**COMPRESION - GZIP**

**Modificación**

Instale 🡪 npm i compression

Importe 🡪 

**Hice dos pruebas con “/info” y con “/api/randoms”**

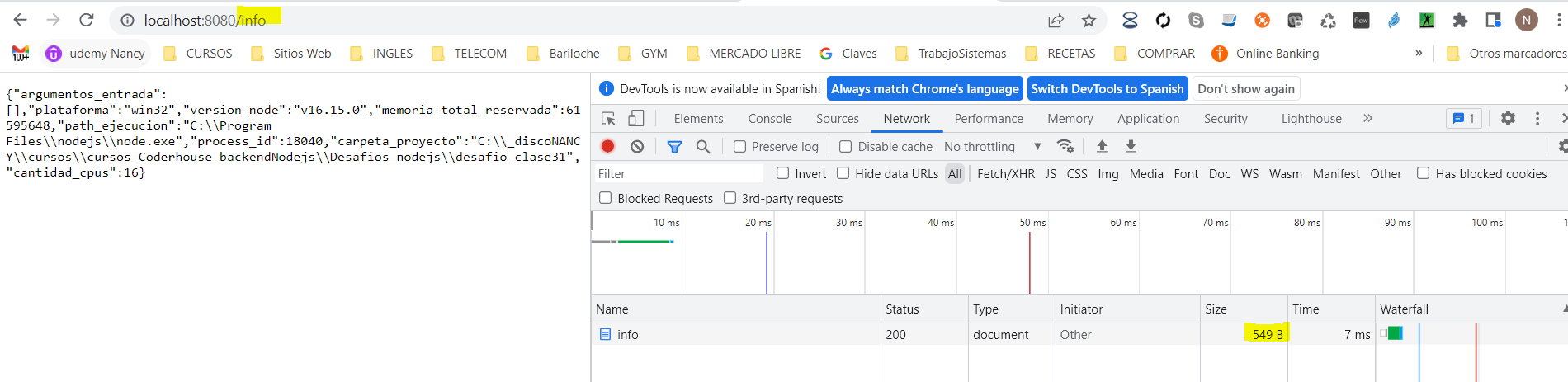
* Con “/info” saque la conclusión que no tiene sentido comprimir ya que sin comprimir es 549B y comprimido 572b
* Con “/api/randoms” saque la conclusión que si tiene sentido comprimir, paso de de 29.7kb paso a 3.7kb

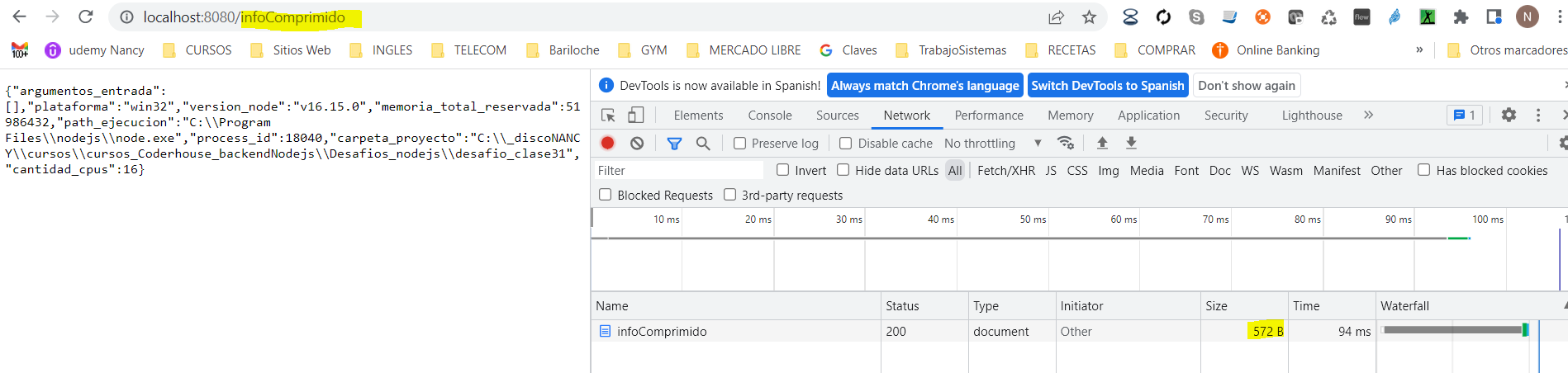
**“/info”**

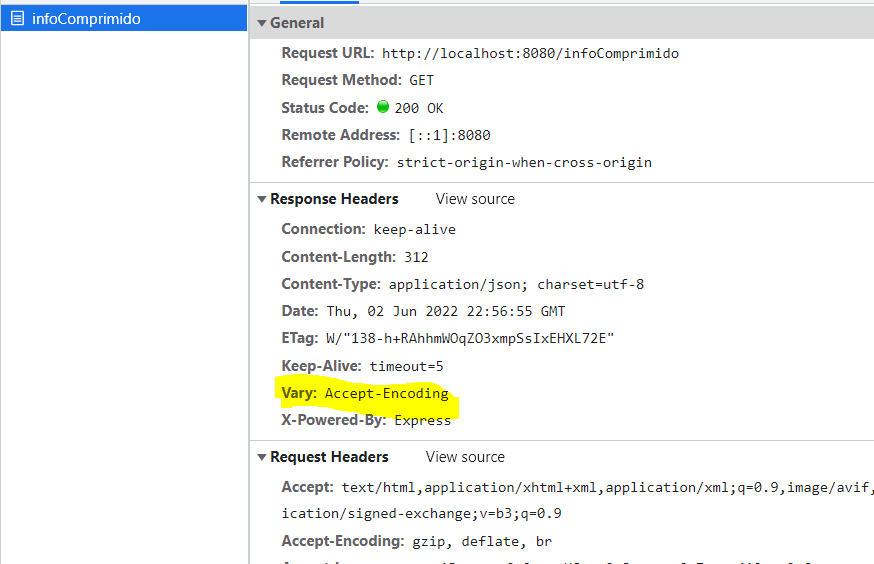
Fue el pedido en el desafio por el cual hice lo siguiente para que quede en el código:

Modififique server.js creando “/infoCompresion” y agregue el MDW comprimir() 🡪 

Resultado 🡪 como dije anteriormente observo que en este caso no tiene sentido comprimir, ya que sin comprimir es 549 B y comprimido 572 B.



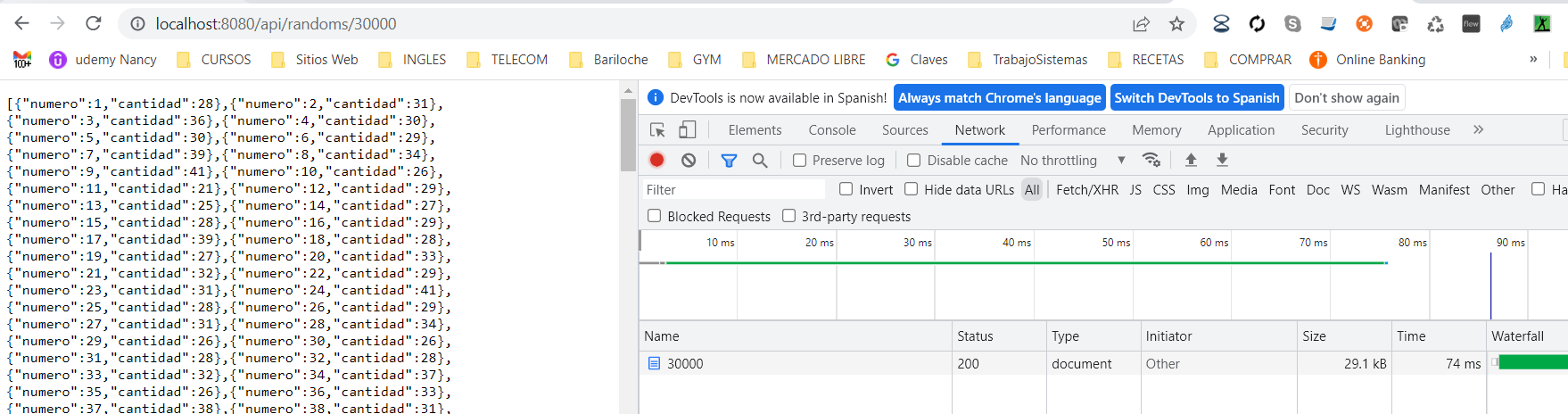




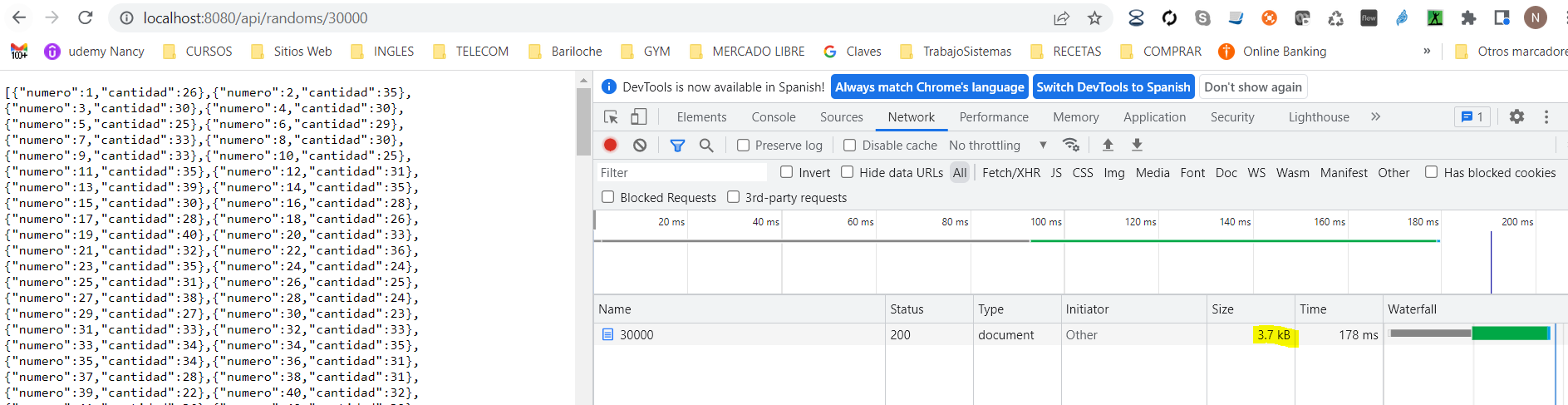
**“/api/randoms”**

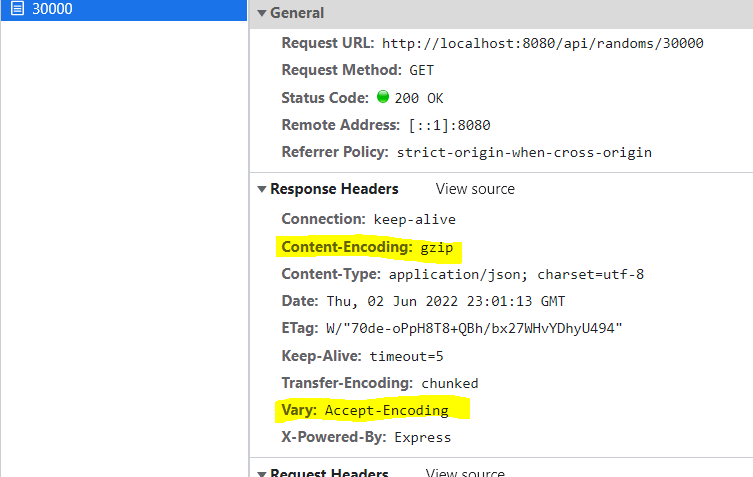
Al probar con /api/randoms y obsevo que si vale la pena comprimir ya que pasa de 29.7kb paso a 3.7kb

Sin comprimir 🡪 



Comprimido 🡪





**LOGGERS**

Configuración

 🡪 se puede observar que guardo en archivo en el caso de warn y de error.

import log4js from 'log4js';

import config from '../config/config.js';

log4js.configure({

  appenders: {

    consola: { type: 'console' },

    archivoErrores: { type: 'file', filename: 'error.log' },

    archivoWarning: { type: 'file', filename: 'warning.log' },

    loggerConsola: {

      type: 'logLevelFilter',

      appender: 'consola',

      level: 'info',

    },

    loggerArchivoWarning: {

      type: 'logLevelFilter',

      appender: 'archivoWarning',

      level: 'warn',

    },

    loggerArchivoErrores: {

      type: 'logLevelFilter',

      appender: 'archivoErrores',

      level: 'error',

    }

  },

  categories: {

    default: {

      appenders: ['loggerConsola', 'loggerArchivoErrores', 'loggerArchivoWarning'],

      level: 'all',

    },

    prod: {

      appenders: ['loggerConsola', 'loggerArchivoErrores', 'loggerArchivoWarning'],

      level: 'all',

    },

  },

})

let logger = null

if (config.NODE\_ENV === 'production') {

  logger = log4js.getLogger('prod')

} else {

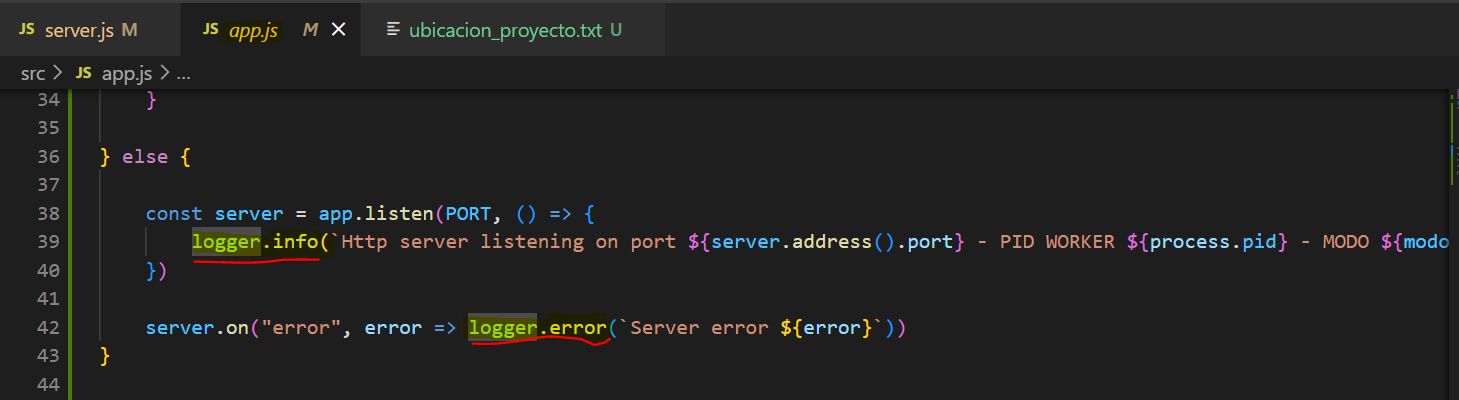
  logger = log4js.getLogger()

}

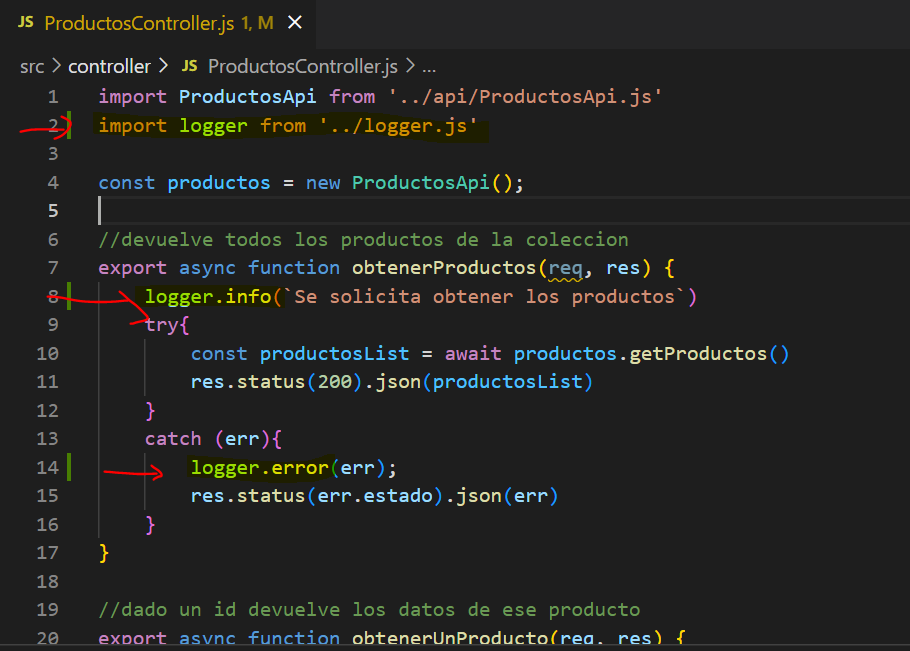
export default logger;

A continuación paso paso un ejemplo de cada uno de los tipos de logueo pedidos: info, error y warning

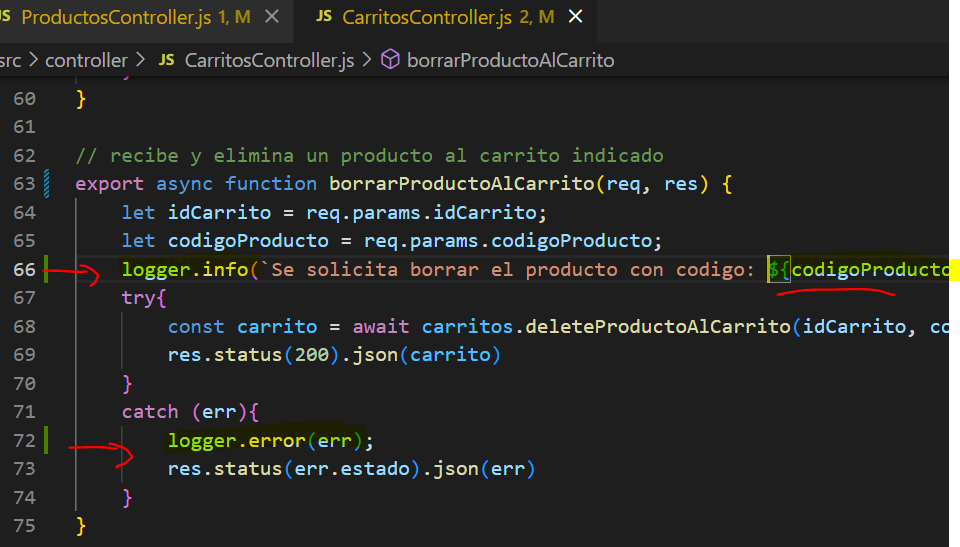
INFO y ERROR Ejemplo 1: conexión servidor.



INFO y ERROR Ejemplo 2: PRODUCTOS



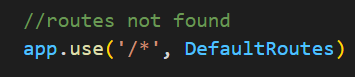
INFO y ERROR Ejemplo 2: CARRITO , grabando también valores que fueron ingresados



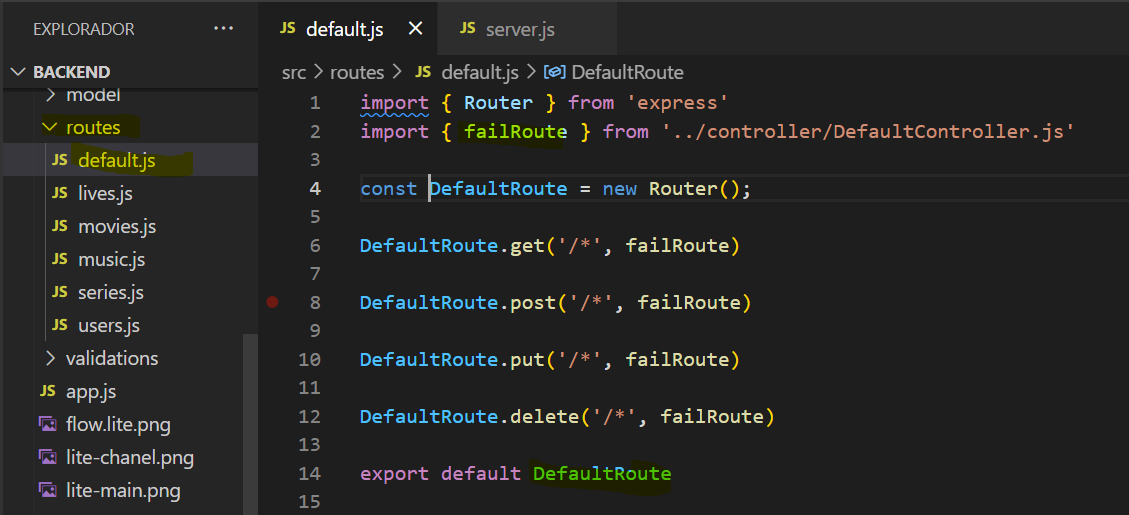
WARN

Ejemplo 🡪 Para el caso de querer acceder a alguna ruta inexistente

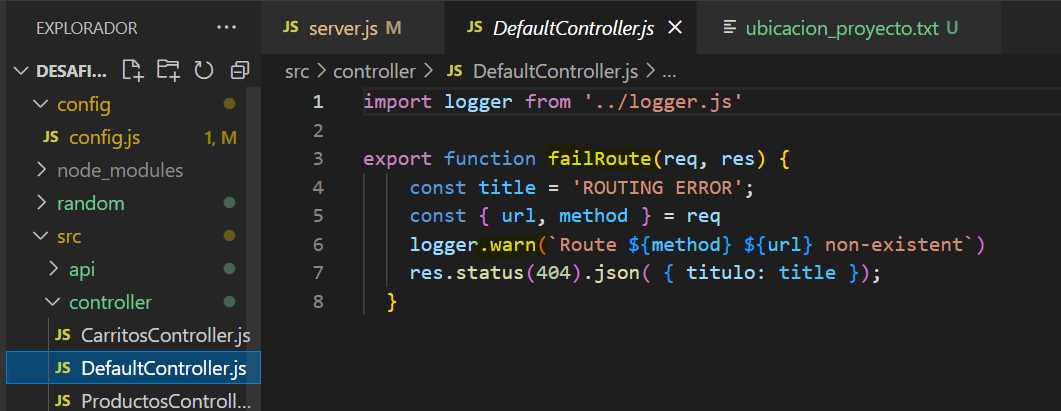
**Server.js**



**Default.js 🡪** Lo hago para todos métodos: get, post, put y delete.

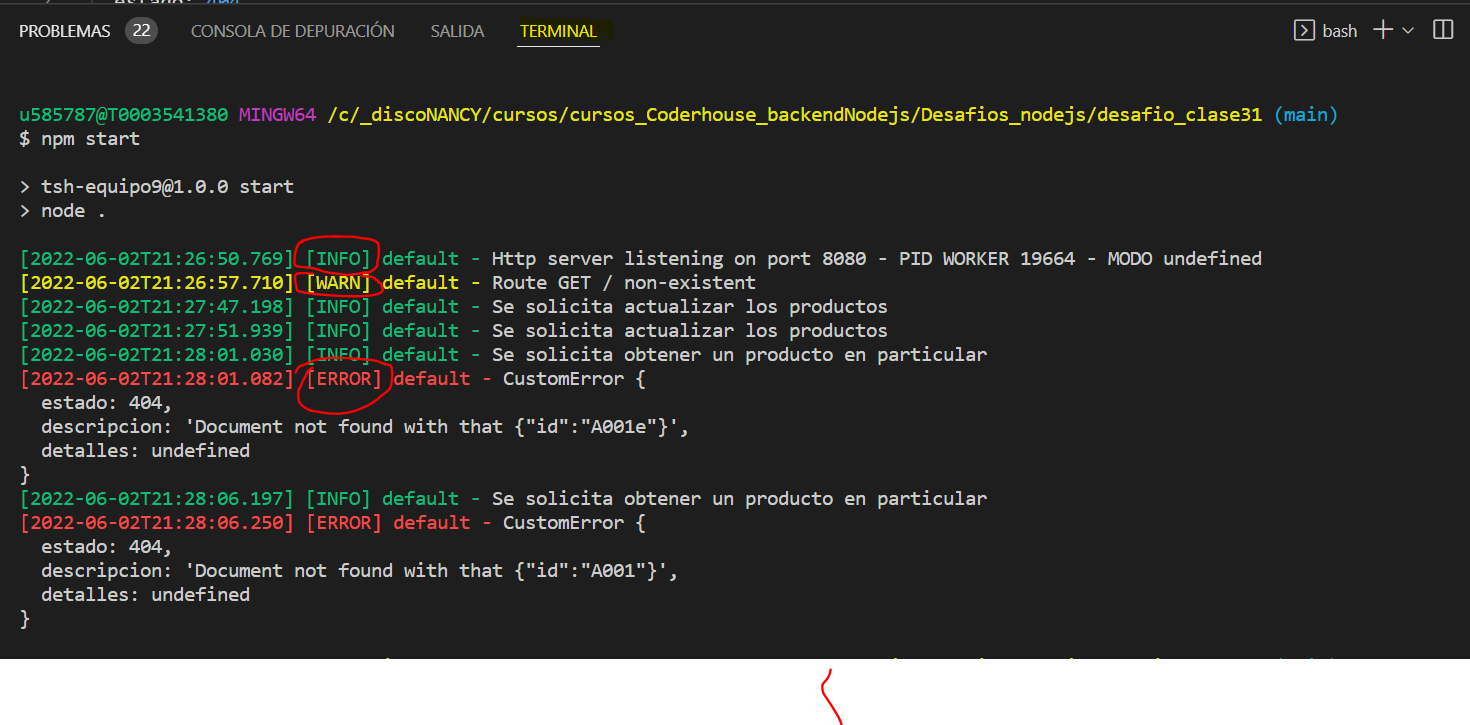


DefaultController.js 🡪 creo log de warning

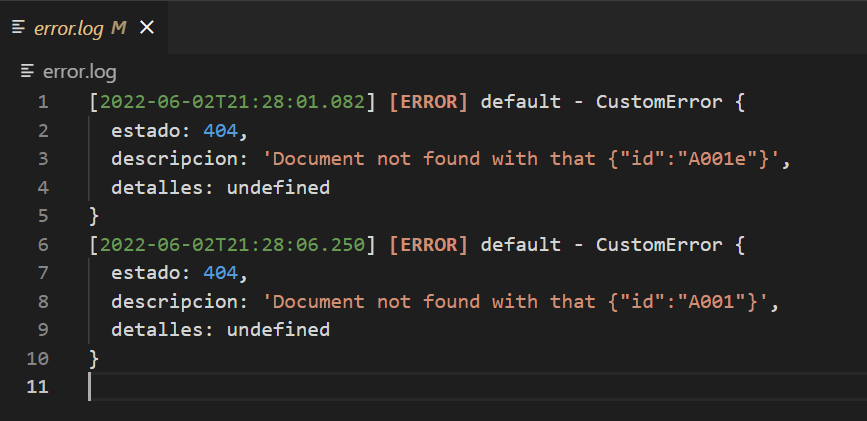


Ejemplo de una ejecución

Resultado por consola



Archivo error.log



Archivo warning.log

