**Clase 32. Logs, profiling & debug - Parte 2(Desafio)**

*ANÁLISIS COMPLETO DE PERFORMANCE*

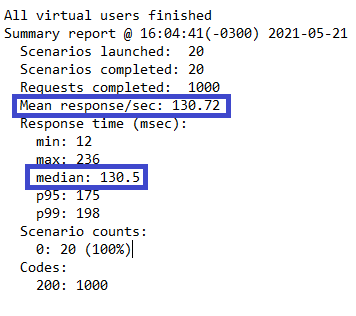
**Punto 1**

Arranque servidor

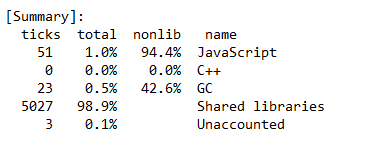
node --prof dist/index.js 12345 315974313260683 1ceb7ea6bd853655fc646f96fb6941cf FORK

**Con Console.log Activado**

artillery quick --count 20 -n 50 http://localhost:12345/info > result\_console.txt

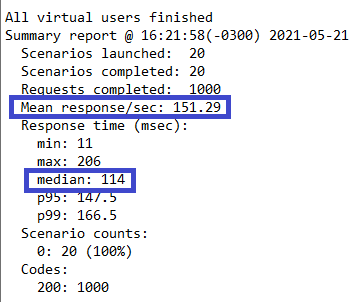


node --prof-process ConConsola-v8.log > result\_ConConsola.txt

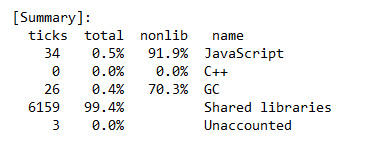


**Con Console.log Desactivado**

artillery quick --count 20 -n 50 http://localhost:12345/info > result\_sconsole.txt

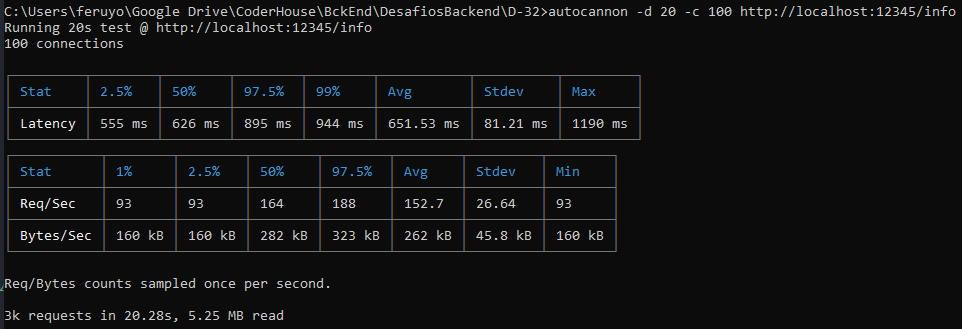


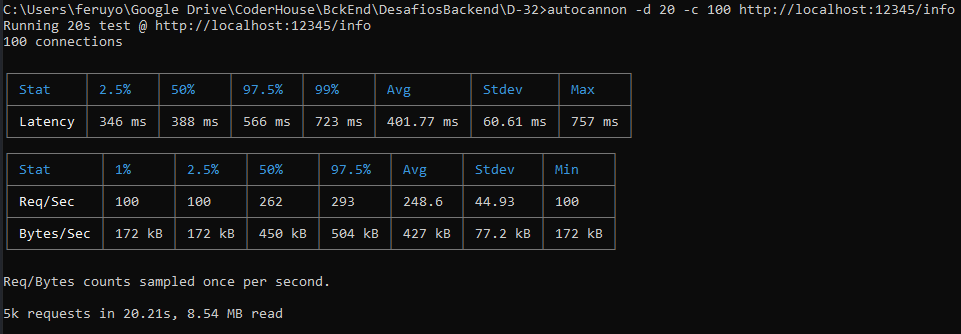
node --prof-process SinConsola-v8.log > result\_SinConsola.txt



**AUTOCANNON**

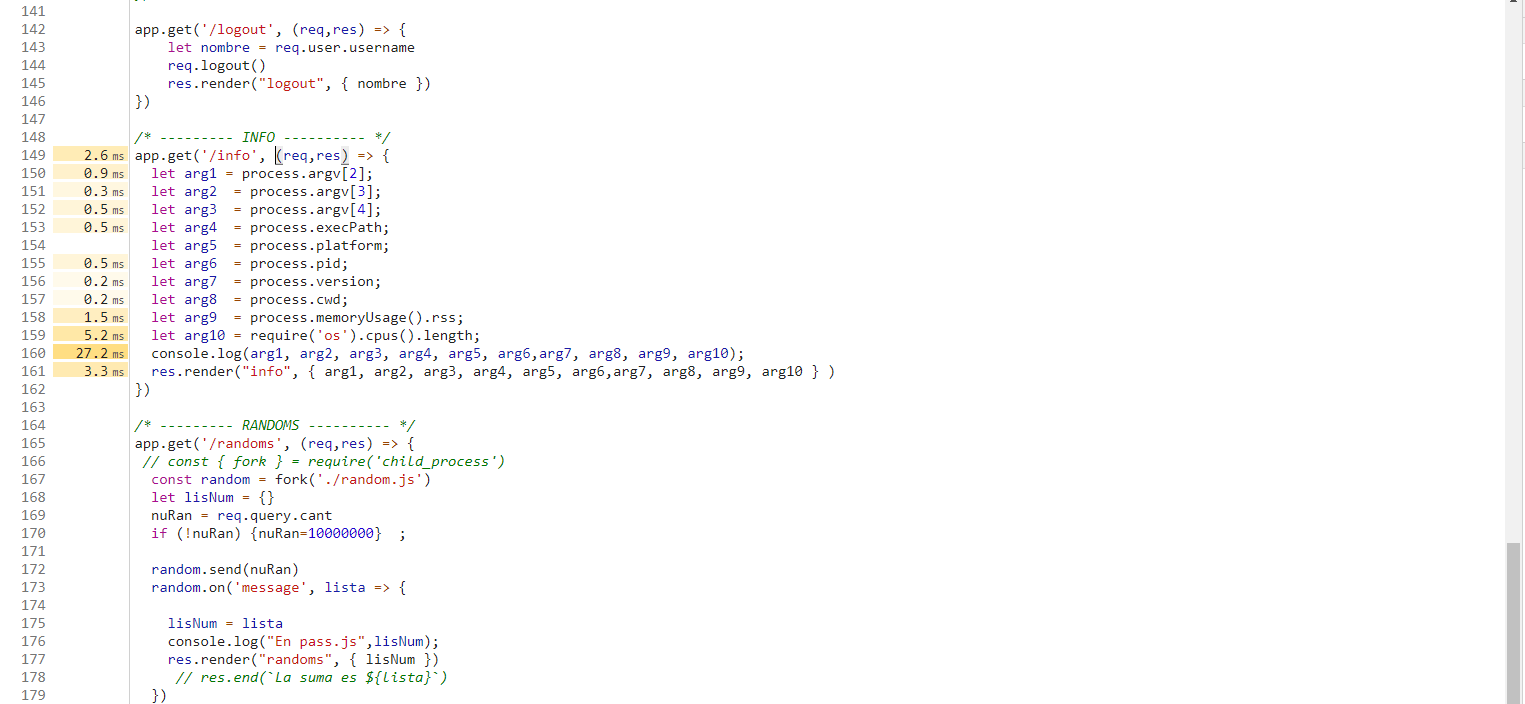
autocannon -d 20 -c 100 http://localhost:12345/info

**Con Console.log Activado**

**Con Console.log Desactivado** 

**Punto 2**

Perfilamiento del servidor con el modo inspector de node.js --inspect. Revisar el tiempo de los procesos menos performantes sobre el archivo fuente de inspección

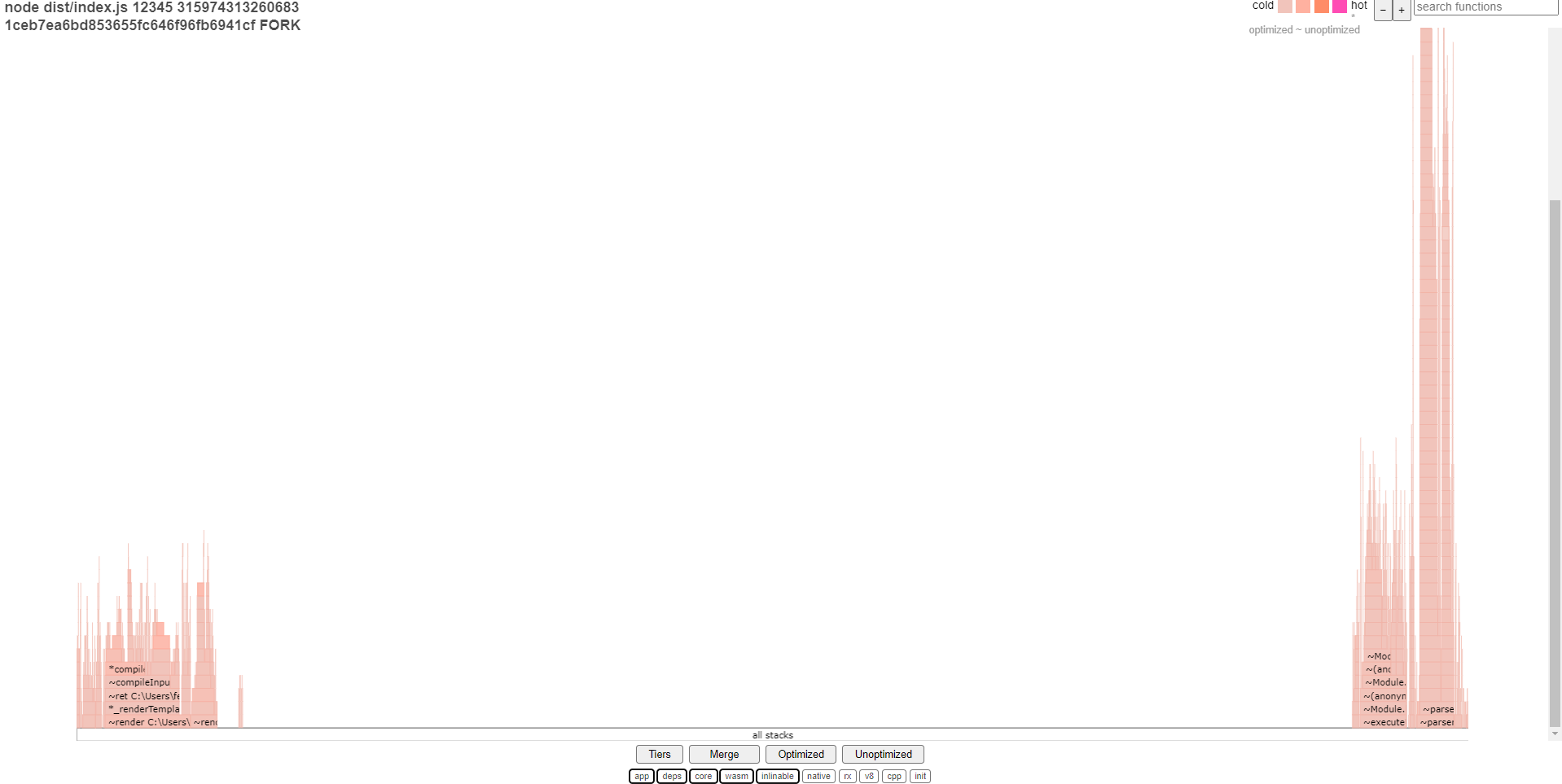


.

**Punto 3**

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

artillery quick --count 20 -n 50 http://localhost:12345/info



**Informe:**

Utilizando el estresando del servidor en Node.js mediante las dependencias de Artillery y Autocannon podemos ver en el punto 1 que es apreciable el incremento de la media de respuestas (Mean response/sec) por segundo en el caso de no usar el console.log de la ruta “/info”.

**Mean response/sec:**

Con console.log 130.72

Sin console.log 151.29

Los milisegundos de latencia (Response time, median) es más alto en el caso de usar el console.log de la ruta “/info”.

**Response time (msec):**

Con console.log 130.5

Sin console.log 114

En el punto 2, observando el perfilamiento del servidor con el modo inspector de node.js –inspect, podemos ver que el tiempo que tarda la instrucción “console.log” de la ruta “/info” es por lejos el mayor tiempo empleado por las instrucciones del servidor( 27,2 ms).

En el punto 3 se puede observar a la izquierda los procesos de render de “Handlebars”, que tienen una magnitud de tiempo empleado comparativa similar a los procesos generados por Artillery, a la izquerda.

Los procesos de Artillery son 50 conexiones concurrentes con 20 request por cada una y “Handlebars” era una sola conexión, lo que nos dice del gran tiempo que nos toma el renderizado de una página.

Aparte de eso, la curva formada por Artillery es bastante optima, ya que para serlo tiene que tener la forma de una aguja, alta pero de poca base(tiempo).