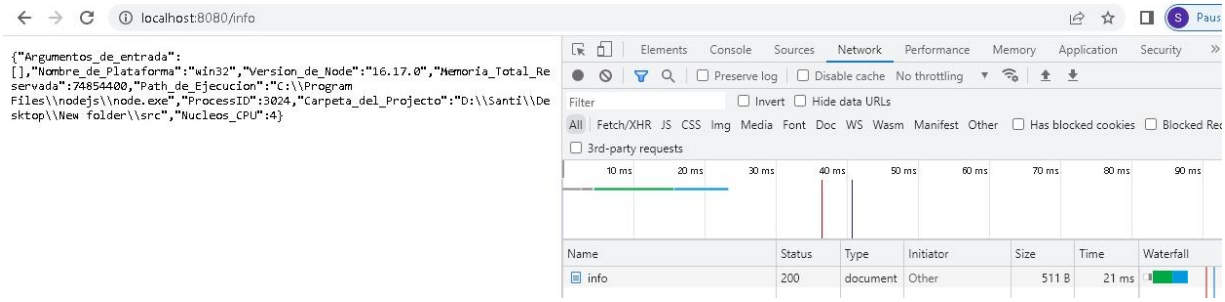
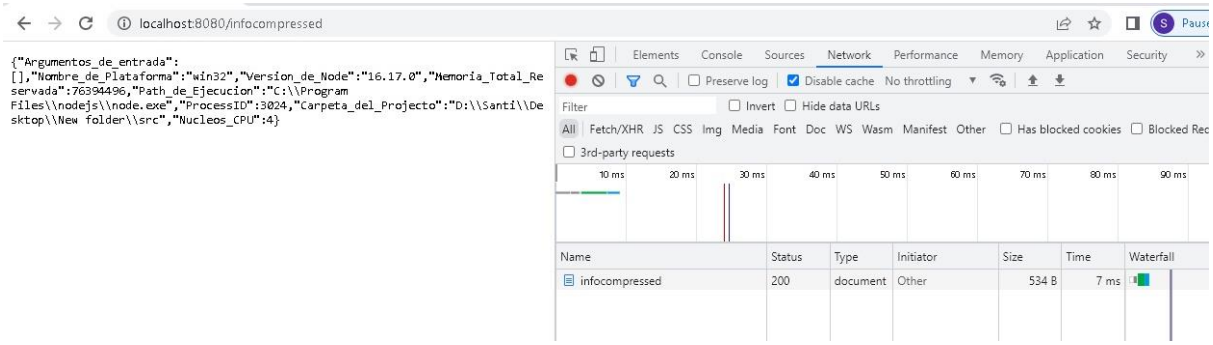
Diferencia de tamaño y tiempo de respuesta en el perfomance de la aplicación.

Compression

Sin comprimir (511B)



Comprimida (534B)



Artillery

Síncrono (6052)

```
[Summary]:
  ticks   total   nonlib   name
     16    0.3%   94.1%  JavaScript
      0    0.0%    0.0%  C++
     21    0.3%  123.5%  GC
   6052   99.7%          Shared libraries
      1    0.0%          Unaccounted
```

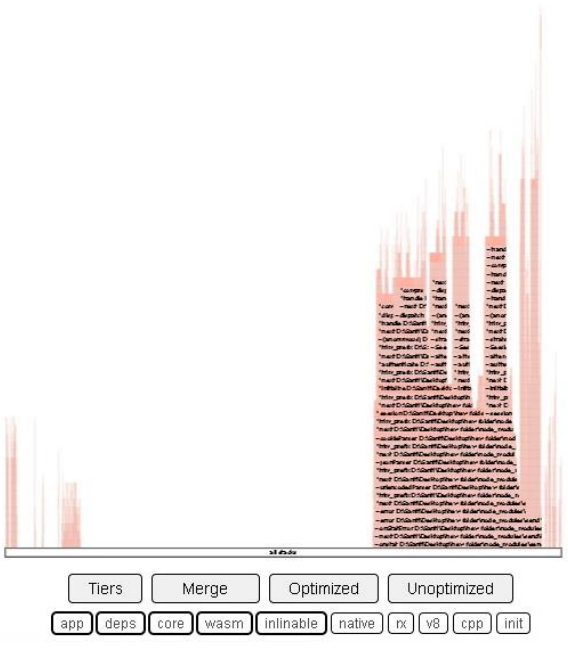
Asíncrono (39347)

```
[Summary]:
  ticks   total   nonlib   name
     21    0.1%  100.0%  JavaScript
      0    0.0%    0.0%  C++
     17    0.0%   81.0%  GC
  39347   99.9%          Shared libraries
```

Autocannon

Node Inspect

0x



Resultados

Se puede observar que, a pesar de la implementación de métodos de compresión y funciones asíncronas, la ruta /info solo ganaba un pequeño porcentaje de peso, tiempo de respuesta o cantidad de procesos para realizar la tarea.

Con la implementación de compression, esta ruta paso de 511B a tener 534B, es decir, aumento un %4.5.

Con la implementación de una función síncrona como lo es console.log(), la ruta paso de usar 6052 procesos para la misma tarea, a usar 39247. Hubo un aumento del %548.5.

En cuanto a tiempo de respuesta evaluado con autocannon y node inspect, se observa que la ruta con procesos síncronos respondía entre los 46 y 48.9 milisegundos, mientras que la asíncrona lo hacia entre los 46.2 y 52.2 milisegundos. Esto equivale a un aumento del %0.4 al %6.7.

La única medición que se noto una mejora al aplicar un proceso asíncrono fue en la realizada con 0x.

En conclusión, estos datos parecen totalmente contradictorios debido a que una ruta sin comprimir y con métodos bloqueantes debería ser mucho más rápida y ligera que una ruta asíncrona y comprimida, pero en este caso la única justificación posible es que de por si las tareas que se realizan en /info son tan básicas que no se aprecia un cambio en la eficiencia de su función. Distinto podría ser si se probara en una ruta como login donde si hay un proceso más complejo y pesado.