'node_modules', '.gitignore', 'OUTLINE', and 'TIMELINE'.

The Terminal pane shows the execution of the 'benchmark.js' script. The output indicates that the benchmarks are running in parallel, with a 20s test at http://localhost:8080/info-nobloq and 100 connections. The results are displayed in a table format.

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	506 ms	1434 ms	1574 ms	1579 ms	1387.43 ms	230.58 ms	1608 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	32	32	54	100	70	25.22	32
Bytes/Sec	23.4 kB	23.4 kB	39.4 kB	73 kB	51.1 kB	18.4 kB	23.4 kB

Req/Bytes counts sampled once per second.
of samples: 20
2k requests in 20.22s, 1.02 MB read
Running 20s test @ http://localhost:8080/info-bloq
100 connections

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1101 ms	1430 ms	1583 ms	1621 ms	1432.28 ms	188.28 ms	1637 ms

Visual Studio Code interface showing the execution of benchmarks in the 'boring-shop' project. The Explorer pane on the left shows the project structure, including files like 'desafio_13', 'desafio_14', 'logs', 'node_modules', 'public', 'views', '.env', 'aleatorio.js', 'apiRandoms.js', 'benchmark.js', 'bloq-v8.log', 'comandos.txt', 'nginx.conf', 'nobloq-v8.log', 'notas.txt', 'package-lock.json', 'package.json', 'result_bloq.txt', 'result_nobloq.txt', 'result_prof-bloq.txt', 'result_prof-nobloq.txt', 'server.js', 'entrega_1', 'entrega_2', 'node_modules', '.gitignore', 'OUTLINE', and 'TIMELINE'.

The Terminal pane shows the execution of the 'benchmark.js' script. The output indicates that the benchmarks are running in parallel, with a 20s test at http://localhost:8080/info-bloq and 100 connections. The results are displayed in a table format.

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1101 ms	1430 ms	1583 ms	1621 ms	1432.28 ms	188.28 ms	1637 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	31	31	65	99	67.5	17.3	31
Bytes/Sec	22.6 kB	22.6 kB	47.5 kB	72.3 kB	49.3 kB	12.6 kB	22.6 kB

Req/Bytes counts sampled once per second.
of samples: 20
1k requests in 20.33s, 986 kB read

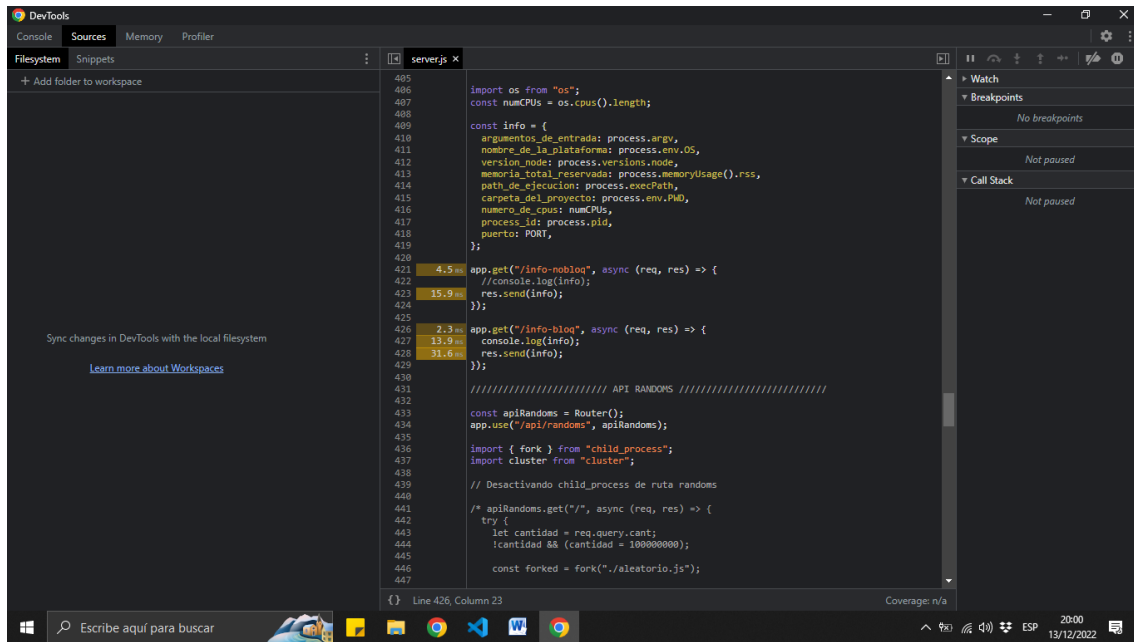
Visual Studio Code interface showing the execution of benchmarks in the 'boring-shop' project. The Explorer pane on the left shows the project structure, including files like 'desafio_13', 'desafio_14', 'logs', 'node_modules', 'public', 'views', '.env', 'aleatorio.js', 'apiRandoms.js', 'benchmark.js', 'bloq-v8.log', 'comandos.txt', 'nginx.conf', 'nobloq-v8.log', 'notas.txt', 'package-lock.json', 'package.json', 'result_bloq.txt', 'result_nobloq.txt', 'result_prof-bloq.txt', 'result_prof-nobloq.txt', 'server.js', 'entrega_1', 'entrega_2', 'node_modules', '.gitignore', 'OUTLINE', and 'TIMELINE'.

The Terminal pane shows the execution of the 'benchmark.js' script. The output indicates that the benchmarks are running in parallel, with a 20s test at http://localhost:8080/info-bloq and 100 connections. The results are displayed in a table format.

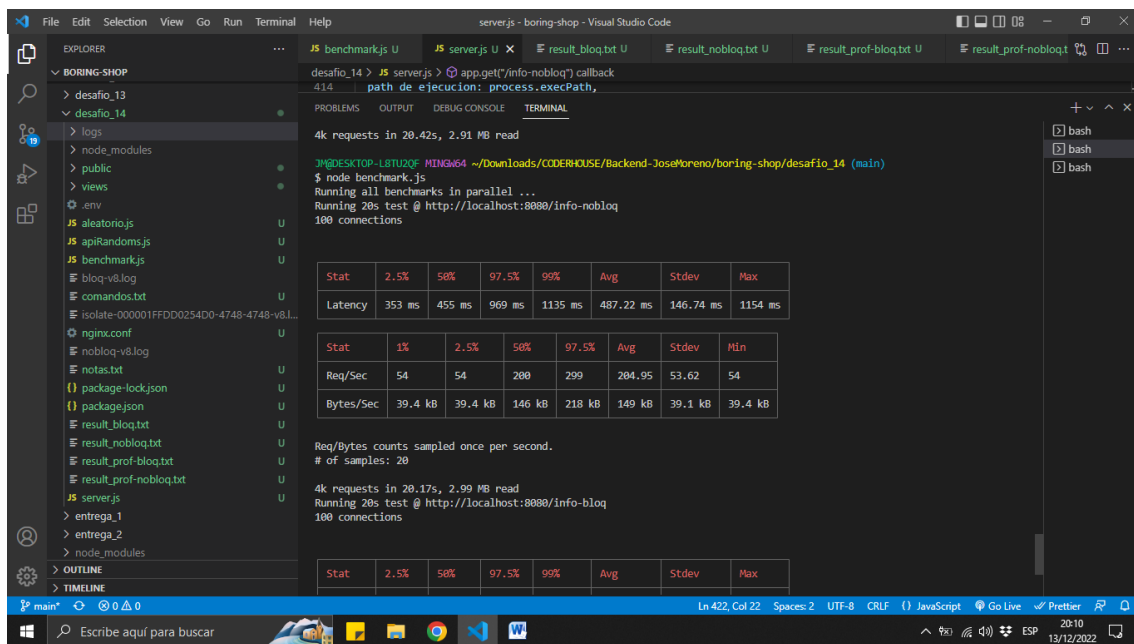
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1101 ms	1430 ms	1583 ms	1621 ms	1432.28 ms	188.28 ms	1637 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	31	31	65	99	67.5	17.3	31
Bytes/Sec	22.6 kB	22.6 kB	47.5 kB	72.3 kB	49.3 kB	12.6 kB	22.6 kB

Req/Bytes counts sampled once per second.
of samples: 20
1k requests in 20.33s, 986 kB read



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



Visual Studio Code interface showing the execution of a Node.js benchmark. The Explorer pane on the left shows the project structure for 'BORING-SHOP', including files like 'logs', 'node_modules', 'public', 'views', 'env', 'aleatorio.js', 'apiRandoms.js', 'benchmark.js', 'bloq-v8.log', 'comandos.txt', 'isolate-00001FFDD0254D0-4748-4748-v8L...', 'nginx.conf', 'nobloq-v8.log', 'notas.txt', 'package-lock.json', 'package.json', 'result_bloq.txt', 'result_nobloq.txt', 'result_prof-bloq.txt', 'result_prof-nobloq.txt', 'server.js', 'entrega_1', 'entrega_2', and 'node_modules'. The main editor shows the 'server.js' file with a benchmark script. The Output pane displays the results of the benchmark, including a table of Req/Sec and Bytes/Sec counts, and a table of latency statistics.

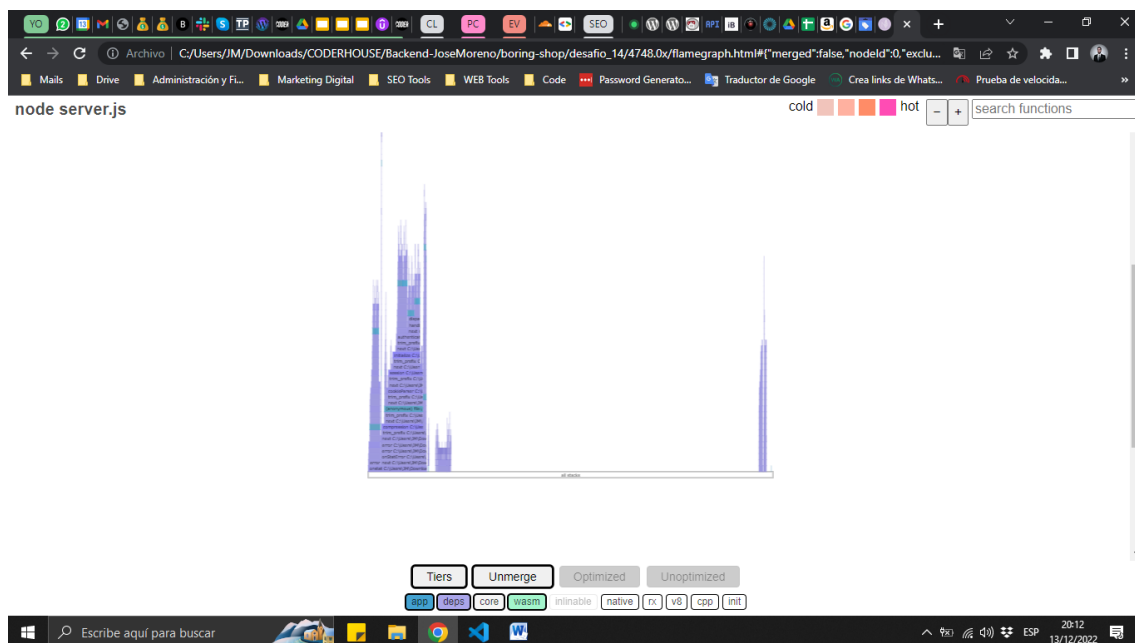
Req/Sec	54	54	200	299	204.95	53.62	54
Bytes/Sec	39.4 kB	39.4 kB	146 kB	218 kB	149 kB	39.1 kB	39.4 kB

Req/Bytes counts sampled once per second.
of samples: 20
4k requests in 20.17s, 2.99 MB read
Running 20s test @ http://localhost:8080/info-bloq
100 connections

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	370 ms	462 ms	932 ms	1160 ms	493.47 ms	145.09 ms	1188 ms

Req/Bytes counts sampled once per second.
of samples: 20
4k requests in 20.42s, 2.92 MB read

Terminal output: `node server.js`



- 4) Conclusión: En todas las pruebas realizadas se confirma que los procesos bloqueantes son menos eficientes y veloces que los procesos no bloqueantes.