

# Curso de Programación Backend. - CODERHOUSE

## (Comisión 32195)

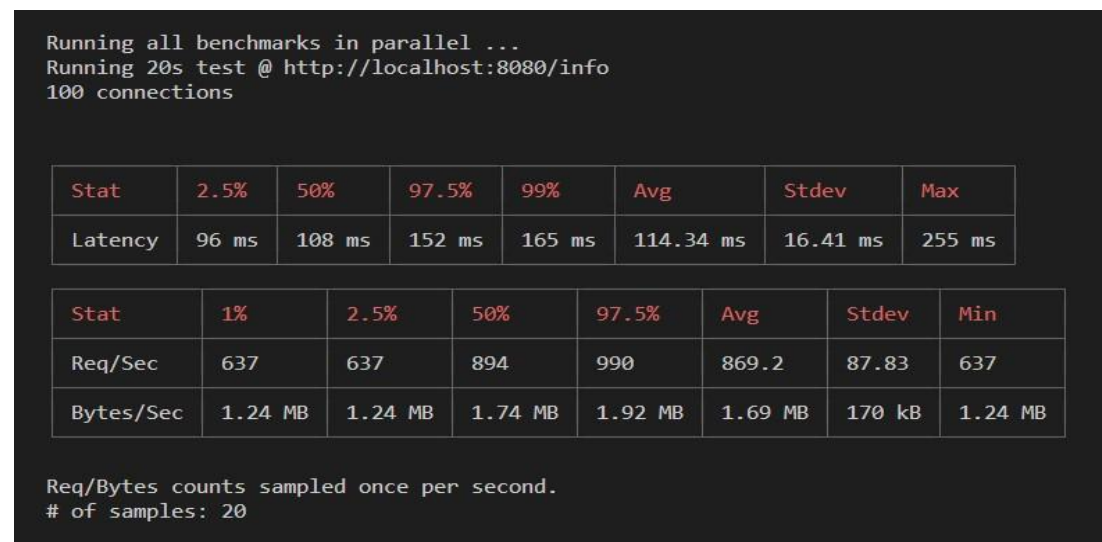
Alumno: Belén Schmid

Desafío Clase 32: Conclusión obtenida a partir de análisis de datos.

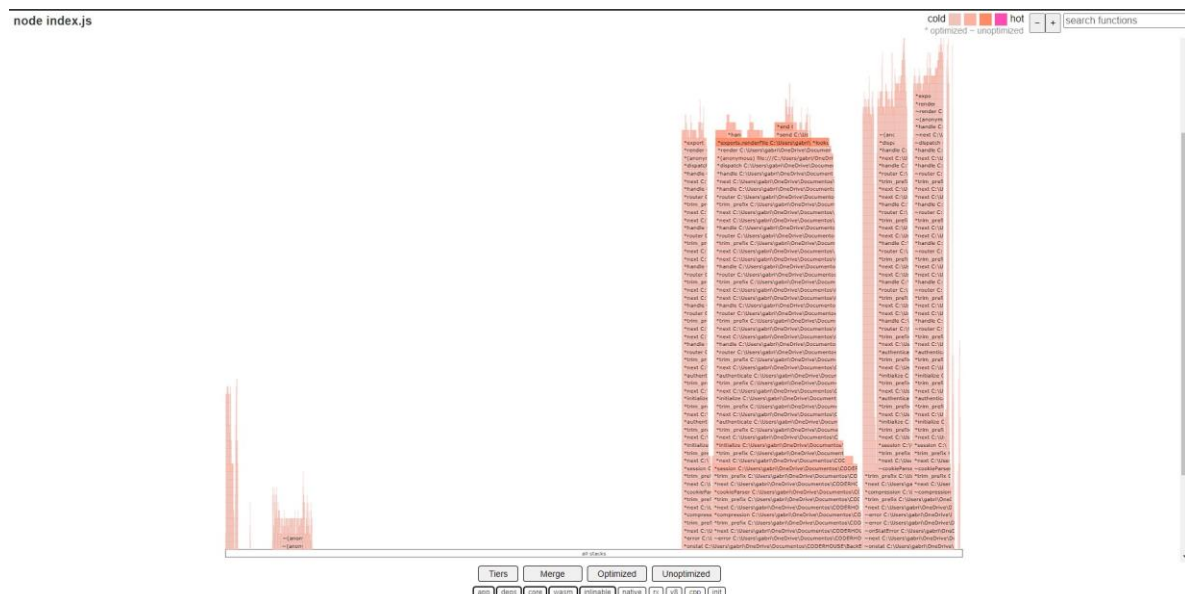
### CONCLUSIÓN

A continuación, conforme lo solicitado en el desafío, presento los print de pantalla con las pruebas realizadas. Las mismas fueron hechas a través del Modo Fork.

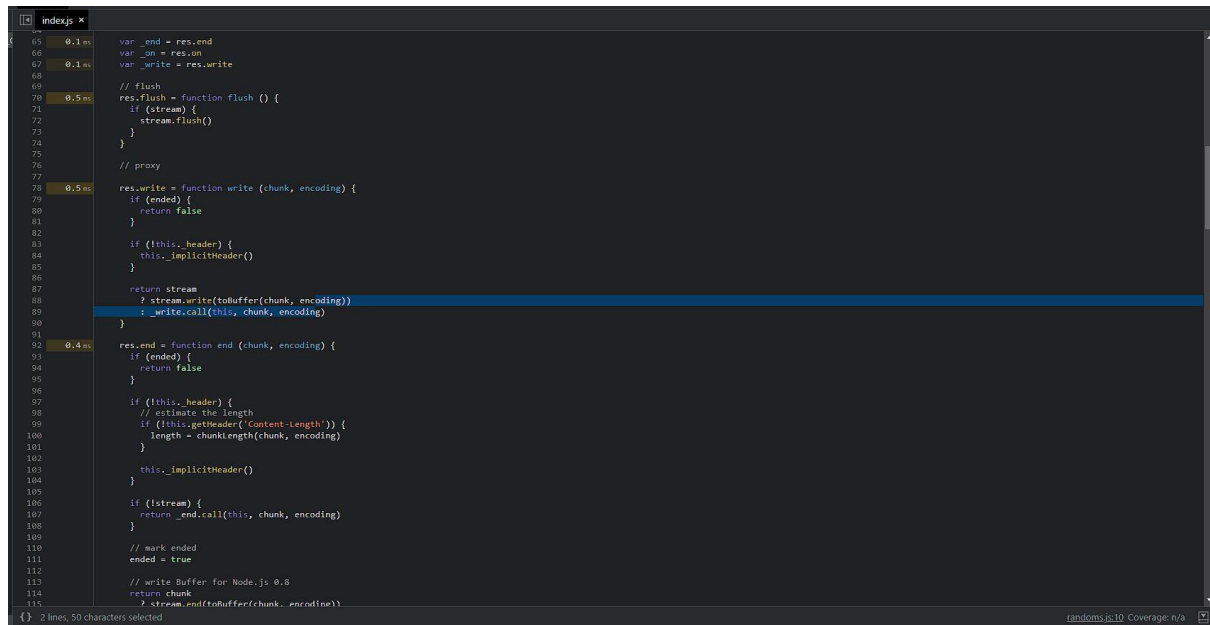
### RESULTADOS DE AUTOCANNON



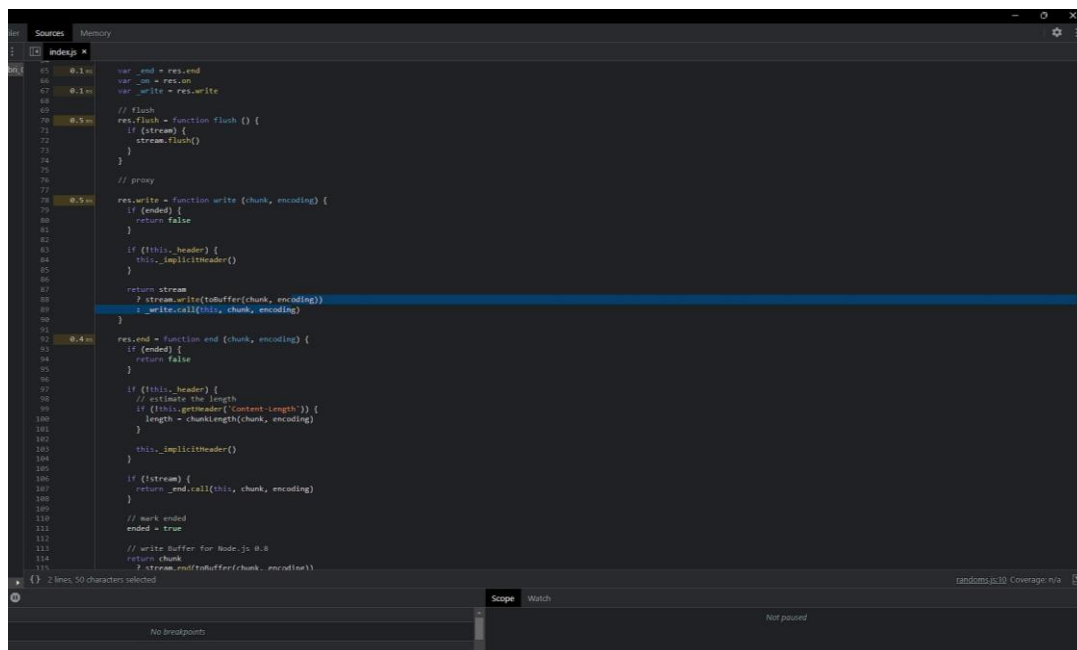
### DIAGRAMA DE FLAMA 0x



# NODE INSPECT



```
65 0.1 ms var _end = res.end
66      var _on = res.on
67 0.1 ms var _write = res.write
68
69
70 0.5 ms // flush
71      res.flush = function flush () {
72          if (!stream) {
73              stream.flush()
74          }
75      }
76
77      // proxy
78 0.5 ms res.write = function write (chunk, encoding) {
79          if (!ended) {
80              return false
81          }
82
83          if (!this._header) {
84              this._implicitHeader()
85          }
86
87          return stream
88          ? stream.write(toBuffer(chunk, encoding))
89            : _write.call(this, chunk, encoding)
89
90      }
91
92 0.4 ms res.end = function end (chunk, encoding) {
93          if (!ended) {
94              return false
95          }
96
97          if (!this._header) {
98              // estimate the length
99              if (!this.getHeader('Content-length')) {
100                  length = chunklength(chunk, encoding)
101              }
102
103              this._implicitHeader()
104          }
105
106          if (!stream) {
107              return _end.call(this, chunk, encoding)
108          }
109
110          // mark ended
111          ended = true
112
113          // write Buffer for Node.js 0.8
114          return chunk
115              ? stream.end(toBuffer(chunk, encoding))
116              : _end.call(this, chunk, encoding)
116
117      }
118
119      // 2 lines, 50 characters selected
```



```
65 0.1 ms var _end = res.end
66      var _on = res.on
67 0.1 ms var _write = res.write
68
69
70 0.5 ms res.flush = function flush () {
71          if (!stream) {
72              stream.flush()
73          }
74      }
75
76      // proxy
77 0.5 ms res.write = function write (chunk, encoding) {
78          if (!ended) {
79              return false
80          }
81
82          if (!this._header) {
83              this._implicitHeader()
84          }
85
86          return stream
87          ? stream.write(toBuffer(chunk, encoding))
88            : _write.call(this, chunk, encoding)
89
90      }
91
92 0.4 ms res.end = function end (chunk, encoding) {
93          if (!ended) {
94              return false
95          }
96
97          if (!this._header) {
98              // estimate the length
99              if (!this.getHeader('Content-length')) {
100                  length = chunklength(chunk, encoding)
101              }
102
103              this._implicitHeader()
104          }
105
106          if (!stream) {
107              return _end.call(this, chunk, encoding)
108          }
109
110          // mark ended
111          ended = true
112
113          // write Buffer for Node.js 0.8
114          return chunk
115              ? stream.end(toBuffer(chunk, encoding))
116              : _end.call(this, chunk, encoding)
116
117      }
118
119      // 2 lines, 50 characters selected
```