# Desafio 14

## PROF

```
[Summary]:
  ticks  total  nonlib   name
      4   0.0%  100.0%  JavaScript
      0   0.0%    0.0%  C++
      9   0.1%  225.0%  GC
   9324  100.0%          Shared libraries
```

Podemos ver que lo que hace más trabajo son las librerias de node, ya que lo que hicimos es algo muy ligero, no tiene iteraciones ni es un proceso bloqueante.

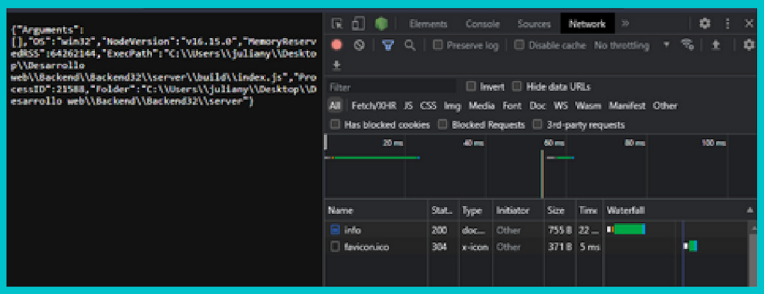
## Artillery

Al ser una ruta con un proceso muy ligero podemos ver que la respuesta media de /info es de 18ms

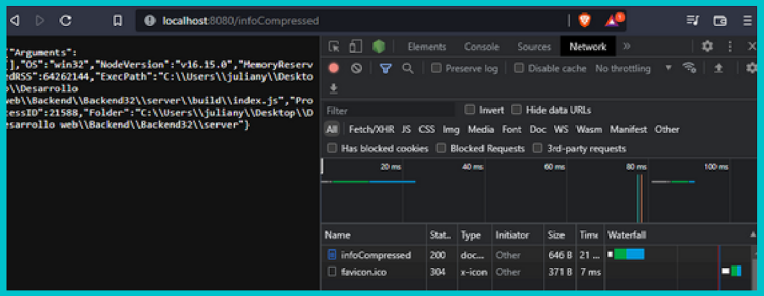
```
Phase started: unnamed (index: 0, duration: 1s) 15:38:29(-0300)

Phase completed: unnamed (index: 0, duration: 1s) 15:38:30(-0300)

-------------------------------------
Metrics for period to: 15:38:40(-0300) (width: 2.059s)
-------------------------------------

http.codes.200: ..... 1000
http.request_rate: ..... 486/sec
http.requests: ..... 1000
http.response_time:
  min: ..... 1
  max: ..... 70
  median: ..... 18
  p95: ..... 34.8
  p99: ..... 46.1
http.responses: ..... 1000
vusers.completed: ..... 20
vusers.created: ..... 19
vusers.created_by_name.0: ..... 19
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 977.2
  max: ..... 1485.5
  median: ..... 1326.4
  p95: ..... 1465.9
  p99: ..... 1465.9
```

## Compression

Acá podemos ver que la ruta con gzip tiene menor tamaño y tardo ligeramente menos, 1ms, que el que no está comprimido.

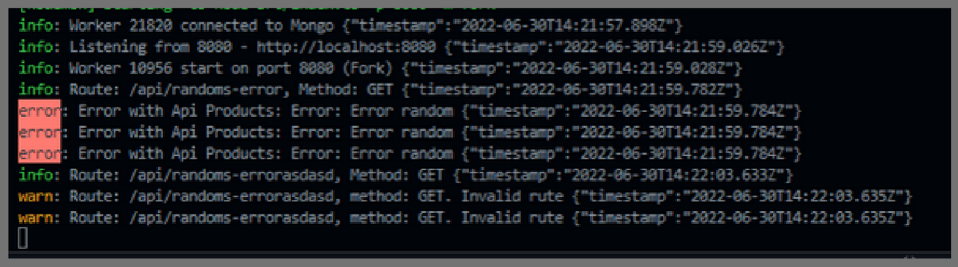



```
// Desafios
const info = {
    Arguments: config.arguments,
    OS: config.os,
    NodeVersion: config.NodeVersion,
    MemoryReservedRSS: config.MemoryReservedRSS,
    ExecPath: config.ExecPath,
    ProcessID: config.ProcessID,
    Folder: config.Folder,
}
app.get('/info', (req, res) => {
    res.send(info)
})

app.get('/infoCompressed', compression(), (req, res) => {
    res.send(info)
})
```



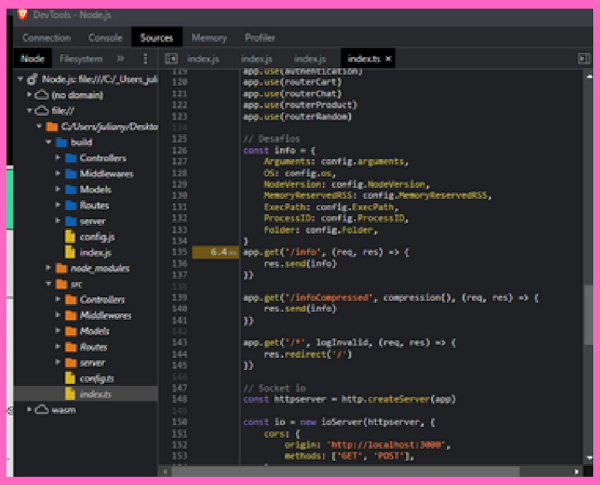
# Logging

Cree la ruta /api/random-error para que genere un error y lo muestre en la consola. Así también, puse un middleware para que detecte las rutas inexistenes.



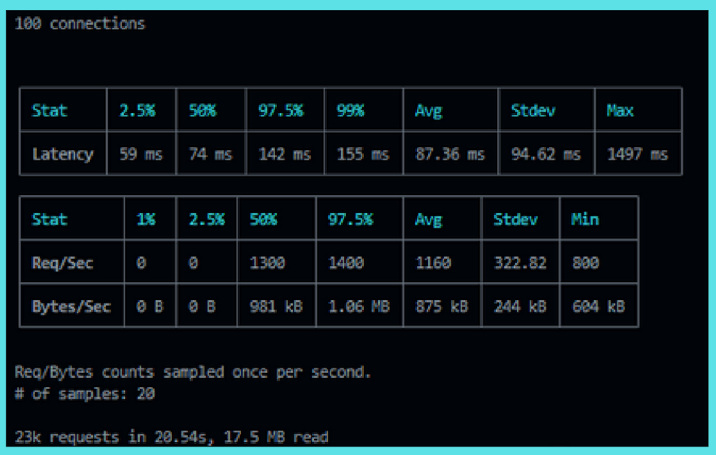

# Inspect

Al tener el código en TypeScript, realmente no se donde tengo que buscar en especifico. Vi los archivos en JS y no me mostraban el tiempo de ejecución pero los archivos de TS si. Podemos ver que la ruta /info tardo 6.4ms en enviar la información.



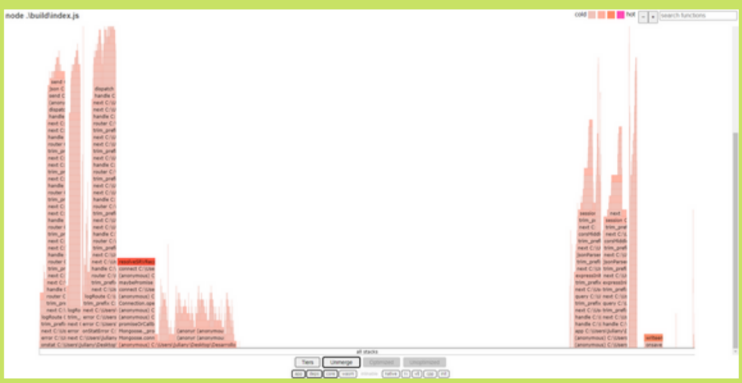
# Autocannon

A medida que va haciendo los request, la latencia sube. En cuanto a los bytes tiene un promedio de 875kB/s.




# 0x

Me sorprende que lo que se ve parezca que tenga procesos bloqueantes, pero en realidad no. No se si puede interferir algo TS, pero realmente no entiendo mucho esta grafica. La mayoría de las cosas que hay en la muestra son librerias de NodeJS



# Conclusión

Tener pruebas de rendimiento en nuestra aplicación nos ayuda bastante para poder mejorar nuestro código y visualizar los posibles errores.
Obviamente, hay que mejorar varios factores, principalmente aprender a leer bien los graficos e identificar los problemas. Ya es cuestión mía para ponerme a investigar más sobre el tema.