Paso a paso para correr el profiling de la aplicación

Precondiciones: tener instalado artillery, 0x y autocannon de manera global
npm i -g artillery
npm i -g autocannon
npm i -g 0x

- npm install
  Este comando es para instalar todas las dependencias del proyecto

- npm run build
  Este comando es para buildear la aplicación que está hecha con typescript y lo transpila a javascript

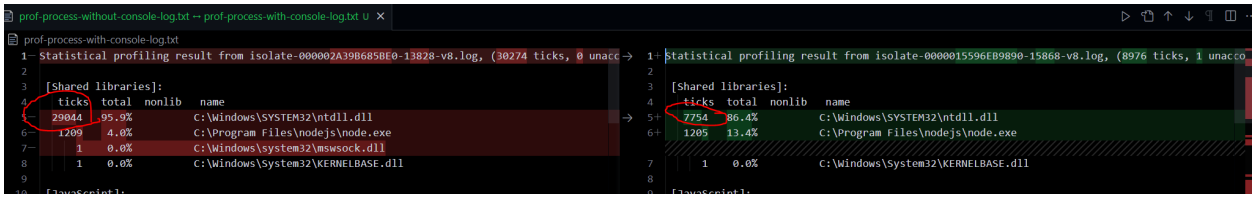
- npm run start:prof
  Este comando corre un "node --prof dist/index.js"
  La opción --prof permite ejecutar una aplicación Node.js con el modo de profiling activado.Esto genera información de perfil en el archivo "isolate-0x...-v8.log" en el directorio actual que puede ser procesada y analizada para ayudar a identificar cuellos de botella en el rendimiento de la aplicación.

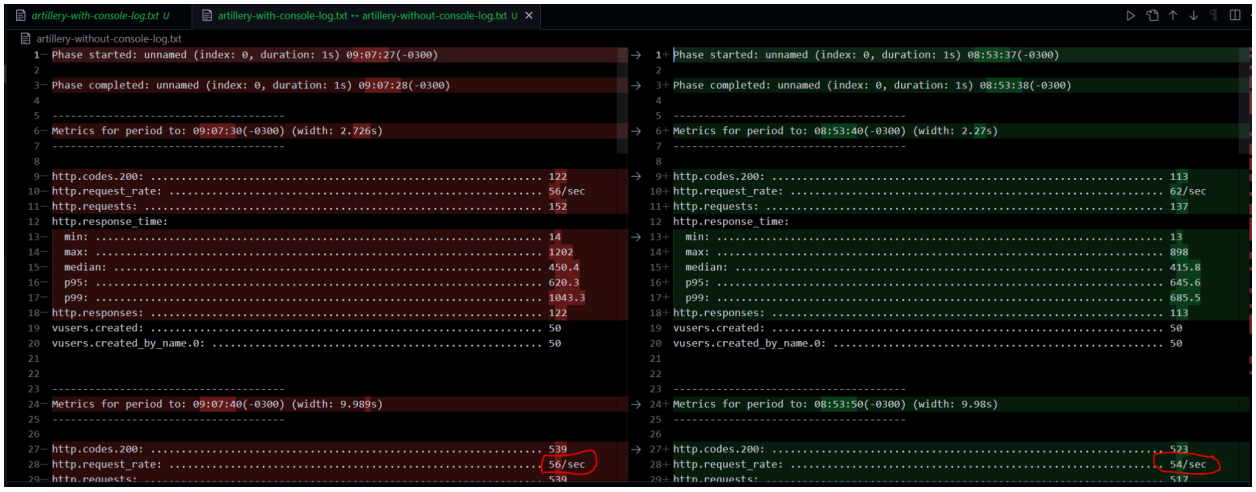
- npm run artillery-without-console-log
  Este comando corre "artillery quick --count 50 -n 20 http://localhost:8080/api/info > artillery-without-console-log.txt" para realizar un test con 50 conexiones concurrentes con 20 request por cada una y el resultado lo guarda en el archivo artillery-without-console-log.txt.

- El paso anterior genera un archivo isolate-000002A39B685BE0-13828-v8.log entonces lo que sigue hacer es procesarlo con el comando
  node --prof-process isolate-000002A39B685BE0-13828-v8.log > prof-process-without-console-log.txt

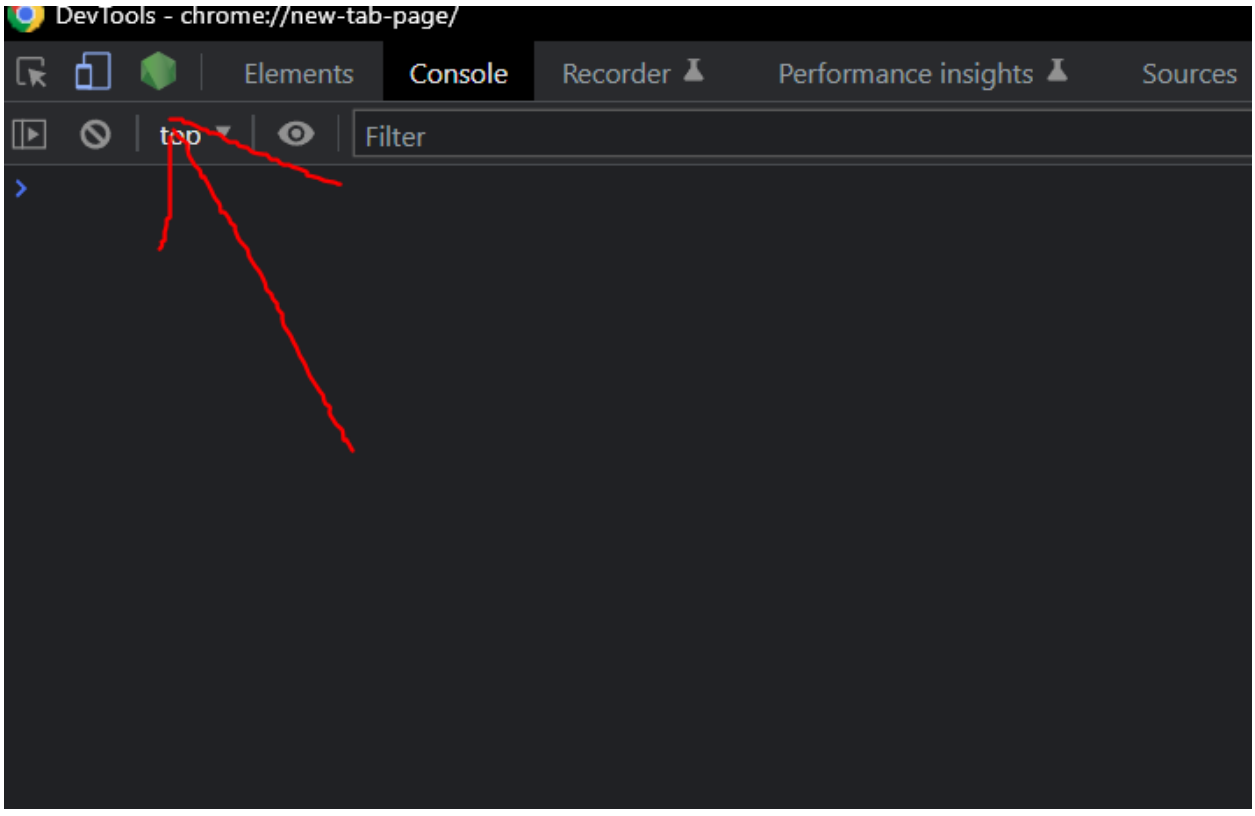
- Mato el proceso y repito a hacer los pasos anteriores pero con la version que loguea con console.log

1. npm run start:prof
2. npm run artillery-with-console-log
3. Mato el proceso
4. Me genera el archivo isolate-0000015596EB9890-15868-v8.log y lo proceso con
   node --prof-process isolate-0000015596EB9890-15868-v8.log > prof-process-with-console-log.txt

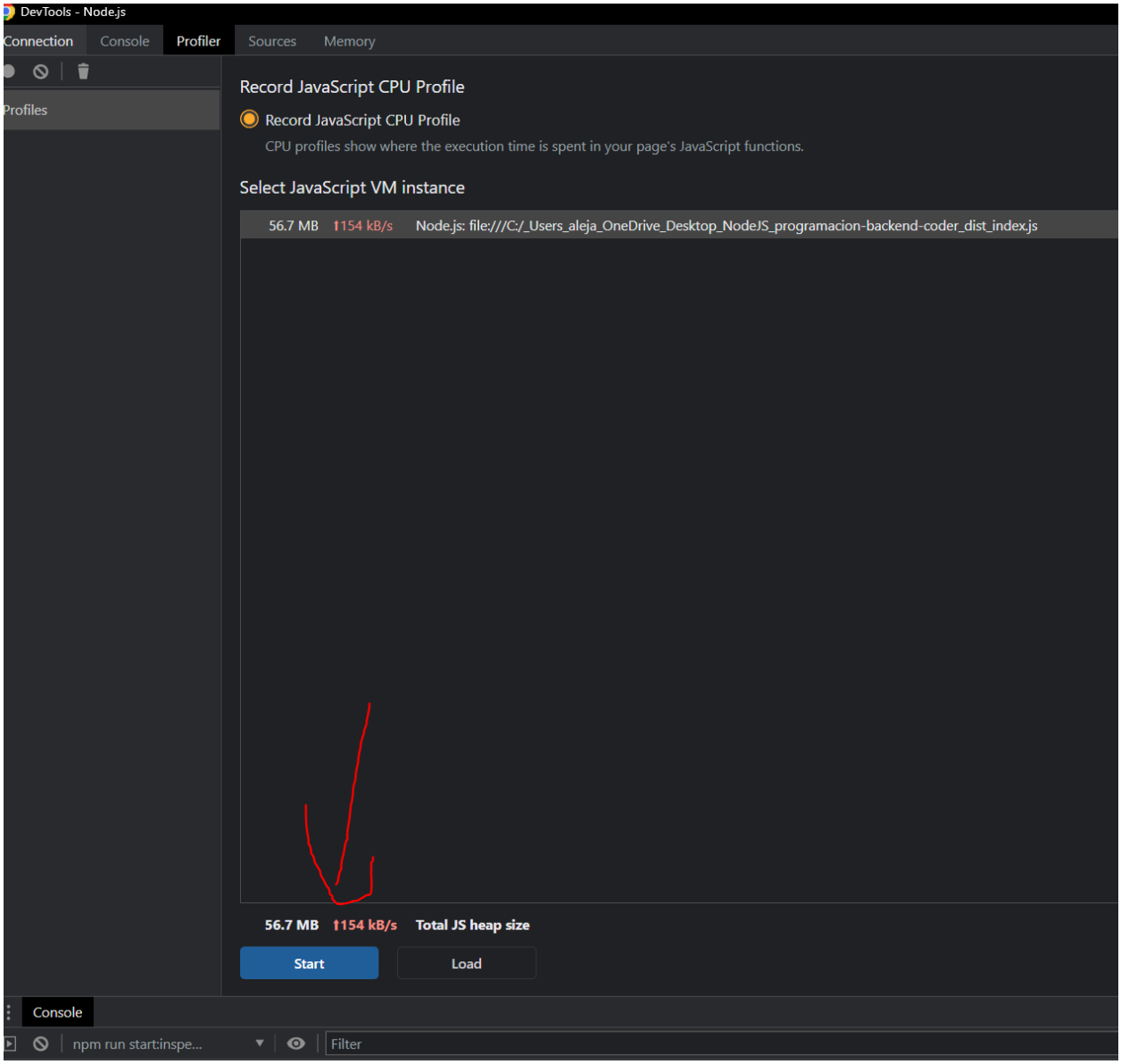




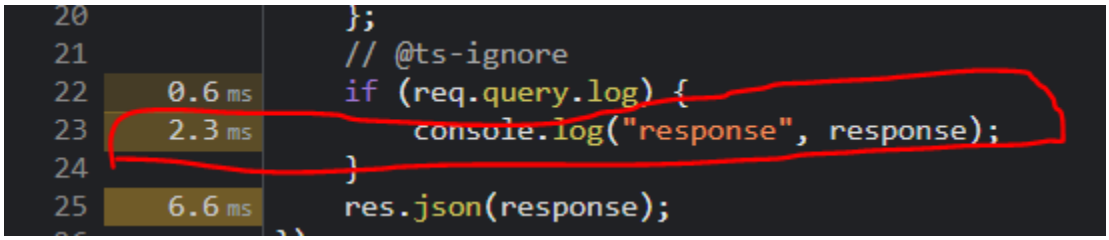
- npm run start:inspect
  este script ejecuta node --inspect dist/index.js para poder hacer el profile con chrome
  Para abrir en chrome abrir las dev tools y hacer click en el ícono verde

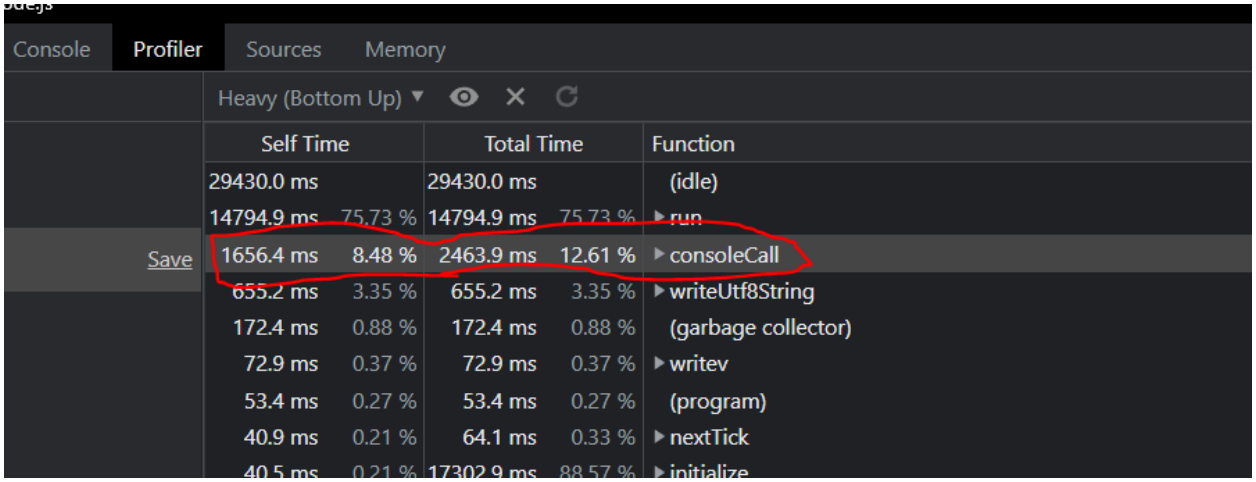


luego en start

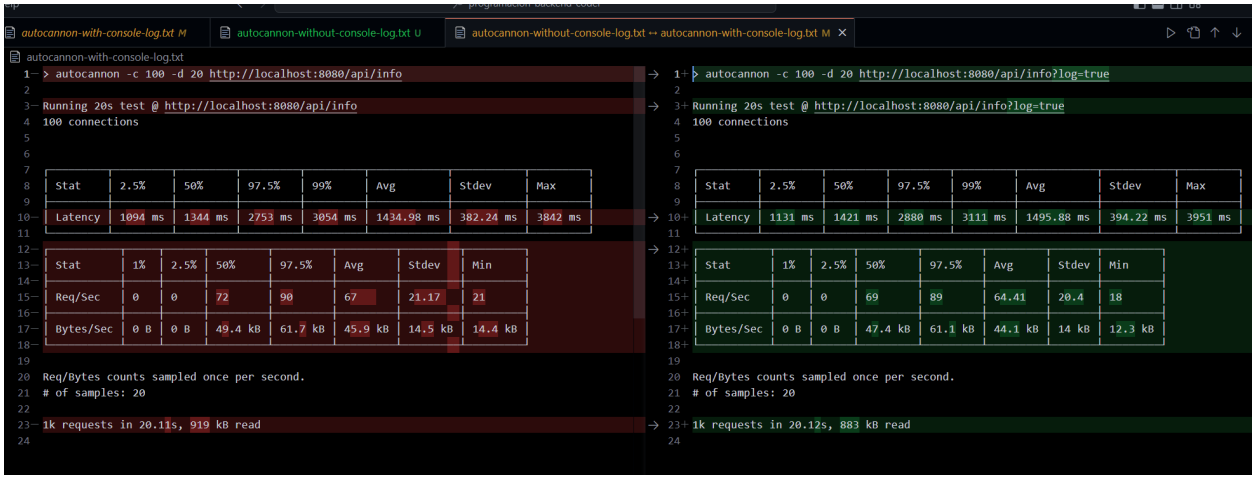


y ejecutar las pruebas con npm run autocannon-with-console-log que es un script que va a ejecutar autocannon -c 100 -d 20 http://localhost:8080/api/info?log=true que lo que va a hacer es testear con 100 hilos concurrentes durante 20 segundos
Luego darle a Stop y podemos observar lo que tarda el console log





- npm run start0x
  este script va a correr 0x dist/index.js
- npm run autocannon-without-console-log y npm run autocannon-with-console-log
  este script va a ejecutar autocannon



El gráfico de flema refleja la carga del CPU en el tiempo. El eje horizontal representa el tiempo mientras que el eje vertical representa el call stack. En los espacios en blanco el procesador estuvo libre.

Conclusión

La versión con console log resultó en 29044 ticks lo que representa más del triple de ticks comparado con los 7754 ticks de la versión sin console log. Cuantos más ticks tenga significa que el procesador estuvo más ocupado. Según los resultados de Artillery, la versión sin console log procesa 57 requests/segundo mientras que la versión con console log procesa 54 requests

por segundo lo que evidencia que es más eficiente la versión sin console log. Los resultados de Autocannon dan que la versión con console log tiene mayor latencia y el promedio de request por segundos da más bajo comparado con la versión sin console log.