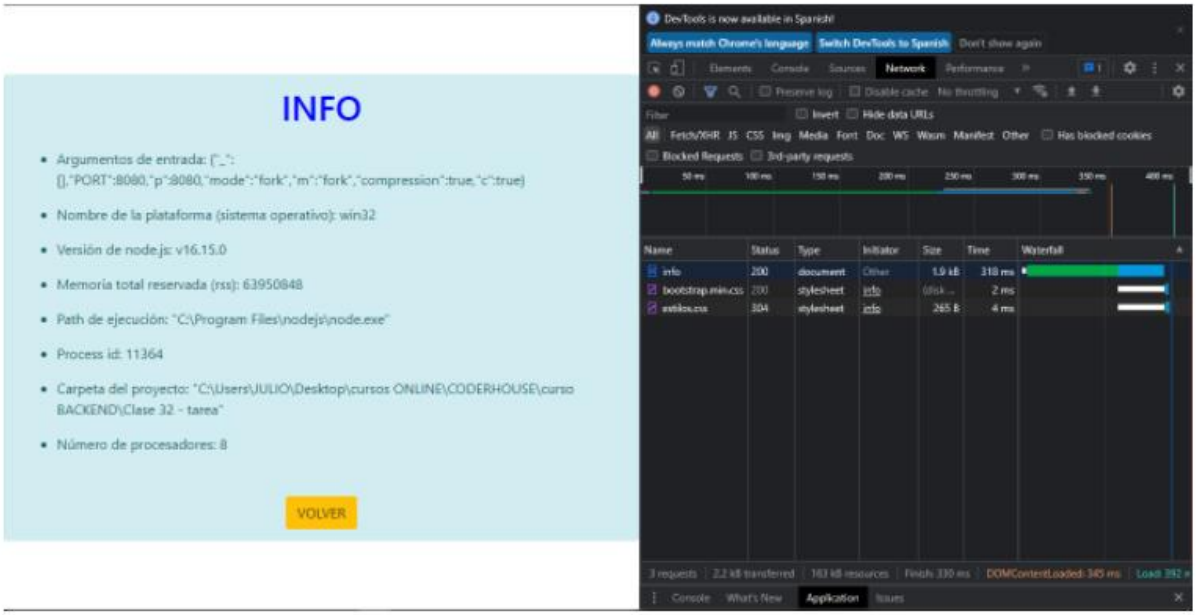
Consigna "LOGGERS Y GZIP":

- Incorporar al proyecto de servidor de trabajo la compresión gzip.
  > ubicación: "./src/routers/app.routes.js"

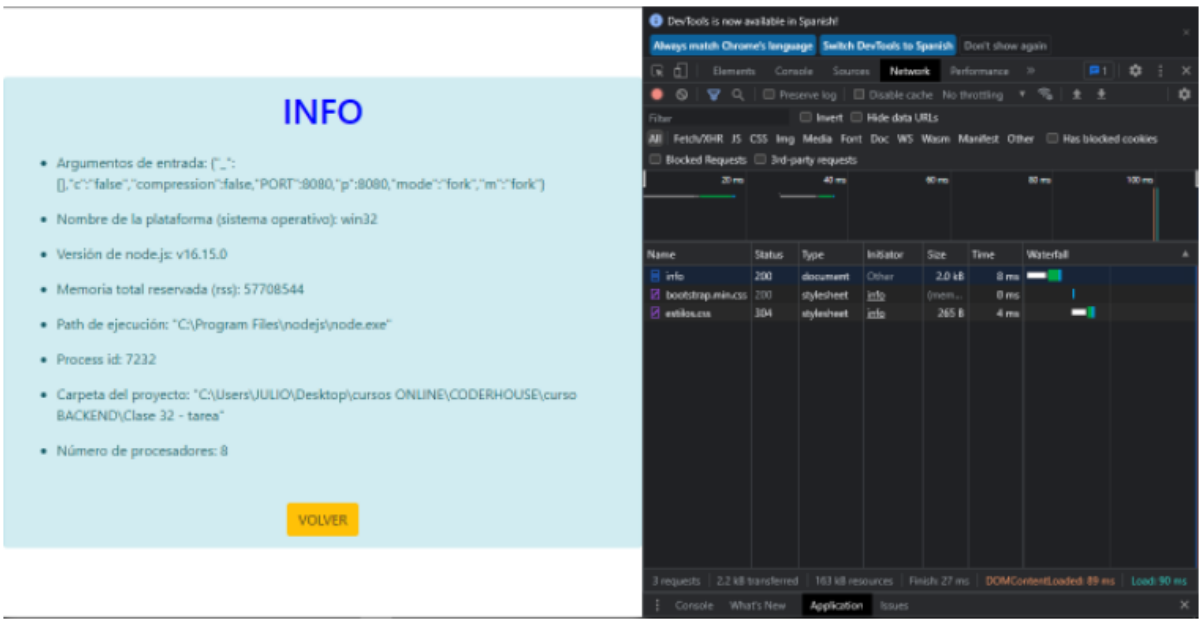
- Verificar sobre la ruta /info con y sin compresión, la diferencia de cantidad de bytes devueltos en un caso y otro.
  - con compresión:

```
npm start
```



  - sin compresión:

```
npm start -- -c false
```



La diferencia es de 0.1kB

- Luego implementar loggueo (con alguna librería vista en clase) que registre lo siguiente:
  - Ruta y método de todas las peticiones recibidas por el servidor (info):
    > ubicación: "./middlewares/loggerInfo.js"
  - Ruta y método de las peticiones a rutas inexistentes en el servidor (warning):
    > ubicación: "./controllers/routes.controller.js"
  - Errores lanzados por las apis de mensajes y productos, únicamente (error):
    > ubicación: "./models/containers" y "./models/daos"

Consigna "ANÁLISIS COMPLETO DE PERFORMANCE":

- Vamos a trabajar sobre la ruta '/info', en modo fork, agregando ó extrayendo un console.log de la información colectada antes de devolverla al cliente. Además desactivaremos el child_process de la ruta '/randoms'. Para ambas condiciones (con o sin console.log) en la ruta '/info' OBTENER:
  - El perfilamiento del servidor, realizando el test con --prof de node.js. Analizar los resultados obtenidos luego de procesarlos con --prof-process.
    > ubicación: "./result-info.txt" y "./result-process-info.txt"
  - Utilizaremos como test de carga Artillery en línea de comandos, emulando 50 conexiones concurrentes con 20 request por cada una. Extraer un reporte con los resultados en archivo de texto.
    > ubicación: "./result-artillery.txt"

- Luego utilizaremos Autocannon en línea de comandos, emulando 100 conexiones concurrentes realizadas en un tiempo de 20 segundos. Extraer un reporte con los resultados (puede ser un print screen de la consola):

```
C:\Users\JULIO\Desktop\cursos ONLINE\CODERHOUSE\curso BACKEND\Clase 32 - tarea>node src/benchmark.js
Running all benchmarks in parallel ...
Running 20s test @ http://localhost:8080/info
100 connections


| Stat    | 2.5%   | 50%    | 97.5%   | 99%     | Avg        | Stdev    | Max     |
| Latency | 852 ms | 991 ms | 1467 ms | 1881 ms | 1021.21 ms | 184.7 ms | 2699 ms |

| Stat      | 1%      | 2.5%    | 50%     | 97.5%   | Avg     | Stdev    | Min     |
| Req/Sec   | 15      | 15      | 100     | 115     | 96.3    | 20.82    | 15      |
| Bytes/Sec | 28.3 kB | 28.3 kB | 189 kB  | 217 kB  | 182 kB  | 39.3 kB  | 28.3 kB |

Req/Bytes counts sampled once per second.
# of samples: 20

2k requests in 20.44s, 3.64 MB read
```

- El 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.

routes.controller.js

```
1        import env from '../src/env.config.js';
2        import { args } from '../src/index.js';
3        import os from 'os';
4        import { warnLogger } from "../utils/logger.utils.js"
5        import { STATUS } from '../constants/api.constants.js';
6        const { NOT_FOUND } = STATUS;
7
8        const getInfo = (req, res) => {
9    1.0 ms  const renderInfo = {
10  25.0 ms    inputArguments: JSON.stringify(args),
11   1.4 ms    platformName: process.platform,
12   0.8 ms    versionNode: process.version,
13   2.4 ms    rss: process.memoryUsage().rss,
14   1.7 ms    path: `"${process.argv[0]}"`,
15   0.5 ms    processId: process.pid,
16   1.2 ms    projectFolder: `"${process.cwd()}"`,
17   4.3 ms    numOfProcessors: os.cpus().length
18         };
19
20         // ↓↓↓ comentar/descomentar según pida la consigna ↓↓↓
21             // console.log(renderInfo);
22         // ↑↑↑ comentar/descomentar según pida la consigna ↑↑↑
23
24  21.6 ms  res.render('info', renderInfo);
25       }
26       const getHome = (req, res) => {
27         const user = req.user.email;
28         res.render('home', { nombre: user });
29       }
30       const getLogout = (req, res) => {
31         const user = req.user.email;
32         req.logout();
33         req.session.destroy(err => {
34           if(err) res.clearCookie(env.SESSION_NAME);
35           res.render("logout", { nombre: user });
36         });
37         console.log(`User ${user} logued out!`);
38         res.clearCookie(env.SESSION_NAME);
39       }
40       const getError = (req, res, page) => {
41         const message = `USER ERROR ${page == "register-error" ? "SIGNUP" : "LOGIN"}`;
```

{} Line 8, Column 17 Coverage: n/a

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