En primer lugar, incorporamos un gzip al proyecto

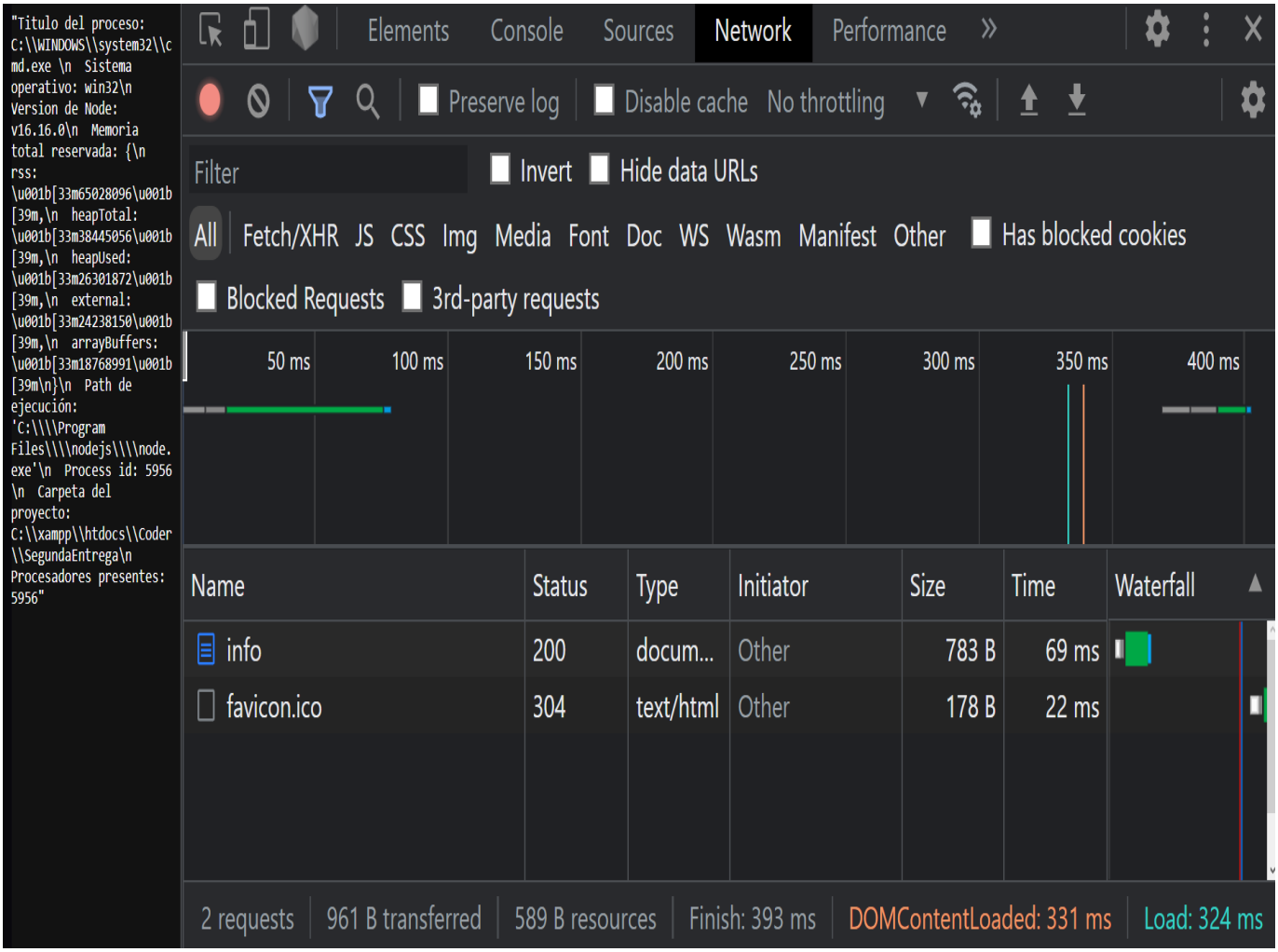
```
//const { fork } = require("child_process");
const isAuth = require ('../middlewares/isAuth.js');
const util = require("util");
const compression = require ("compression");
const logger = require ("../logger.js");

module.exports = function(passport){
  routes.get('/info',compression(), (req,res)=>{util
    res.json(
    `Titulo del proceso: ${process.title}
    Sistema operativo: ${process.platform}
    Version de Node: ${process.version}
    Memoria total reservada: ${util.inspect(process.memoryUsage(), {
      showHidden: false,
      depth: null,
      colors: true})}
    Path de ejecución: ${util.inspect(process.execPath)}
    Process id: ${process.pid}
    Carpeta del proyecto: ${process.cwd()}
    Procesadores presentes: ${process.pid}`
  )});
routes.get('/', isAuth,(req,res)=>{
  res.sendFile(path.join(__dirname, ".././public/home.html"))
})
routes.get('/login',(req, res)=>{
  if(req.isAuthenticated()){
  res.redirect('/')
  }else{
  res.sendFile(path.join(__dirname, ".././public/login.html"));
  }
```

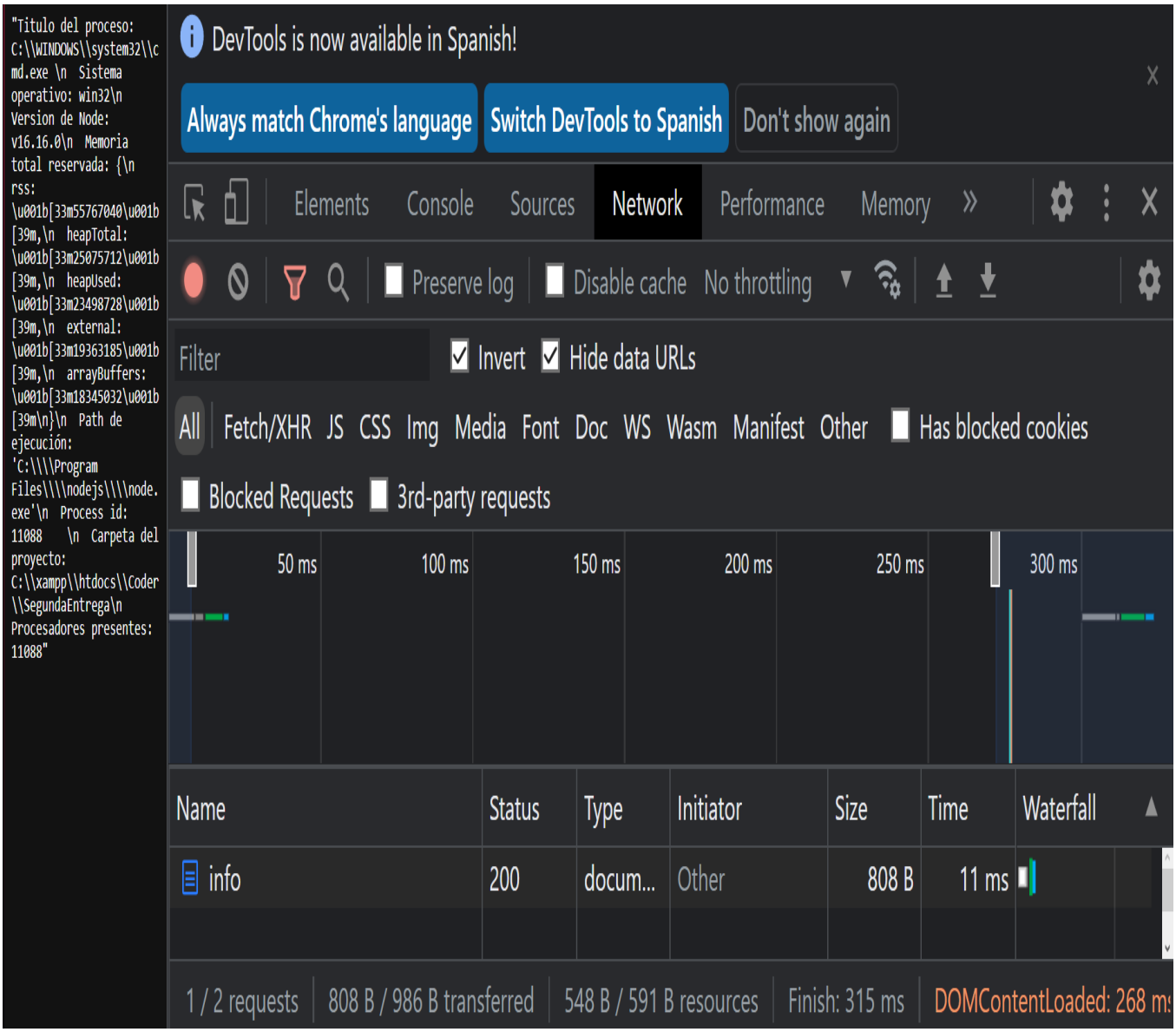
En el proyecto lo deje en el final del export, y comente el child_process

Comprobamos su uso en la ruta /info

Sin comprimir:



Comprimido:



Por alguna razón el archivo incremento su peso. Probé anteriormente con otras rutas y funcionó adecuadamente

Implementado el loggeo con Winston

logger.js > buildProdLogger

```
const winston = require("winston");
require("dotenv").config()

const buildProdLogger = () => {
  const prodLogger = winston.createLogger({
    transports: [
      new winston.transports.File({ filename: "warn.log", level: "warn" }),
      new winston.transports.File({ filename: "error.log", level: "error" }),
    ],
  });

  return prodLogger;
};

const buildDevLogger = () => {
  const devLogger = winston.createLogger({
    transports: [new winston.transports.Console({ level: "info" })],
  });

  return devLogger;
};

let logger;

if (process.env.NODE_ENV.toLocaleUpperCase() === "PROD") {
  logger = buildProdLogger();
} else {
  logger = buildDevLogger();
}

module.exports = logger;
```

Prender el servidor en modo prof

```
PS C:\xampp\htdocs\Coder\SegundaEntrega> node --prof server.js
8080
{"level":"info","message":"Servidor http escuchando en el puerto 8080 - PID 10540"}
Servidor escuchando: 8080
```

Uso de artillery con 50 conexiones y 20 request

```
PS C:\xampp\htdocs\Coder\SegundaEntrega> artillery quick --count 50 --num 20 http://localhost:8080/info > result_fork.txt
PS C:\xampp\htdocs\Coder\SegundaEntrega>
```

Uso de –prof-process

```
PS C:\xampp\htdocs\Coder\SegundaEntrega> node --prof-process prof_info.log > re
sult_prof.txt
(node:4532) ExperimentalWarning: VM Modules is an experimental feature. This fe
ature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
Could not find function 0xec540cf90
Could not find function 0xec540cf90
Could not find function 0xec540d408
Could not find function 0x30482ffce70
Could not find function 0xec540d458
Could not find function 0xec540cf90
Could not find function 0xec540d408
Could not find function 0x30482ffce70
Could not find function 0xec540d458
Could not find function 0xec540cf90
Could not find function 0xec540d408
Could not find function 0x30482ffce70
Could not find function 0xec540d408
Could not find function 0x3fda9e5940
Could not find function 0xec540cf90
Could not find function 0xec540d408
Could not find function 0x30482ffce70
Could not find function 0x30482fcf918
Could not find function 0x15df9578fc0
Could not find function 0xad5e051570
Could not find function 0xad5e0509a0
Could not find function 0xad5e051018
Could not find function 0x15df9578fc0
Could not find function 0xad5e06ad10
Could not find function 0xad5e064ce0
Could not find function 0xad5e050700
Could not find function 0xad5e0445a0
Could not find function 0xad5e052208
PS C:\xampp\htdocs\Coder\SegundaEntrega>
```

Uso de autocannon, emulando 100 conexiones en un lapso de 20 segundos

```
PS C:\xampp\htdocs\Coder\SegundaEntrega> npm test
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> backend@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	82 ms	122 ms	467 ms	664 ms	160.1 ms	108.86 ms	795 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	197	197	571	1062	623.1	278.78	197
Bytes/Sec	165 kB	165 kB	479 kB	891 kB	523 kB	234 kB	165 kB

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

13k requests in 20.28s, 10.5 MB read
PS C:\xampp\htdocs\Coder\SegundaEntrega>
```

Configuración encontrada en benchmark.js

Prender el servidor mediante inspect

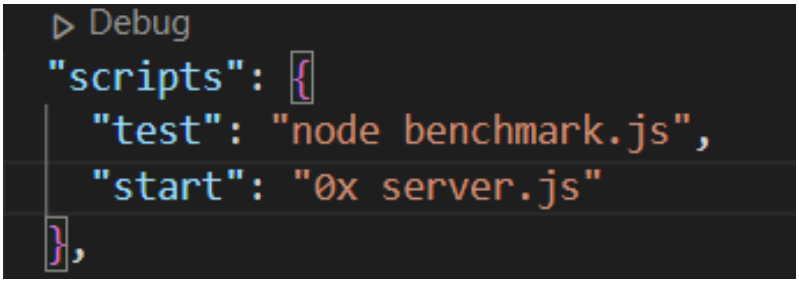
```
PS C:\xampp\htdocs\Coder\SegundaEntrega> node --inspect server.js
Debugger listening on ws://127.0.0.1:9229/b6898b3e-9606-458f-9e11-4a9233eb43f9
For help, see: https://nodejs.org/en/docs/inspector
8080
{"level":"info","message":"Servidor http escuchando en el puerto 8080 - PID 664"}
Servidor escuchando: 8080
```

Pantallazo de Chrome://inspect

Self Time		Total Time		Function	
00509.6 ms		00509.6 ms		(idle)	
31.3 ms	21.43 %	78.3 ms	53.57 %	▸ deserializeObject	deserializer.ts:117
21.0 ms	14.37 %	21.0 ms	14.37 %	(program)	
5.9 ms	4.05 %	11.0 ms	7.54 %	▸ serialize	bson.ts:137
5.4 ms	3.73 %	6.0 ms	4.13 %	▸ writeBuffer	
4.4 ms	3.02 %	4.4 ms	3.02 %	▸ Long	long.ts:136
3.5 ms	2.38 %	5.1 ms	3.49 %	▸ nextTick	node:internal/p...ask_queues:104
3.2 ms	2.22 %	88.1 ms	60.32 %	callbackTrampoline	node:internal/async_hooks:118
3.2 ms	2.22 %	4.3 ms	2.94 %	emitHook	node:internal/async_hooks:228
2.6 ms	1.75 %	2.6 ms	1.75 %	▸ utf8Slice	
2.3 ms	1.59 %	4.9 ms	3.33 %	▸ slice	node:buffer:593
1.9 ms	1.27 %	3.2 ms	2.22 %	▸ emitInitScript	node:internal/async_hooks:485
1.7 ms	1.19 %	83.4 ms	57.06 %	▸ onStreamRead	node:internal/s...se_commons:171
1.6 ms	1.11 %	1.6 ms	1.11 %	(garbage collector)	
1.6 ms	1.11 %	76.0 ms	51.98 %	▸ processIncomingData	message_stream.ts:135
1.6 ms	1.11 %	68.4 ms	46.83 %	▸ (anonymous)	connection.ts:256
1.6 ms	1.11 %	90.1 ms	61.67 %	▸ addChunk	node:internal/s...s/readable:304
1.5 ms	1.03 %	6.0 ms	4.13 %	▸ serializeInto	serializer.ts:750
1.4 ms	0.95 %	1.7 ms	1.19 %	▸ slice	node:buffer:1115
1.3 ms	0.87 %	51.6 ms	35.32 %	▾ parse	commands.ts:643
1.3 ms	0.87 %	51.6 ms	35.32 %	▸ onMessage	connection.ts:381
1.3 ms	0.87 %	5.2 ms	3.57 %	▸ Long.comp	long.ts:420
1.0 ms	0.71 %	1.0 ms	0.71 %	▸ FastBuffer	node:internal/buffer:958
1.0 ms	0.71 %	171.2 ms	117.14 %	▸ emit	node:events:340
1.0 ms	0.71 %	66.8 ms	45.71 %	▸ onMessage	connection.ts:381
1.0 ms	0.71 %	6.3 ms	4.29 %	▸ serverUpdateHandler	topology.ts:608
1.0 ms	0.71 %	3.8 ms	2.62 %	▸ Long.compare	long.ts:403
0.9 ms	0.63 %	5.1 ms	3.49 %	processTicksAndRejections	node:internal/p...task_queues:68
0.9 ms	0.63 %	3.4 ms	2.30 %	▸ onwrite	node:internal/s...s/writable:425

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

Aquí supondré que se refiere iniciar el servidor con 0x, como lo deje configurado en el package.json




```
PS C:\xampp\htdocs\Coder\SegundaEntrega> npm start
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> backend@1.0.0 start
> 0x server.js

Profiling8080
{"level":"info","message":"Servidor http escuchando en el puerto 8080 - PID 5876"}
Servidor escuchando: 8080
```