

>> Consigna: 1

Incorporar al proyecto de servidor de trabajo la compresión gzip.

Verificar sobre la ruta /info con y sin compresión, la diferencia de cantidad de bytes devueltos en un caso y otro.

Ruta “/info”, servidor sin y con “compression”:

Firefox:

- Sin Compresión:

Filtrar URLs

||

+

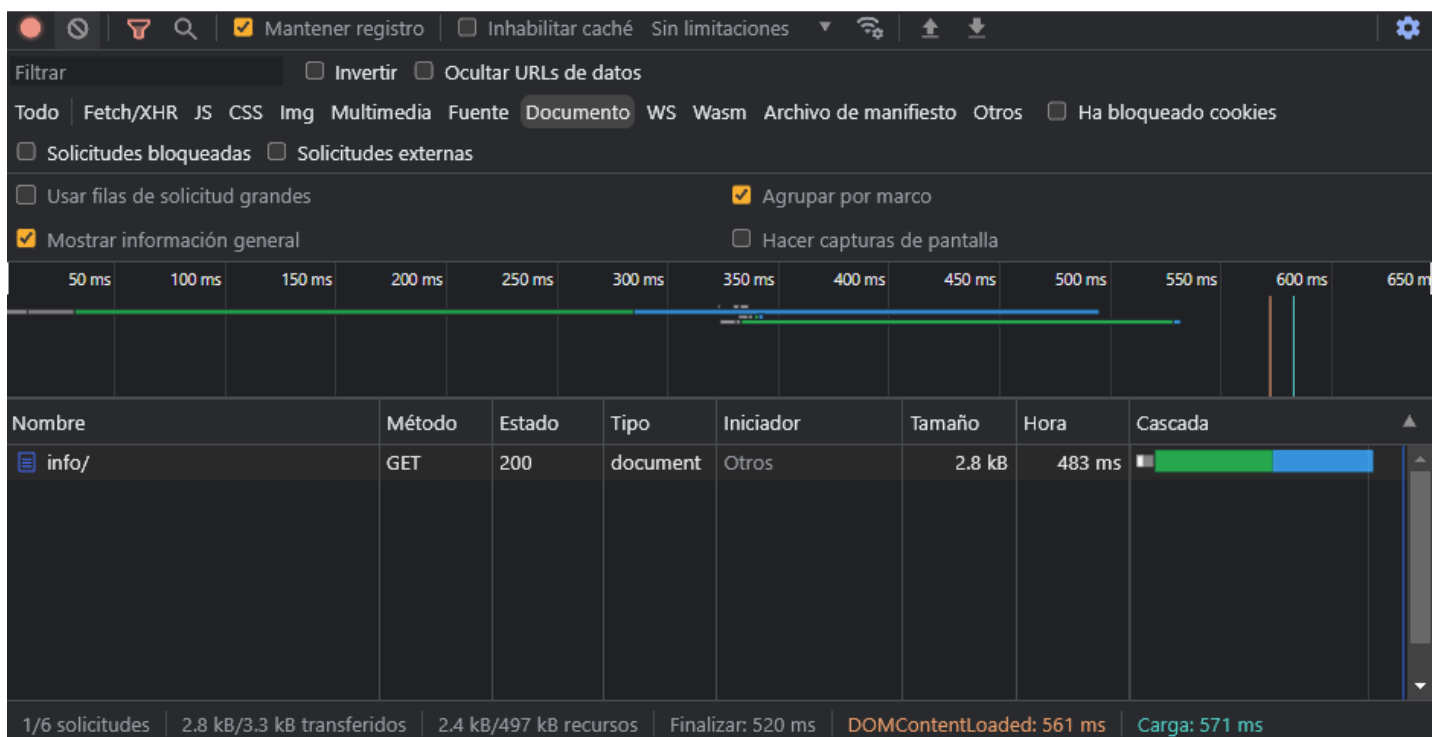
</

- Con Compresión:

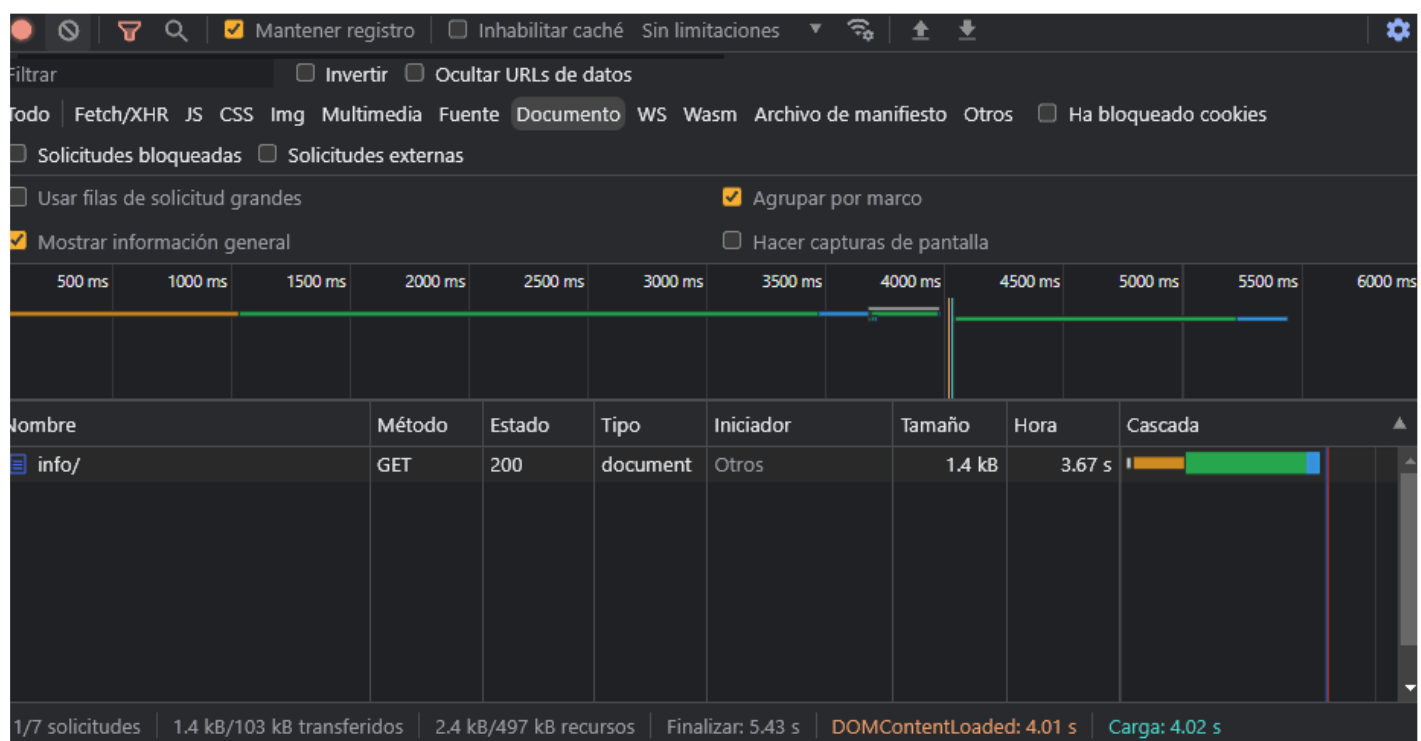
Filtrar URLs

Chrome:

- Sin compresión:



- Con Compresión:



Luego implementar logguego (con alguna librería vista en clase) que registre lo siguiente:

Winston: (<https://www.npmjs.com/package/winston>)

Ruta y método de todas las peticiones recibidas por el servidor (**info**)

- [server.js línea 80](#)

Ruta y método de las peticiones a rutas inexistentes en el servidor (**warning**)

- [server.js línea 129](#)

Errores lanzados por las apis de mensajes y productos, únicamente (**error**)

- [server.js línea 123](#)

Mensajes:

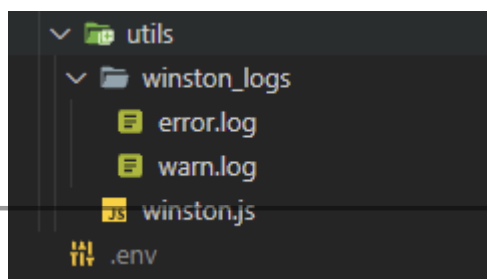
- [/src/routes/router.mensajes.js](#)
- [/src/controllers/contenedorMensajes.js](#)

Productos:

- [/src/controllers/contenedorProductos.js](#)
- [/src/routes/router.productos.js](#)

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> node server
YARGS: { port: 8080, debug: false, modo: 'FORK', pid: 14948, otros: [] }
info: Escuchando en el Puerto: 8080 - MODO: FORK - PID: 14948 - fyh: 15/8/2022, 6:55:53
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /api/mensajes y Metodo: GET solicitado
info: Ruta: /api/mensajes/344 y Metodo: GET solicitado
error: Error Metodo: GET Mensaje 344 no existe
info: Ruta: /api/asdas y Metodo: GET solicitado
warn: Ruta /api/asdas y metodo GET no implementada - fyh: 15/8/2022, 6:56:27
█
```



>> Consigna: 2

Profile con ruta “/info” con y sin console.log

1) node --prof server.js

Get a “/info” con console, detener servidor, re nombrar archivo a “info_console.log”

2) node --prof server.js

Get a “/info” , detener servidor, re nombrar archivo a “info.log”

3) node --prof-process info_console.log > result_info_console.log

```
EXPLOADOR
...
server.js info.js result_info_console.log X TC New Request

DESAFIO-32
> nginx-1.23.1
> node_modules
> public
> src
  > controllers
  > db
  > mocks
  > routes
    > randoms
      info.js
      passport-local.js
      router.js
      router.mensajes.js
      router.productos.js
      routes.js
      yarg-cli.js
    > utils
      .env
      .gitignore
      info_console.log
      info.log
      package-lock.json
      package.json
      README.md
      result_info_console.log
      result_info.log
      server.js

result_info_console.log
1 Statistical profiling result from info_console.log, (985 ticks, 0 unaccounted, 0 excluded).
2
3 [Shared libraries]:
4   ticks total nonlib name
5     585  59.4%
6     384  39.0% C:\Program Files\nodejs\node.exe
7       1   0.1% C:\WINDOWS\System32\KERNELBASE.dll
8       1   0.1% C:\WINDOWS\System32\KERNEL32.DLL
9
10 [JavaScript]:
11   ticks total nonlib name
12     6    0.6% 42.9% LazyCompile: *resolve node:path:158:10
13     1    0.1% 7.1% LazyCompile: *isFileType node:fs:205:20
14     1    0.1% 7.1% LazyCompile: *dirname node:path:653:10
15     1    0.1% 7.1% LazyCompile: *Module._nodeModulePaths node:internal/modules/cjs/loader:400:23
16     1    0.1% 7.1% Function: ^tryExtensions node:internal/modules/cjs/loader:402:13
17     1    0.1% 7.1% Function: ^isAbsolute node:path:402:13
18     1    0.1% 7.1% Function: ^hrtime node:internal/process/per_thread:79:16
19     1    0.1% 7.1% Function: ^Module._findPath node:internal/modules/cjs/loader:494:28
20     1    0.1% 7.1% Function: ^<anonymous> node:vm:313:27
21
22 [C++]:
23   ticks total nonlib name
24
25 [Summary]:
26   ticks total nonlib name
27     14    1.4% 100.0% JavaScript
28     0     0.0%  0.0% C++
29     2     0.2% 14.3% GC
30    971   98.6%   Shared libraries
31
32 [C++ entry points]:
33
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> node server.js -p 8080 -m FORK
YARGS: { port: 8080, debug: false, modo: 'FORK', pid: 9120, otros: [] }
{"level":"info","message":"Escuchando en el Puerto: 8080 - MOD0: FORK - PID: 9120 - fyh: 15/8/2022, 0:37:53"}
{"level":"info","message":"Ruta: /info y Metodo: GET solicitado"}
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32>
```

4) node --prof-process info.log > result_info.log

EXPLORADOR

DESAFIO-32

- nginx-1.23.1
- node_modules
- public
- src
 - controllers
 - db
 - mocks
 - routes
 - randoms
 - info.js
 - passport-local.js
 - router.js
 - router.mensajes.js
 - router.productos.js
 - routes.js
 - yarg-cli.js
 - utils
 - .env
 - .gitignore
 - info_console.log
 - info.log
 - package-lock.json
 - package.json
 - README.md
 - result_info_console.log
 - result_info.log
 - server.js

result_info.log

```
1 Statistical profiling result from info.log, (765 ticks, 0 unaccounted, 0 excluded).
2
3 [Shared libraries]:
4   ticks total nonlib name
5   378   49.4%   C:\WINDOWS\SYSTEM32\ntdll.dll
6   372   48.6%   C:\Program Files\nodejs\node.exe
7    2    0.3%   C:\WINDOWS\System32\KERNEL32.DLL
8
9 [JavaScript]:
10  ticks total nonlib name
11   6    0.8%  46.2% LazyCompile: *resolve node:path:158:10
12   1    0.1%  7.7% LazyCompile: *toNamespacedPath node:path:618:19
13   1    0.1%  7.7% LazyCompile: *stat node:internal/modules/cjs/loader:151:14
14   1    0.1%  7.7% LazyCompile: *nextPart node:fs:2401:31
15   1    0.1%  7.7% LazyCompile: *dirname node:path:653:10
16   1    0.1%  7.7% Function: ^retry D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32\
17   1    0.1%  7.7% Function: ^internalBinding node:internal/bootstrap/loaders:164:45
18   1    0.1%  7.7% Function: ^basename node:path:749:11
19
20 [C++]:
21  ticks total nonlib name
22
23 [Summary]:
24  ticks total nonlib name
25   13    1.7% 100.0% JavaScript
26    0    0.0%  0.0% C++
27    4    0.5% 30.8% GC
28   752   98.3%   Shared libraries
29
30 [C++ entry points]:
31  ticks cpp total name
32
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> node server.js -p 8080 -m FORK
YARGS: { port: 8080, debug: false, modo: 'FORK', pid: 1848, otros: [] }
{"level":"info","message":"Escuchando en el Puerto: 8080 - MODULO: FORK - PID: 1848 - fyh: 15/8/2022, 0:41:53"}
{"level":"info","message":"Ruta: /info y Metodo: GET solicitado"}
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> ^C
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32>
```

Utilizaremos como test de carga Artillery en línea de comandos, emulando 50 conexiones concurrentes con 20 request por cada una. Extraer un reporte con los resultados en archivo de texto.

`npm install -g artillery`

1) `node server.js -p 8080 -m FORK`

2) `artillery quick --count 50 -n 20 "http://localhost:8080/info" > artillery/result_artillery_info_console.txt`

The screenshot shows a VS Code editor with three panels. The left panel displays `server.js` with a REST client for the `GET /info` endpoint. The middle panel shows the `result_artillery_info_console.txt` file, which contains the output of an Artillery load test. The right panel shows the terminal window with the command `artillery quick --count 50 -n 20 "http://localhost:8080/info" > result_artillery_info_console.txt` and its output, including the process ID, directory, and port.

```
src > routes > info.js > routerInfo.get("/") callback
6
7 const info = {
8   "Argumentos de entrada": process.argv,
9   "Nombre de la plataforma": process.platform,
10  "Versión de Node.js": process.version,
11  "Path de ejecución": process.execPath,
12  "Memoria total de reservada": process.memoryUsage().heapTotal,
13  "Process ID": process.pid,
14  "Directorio actual del trabajo": process.cwd()
15 }
16
17
18 routerInfo.get('/', (req, res) => {
19   const PORT = req.socket.localPort;
20   const infoProyecto = {
21     argumentos: process.argv, // "Argumentos de entrada"
22     plataforma: process.platform, // "Nombre de la plataforma"
23     versionNode: process.version, // "Versión de Node.js"
24     pathEjecucion: process.execPath, // "Path de ejecución"
25     memoriaTotalReservada: Math.floor(process.memoryUsage().heapTotal / 1024), // "Memoria total de reservada"
26     processId: process.pid, // "Process ID"
27     directorioActualTrabajo: process.cwd(), // "Directorio actual del trabajo"
28     numProcesadores: os.cpus().length, // "Número de procesadores"
29     PORT: PORT,
30   };
31   console.log(infoProyecto);
32   res.render('info', { title: "Info", infoProyecto });
33 });
34
35
36 module.exports = routerInfo;
```

```
1 Running scenarios...
2 Phase started: unnamed (index: 0, duration: 1s) 03:35:28(-0300)
3
4 Phase completed: unnamed (index: 0, duration: 1s) 03:35:29(-0300)
5
6 -----
7 Metrics for period to: 03:35:30(-0300) (width: 1.283s)
8 -----
9
10 http.codes.200: ..... 50
11 http.request_rate: ..... 43/sec
12 http.requests: ..... 55
13 http.response_time:
14   min: ..... 9
15   max: ..... 437
16   median: ..... 135.7
17   p95: ..... 354.3
18   p99: ..... 354.3
19 http.responses: ..... 50
20 vusers.created: ..... 50
21 vusers.created_by_name.0: ..... 50
22
23 -----
24 Metrics for period to: 03:35:40(-0300) (width: 9.917s)
25 -----
26
27 http.codes.200: ..... 241
28 http.request_rate: ..... 27/sec
29 http.requests: ..... 271
30 http.response_time:
31   min: ..... 601
```

```
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> artillery quick --count 50 -n 20 "http://localhost:8080/info" > result_artillery_info_console.txt
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32>
```

```
memoriaTotalReservada: 150,
processId: 14740,
directorioActualTrabajo: 'D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32',
numProcesadores: 4,
PORT: 8080
}
info: Ruta: /info y Metodo: GET solicitado
{
  argumentos: [
    'C:\Program Files\nodejs\node.exe',
    'D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32\server.js',
    '-p',
    '8080',
    '-m'
  ]
}
```

```
3) artillery quick --count 50 -n 20 "http://localhost:8080/info" >
   result_artillery_info.txt
```

[illegible]

>> Consigna: 3

Luego utilizaremos Autocannon en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20 segundos. Extraer un reporte con los resultados (puede ser un print screen de la consola)

1 - npm start

2 - npm run test (sin console.log)

```
PROBLEMAS      SALIDA      CONSOLA DE DEPURACIÓN    TERMINAL

info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
info: Ruta: /info y Metodo: GET solicitado
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32>
```

```
> desafio-32@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel...g:load:flatten Completed in 4ms
Running 20s test @ http://localhost:8080/info?latten Completed in 4ms
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	888 ms	1022 ms	4215 ms	4767 ms	1342.53 ms	810.44 ms	5696 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1	1	92	102	71.75	33.18	1
Bytes/Sec	3.02 kB	3.02 kB	278 kB	308 kB	217 kB	100 kB	3.02 kB

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

2k requests in 20.18s, 4.34 MB read
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32>
```



```
> desafio-32@1.0.0 test
> node benchmark.js
```

```
Running all benchmarks in parallel...g:load:flatten Completed in 4ms
Running 20s test @ http://localhost:8080/infoflatten Completed in 4ms
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	888 ms	1022 ms	4215 ms	4767 ms	1342.53 ms	810.44 ms	5696 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1	1	92	102	71.75	33.18	1
Bytes/Sec	3.02 kB	3.02 kB	278 kB	308 kB	217 kB	100 kB	3.02 kB

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

```
2k requests in 20.18s, 4.34 MB read
```

```
PS D:\CODERHOUSE\CODERHOUSE\BACKEND\DESAFIOS\Desafio-32> 
```

1 - npm start

2 - npm run test (con console.log)

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

```
    ],
    plataforma: 'win32',
    versionNode: 'v16.16.0',
    pathEjecucion: 'C:\\Program Files\\nodejs\\node.exe',
    memoriaTotalReservada: 217,
    processId: 14652,
    directorioActualTrabajo: 'D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32',
    numProcesadores: 4,
    PORT: 8080
  }
}
info: Ruta: /info y Metodo: GET solicitado
{
  argumentos: [
    'C:\\Program Files\\nodejs\\node.exe',
    'D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32\\server.js'
  ],
  plataforma: 'win32',
  versionNode: 'v16.16.0',
  pathEjecucion: 'C:\\Program Files\\nodejs\\node.exe',
  memoriaTotalReservada: 217,
  processId: 14652,
  directorioActualTrabajo: 'D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32',
  numProcesadores: 4,
  PORT: 8080
}
PS D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32>
```

```
> desafio-32@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel...g:load:flatten Completed in 5ms
Running 20s test @ http://localhost:8080/info:load:flatten Completed in 5ms
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1223 ms	1307 ms	4946 ms	5361 ms	1742.52 ms	928.99 ms	6319 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	72	80	55.75	26.73	5
Bytes/Sec	0 B	0 B	218 kB	242 kB	169 kB	80.8 kB	15.1 kB

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

1k requests in 20.16s, 3.37 MB read

```
PS D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32>
```

```
> desafio-32@1.0.0 test
> node benchmark.js
```

```
Running all benchmarks in parallel...g:load:flatten Completed in 5ms
Running 20s test @ http://localhost:8080/info:load:flatten Completed in 5ms
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	1223 ms	1307 ms	4946 ms	5361 ms	1742.52 ms	928.99 ms	6319 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	0	0	72	80	55.75	26.73	5
Bytes/Sec	0 B	0 B	218 kB	242 kB	169 kB	80.8 kB	15.1 kB

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

1k requests in 20.16s, 3.37 MB read

```
PS D:\\CODERHOUSE\\CODERHOUSE\\BACKEND\\DESAFIOS\\Desafio-32>
```

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.

- Ruta “/info” con console.log:
 - 1) node --inspect server.js
 - 2) chrome://inspect/#devices

ProfilerConsolaFuentesMemoria

Sistema de archivos>>server.jsinfo.js x

+ Añadir carpeta a espacio de trab

Sincronizar cambios en DevTools con el sistema de archivos local

Más información sobre los espacios de trabajo

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

192.4 ms

200.1 ms

210.7 ms

220.7 ms

230.4 ms

240.3 ms

251.2 ms

260.1 ms

270.5 ms

282.3 ms

290.1 ms

30

318.5 ms

3215.6 ms

33

34

35

36

const Router = require('express').Router;

const os = require('os');

const routerInfo = Router();

const info = {

"Argumentos de entrada": process.argv,

"Nombre de la plataforma": process.platform,

"Versión de Node.js": process.version,

"Path de ejecución": process.execPath,

"Memoria total de reservada": process.memoryUsage().rss,

"Process ID": process.pid,

"Directorio actual del trabajo": process.cwd(),

}

routerInfo.get('/', (req, res) => {

const PORT = req.socket.localPort;

const infoProyecto = {

argumentos: process.argv, // "Argumentos de entrada"

plataforma: process.platform, // "Nombre de la plat"

versionNode: process.version, // "Versión de Node.js"

pathEjecucion: process.execPath, // "Path de ejecu"

memoriaTotalReservada: Math.floor(process.memoryUsa

processId: process.pid, // "Process ID": process.pi

directorioActualTrabajo: process.cwd(), // "Directo

numProcesadores: os.cpus().length, // "Número de pr

PORT: PORT,

};

console.log(infoProyecto);

res.render('info', { title: "Info" , infoProyecto});

});

module.exports = routerInfo;

- Ruta “/info”:
1) node --inspect server.js
2) chrome://inspect/#devices

The screenshot shows the Chrome DevTools interface with the 'Fuentes' (Sources) tab selected. The file 'server.js' is open, and the 'info.js' file is also visible. The code defines a REST API endpoint for '/info' that returns project information. Performance data is shown for the execution of this endpoint.

Linea	Time	Code
19	2.6 ms	const PORT = req.socket.localPort;
20	0.1 ms	const infoProyecto = {
21	1.4 ms	argumentos: process.argv, // "Argumentos de entrada": process.
22	0.1 ms	plataforma: process.platform, // "Nombre de la plataforma": pr
23		versionNode: process.version, // "Versión de Node.js": process
24		pathEjecucion: process.execPath, // "Path de ejecución": proce
25	1.9 ms	memoriaTotalReservada: Math.floor(process.memoryUsage().rss /
26	0.6 ms	processId: process.pid, // "Process ID": process.pid,
27	0.3 ms	directorioActualTrabajo: process.cwd(), // "Directorio actual
28	3.2 ms	numProcesadores: os.cpus().length, // "Número de procesadores"
29		PORT: PORT,
30		};
31		// console.log(infoProyecto);
32	19.4 ms	res.render('info', { title: "Info" , infoProyecto});
33		});
34		
35		
36		module.exports = routerInfo;

Additional information on the left sidebar:

- ± Añadir carpeta a espacio de trab
- Sincronizar cambios en DevTools con el sistema de archivos local
- [Más información sobre los espacios de trabajo](#)

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

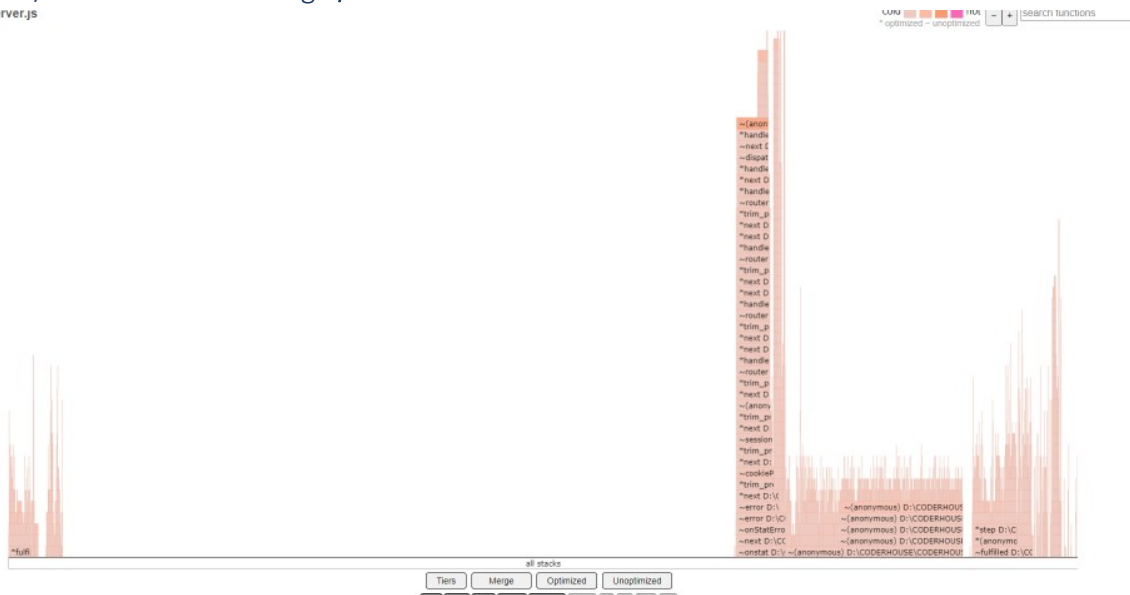
1- npm run ox (0x)

2 – npm run test (autocanon)

3 – Detener servidor

Ruta “/info” con console.log : /9456.0x

node server.js



Ruta “/info” sin console.log : /12216.0x

node server.js

