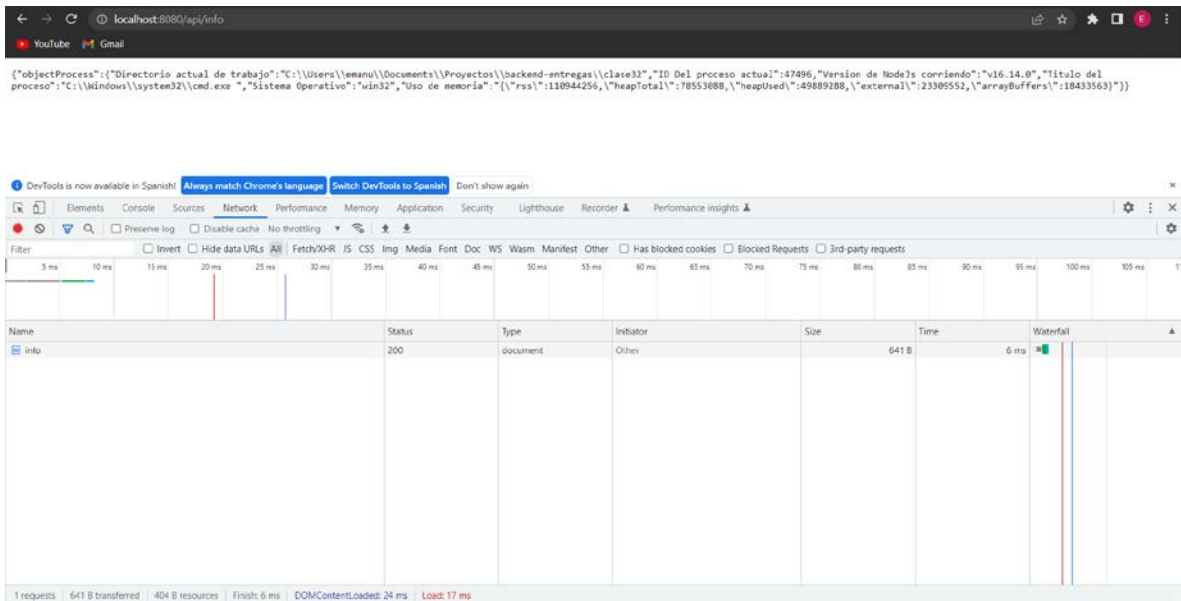
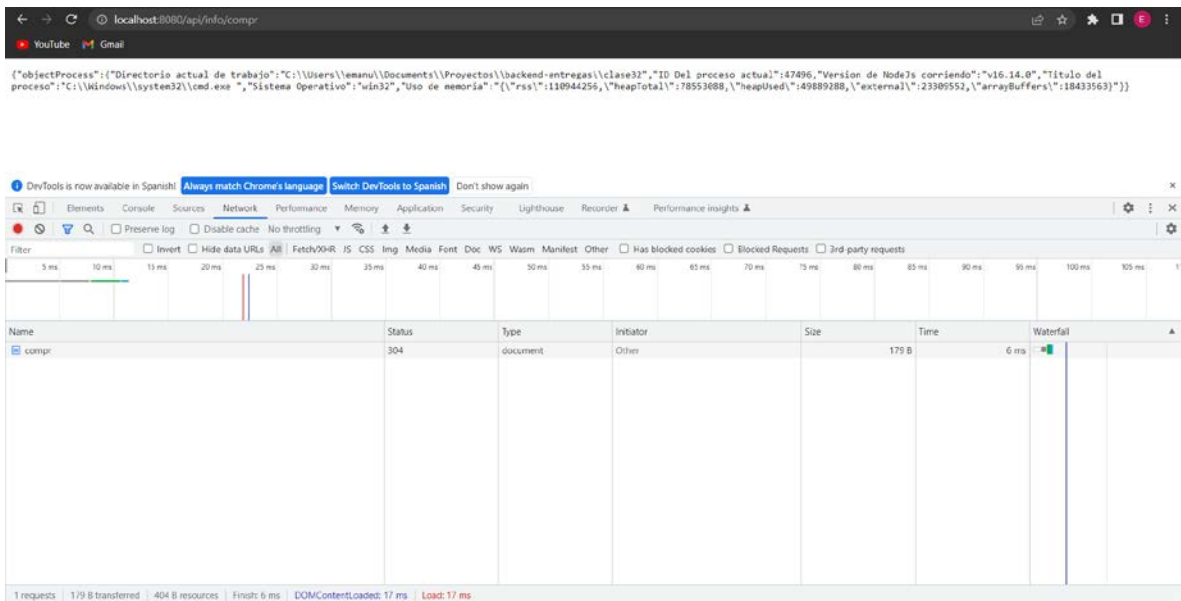


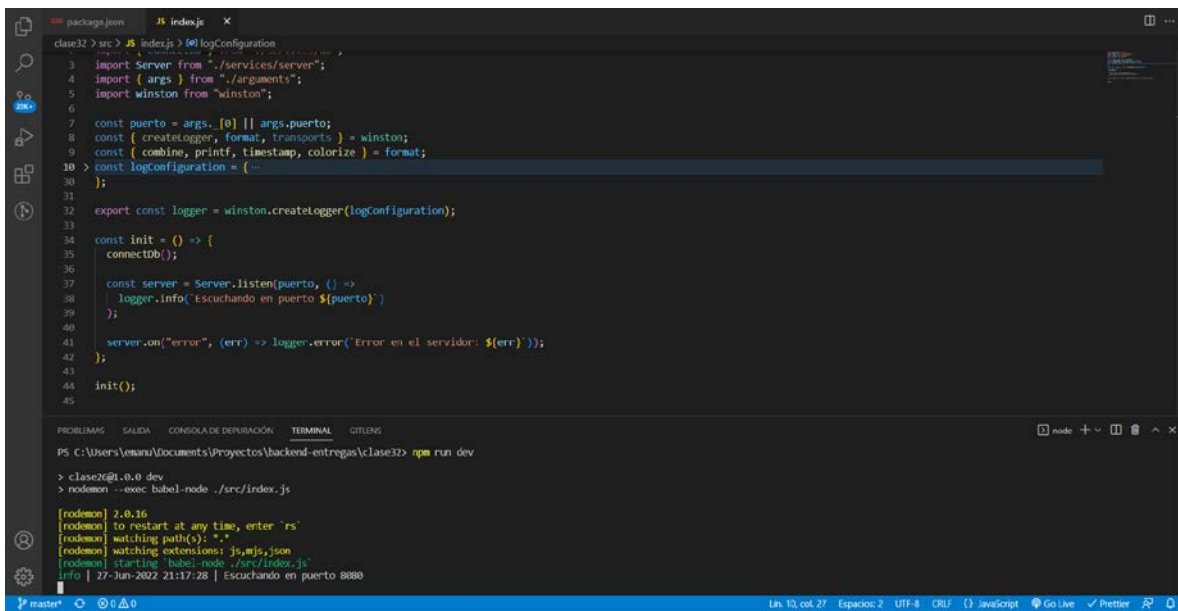
1) “/info”



2) “/info” (gzip)

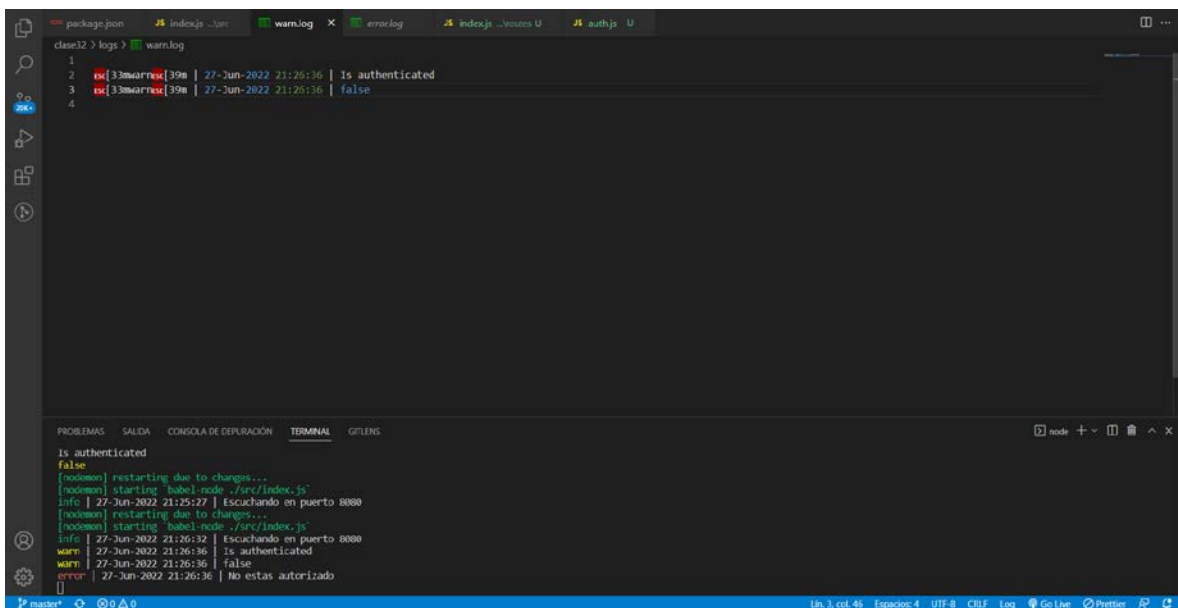


3) Winston



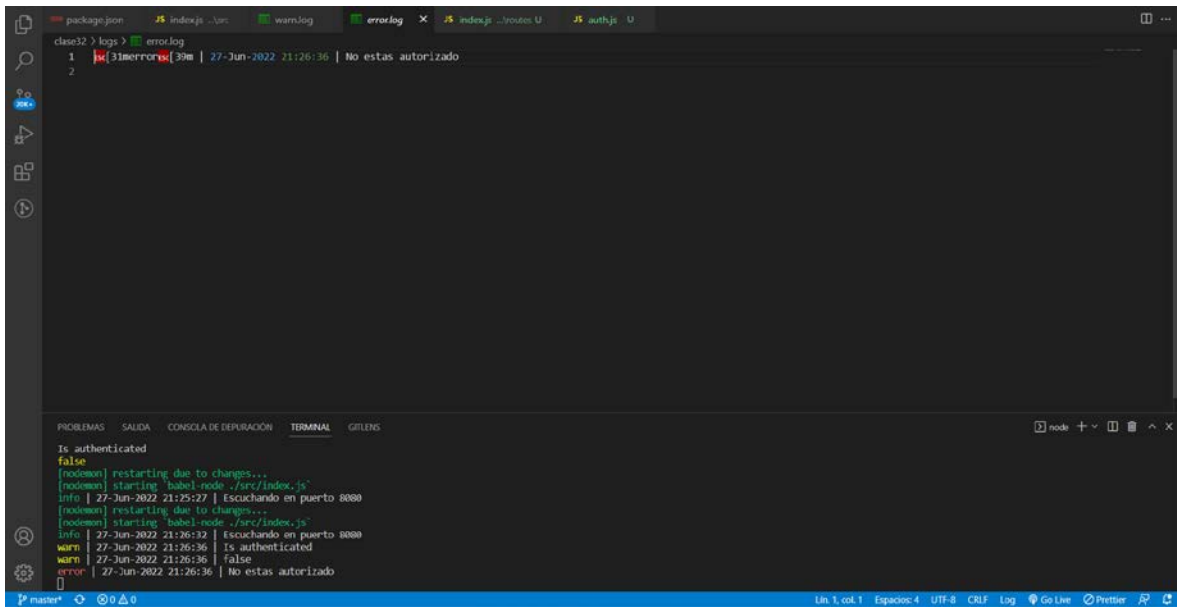
```
package.json  JS index.js  X
class2 > src > JS index.js > @logConfiguration
1 // ...
2
3 import Server from './services/server';
4 import { args } from './arguments';
5 import winston from 'winston';
6
7 const puerto = args[0] || args.puerto;
8 const { createLogger, format, transports } = winston;
9 const { combine, printf, timestamp, colorize } = format;
10 const logConfiguration = {
11   // ...
12 };
13
14 export const logger = winston.createLogger(logConfiguration);
15
16 const init = () => {
17   connectDb();
18
19   const server = Server.listen(puerto, () => {
20     logger.info('Escuchando en puerto ${puerto}');
21   });
22
23   server.on('error', (err) => logger.error('Error en el servidor: ${err}'));
24 };
25
26 init();
27
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
PS C:\Users\Yohana\Documents\Proyectos\backend-entregas\class2> npm run dev
> class2@1.0.0 dev
> nodemon --exec babel-node ./src/index.js
[nodemon] 2.0.16
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'babel-node ./src/index.js'
info | 27-Jun-2022 21:17:28 | Escuchando en puerto 8080
```

4) Winston (warn)



```
package.json  JS index.js ...  warn.log  X  error.log  JS index.js ...  JS auth.js  U
class2 > logs > warn.log
1
2 [warn] 2022-06-27 21:25:36 | Is authenticated
3 [warn] 2022-06-27 21:25:36 | false
4
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  GITLENS
Is authenticated
false
[nodemon] restarting due to changes...
[nodemon] starting 'babel-node ./src/index.js'
info | 27-Jun-2022 21:25:27 | Escuchando en puerto 8080
[nodemon] restarting due to changes...
[nodemon] starting 'babel-node ./src/index.js'
info | 27-Jun-2022 21:26:32 | Escuchando en puerto 8080
warn | 27-Jun-2022 21:26:36 | Is authenticated
warn | 27-Jun-2022 21:26:36 | false
error | 27-Jun-2022 21:26:36 | No estas autorizado
[]
```

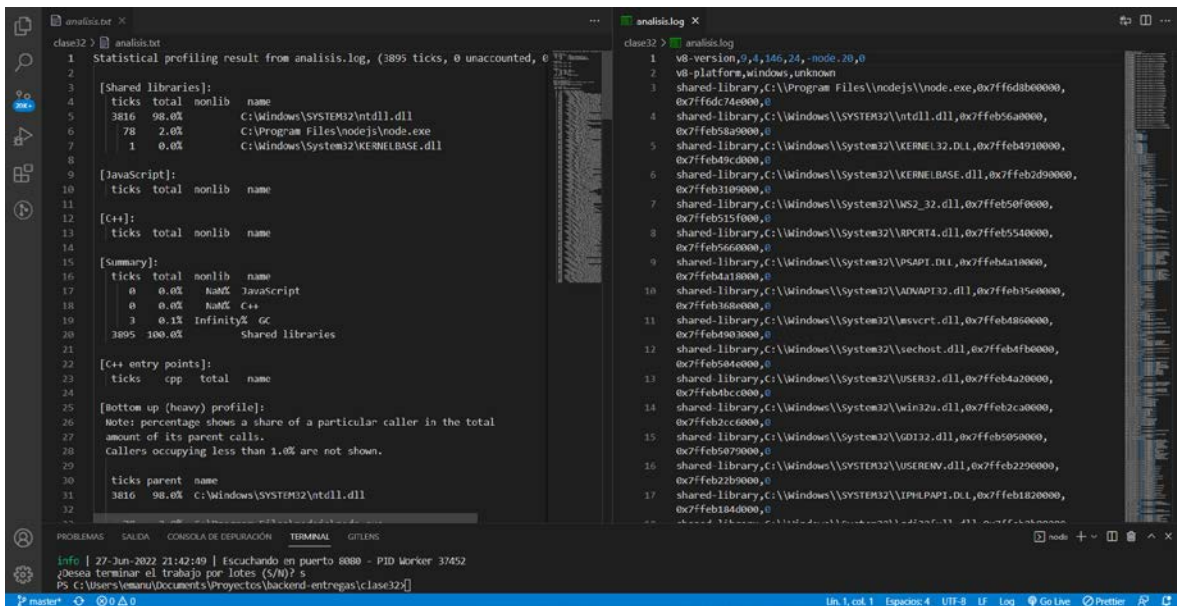
5) Winston (error)



The screenshot shows the VS Code interface with the 'TERMINAL' tab active. The terminal displays the output of a Node.js application using Winston for logging. The logs show a sequence of events: a warning about 31 errors, a message 'No estas autorizado', and a series of log messages from a 'nodemon' process. The logs indicate that the application is starting, listening on port 8080, and encountering an authentication error.

```
class2 > logs > error.log
1 [warn] 31errors:39m | 27-Jun-2022 21:26:36 | No estas autorizado
2
[nodemon] restarting due to changes...
[nodemon] starting 'babel-node ./src/index.js'
info | 27-Jun-2022 21:25:27 | Escuchando en puerto 8080
[nodemon] restarting due to changes...
[nodemon] starting 'babel-node ./src/index.js'
info | 27-Jun-2022 21:26:32 | Escuchando en puerto 8080
warn | 27-Jun-2022 21:26:36 | Is authenticated
warn | 27-Jun-2022 21:26:36 | false
error | 27-Jun-2022 21:26:36 | No estas autorizado
```

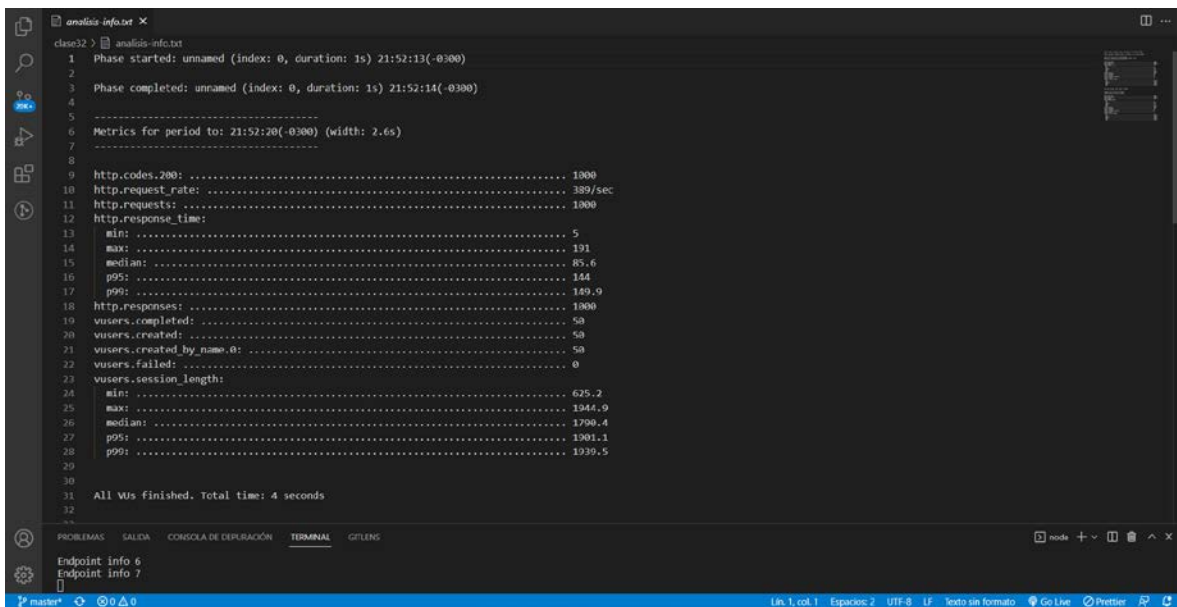
6) -prof-process



The screenshot shows the VS Code interface with the 'TERMINAL' tab active. The terminal displays the output of a Node.js application using the '-prof-process' flag. The output shows a detailed statistical profiling result from 'analysis.log', including shared libraries, JavaScript ticks, C++ ticks, and a bottom-up (heavy) profile. The logs also show the application listening on port 8080 and a message from the user 'Desee terminar el trabajo por lotes (S/N)? s'.

```
class2 > analysis.log
1 Statistical profiling result from analysis.log, (3895 ticks, 0 unaccounted, 0
2
3 [Shared libraries]:
4 ticks total nonlib name
5 3816 98.0% C:\Windows\SYSTEM32\ntdll.dll
6 78 2.0% C:\Program Files\nodejs\node.exe
7 1 0.0% C:\Windows\System32\KERNELBASE.dll
8
9 [JavaScript]:
10 ticks total nonlib name
11
12 [C++]:
13 ticks total nonlib name
14
15 [Summary]:
16 ticks total nonlib name
17 0 0.0% NaN% JavaScript
18 0 0.0% NaN% C++
19 3 0.1% Infinity% GC
20 3895 100.0% Shared libraries
21
22 [C++ entry points]:
23 ticks cpp total name
24
25 [Bottom up (heavy) profile]:
26 Note: percentage shows a share of a particular caller in the total
27 amount of its parent calls.
28 callers occupying less than 1.0% are not shown.
29
30 ticks parent name
31 3816 98.0% C:\Windows\SYSTEM32\ntdll.dll
32
33
34 info | 27-Jun-2022 21:42:49 | Escuchando en puerto 8080 - PID Worker 37452
35 Desee terminar el trabajo por lotes (S/N)? s
36 C:\Users\Venana\Documents\Proyectos\backend-entregas\class2>
```

7) Artillery

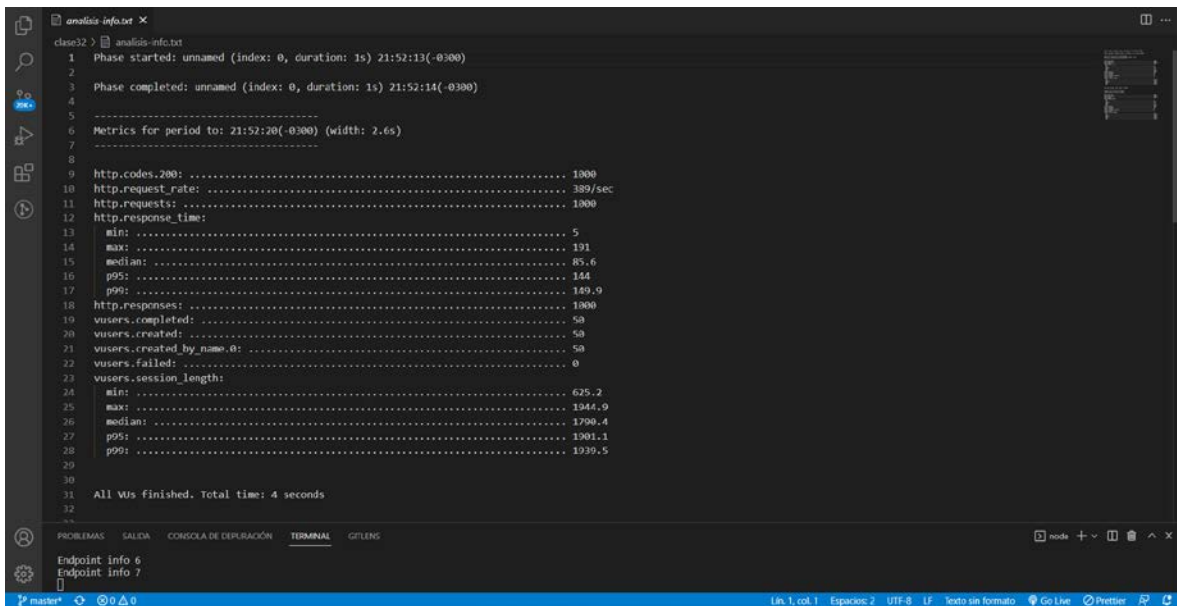


The screenshot shows a VS Code editor with a terminal window displaying the output of an Artillery test. The test file is named `analisis-info.txt`. The output shows the test started at 21:52:13 and completed at 21:52:14. The metrics for the period 21:52:20(-0300) (width: 2.6s) are as follows:

Metric	Value
http.codes.200	1000
http.request_rate	389/sec
http.requests	1000
http.response_time	
min	5
max	191
median	85.6
p95	144
p99	149.9
http.responses	1000
users.completed	50
users.created	50
users.created_by_name.0	50
users.failed	0
users.session_length	
min	625.2
max	1944.9
median	1790.4
p95	1901.1
p99	1939.5

The test finished with the message: "All WUs finished. Total time: 4 seconds".

8) Autocannon



The screenshot shows a VS Code editor with a terminal window displaying the output of an Autocannon test. The test file is named `analisis-info.txt`. The output shows the test started at 21:52:13 and completed at 21:52:14. The metrics for the period 21:52:20(-0300) (width: 2.6s) are as follows:

Metric	Value
http.codes.200	1000
http.request_rate	389/sec
http.requests	1000
http.response_time	
min	5
max	191
median	85.6
p95	144
p99	149.9
http.responses	1000
users.completed	50
users.created	50
users.created_by_name.0	50
users.failed	0
users.session_length	
min	625.2
max	1944.9
median	1790.4
p95	1901.1
p99	1939.5

The test finished with the message: "All WUs finished. Total time: 4 seconds".

9) Autocannon

```
C:\Windows\system32\cmd.exe
C:\Users\emana\Documents\Proyectos\backend-entregas\clase32>autocannon -c 100 -d 20 "http://localhost:8080/api/info"
Running 20s test @ http://c-100,http://localhost:8080/api/info
10 connections

Stat: 2.5% 50% 97.5% 99% Avg Stddev Max
Latency 1 ms 9 ms 28 ms 48 ms 11.79 ms 80.96 ms 4668 ms

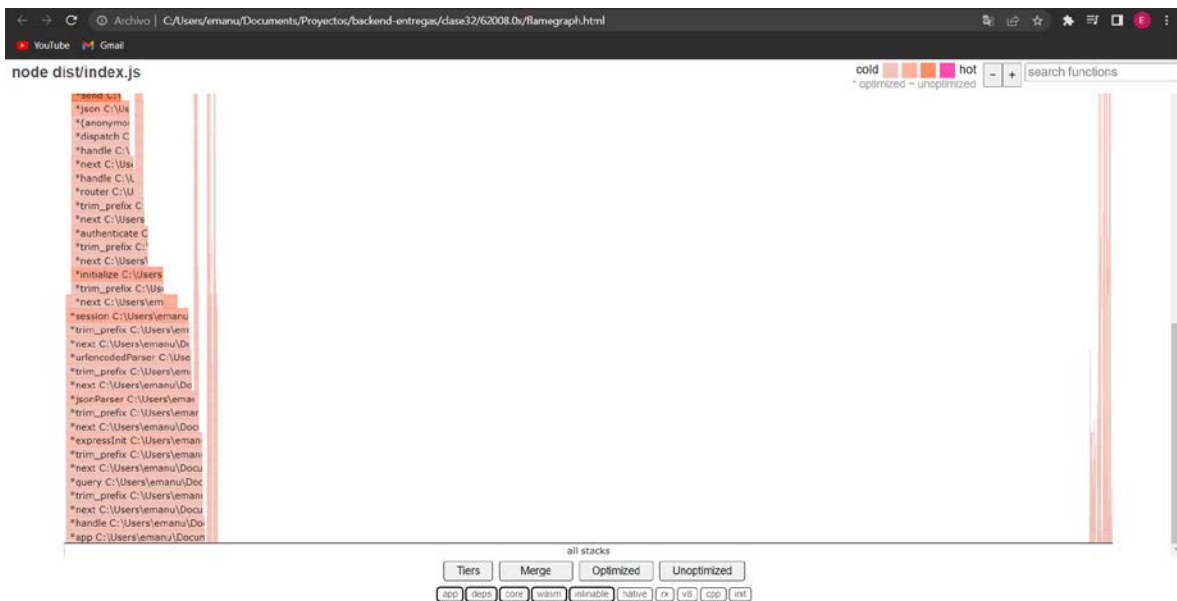
Stat: 1% 2.5% 50% 97.5% Avg Stddev Min
Req/Sec 306 306 379 600 407.4 87.55 306
Bytes/Sec 196 kB 196 kB 243 kB 384 kB 261 kB 56 kB 196 kB

Req/Bytes counts sampled once per second.
# of samples: 20
8k requests in 20.08s, 5.21 MB read
10 errors (0 timeouts)
C:\Users\emana\Documents\Proyectos\backend-entregas\clase32>
```

10) Inspect

```
DevTools is now available in Spanish! Always match Chrome's language Switch DevTools to Spanish Don't show again
Connection Console Sources Memory Profiler
Node Filesystem Snippets index.js x
Node.js file:///C:/Users/emana/De
88 router.get("/logout", _auth.isLoggedIn, function (req, res) {
89   req.logout(function (err) {
90     if (err) {
91       _logger.error(err);
92     }
93     return next(err);
94   }
95   });
96   res.render("login");
97 }
98 router.get("/hello", function (req, res) {
99   res.json({
100     msg: "Hola",
101     session: req.session
102   });
103 });
104 router.get("/info", function (req, res) {
105   console.log("Endpoint info");
106   console.log("Endpoint info 2");
107   res.json({
108     objectProcess: _arguments.objectProcess
109   });
110 });
111 router.get("/comp", (0, _compression["default"])(), function (req, res) {
112   res.json({
113     objectProcess: _arguments.objectProcess
114   });
115 });
116 var scriptPath = _path["default"].resolve(_dirname, "../randoms/index.js");
117
118 router.post("/randoms", function (req, res) {
119   var cantidad = req.query.cantidad;
120   var randoms = (0, _child_process.fork)(scriptPath);
121   if (cantidad) {
122     randoms.send(cantidad);
123   } else {
124     randoms.send(10000);
125   }
126   randoms.on("message", function (cantidad) {
127     res.json({
128       values: cantidad
129     });
130   });
131 });
132
133
134 }
Line 104, Column 30 Coverage: n/a
```

11) 0x



Se puede ver que al usar compression, el tamaño del endpoint se redujo de 641 KB a 179 KB.

Con artillery se probaron 50 conexiones con 20 request, dando como resultado 1000 http.codes.200, significa que todas las request fueron respondidas, en un tiempo max de casi 0.2 segundos y de 389 request por segundo.

Se prueba con autocannon, 100 conexiones de 20 segundos y se detectaron 16 errores y la finalización de la aplicación.

Con inspect de node se observa que el proceso que lleva más tiempo es la respuesta del endpoint, al solicitarse la información del proceso, seguido de los dos console.log.