Sesión # 16 Componente Práctico

Introducción a DevOps

En esta sesión vamos añadir un monitor de estado a nuestra aplicación Express para tener un seguimiento del estado y consumo de recursos de está misma cada vez que se realice una petición. Lastimosamente, Express no proporciona una forma de rastrear el estado de nuestra aplicación de forma predeterminada. Sin embargo, podemos agregar un paquete para hacer esto.

A continuación veremos el paquete express-status-monitor y cómo usarlo. Lo importante aquí es cómo monitorear los procesos y el uso de NodeJs. Instalalo en nuestro proyecto de la sesión anterior y mira que pasa.

1) Crea un nuevo proyecto/carpeta para tu solución
2) Accede desde la terminal a la ubicación del proyecto y ejecuta
   `npm init -y`
3) Como puedes observar se ha generado tu archivo package.json
4) Ahora instala las dependencias a utilizar:
   ```
   npm i -g express-generator
   npm i express --no-view --git . (presiona y)
   ```

   Este último comando nos ayudará a crear la estructura de nuestro proyecto de solución.

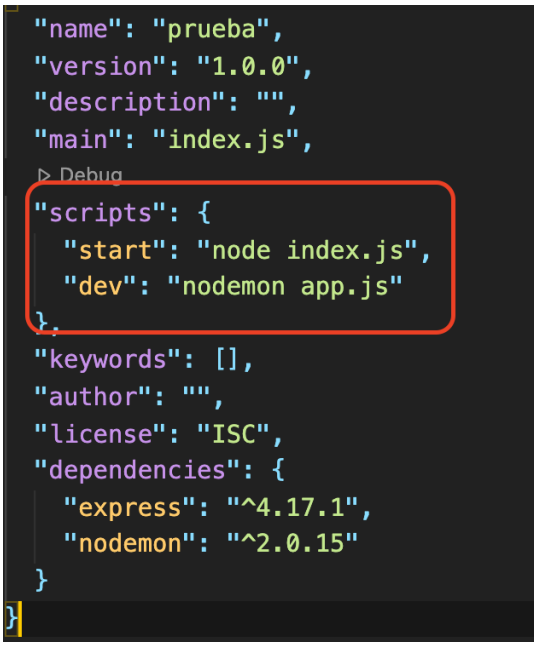
5) Crea una carpeta llamada routes y dentro dos archivos llamados index.js y users.js
6) En la raíz del proyecto crea un nuevo archivo llamado app.js y añade el siguiente código:

```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");

var indexRouter = require("./routes/index");
var usersRouter = require("./routes/users");
var app = express();
```

```
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));
app.use("/", indexRouter);
app.use("/users", usersRouter);

app.listen(3000, () => {
  console.log("Conectado al puerto 3000");
});
```

7) Guarda todo y ejecuta el comando `npm i -D nodemon` esto permitirá que nuestro servidor se actualice cada vez que hagamos cambios sobre él.

8) Modifica tu package.json y añade lo siguiente. Eso permitirá ejecutar el proyecto posteriormente.



9) Instala estas otras dependencias:

```
npm i cookie-parser
```

Ten en cuenta que express-status-monitor está disponible como paquete Node. Podemos instalarlo ejecutando:

```
npm i express-status-monitor
```

Eso nos ayudará a medir el rendimiento de nuestra aplicación.

10) Ahora con la configuración lista, implementemos una petición GET sencilla donde se obtenga un objeto de usuarios. En el archivo users.js añade el siguiente código:

```
var express = require("express");
var router = express.Router();
const users = [
  { id: 0, username: "maria", url: "wedeycode.com" },
  { id: 1, username: "juanito", url: "https://google.com" },
  { id: 2, username: "esteban", url: "http://facebook.com" },
  { id: 3, username: "amina", url: "www.wedeycode.com" },
];
/* GET users listing. */
router.get("/", function (req, res) {
  res.status(200).json(users);
});

module.exports = router;
```

11) En el archivo index.js añade el siguiente código:

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

12) Ahora, para correr el proyecto ejecuta desde terminal `npm i` y posteriormente `npm run dev`

```
❯ npm run dev

> prueba@1.0.0 dev /Users/kristell/Documents/prueba
> nodemon app.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Conectado al puerto 3000
```

13) En este punto ya contamos con una petición y con la dependencia express-status-monitor por lo que podemos proceder a utilizarlo para medir el rendimiento. Para ello, ve al archivo app.js y antes de cualquier otro middleware o router añade la siguiente línea:

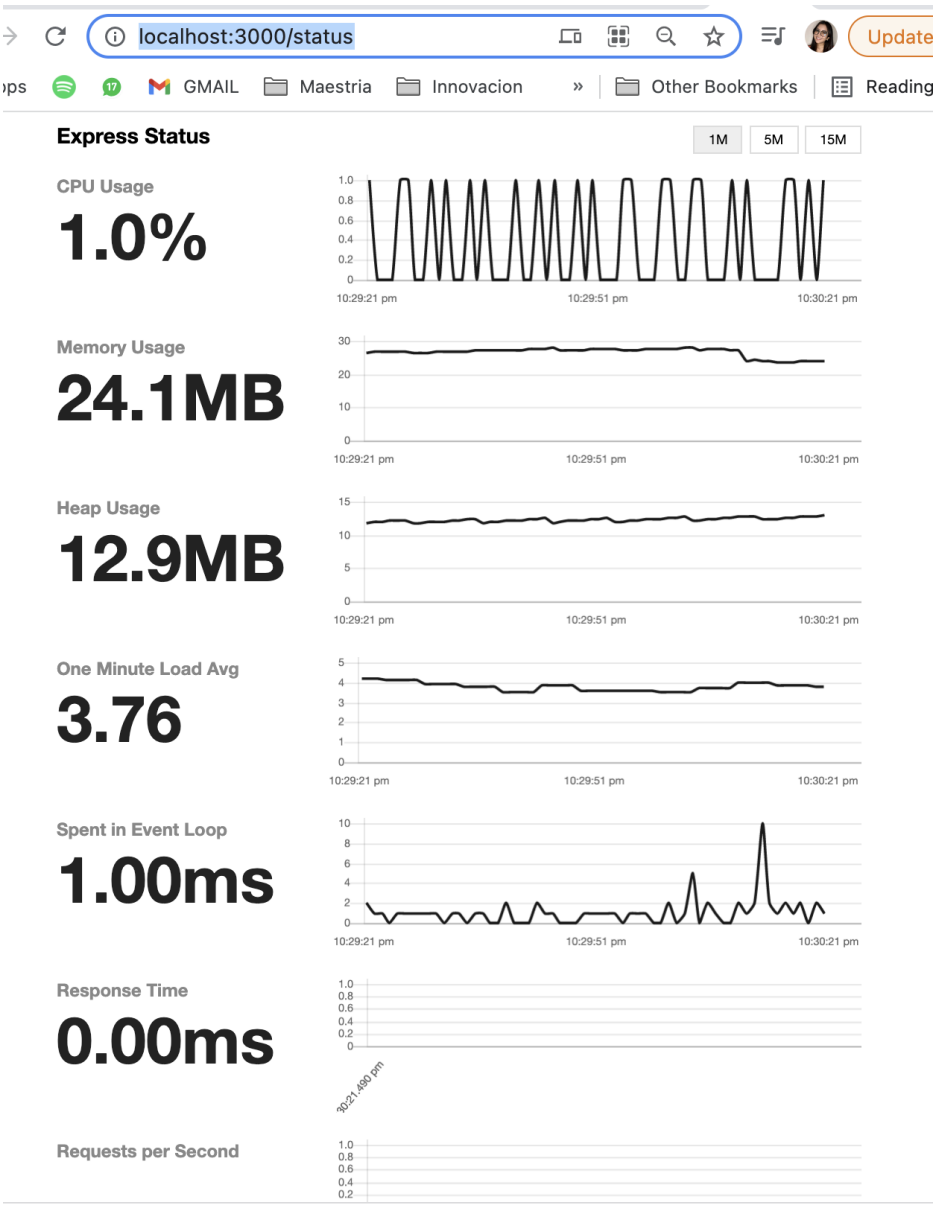
```
app.use(require('express-status-monitor')());
```

al final nuestro archivo debe quedar:

```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");

var indexRouter = require("./routes/index");
var usersRouter = require("./routes/users");
const expressStatusmonitor =
require("express-status-monitor");

var app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));
app.use(expressStatusmonitor());
```

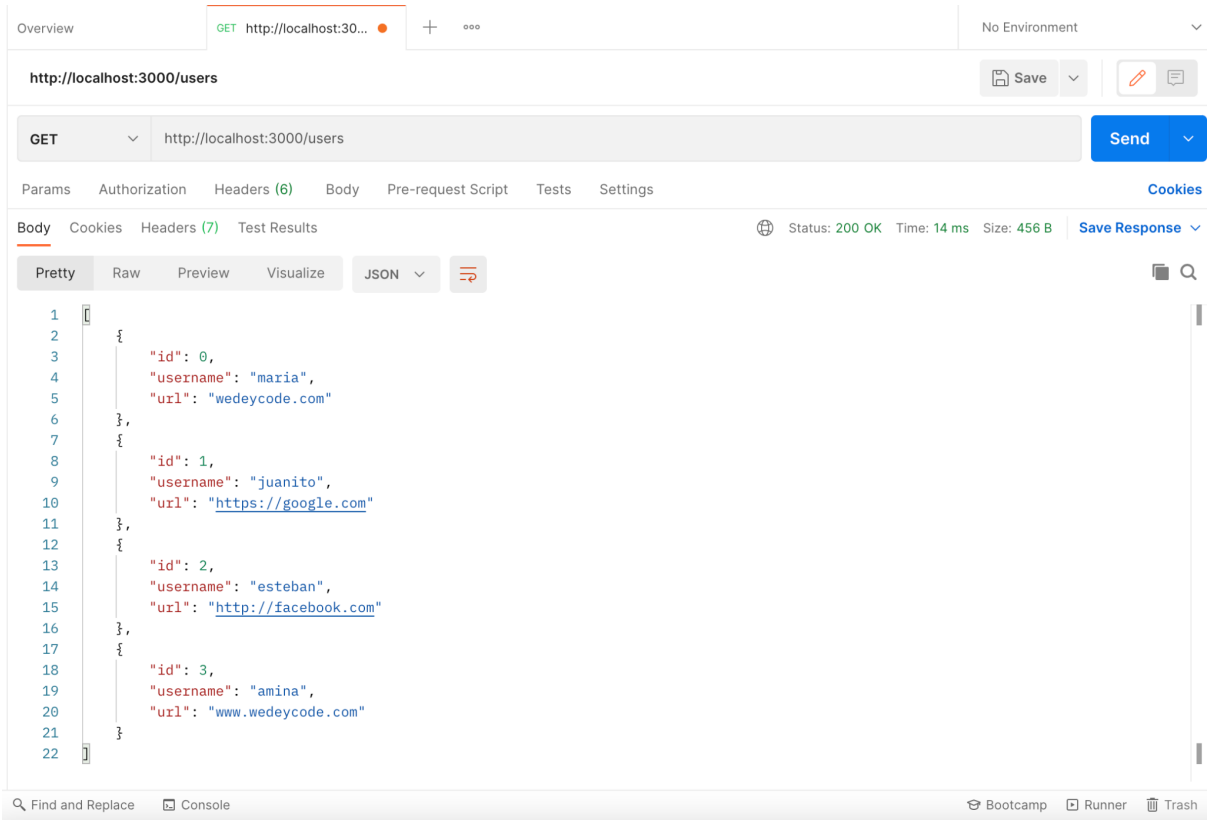
```
app.use("/", indexRouter);
app.use("/users", usersRouter);

app.listen(3000, () => {
  console.log("Conectado al puerto 3000");
});
```

14) Ahora, para probar el monitor de estado inicial reinicia tu servidor (npm run dev) y coloca el siguiente link en el navegador http://localhost:3000/status y verás algo similar a esto:

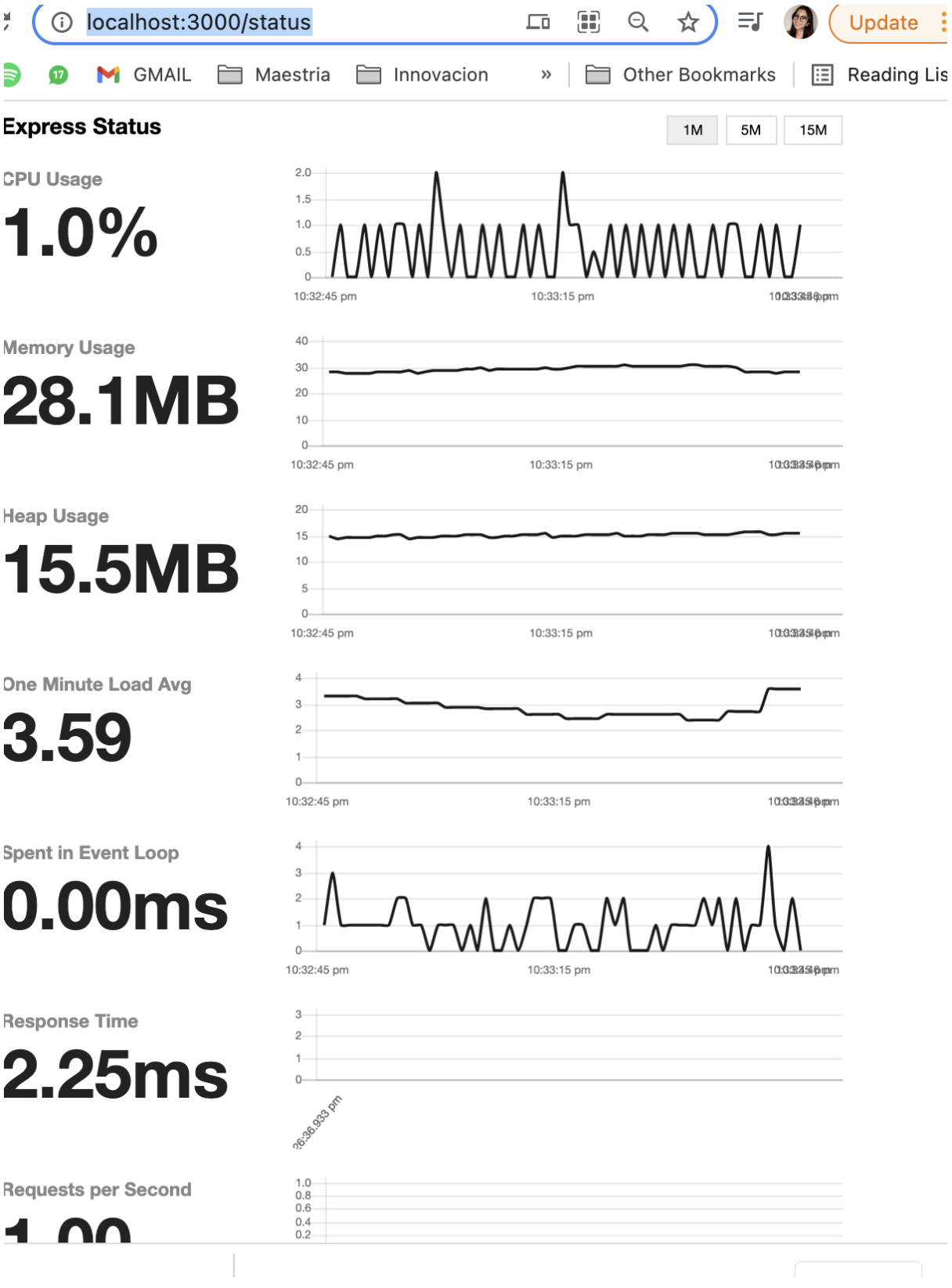


Como puedes notar, en nuestro proyecto no existe como tal ninguna vista llamada status y esta palabra sólo aparece en importaciones, todo esto es proporcionado por la dependencia.

15) Ahora, creemos un poco de tráfico en nuestro servidor. Usando postman, crea una nueva request de tipo GET para obtener los usuarios:



16) Presiona Send y analiza los resultados.



Como puedes ver obtuvimos un pico en nuestra gráfica lo que indica cambios y tráfico o movimiento en nuestra aplicación.

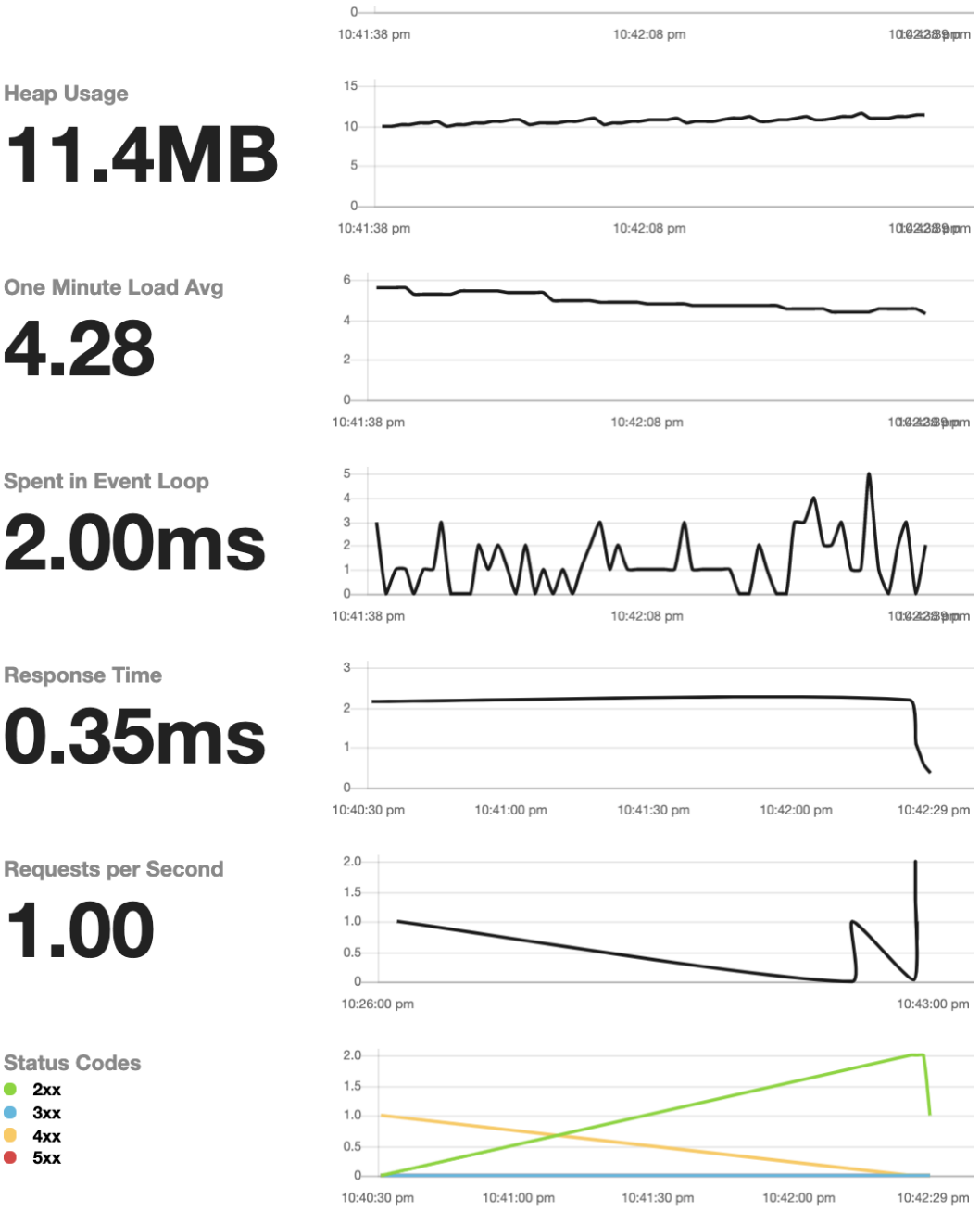
17) Esta librería nos otorga una serie de controles de salud que aparecen debajo de otras estadísticas. Ten en cuenta que una verificación de estado es exitosa si devuelve un código de estado 200.

18) Una vez probado que el template funcione , añadamos en app.js una serie de comprobaciones de salud que aparecerán debajo de otras estadísticas.

```
// config
healthChecks: [{
  protocol: 'http',
  host: 'localhost',
  path: '/admin/health/ex1',
  port: '3000'
}, {
  protocol: 'http',
  host: 'localhost',
  path: '/admin/health/ex2',
  port: '3000'
}]
```

Añadimos las líneas de código (corresponden a las comprobaciones de salud) antes de **app.use(expressStatusmonitor());**

19) Ahora guarda tus cambios y envía repetitivamente varias peticiones desde postman (presionando send en la petición creada en el paso 15) y analiza el status medido.



20) Listo! Hemos creado un tablero para medir el rendimiento de nuestra app. Pero no te quedes solo con esto, investiga qué otras herramientas existen para monitorear el rendimiento de un servidor en node.

Documentación: https://github.com/RafalWilinski/express-status-monitor
Video guia: https://www.youtube.com/watch?v=LwgiZxFr4lM
Solución: https://github.com/Misiontic-Ciclo-4A/sesion16-solucion