Entrega 16 - Coderhouse

```
Compresión:

/info sin comprimir: 621B
/info con compresión: 640B

Esto se debe a que gzip tiene un tamaño mínimo de archivo a la hora de
comprimir y en archivos tan pequeños puede llevar a este caso. Si lo
enviado en /info fuera de 1KB, por ejemplo, sí que se notaría la
compresión.

Performance con --prof de node.js y artillery:

Con console.log:

 [Shared libraries]:
   ticks  total  nonlib   name
   4412   65.0%          C:\WINDOWS\SYSTEM32\ntdll.dll
   2290   33.7%          C:\Program Files\nodejs\node.exe
      3    0.0%          C:\WINDOWS\System32\KERNELBASE.dll
      2    0.0%          C:\WINDOWS\System32\WS2_32.dll
      2    0.0%          C:\WINDOWS\System32\KERNEL32.DLL

Sin console.log:

 [Shared libraries]:
   ticks  total  nonlib   name
   4074   68.7%          C:\WINDOWS\SYSTEM32\ntdll.dll
   1792   30.2%          C:\Program Files\nodejs\node.exe
      2    0.0%          C:\WINDOWS\System32\KERNELBASE.dll

Se puede apreciar que, con tan sólo un console.log el uso de node ha sido
un 3,5% mayor.

En las carpetas Diagrama y DiagramaConsole se pueden observar los
gráficos y, bajo mi punto de vista, destaca el proceso compression en
ambos.
```

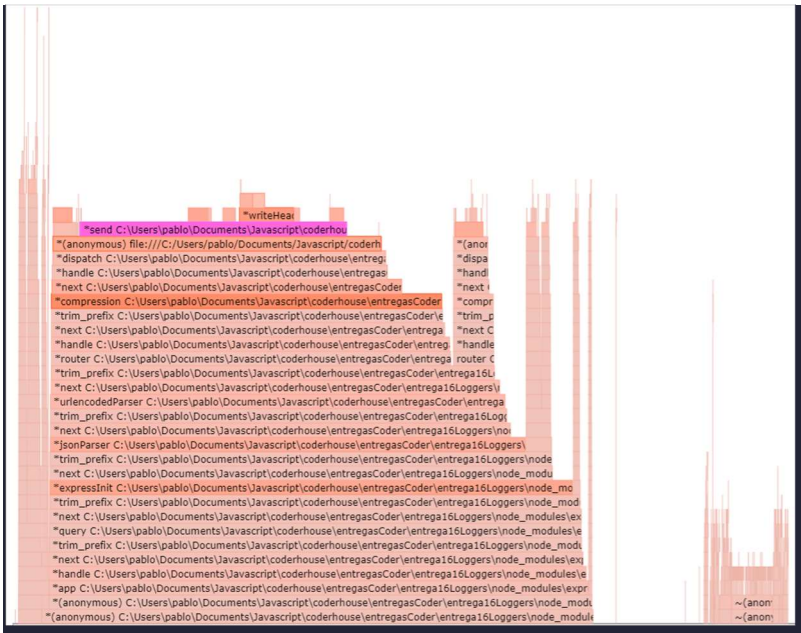


Diagrama de consola

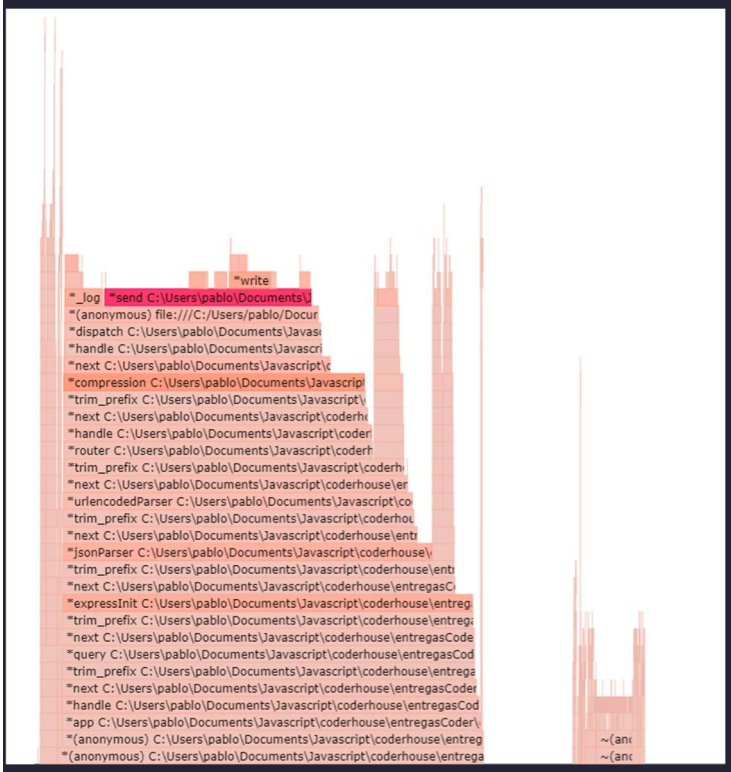
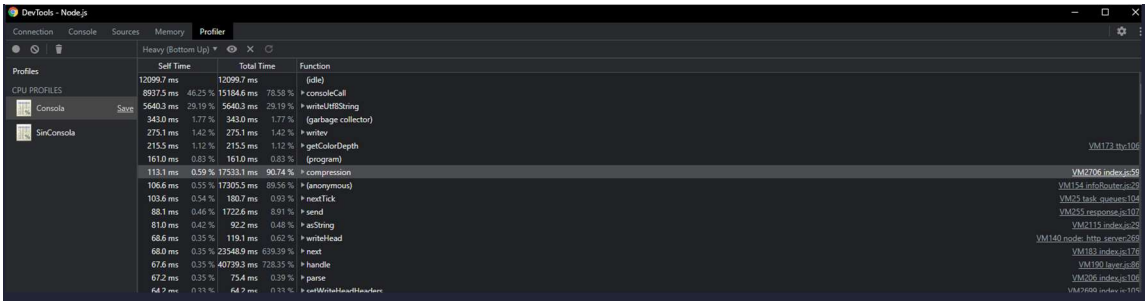


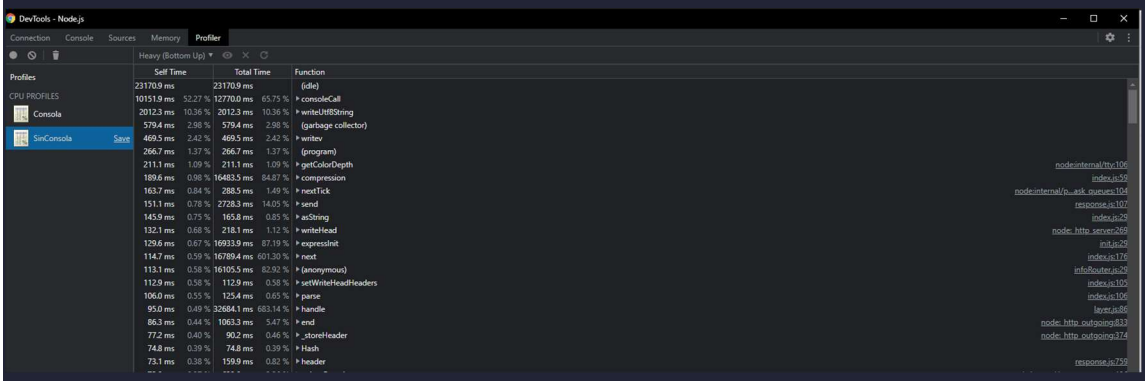
Diagrama sin consola

En ambos además se observa que el proceso "send" es un punto crítico, pero que el proceso es más corto sin la consola.

Usando el insect y las tolos de Chrome, vemos el perfilado es mucho más claro y la consola consume mucho más



Sin la consola:



Es decir, con consola la función consoleCall usa un 78,58% del tiempo y sin console.log, nos vamos a un 65,75%.