Desafío: Loggers, gzip y análisis de performance

Nombre: Eduardo Granero

Información Preliminar:

Ejecutar mongo desde el archivo INICIAR MONGODB.bat ubicado en la carpeta /node/public

Para la realización de este desafío se utilizaron los siguientes comandos por consola:

Artillery:
artillery quick --count 50 -n 20 http://localhost:8000/info > result_conConsoleLog.txt
artillery quick --count 50 -n 20 http://localhost:8000/info > result_sinConsoleLog.txt

Profiling:
node --prof-process isolate-conConsoleLog.log > result_conConsoleLog_prof.txt
node --prof-process isolate-sinConsoleLog.log > result_sinConsoleLog_prof.txt

Artillery con Inspect:
artillery quick --count 50 -n 20 http://localhost:8000/info > result_conConsoleLog_conInspect.txt
artillery quick --count 50 -n 20 http://localhost:8000/info > result_sinConsoleLog_sinInspect.txt

Codigo con/sin console.log:

```
127  app.get("/info",(req,res)=>{
128    const informacion = {
129      Argumentos: args,
130      Plataforma: process.platform,
131      ID: process.pid,
132      Version: process.version,
133      Memoria:process.memoryUsage(),
134      Path:process.execPath,
135      Carpeta: process.cwd()
136    }
137    console.log(informacion)//esta es la linea que agrego
138    res.json(informacion);
139  })
```

Conclusión:

Puede verse por los resultados obtenidos que el rendimiento se ve afectado por el agregado de cualquier función, particularmente las que están relacionadas con la función process (como console.log).

Si bien los resultados no son totalmente concluyentes al aplicar artillery puede verse en el inspect que la función console.log agrega tiempo adicional.

Resultado usando Artillery

Sin console.log

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 21:45:11(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 21:45:12(-0300)

----------------------------------------
Metrics for period to: 21:45:20(-0300) (width: 8.527s)
----------------------------------------

http.codes.200: ..... 415
http.request_rate: ..... 53/sec
http.requests: ..... 445
http.response_time:
  min: ..... 3
  max: ..... 1840
  median: ..... 424.2
  p95: ..... 788.5
  p99: ..... 1153.1
http.responses: ..... 415
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50


All VUs finished. Total time: 25 seconds

--------------------------------
Summary report @ 21:45:34(-0300)
--------------------------------

http.codes.200: ..... 1000
http.request_rate: ..... 22/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 3
  max: ..... 2492
  median: ..... 468.8
  p95: ..... 907
  p99: ..... 1556.5
http.responses: ..... 1000
vusers.completed: ..... 50
vusers.created: ..... 50
vusers.created_by_name.0: ..... 50
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 18683.7
  max: ..... 21720.1
  median: ..... 19737.6
  p95: ..... 20958.1
  p99: ..... 21381.5
```

Con console.log

```
Running scenarios...
Phase started: unnamed (index: 0, duration: 1s) 21:29:29(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 21:29:30(-0300)

---------------------------------------
Metrics for period to: 21:29:30(-0300) (width: 0.918s)
---------------------------------------

http.codes.200: ..................................................... 50
http.request_rate: ..................................................... 69/sec
http.requests: ..................................................... 69
http.response_time:
  min: ..................................................... 1
  max: ..................................................... 333
  median: ..................................................... 2
  p95: ..................................................... 327.1
  p99: ..................................................... 327.1
http.responses: ..................................................... 50
vusers.created: ..................................................... 47
vusers.created_by_name.0: ..................................................... 47


All VUs finished. Total time: 25 seconds

--------------------------------
Summary report @ 21:29:52(-0300)
--------------------------------

http.codes.200: ..................................................... 1000
http.request_rate: ..................................................... 29/sec
http.requests: ..................................................... 1000
http.response_time:
  min: ..................................................... 1
  max: ..................................................... 2424
  median: ..................................................... 376.2
  p95: ..................................................... 925.4
  p99: ..................................................... 1043.3
http.responses: ..................................................... 1000
vusers.completed: ..................................................... 50
vusers.created: ..................................................... 50
vusers.created_by_name.0: ..................................................... 50
vusers.failed: ..................................................... 0
vusers.session_length:
  min: ..................................................... 17375.4
  max: ..................................................... 19426.7
  median: ..................................................... 18588.1
  p95: ..................................................... 19346.7
  p99: ..................................................... 19346.7
```

Resultados usando Inspect+Artillery

Con Console.Log

```
101          app.use(function(req,res,next){
102   22.2 ms    req.session._garbage = Date();
103    0.9 ms    req.session.touch();
104              next()
105          })
106
107          app.use(function (req, res, next) {
108    6.4 ms    logger.info(req.method + " " + req.url);
109    1.6 ms    next();
110          });
111
112          app.use('/api/productos',routerProductos);
113          app.use('/api/productos-test',routerProductostest);
114          app.use('/api/randoms',randoms);
115          app.use('/login',routerLogin);
116
117          app.get('/', async (req, res) => {
118            if (req.session.user) res.redirect("/login")
119            else res.redirect("/login")
120          });
121
122
123          app.get("/info",(req,res)=>{
124    0.9 ms    const informacion = {
125    0.3 ms      Argumentos: args,
126    0.6 ms      Plataforma: process.platform,
127    0.3 ms      ID: process.pid,
128    0.7 ms      Version: process.version,
129    1.6 ms      Memoria:process.memoryUsage(),
130    0.6 ms      Path:process.execPath,
131    0.6 ms      Carpeta: process.cwd()
132            }
133    8.0 ms    console.log(informacion)//esta es la linea que agrego
134   17.6 ms    res.json(informacion);
135              //res.send("asdasd".repeat(1000))
136          })
137
138          app.get('*', (req, res)=>{
139            logger.warn(req.method + " " + req.url);
140            res.status(404).send('sitio no encotrado');
141          });
```

Sin Console.Log

```
101          app.use(function(req,res,next){
102   20.8 ms    req.session._garbage = Date();
103    0.6 ms    req.session.touch();
104              next()
105          })
106
107          app.use(function (req, res, next) {
108    4.1 ms    logger.info(req.method + " " + req.url);
109    2.1 ms    next();
110          });
111
112          app.use('/api/productos',routerProductos);
113          app.use('/api/productos-test',routerProductostest);
114          app.use('/api/randoms',randoms);
115          app.use('/login',routerLogin);
116
117          app.get('/', async (req, res) => {
118            if (req.session.user) res.redirect("/login")
119            else res.redirect("/login")
120          });
121
122
123          app.get("/info",(req,res)=>{
124    0.4 ms    const informacion = {
125    0.6 ms      Argumentos: args,
126    0.5 ms      Plataforma: process.platform,
127    0.4 ms      ID: process.pid,
128    0.4 ms      Version: process.version,
129    1.3 ms      Memoria:process.memoryUsage(),
130    0.3 ms      Path:process.execPath,
131    0.5 ms      Carpeta: process.cwd()
132              }
133              //console.log(informacion)//esta es la linea que agrego
134    9.4 ms    res.json(informacion);
135              //res.send("asdasd".repeat(1000))
136          })
```

Resultados Obtenidos con Autocannon

Con console.log

```
Eduardo@DESKTOP-JH2OVV1 MINGW64 ~/Documents/Documentacion personal/Cursos/CoderHouse/Back End/Desafio-BackEnd/
main)
$ npm test
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> desafio-backend@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8000/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	830 ms	1004 ms	1844 ms	2004 ms	1057.43 ms	237.08 ms	2849 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	20	20	95	99	90.95	16.36	20
Bytes/Sec	8.36 kB	8.36 kB	39.8 kB	41.4 kB	38.1 kB	6.85 kB	8.36 kB

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

2k requests in 20.28s, 761 kB read
```

Sin console.log

```
Eduardo@DESKTOP-JH2OVV1 MINGW64 ~/Documents/Documentacion personal/Cursos/CoderHouse/Back End/Desafio-BackEnd/no
de/public (main)
$ npm test
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

> desafio-backend@1.0.0 test
> node benchmark.js

Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8000/info
100 connections
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	334 ms	1017 ms	1982 ms	2136 ms	1097.58 ms	388.11 ms	3497 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	15	15	95	110	87.3	21.99	15
Bytes/Sec	12.4 kB	12.4 kB	78.8 kB	91.2 kB	72.3 kB	18.2 kB	12.4 kB

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

2k requests in 20.24s, 1.45 MB read
```