## Preguntas

¿Qué son los buffers en NodeJs?

Describa 2 maneras de crear buffers en NodeJs

¿Qué método se usa para la escritura de buffers y que parámetros emplea?

Describa cada uno de los parámetros empleados para la escritura de Buffers en NodeJs

¿Como se hace la lectura de buffers en NodeJs?

Describa cada uno de los parámetros empleados para la lectura de Buffers en NodeJs

## Ejercicio

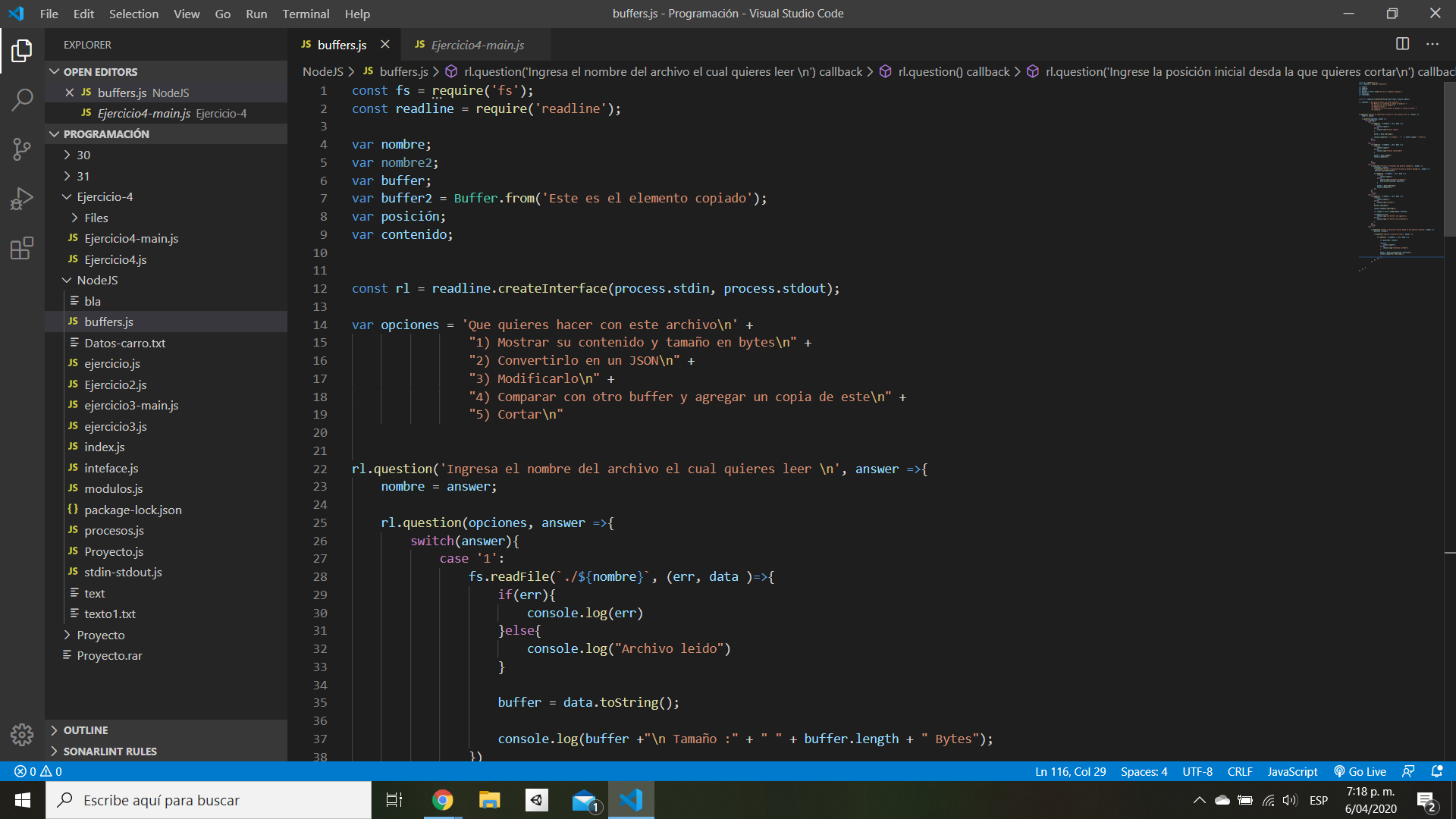
Desarrollar un programa en NodeJs que emplee el uso de Buffers usando los siguientes métodos pertenecientes al objeto Buffer de NodeJs:

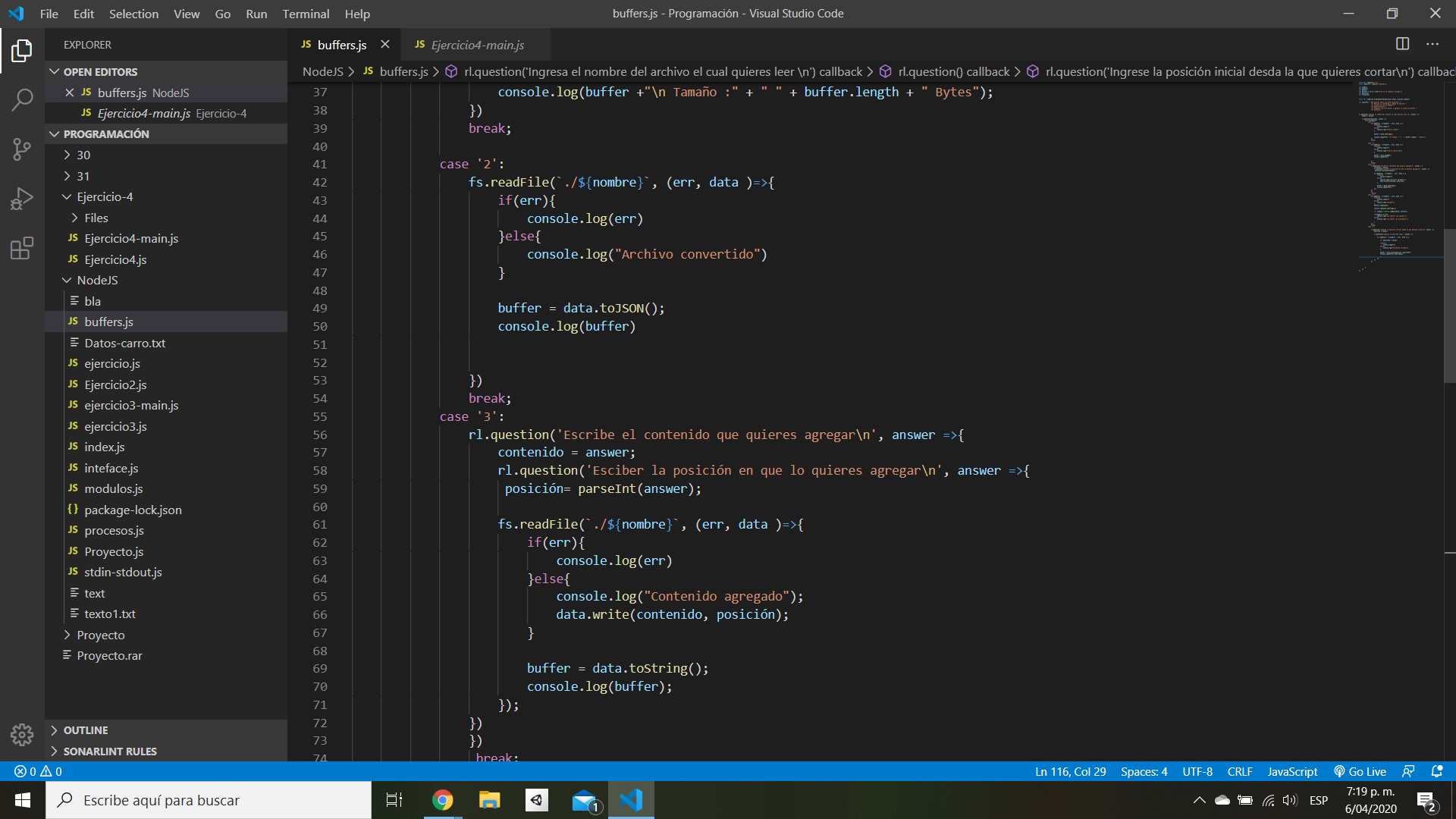
* Write
* toString
* toJSON
* concat
* compare
* copy
* slice
* length

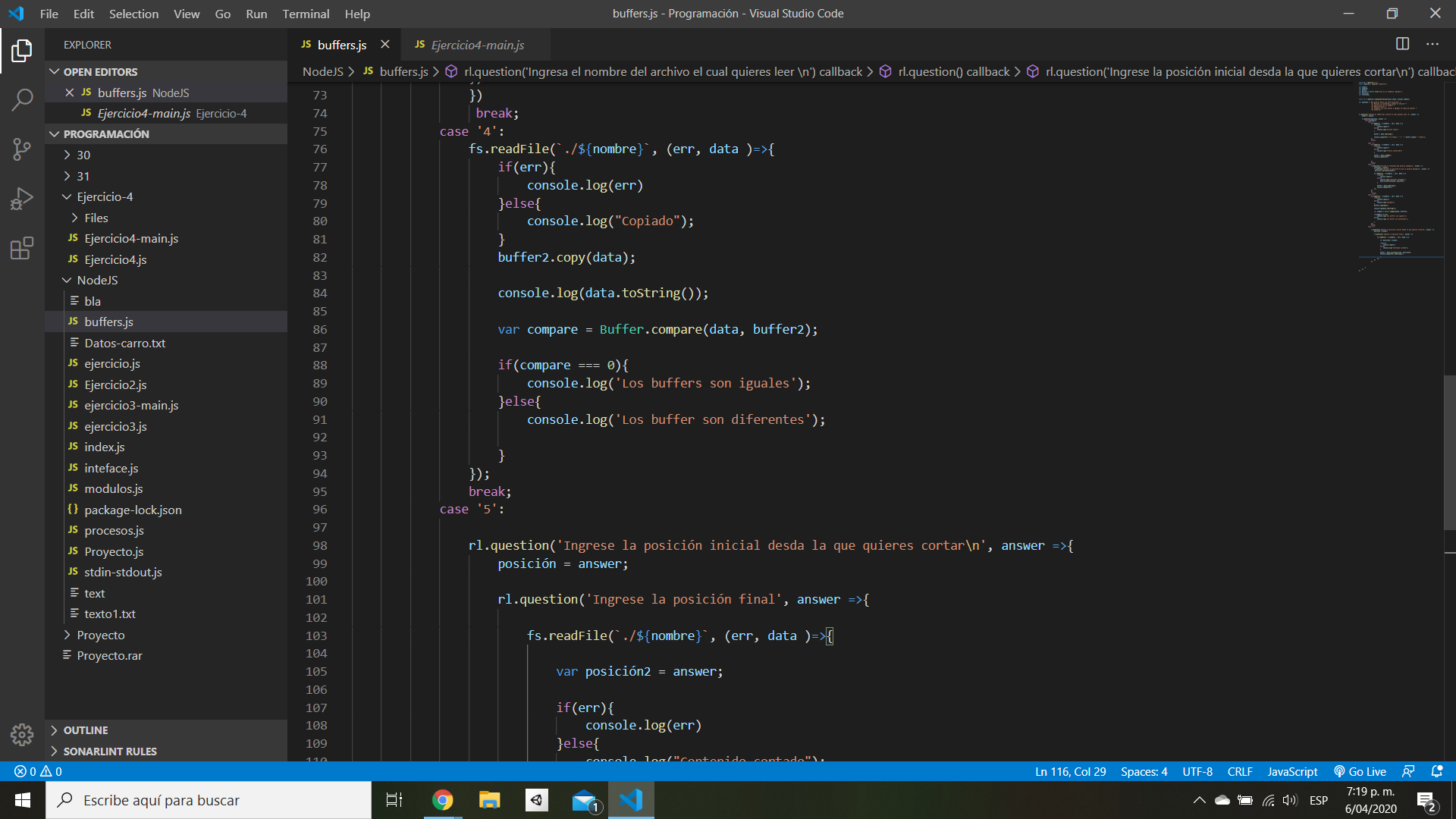
**Desarrollo**

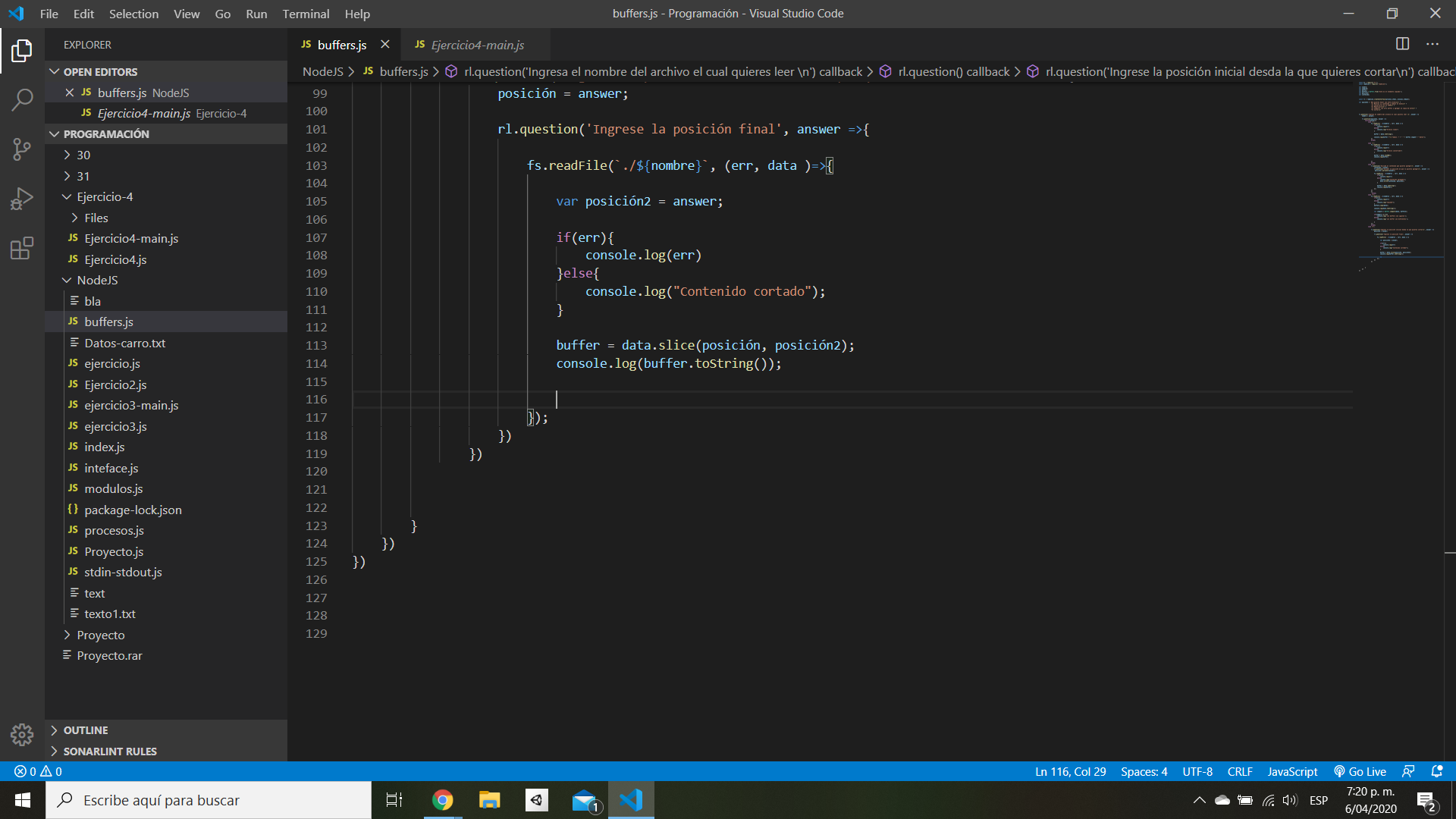
1. En node js un buffer es la forma con la que se representan los datos binarios de una manera similar a los arrays de enteros, en los que tenemos secuencias de bytes en memoria fuera del motor v8.
2. Para crear un buffer podemos utilizar alguna de estas dos formas.
   1. var buffer1 = Buffer.alloc(10); esto crearía un buffer con un tamaño de 20 bytes.
   2. var buffer2 = Buffer.from(‘Hola’); el método from() nos permite crear un buffer en el que le asignaremos de una vez su contenido.
3. Para la escritura de buffers empleamos el método buffer.write(), al cual le debemos pasar de manera obligatoria el parámetro value y de manera opcional podemos pasar los parámetros offset, length y encoding.
4. En el parámetro value debemos pasarle un string el cual debe contener lo que queremos escribir en el buffer, en offset le podemos pasar el lugar en el que queremos comenzar a escribir, similar a un array contamos las posiciones desde el 0, siendo por defecto la posición 0 en caso de que no se le pase algún valor. En length podemos pasarle el tamaño máximo en bytes que queremos para nuestro buffer y en encoding le podemos pasar la codificación de caracteres que deseemos utilizar, siendo en un principio por defecto ‘utf8’
5. La forma más común de leer buffers es mediante el método toString(), aunque hay otros métodos que podríamos utilizar según nuestras necesidades.
6. Si utilizamos el método toString() podremos utilizar 3 parámetros opcionales.
   1. Encoding, al que le pasaremos la codificación de caracteres con la cual queremos que nos devuelva el valor.
   2. start, al que le pasaremos desde cual posición queremos que empiece la lectura.
   3. end, al cual le pasaremos en donde queremos terminar la lectura.

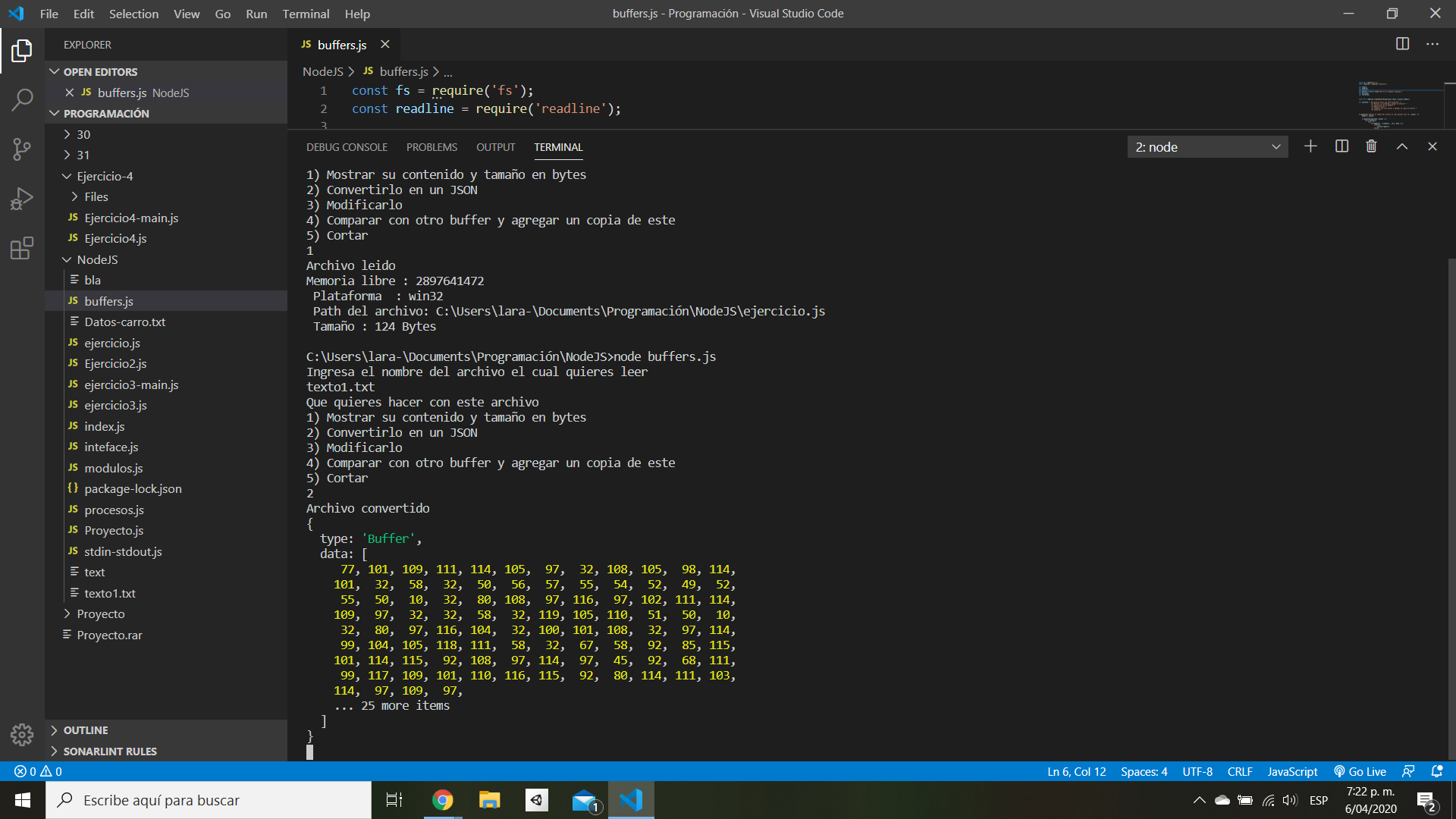
**Ejercicio.**

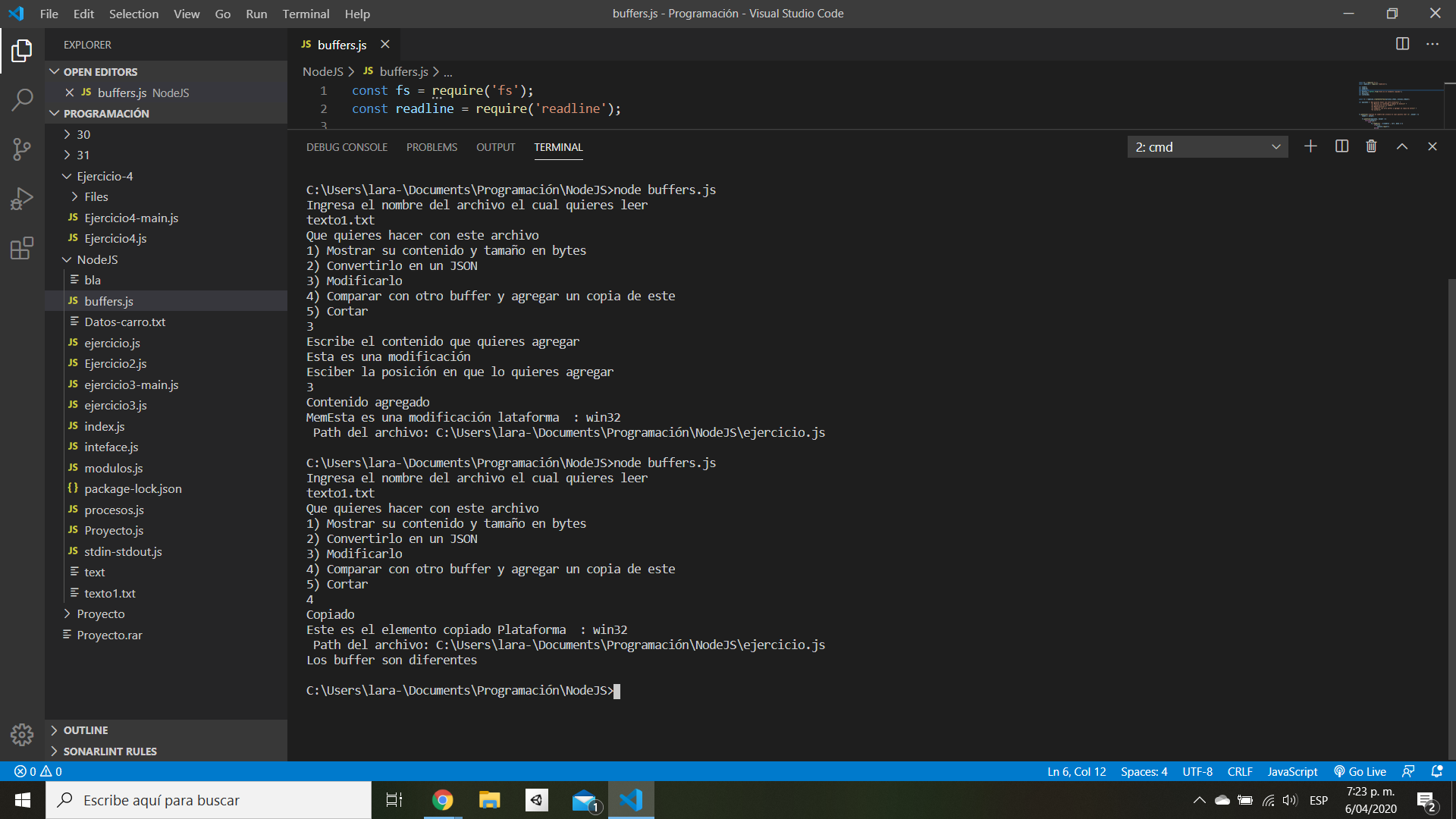
****

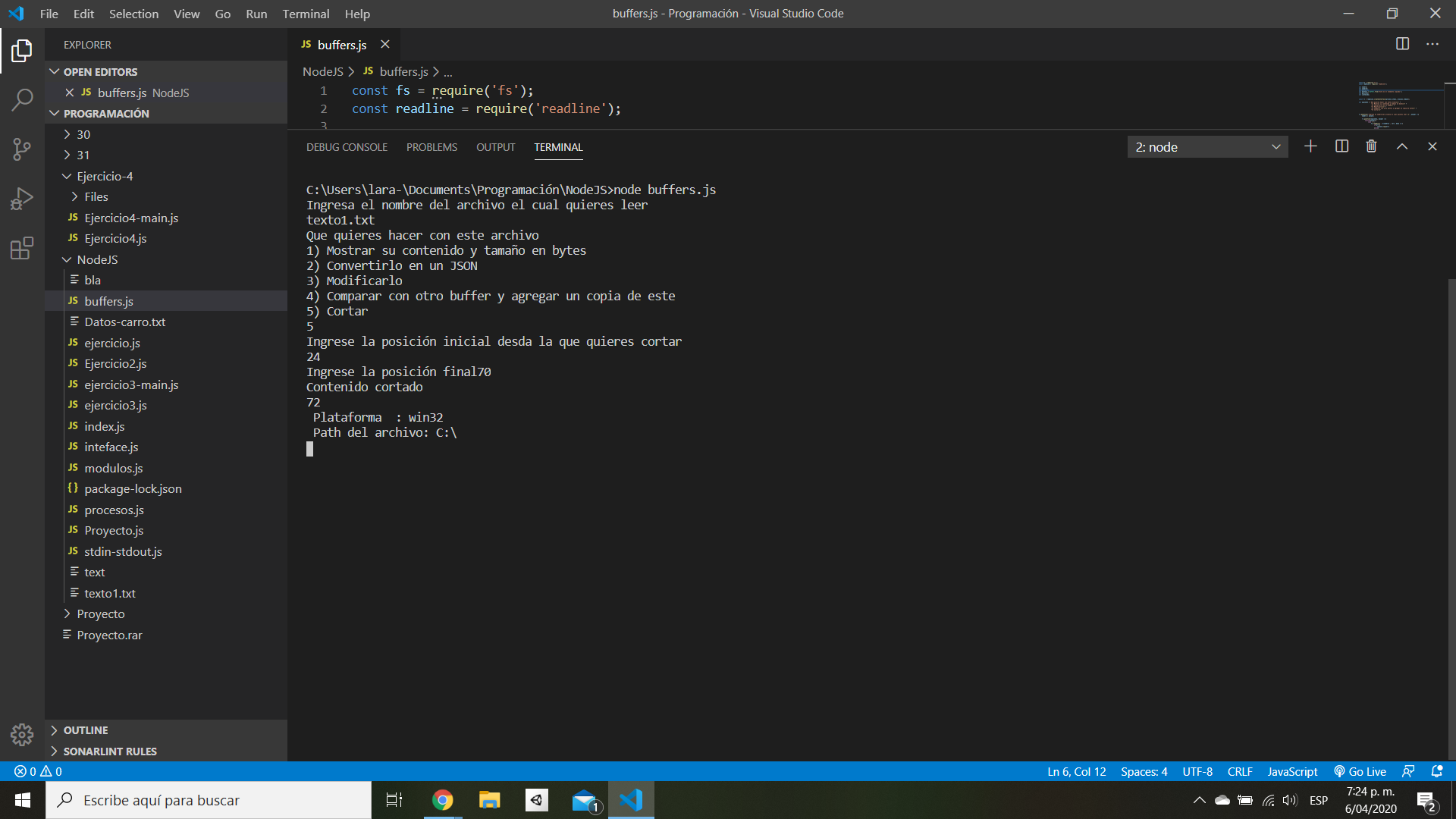
****

****

****

****

****

****