**QUE ES NODE.JS**

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8

Es decir, es un ambiente asíncrono manejado por eventos que se disparan en el que puedes corres Javascript.

Se puede correr archivos de JavaScript si tener que estar conectado a un navegador

**Sincrono**

Tarea 1

Tarea 2

Tarea 3

**Asincrono**

Tarea1

Tarea 3

Tarea2

NODE.js Tiene su propio gestor de paquetes considerado el más grande de código abierto. => Node Package Manager **(NPM)**

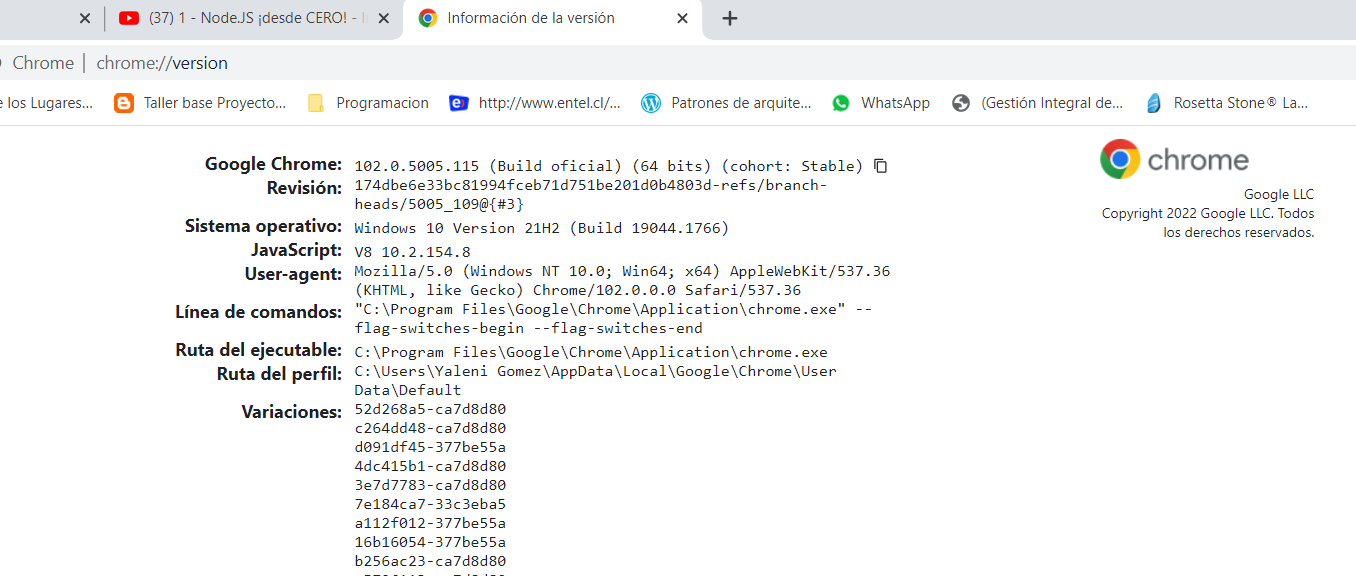
Esto nos permite:

.- Instalar librerías, programas, etc.

.- Administrar dependencias

.- Distribuir código

Es especialista para aplicaciones en tiempo real como juegos en línea etc..



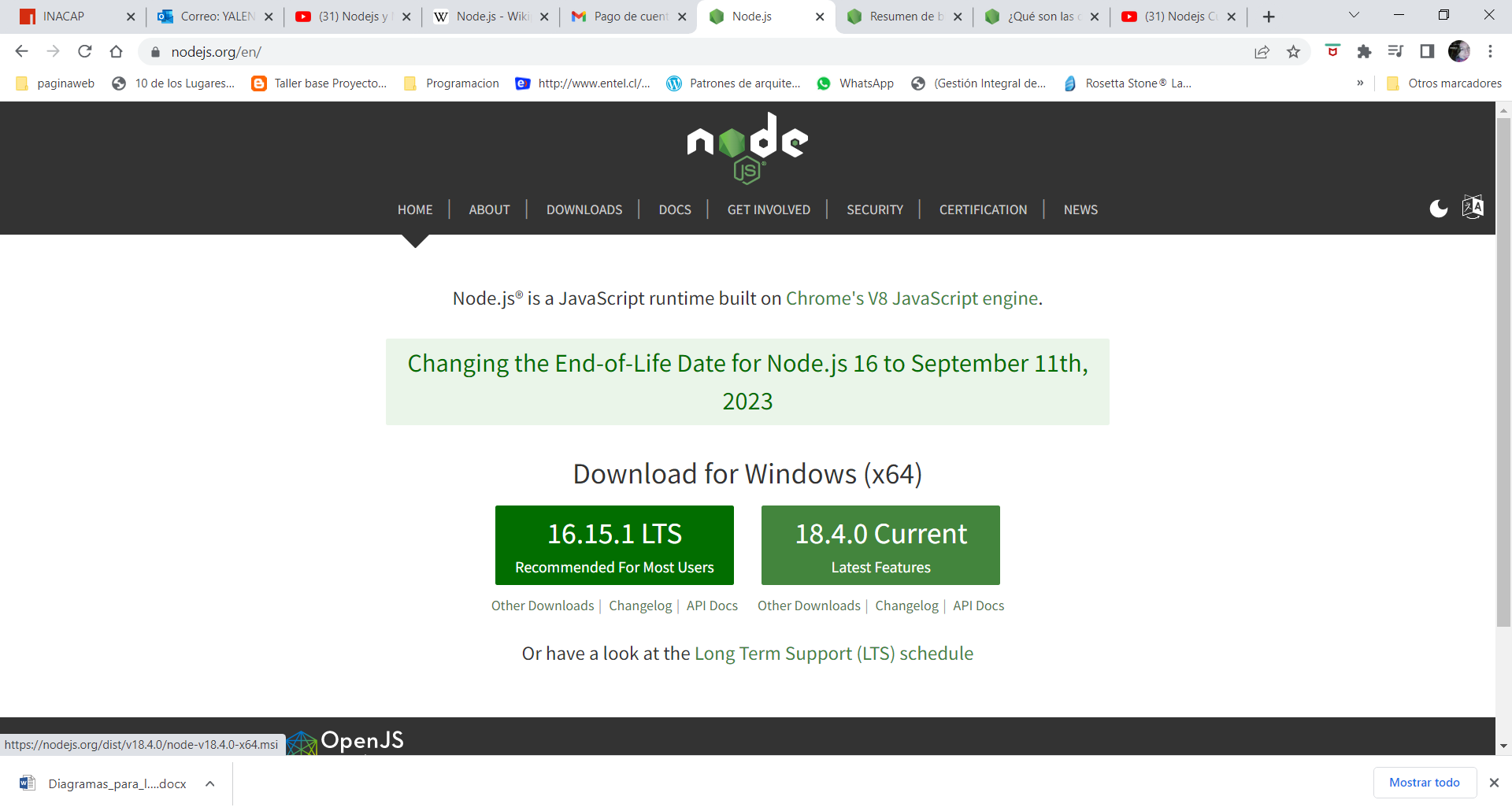
¿Qué es V8 en programación?

V8 es un motor open-source escrito en C++ para compilar JavaScript y WebAssembly en código máquina. Esto quiere decir que traduce JavasScript a un código puramente digital capaz de ser interpretado por la CPU donde se ejecuta

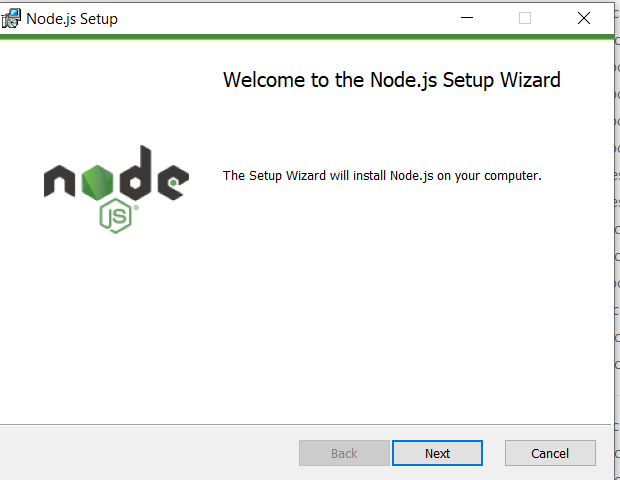
INSTALACION DE NODE.JS (Es multiplataforma dado que se puede utilizar en Linux, Windows)

<https://nodejs.org/en/download/>

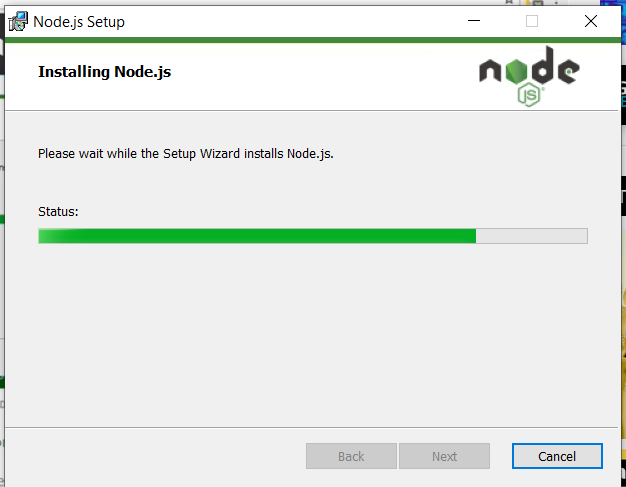
Seleccionar la versión actual(Que hace esto=> es descargar un instalador

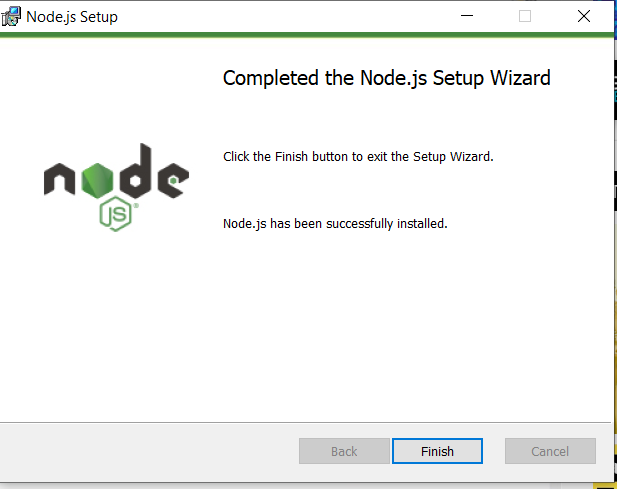


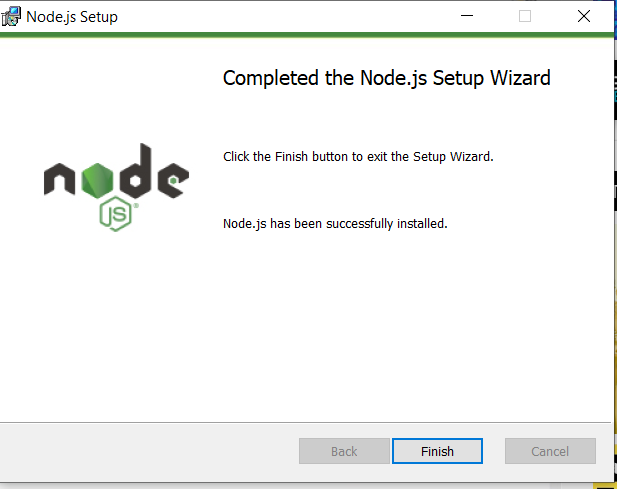
Ejecutar el archivo **node-v18.4.0-x64.msi**



**Se le da siguiente, se debe aceptar las condiciones y siguiente hasta el final.**



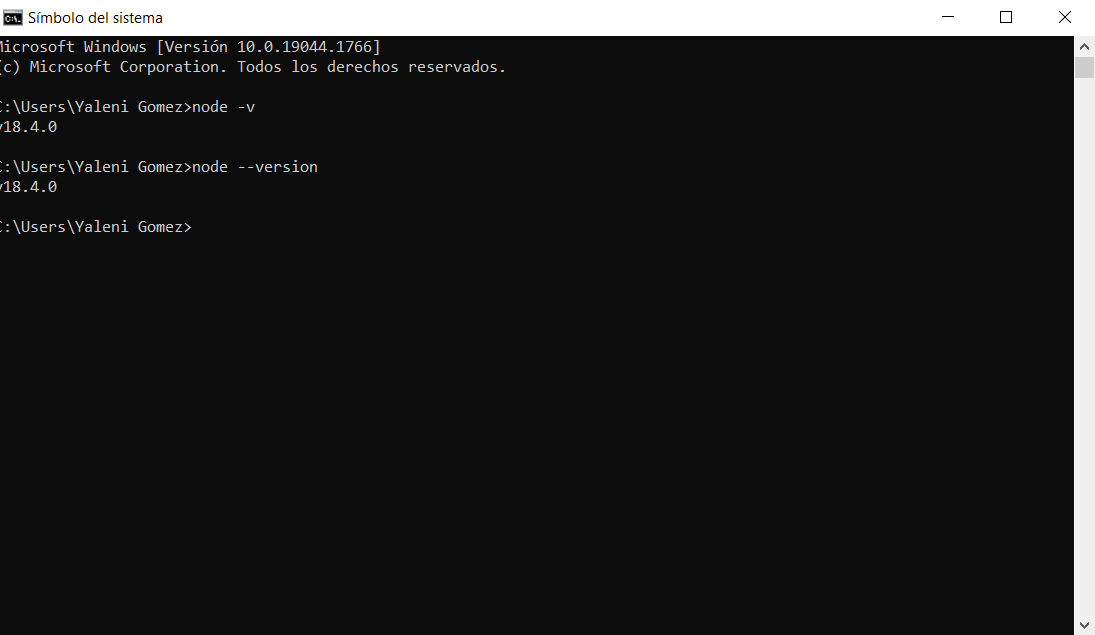


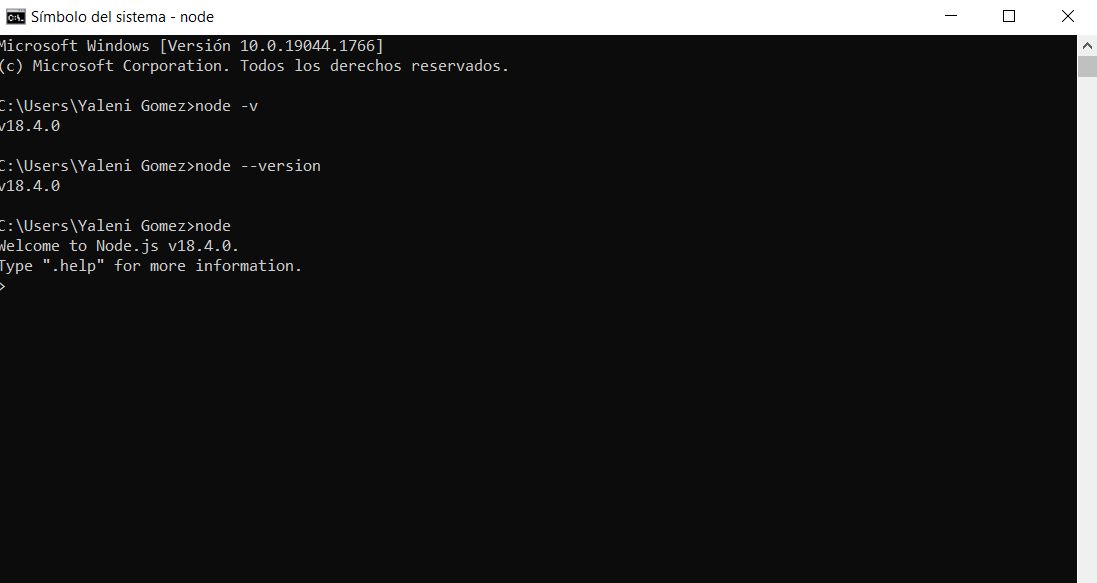


Verificar instalación de Node.js

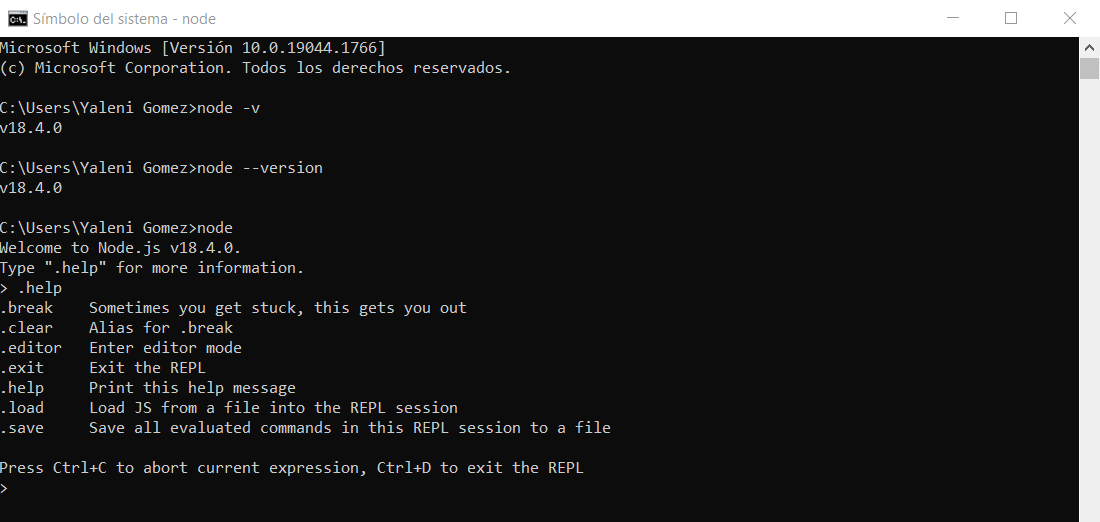
Para validar si quedo instalado Node.js se debe abrir la consola de Windows cmd

Digitar **node –v** o **node –version**

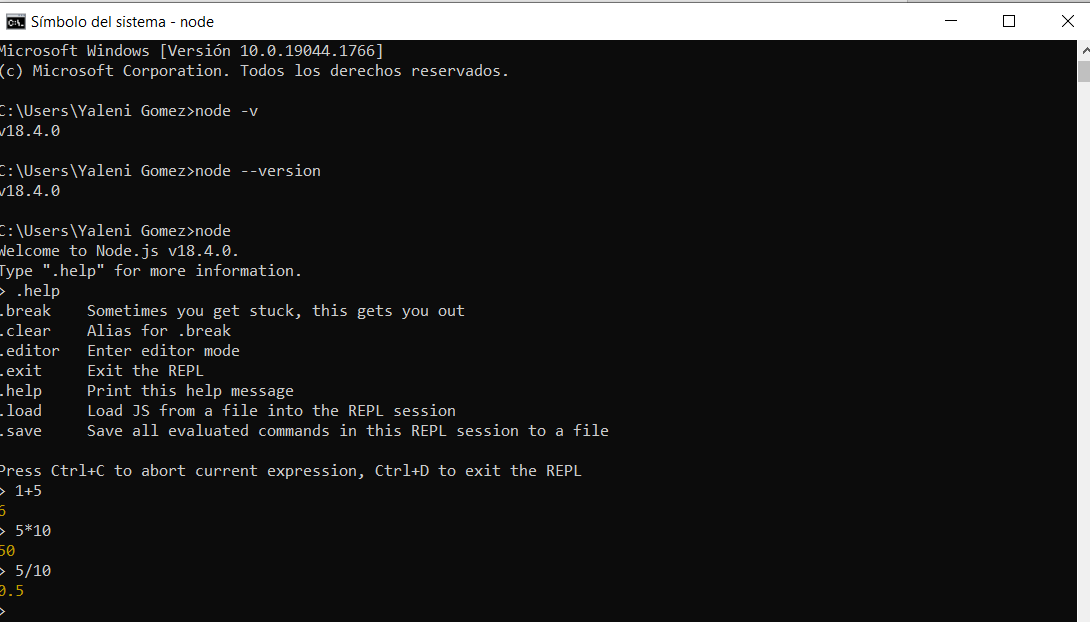




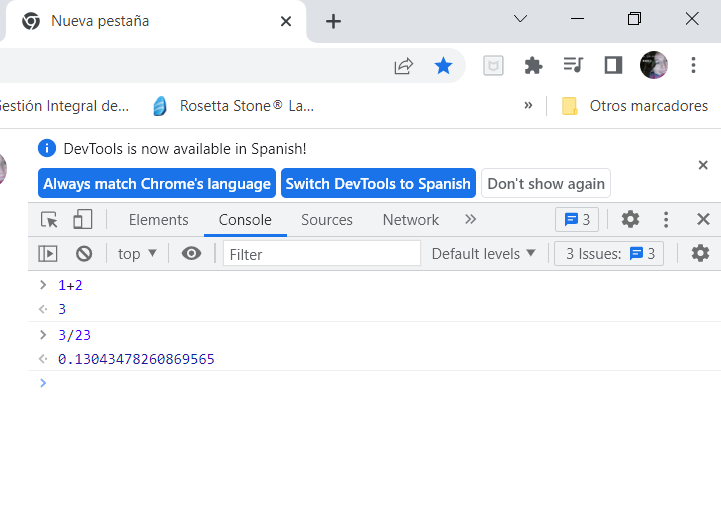
Dentro de Node escribimos una línea de comando “**.help”,** nos entrega información de algunos comandos.



En este entorno de programación podemos hacer varias cosas tales como: Sumar, multiplicar, dividir





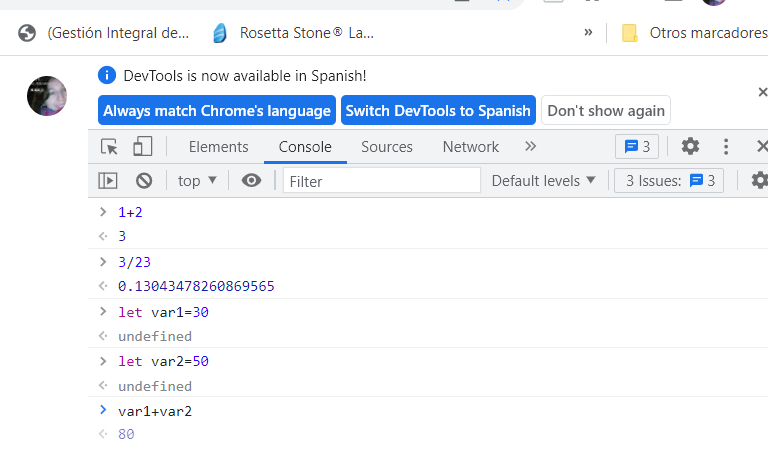


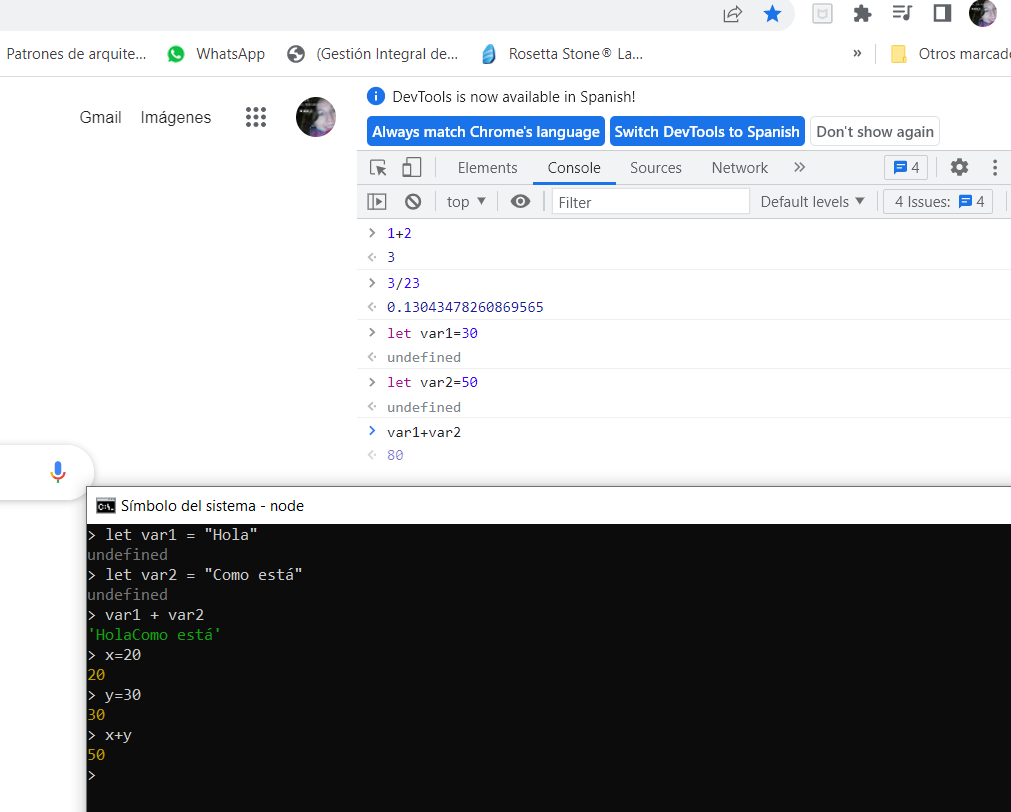
**La tecla Control + L= limpia la pantalla**

**Definiendo variable y mostrando la información de estas variables**

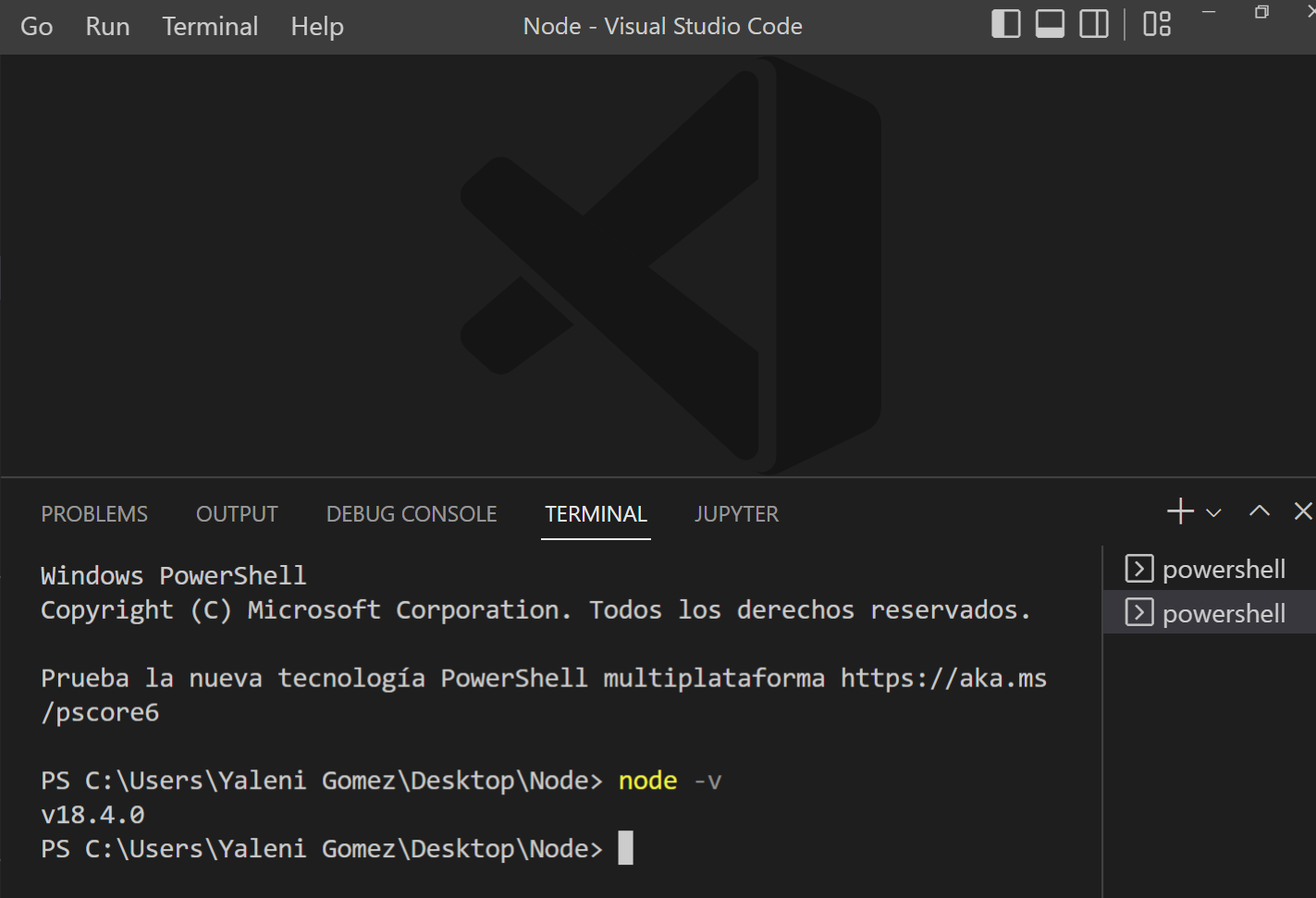






**La consola la podemos mostrar de diferentes maneras. Ejemplo**

**Otra consola en la que podemos trabajar en en Visual Code =>Terminal**



Para salid de la consola node presionar **tecla control + d**

**Conociendo NPM**

Es un administrador de paquetes de node.js, el cual sirve para instalar versiones de paquetes y librerías de javascript

<https://www.npmjs.com/>

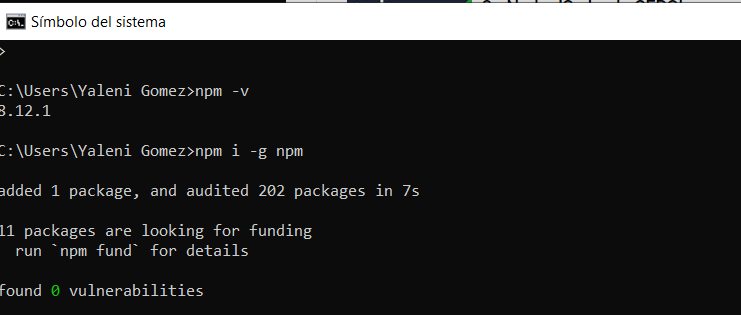
Posicionados en la consola de Windows

Digitar **npm –v** este comando nos permite ver la versión de **npm**

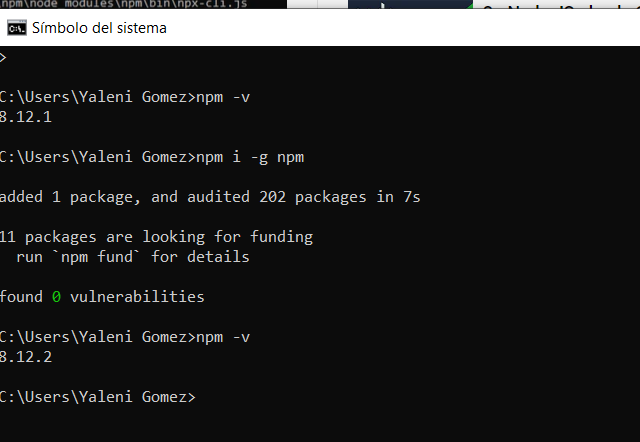


Para actualizar la versión debemos digitar la siguiente línea de comando:

**“” npm i –g npm** “”

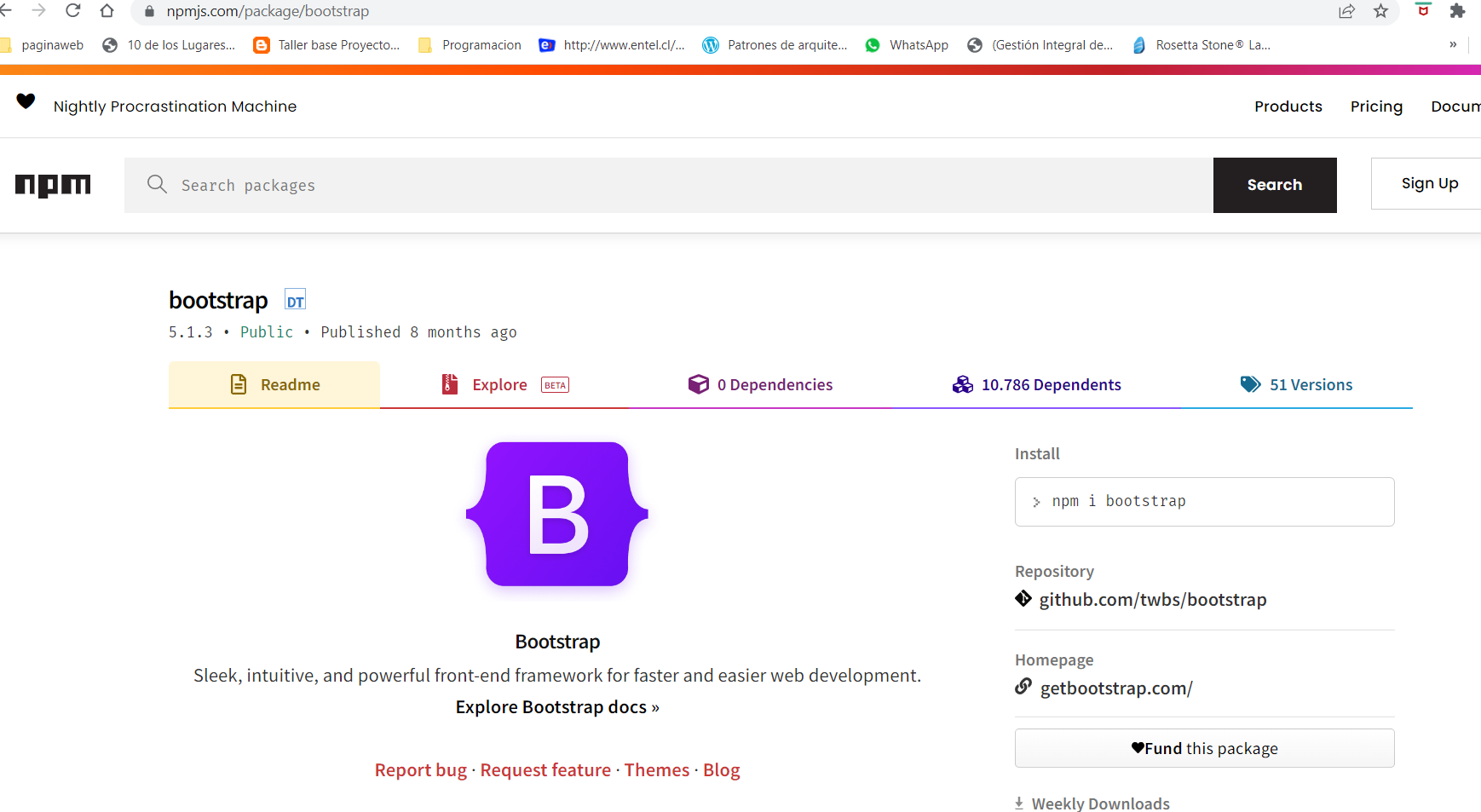


Actualizado la última versión

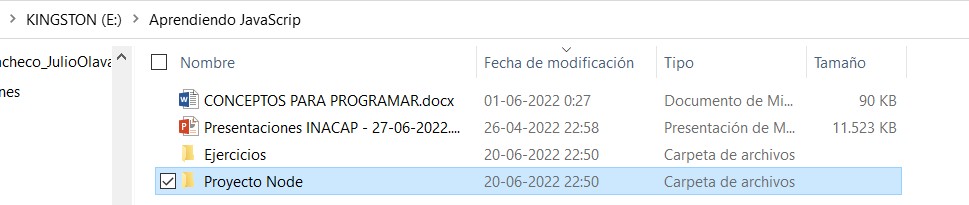


<https://www.npmjs.com/>

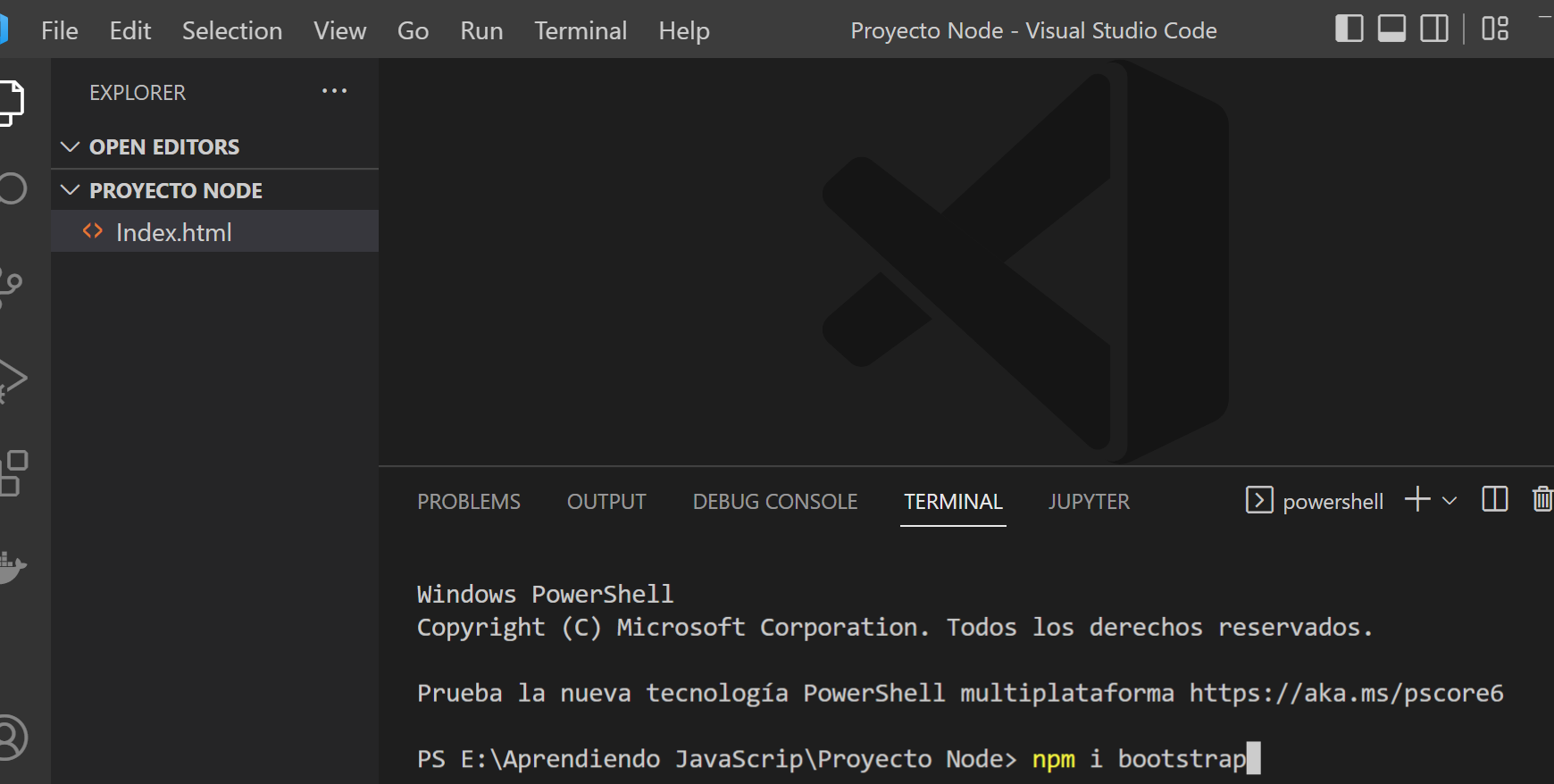
Si queremos instalar boostrap, en este apartado nos indica como instalarlo



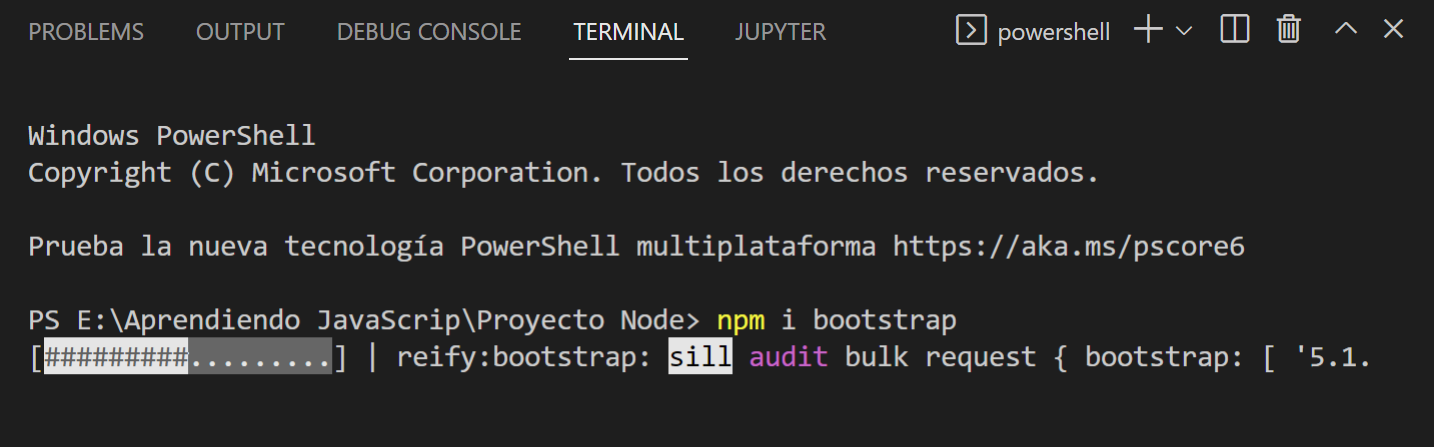
Ejercicio Crear una carpeta **Proyecto Node**



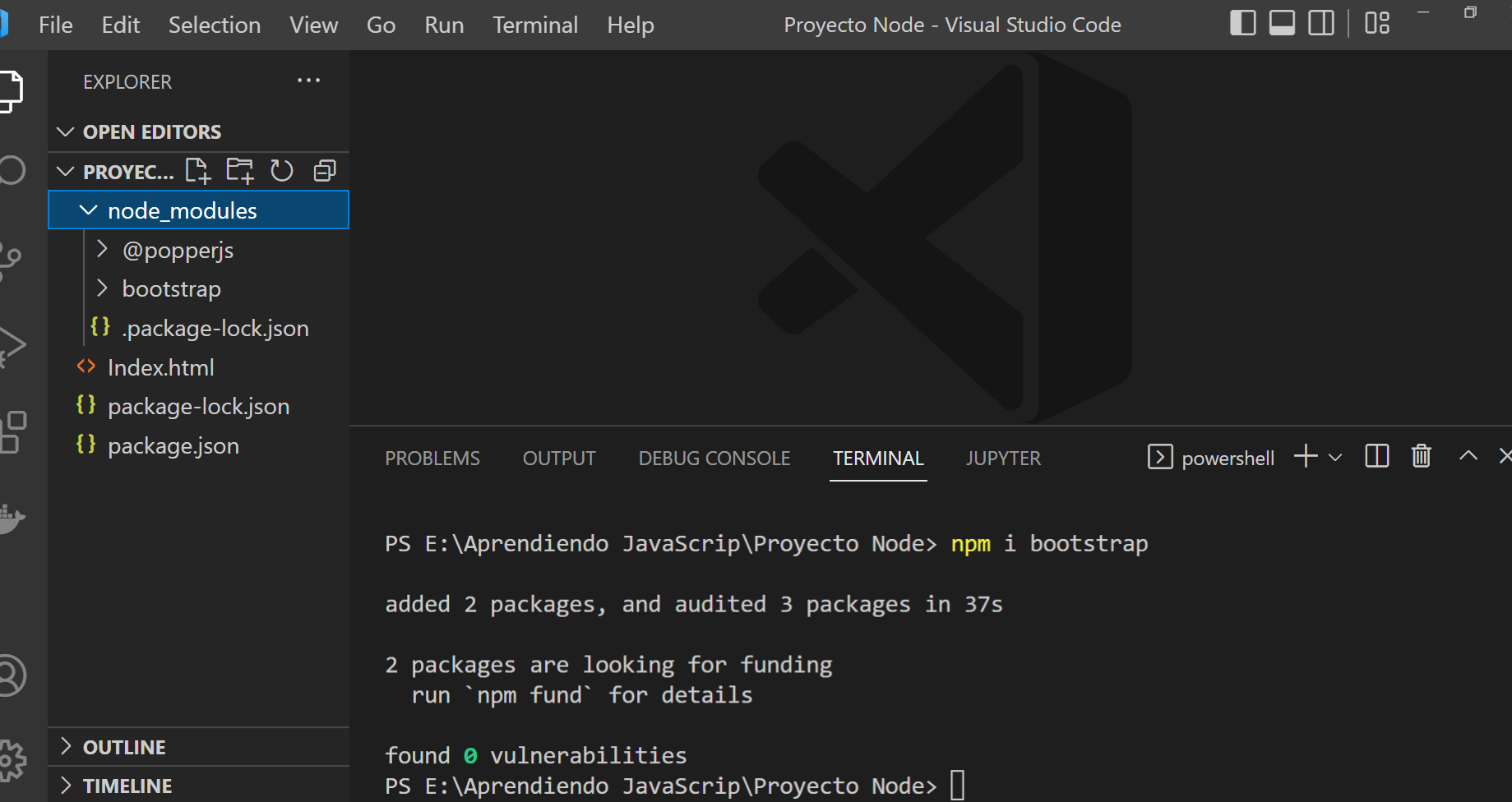
Dentro del proyecto creamos un html como se indica en la imagen



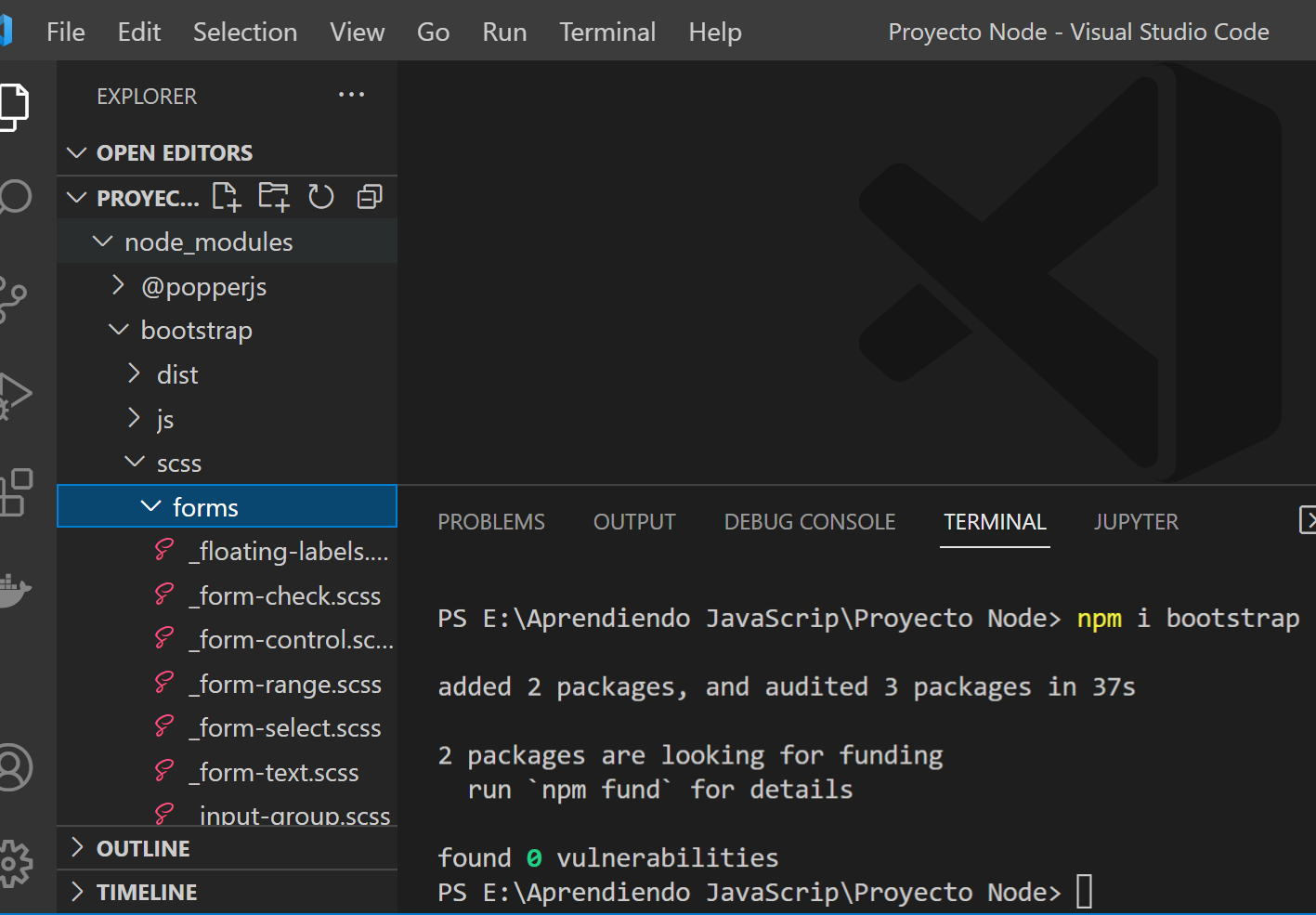
Luego en la consola digitamos el siguiente comando **npm i boostrap** luego damos enter



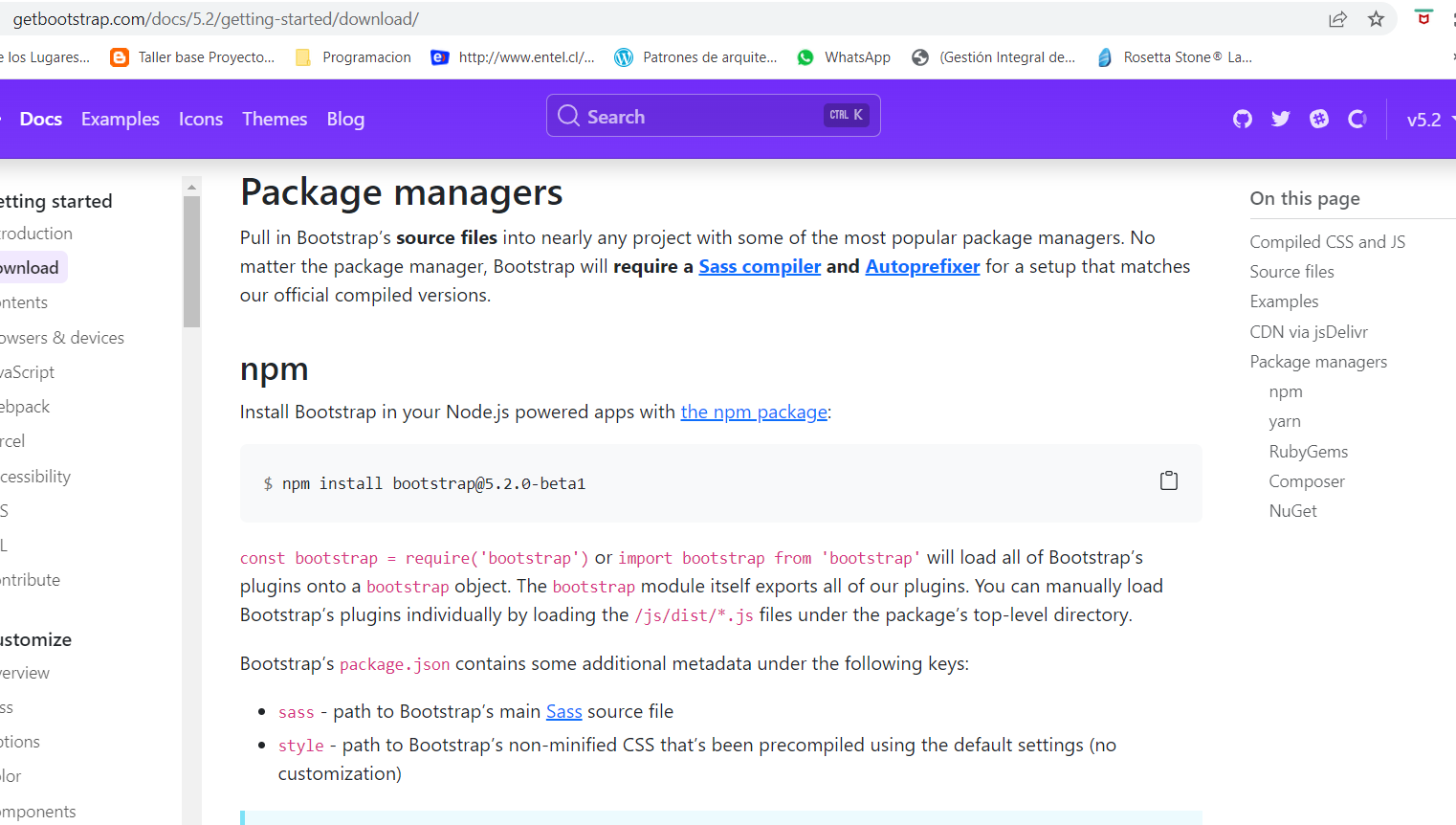
Luego del enter se va a crear una carpeta llamada **node\_module**



Revisando la carpeta **node\_modules**



Al revisar la documentación de boostrap nos entrega la misma información que ejecutamos en la consola

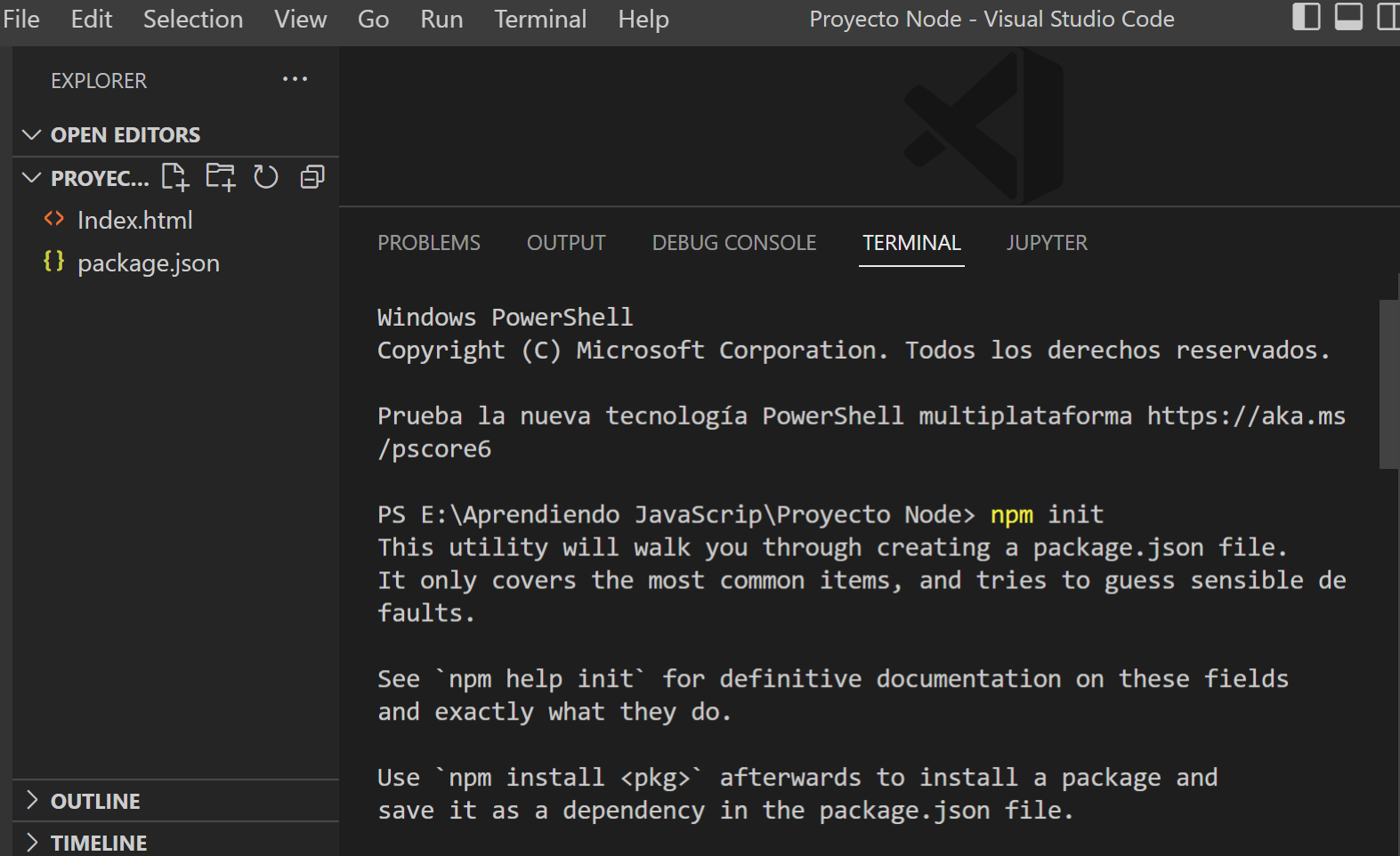


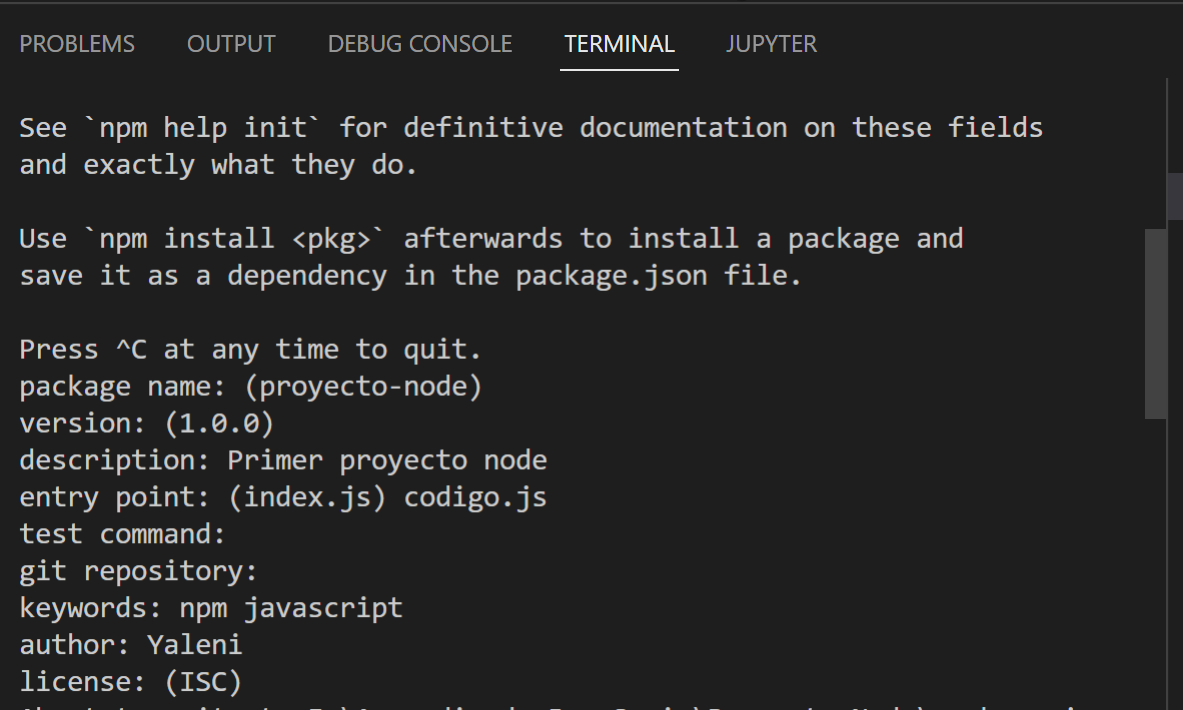
**En este ejercicio veremos el package.json**

Instalando paquetes y comandos básicos

Digitamos **npm init**

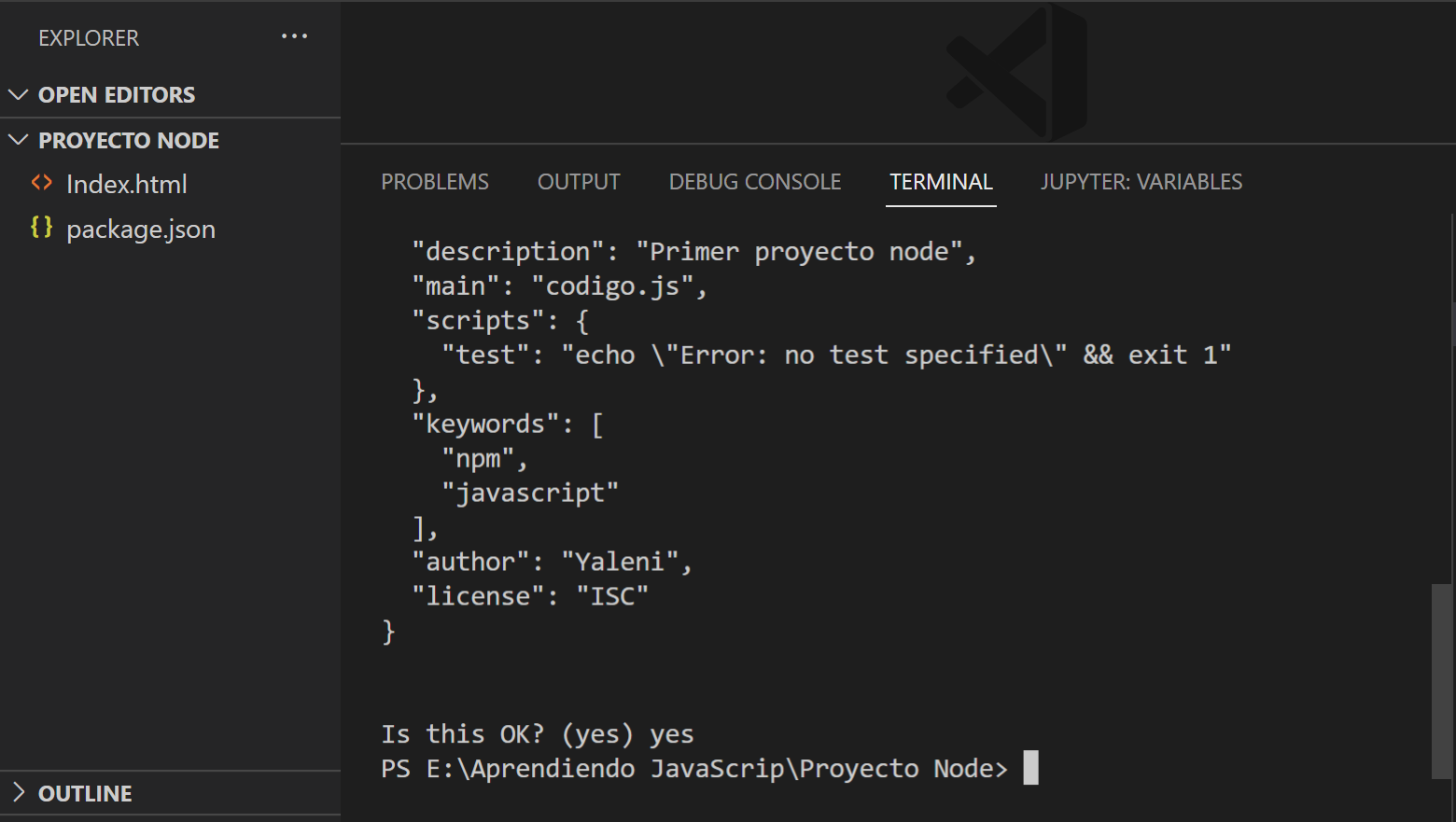
Este comando creara un archivo **package.json**





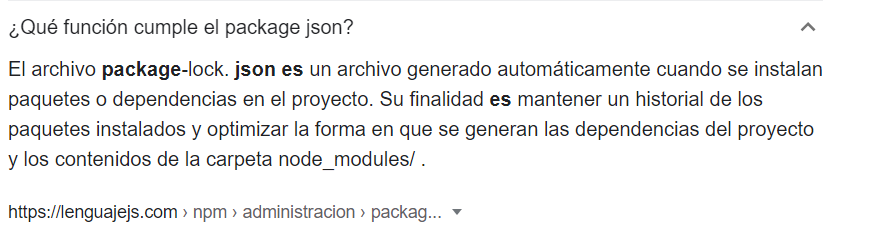
El resultado es:





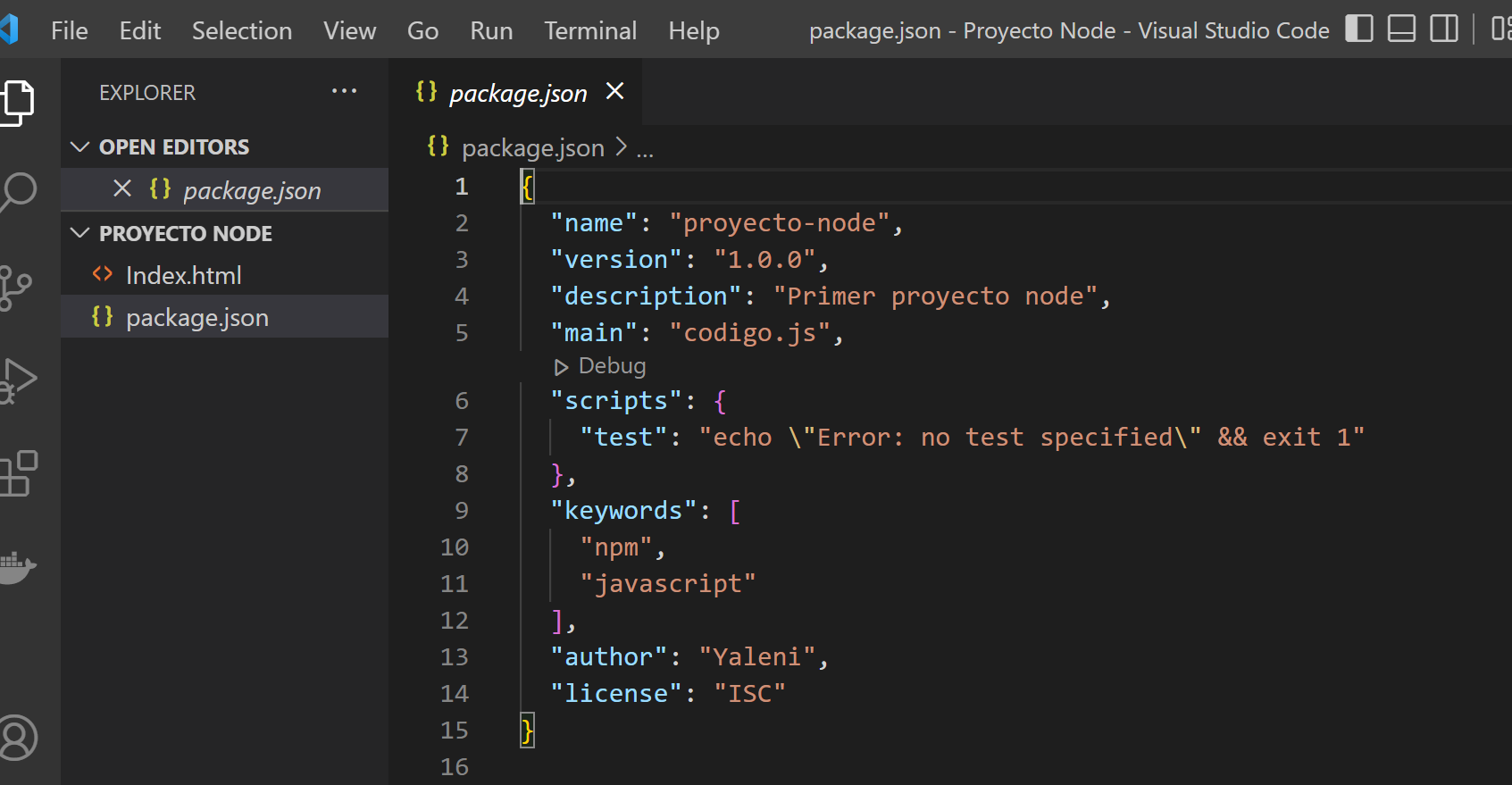
**Se creó el archivo package.json**

**Observación:**



El archivo **package**. **json contiene** todos los metadatos acerca del proyecto tal como descripción, licencia, dependencias y scripts. En la mayoría de los casos **es** sencillo encontrar que módulos son requeridos y cuales son los comandos principales y archivos al solo ver la estructura del archivo **package**. **json**

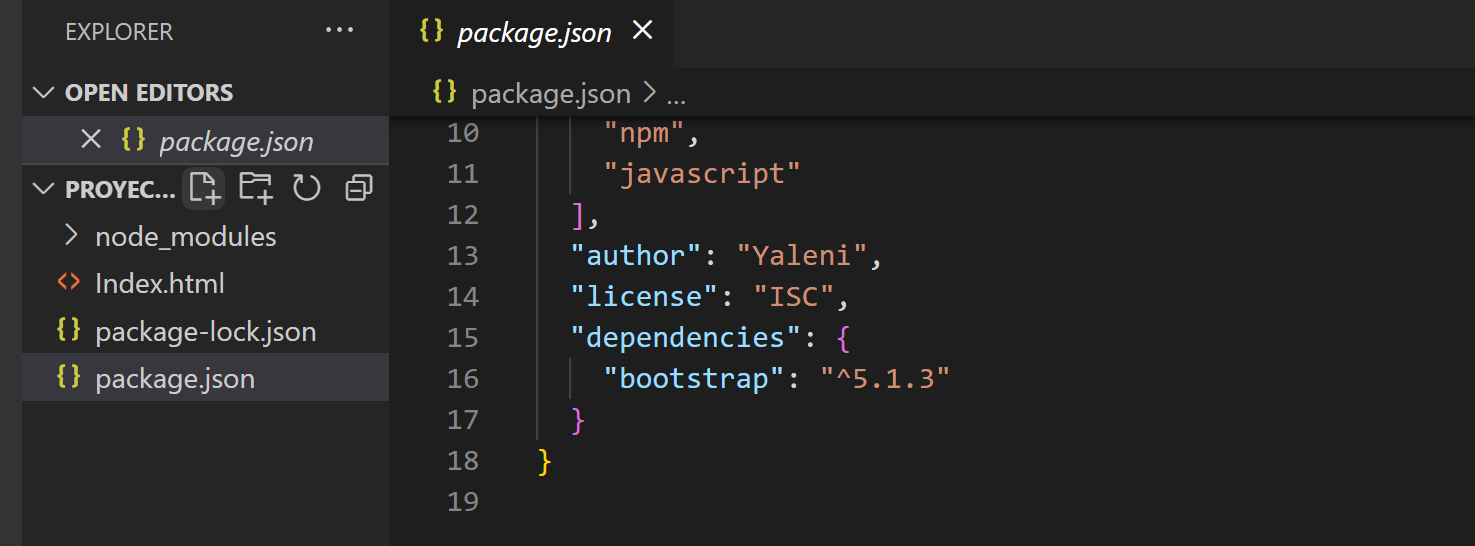
**Resultado**



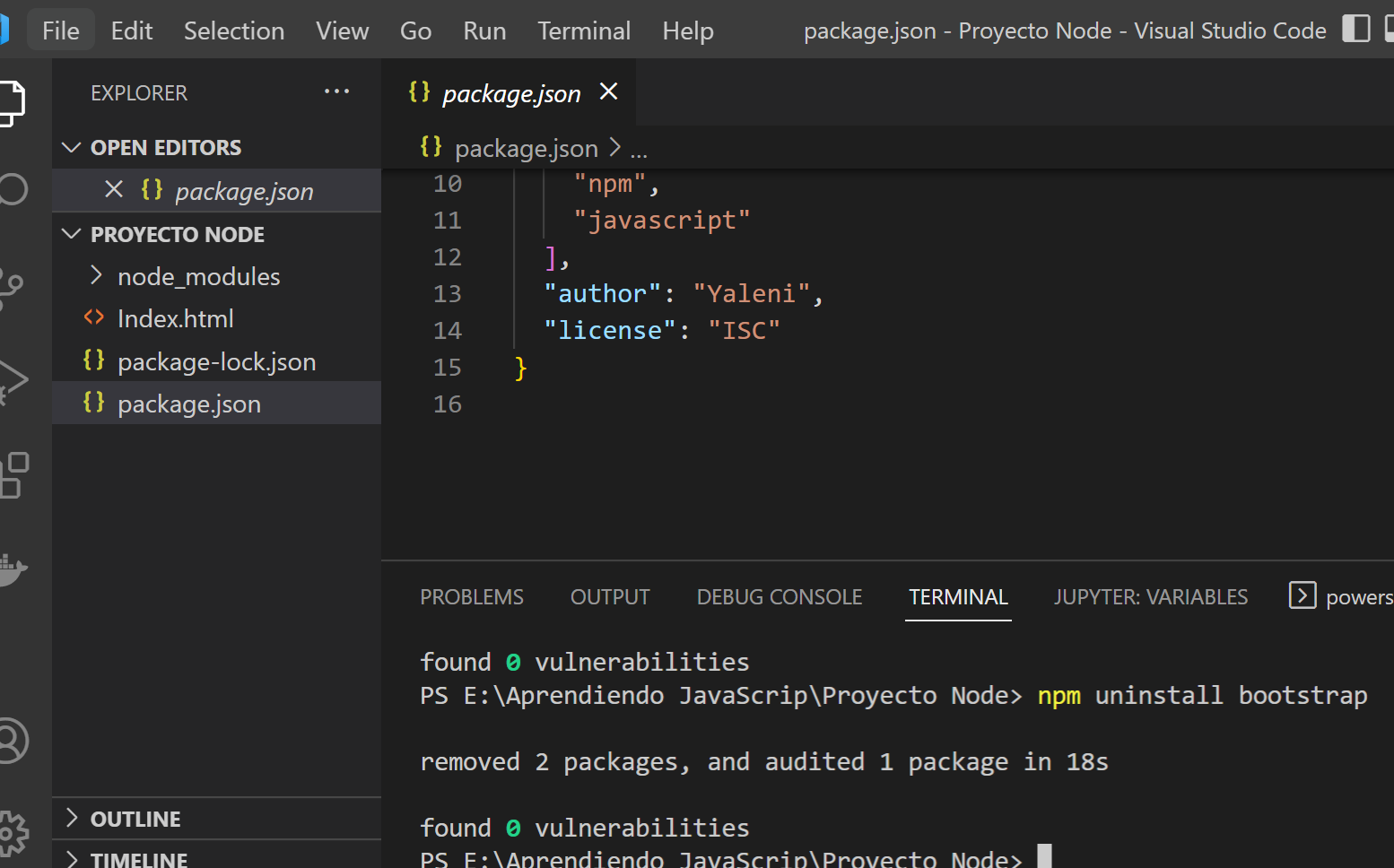
¿Cuáles son las principales propiedades del archivo package.json?

* name (nombre)
* version number (version)
* dependencies (dependencias)
* license (licencia)
* scripts (comandos)

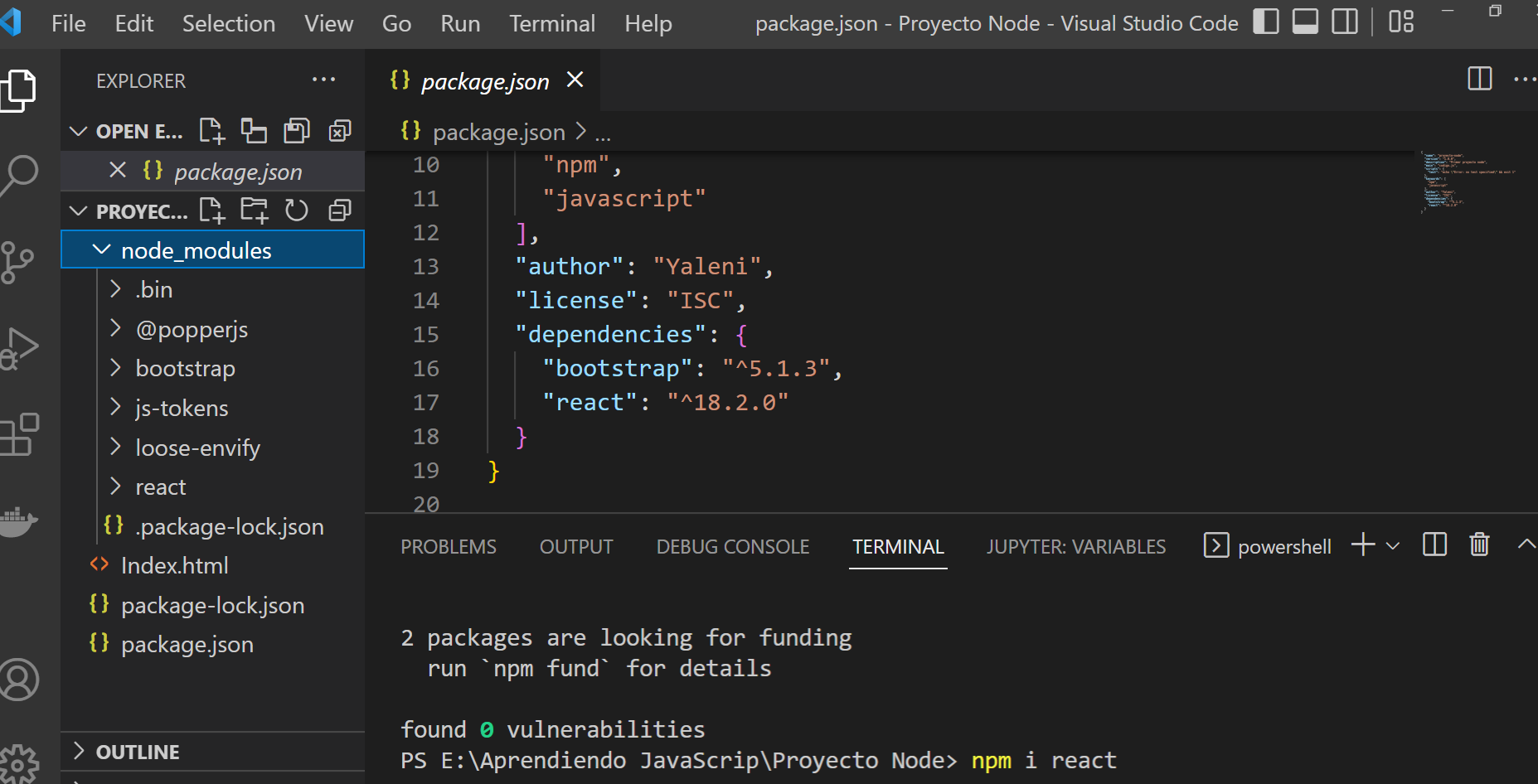
Instalando Bootstrap y si revisamos el archivo package.json nos muestra los cambios que ha sufrido nuestro proyecto se agregó dependencia Bootstrap



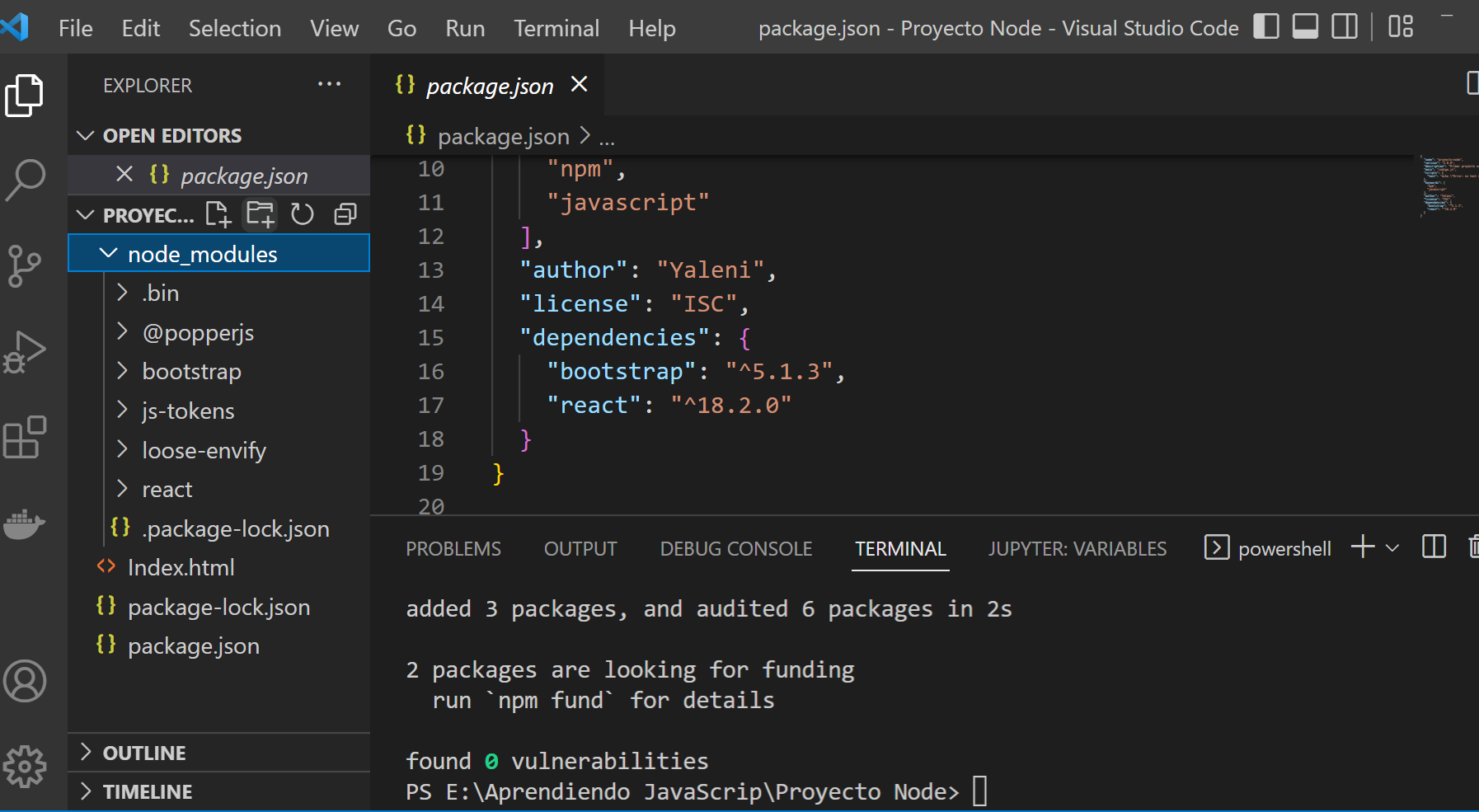
Eliminando Bootstrap con el comando **npm uninstall bootstrap**



**Instalando react npm i react**



Ahora se encuentra instalado Bootstrap y React

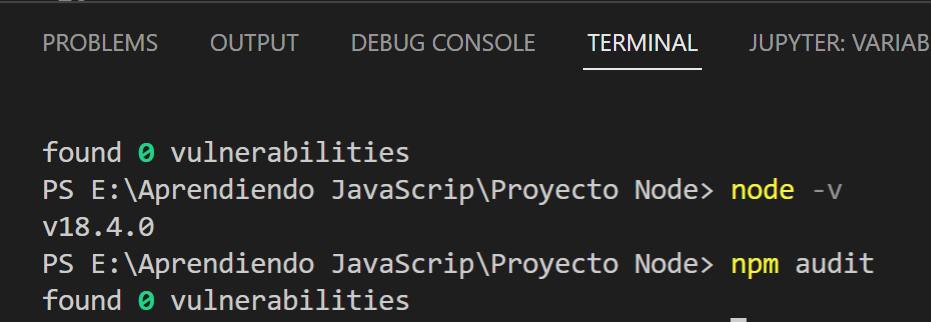


**Automatizar tareas con node.js**

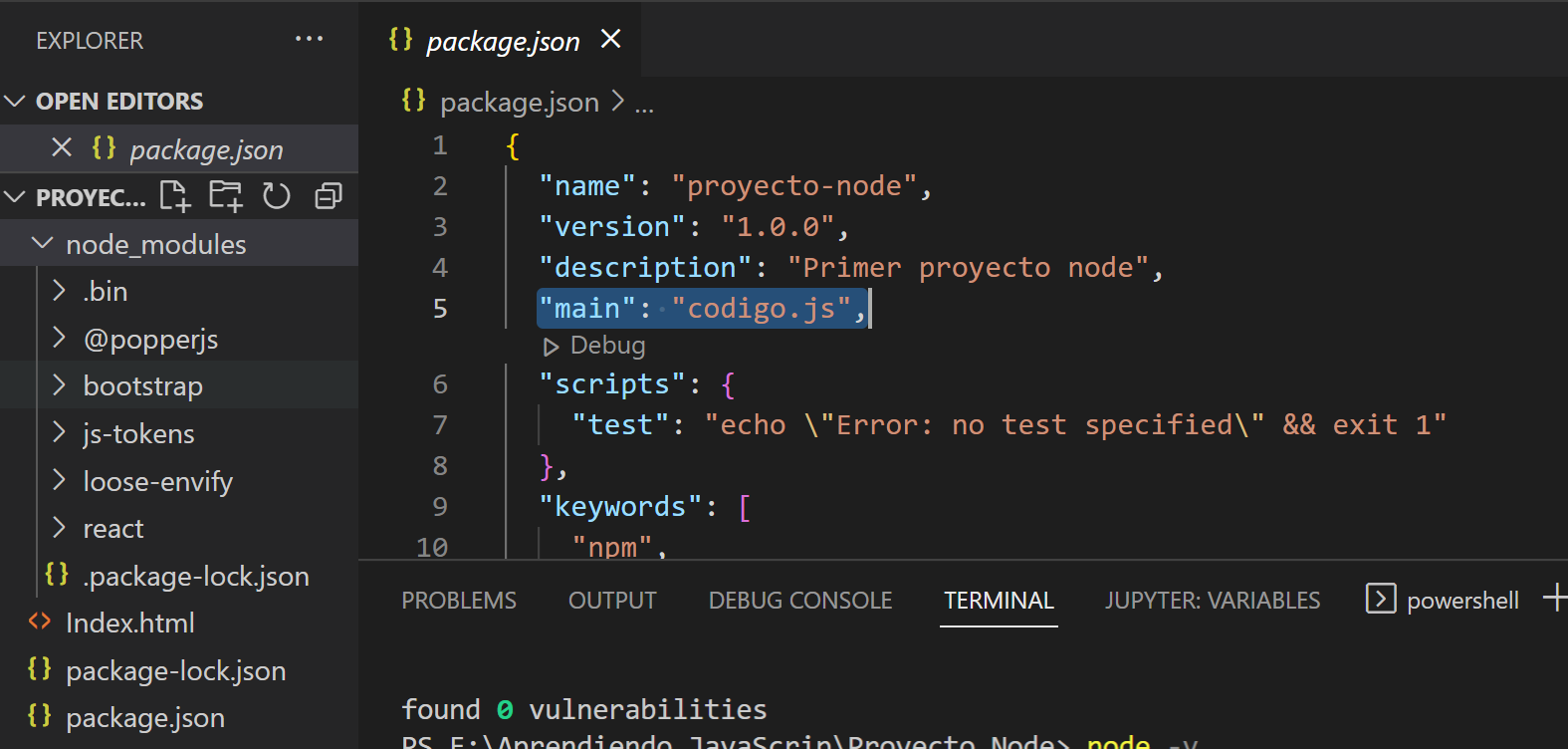
Comando node –v muestra la versión de node.

npm audit muestra las vulnerabilidades que hay en nuestro

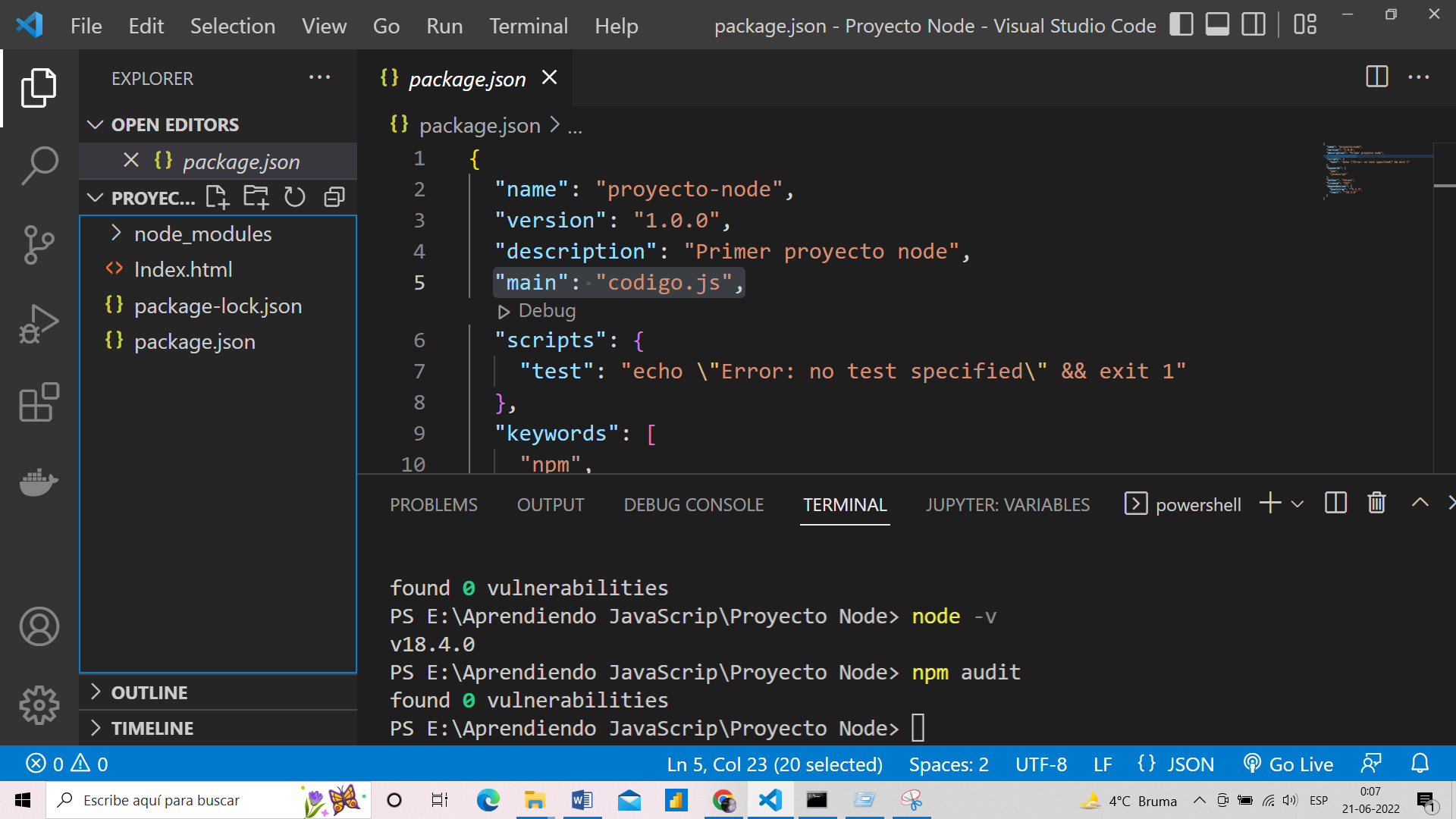
proyecto.



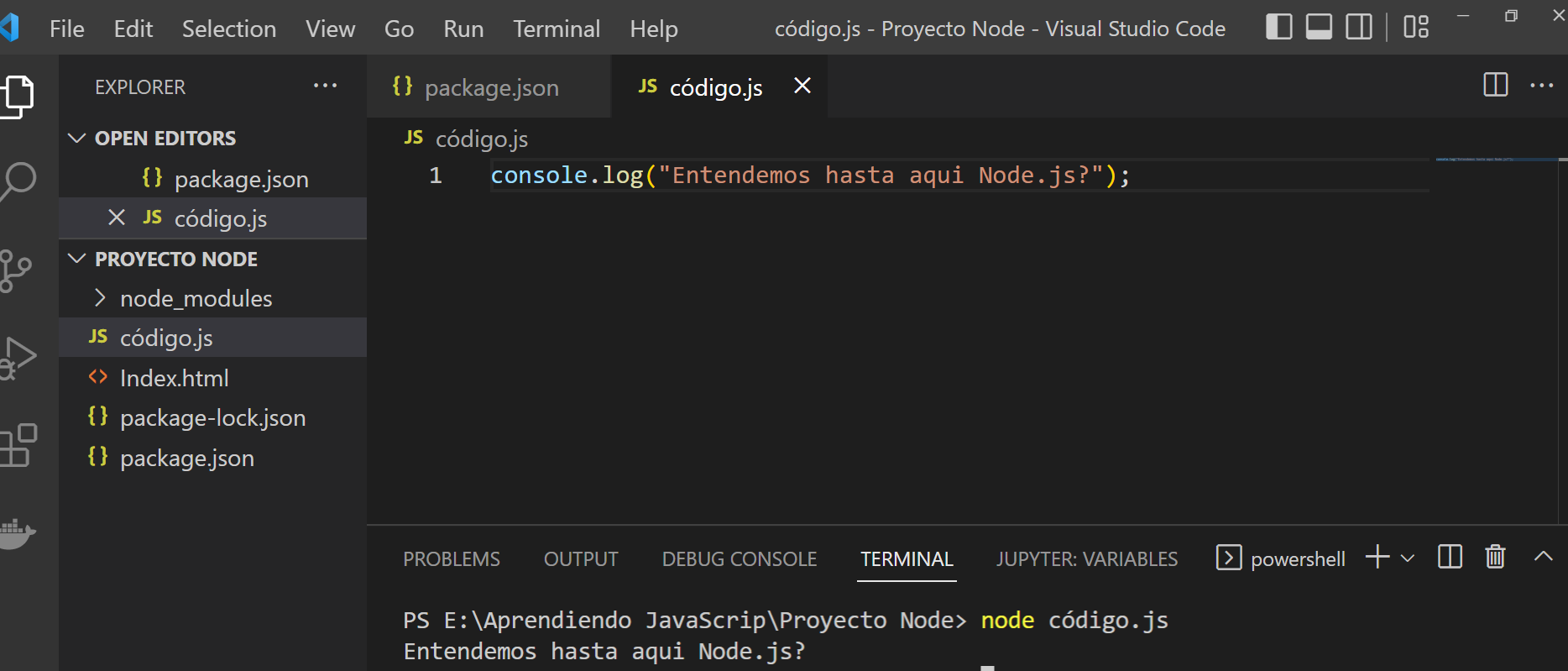
Ahora vamos a ejecutar el archivo principal código.js



Pero el archivo no se encuentra



Creando un archivo “código.js” en el cual imprimimos por consola un mensaje

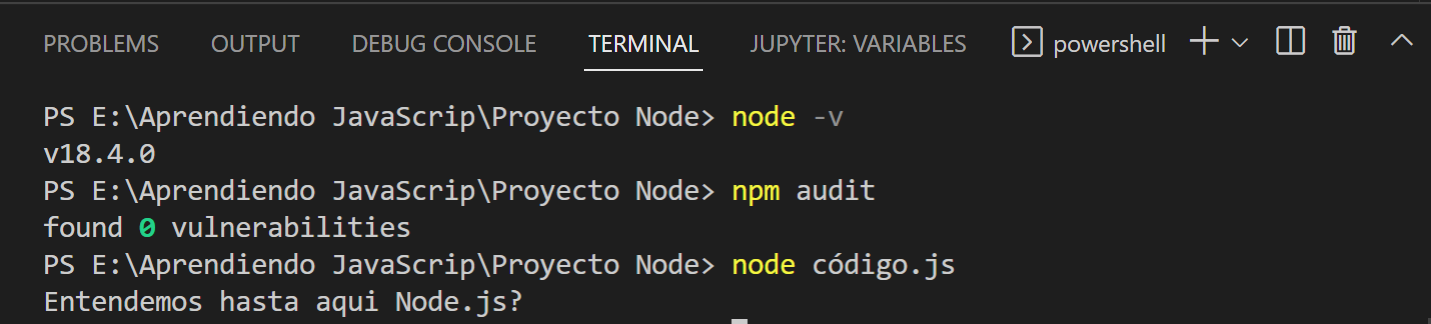


**RESUMIENDO**

.- node -v

.- npm audit

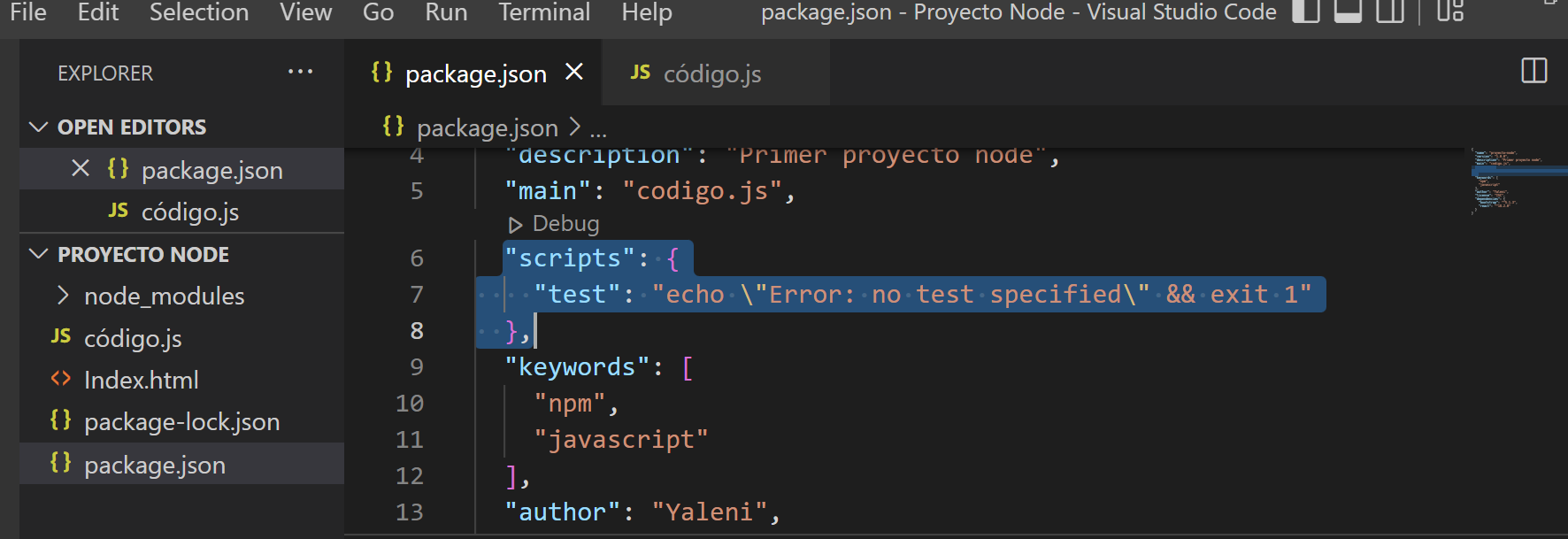
.- node código.js



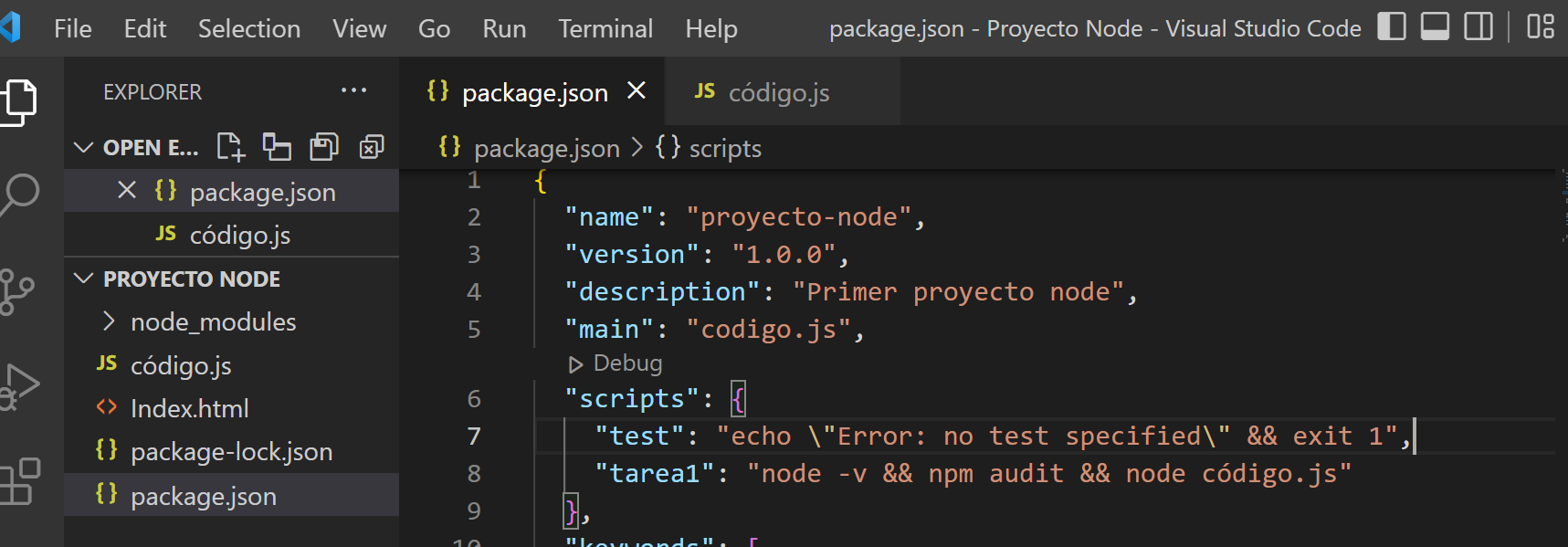
Ahora estas 3 tareas las uniremos y la dejaremos como una sola

Vamos al archivo package.json

Dentro de la línea 6 “script” en especial debajo de test agregaremos una **tarea1**



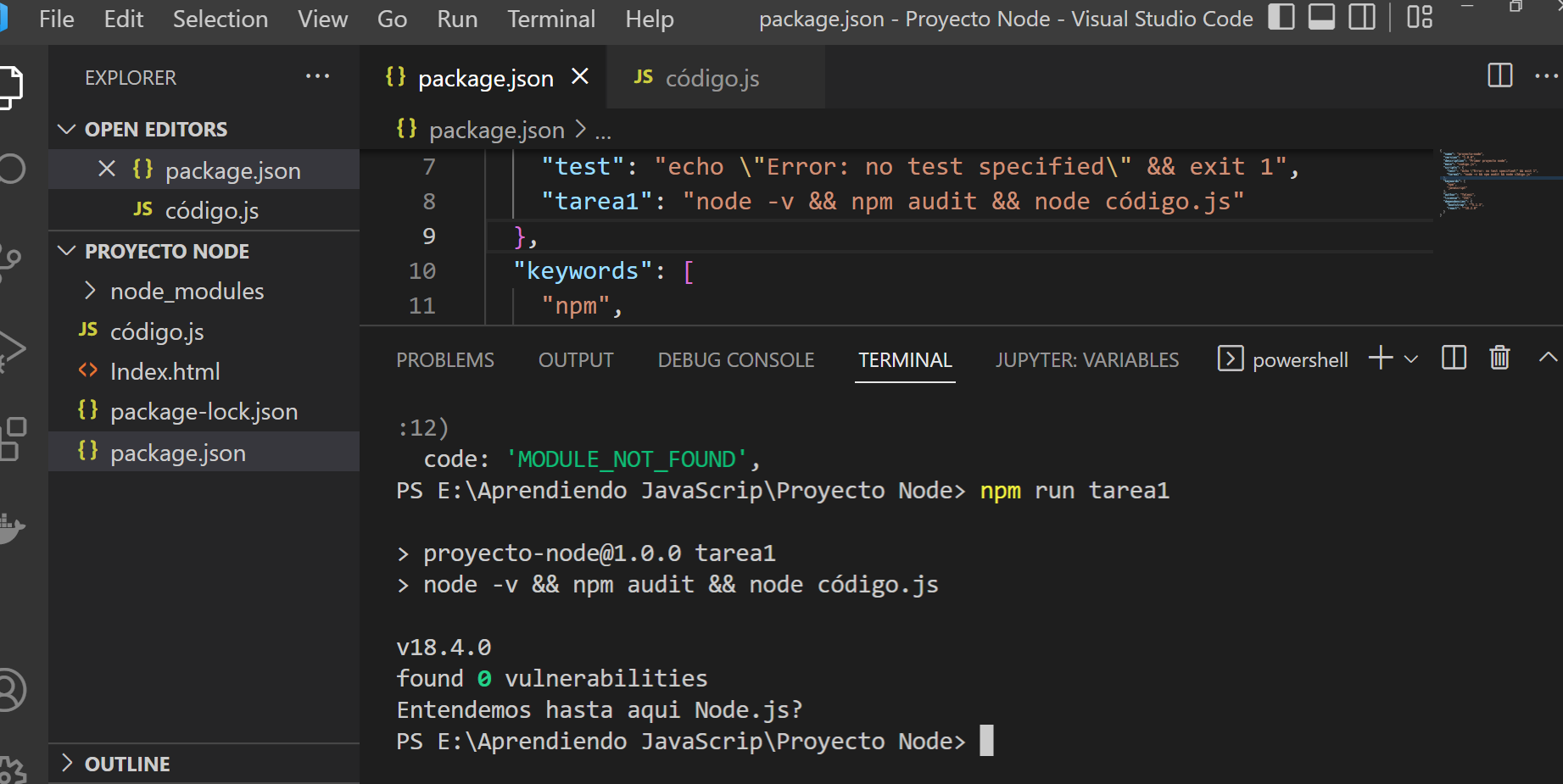
Creando **tarea1**



Ejecutando la tarea1

Digitamos el siguiente comando

**npm run** **tarea1**

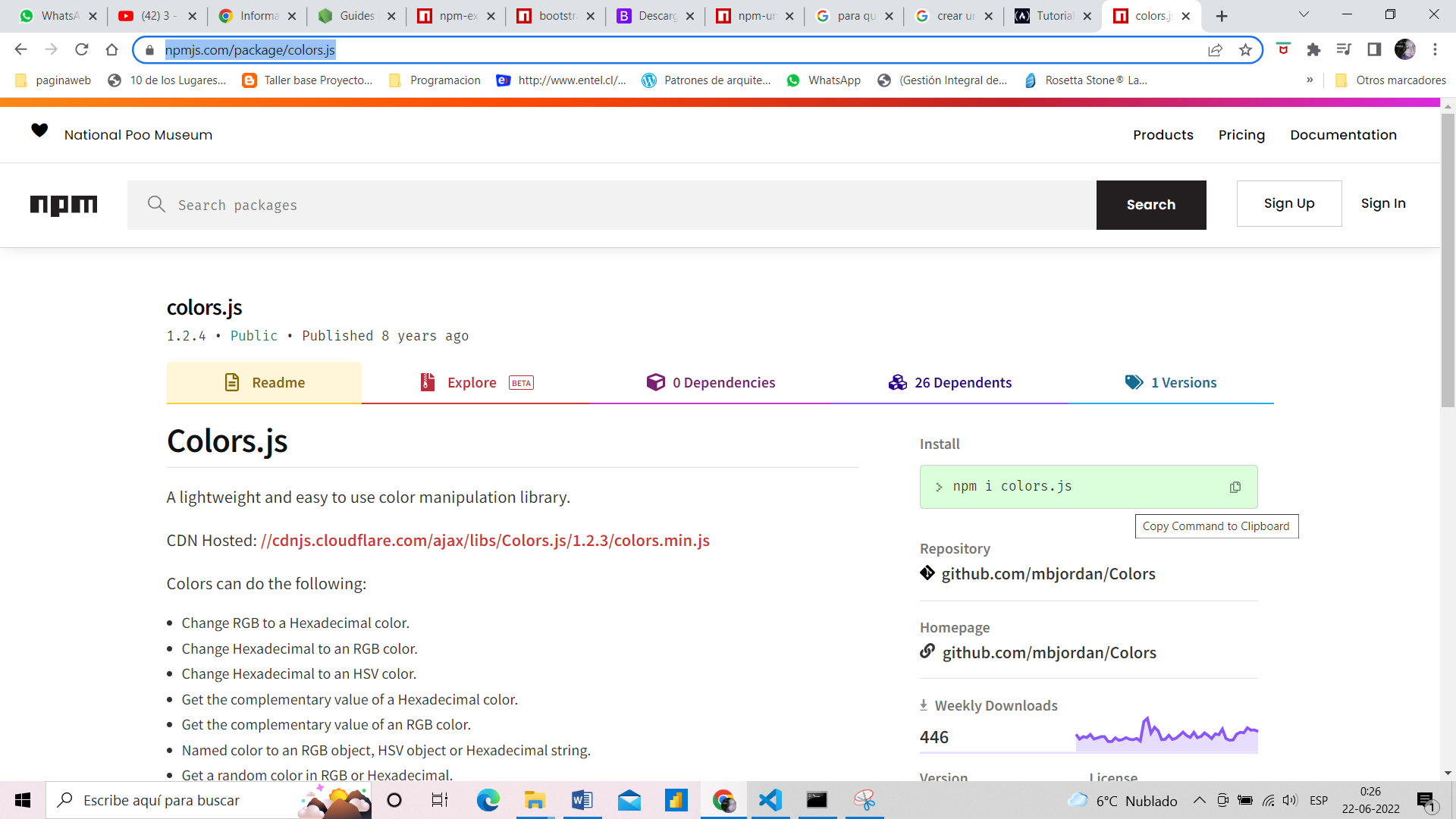


**JAVASCRIPT Requires and Export**

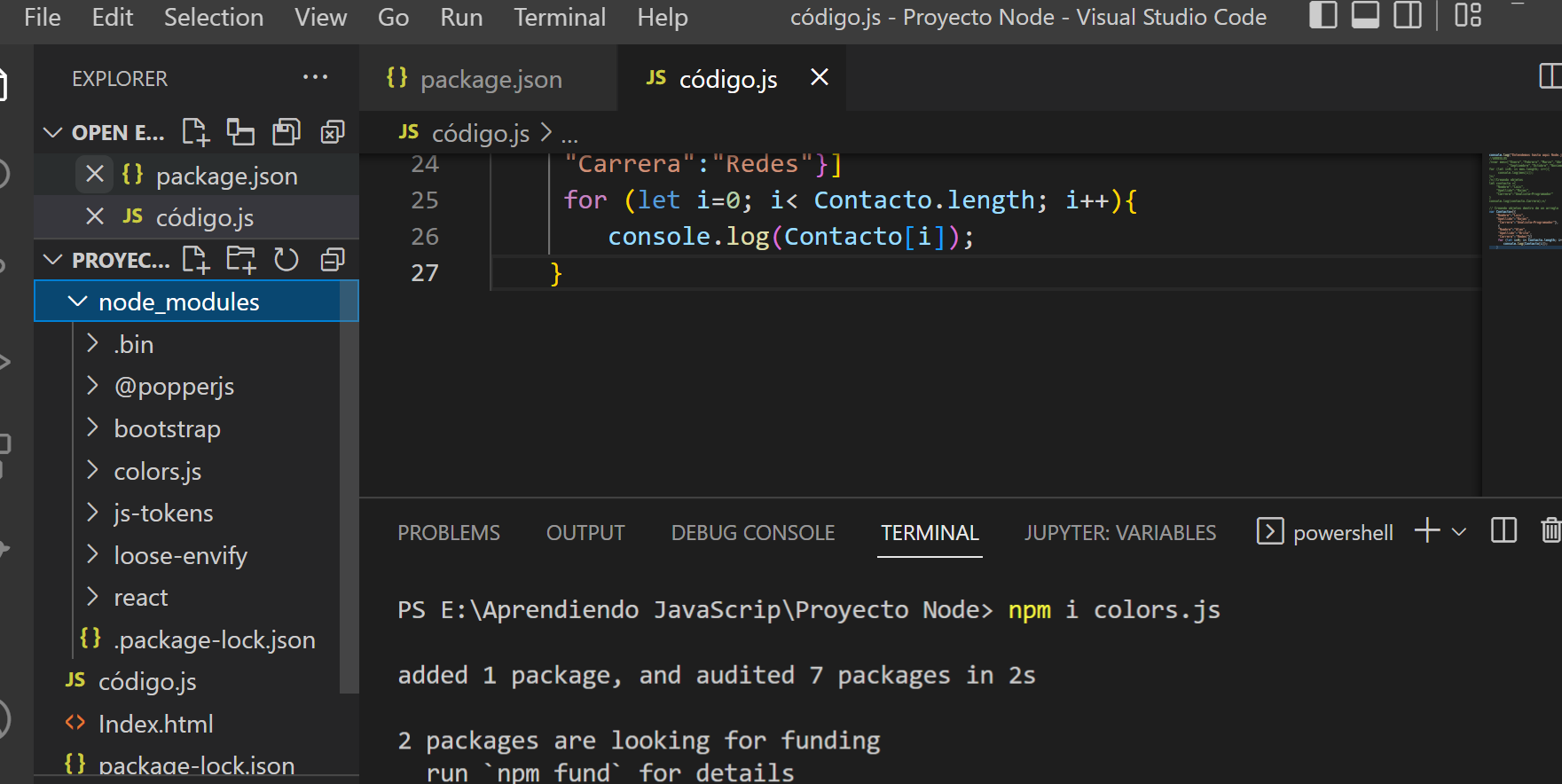
Palabras reservadas **require**

.- 1ero instalaremos un paquete **colors.js**

<https://www.npmjs.com/package/colors.js>



Copiamos el comando **npm i colors.js,** después lo ejecutamos en la consola de visual code.

Resultado:

Como funciona Require

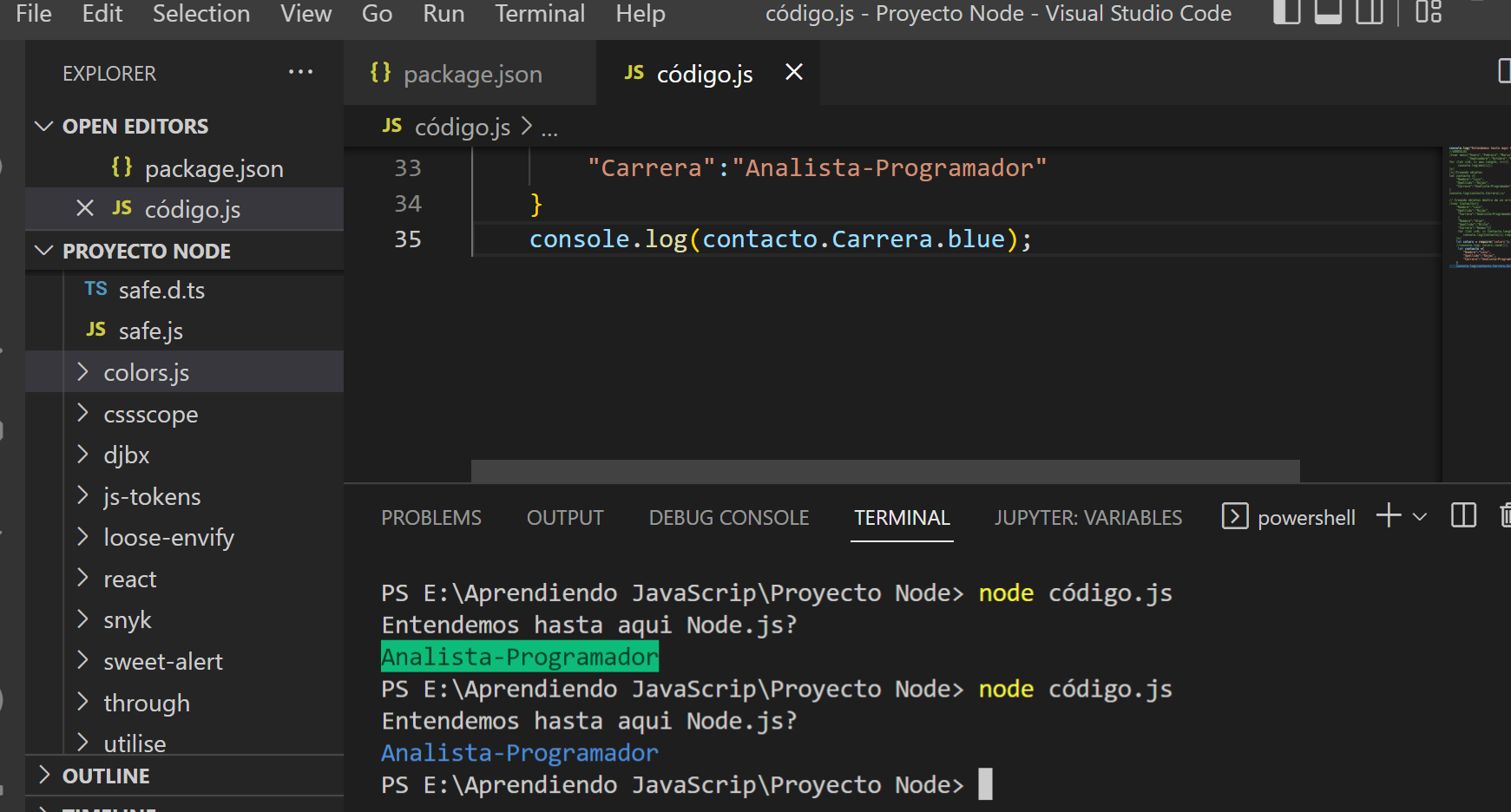
1.- Creamos una variable con el mismo nombre de la librería “Colors”

1.- Creamos una variable con el mismo nombre de la librería **Colors**

2.- Seteamos la variable creada con la función require('colors')

3.- creamos un objeto llamado contacto { } con sus respectivos clave-valor





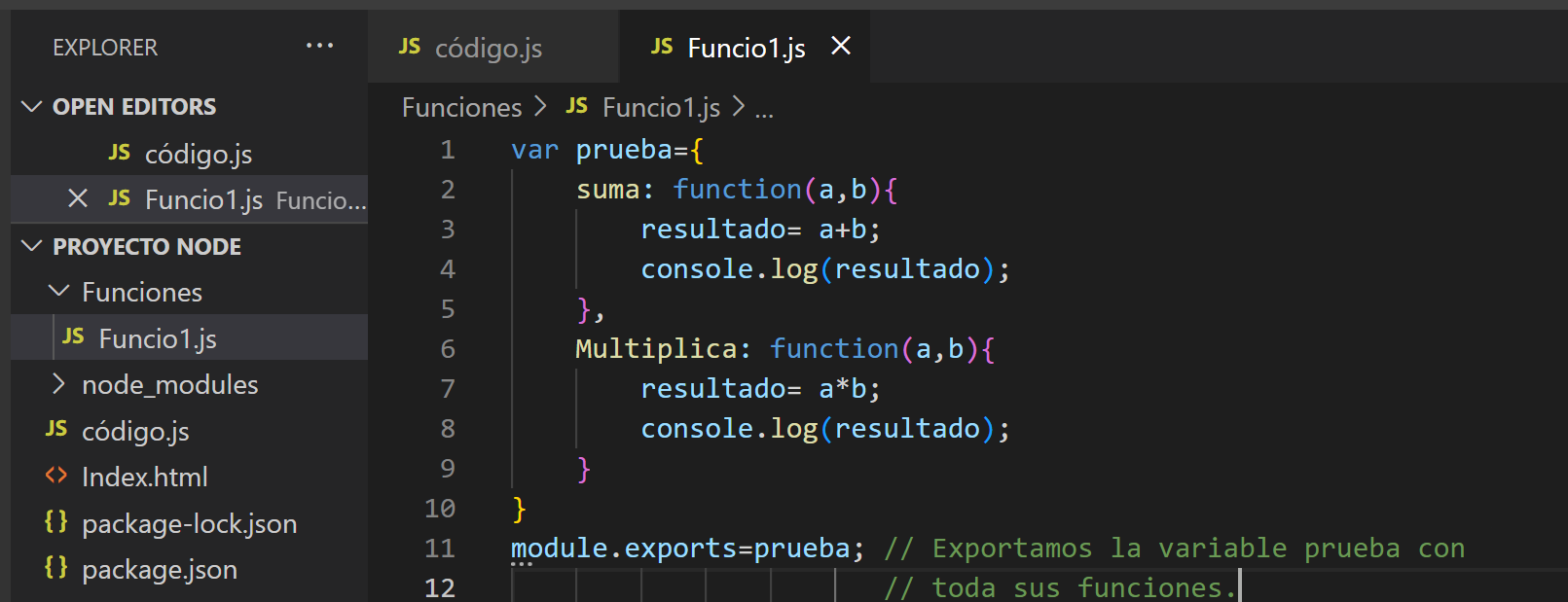
**Utilizando Export**

1.- Creamos una carpeta llamada **Funcion**

2.- Creamos un archivo llamado **Funcion1.js**



En el archivo **Funcion1.js** escribimos el siguiente código



**Nota: para poder utilizar el código que hemos escrito, debemos exportarlo**

**Con la siguiente línea de código:**

module.exports=prueba; // Exportamos la variable prueba con

                       // toda sus funciones.

Luego debemos ir al archivo principal llamado **“codigo.js”**

Debemos agregar la siguiente línea de comando:

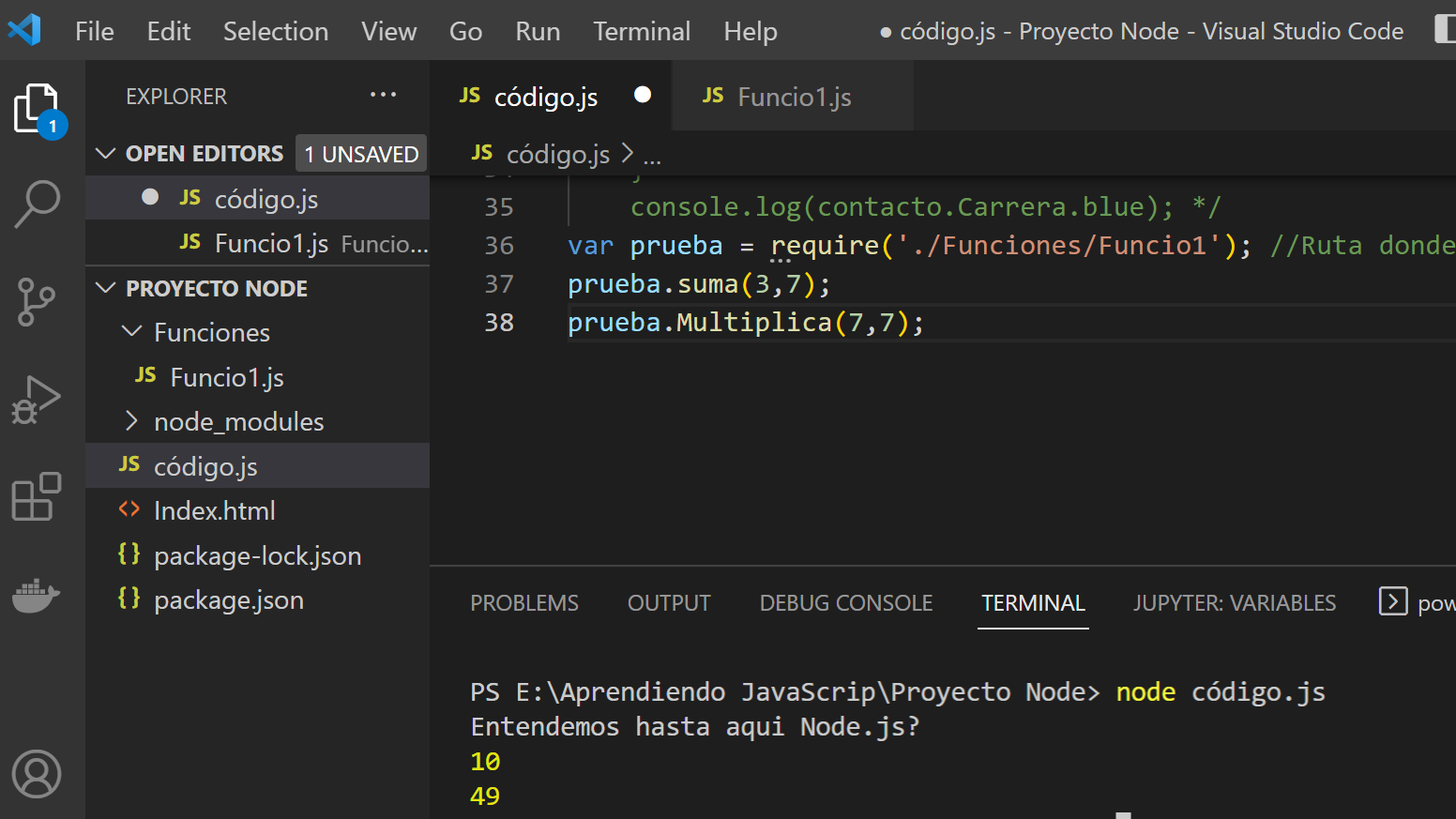
var prueba = require('./Funciones/Funcio1'); //Ruta donde se encuentra el archivo

prueba.suma(3,7);

prueba.Multiplica(7,7);

Posteriormente en la consola debemos ejecutar el código con la siguiente línea de comando:

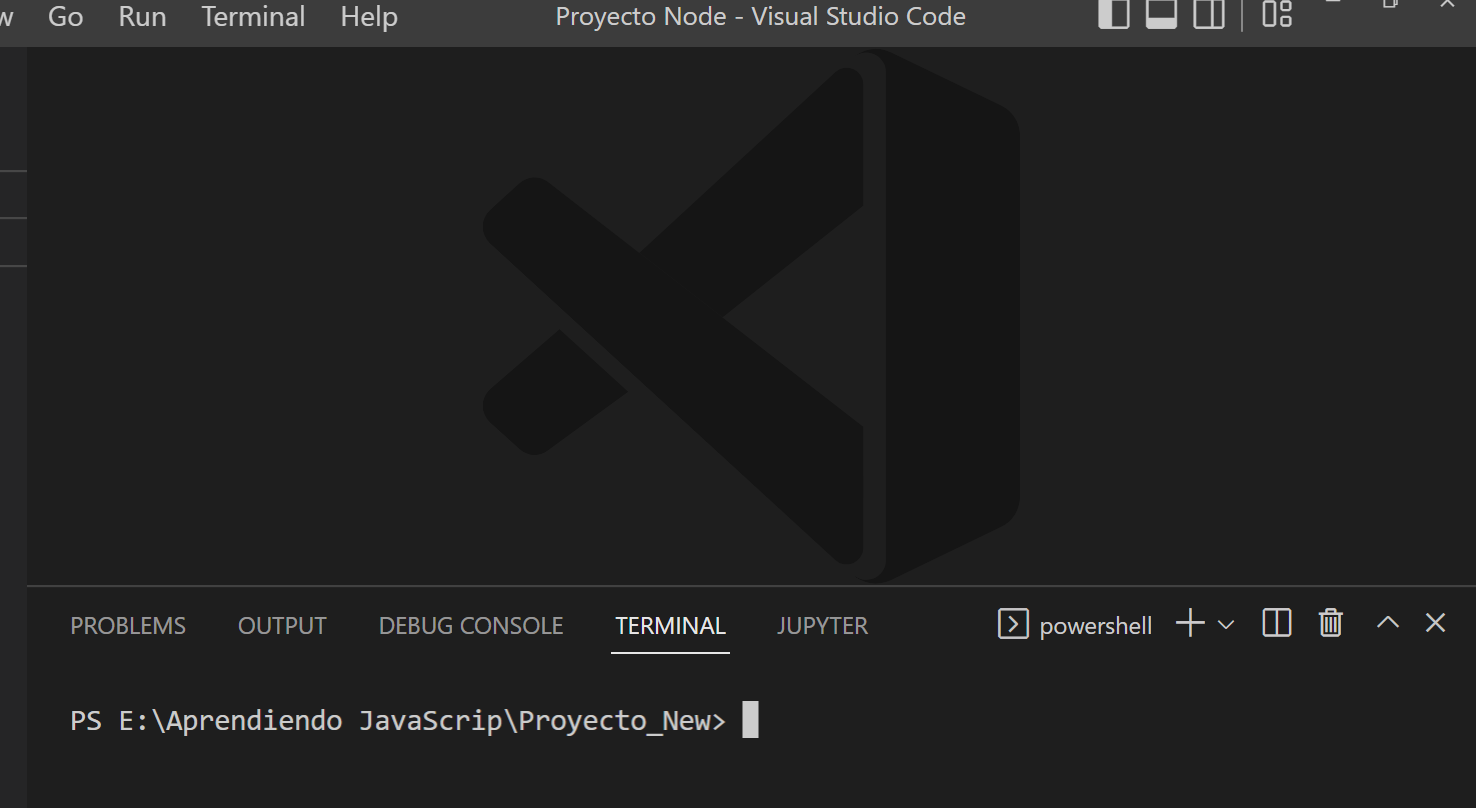
**node código.js**



Creando un proyecto desde Cero **“Servidor Web Local”,** pueden ser servidor de correo etc. (Cliente Servidor)

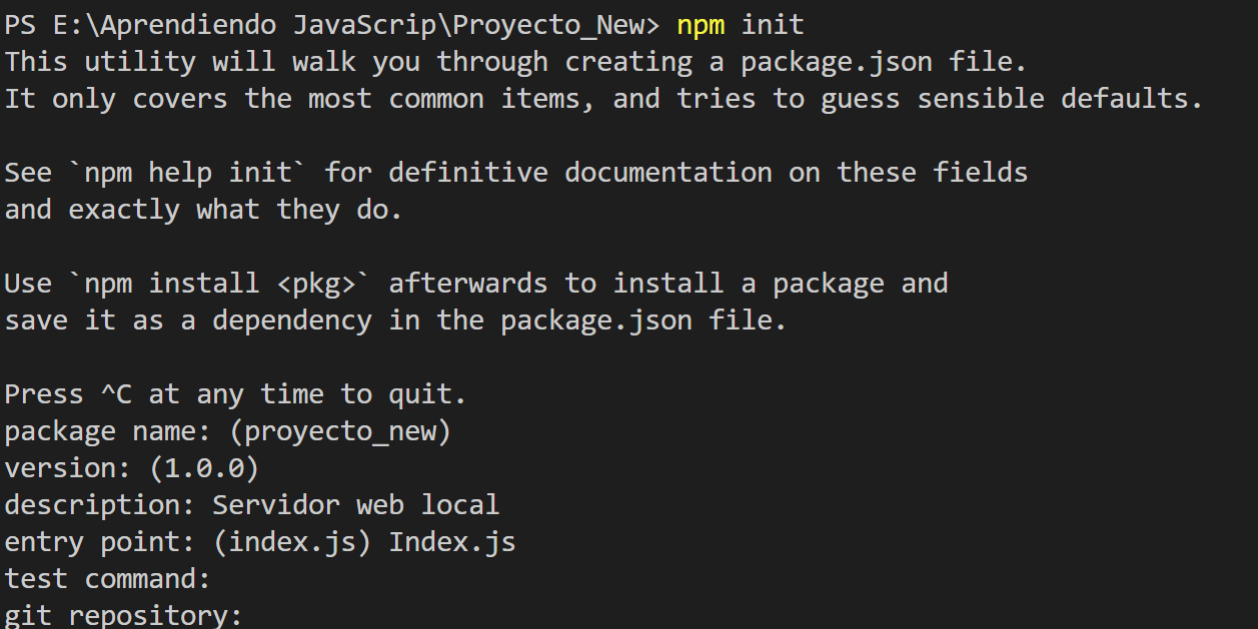
.- Abrir Visual Code

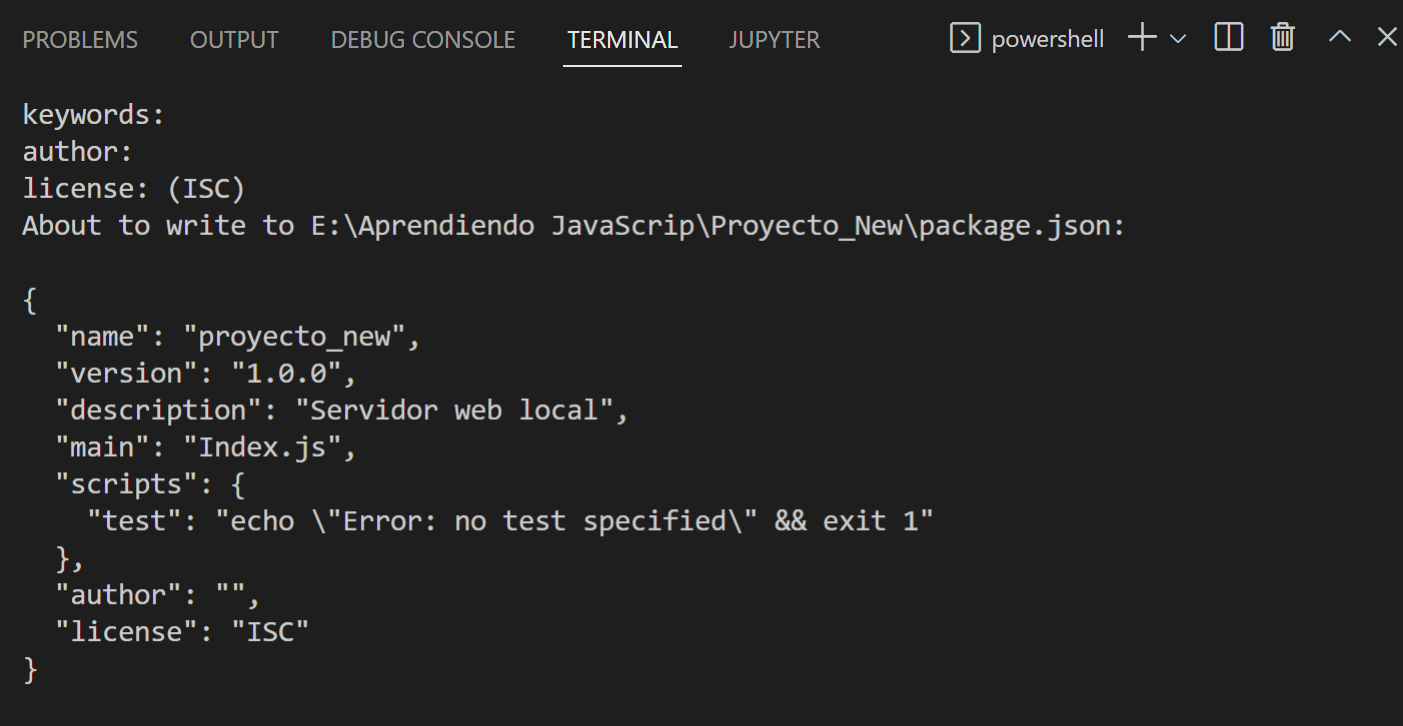
.- Abrir consola e indicar nueva terminal

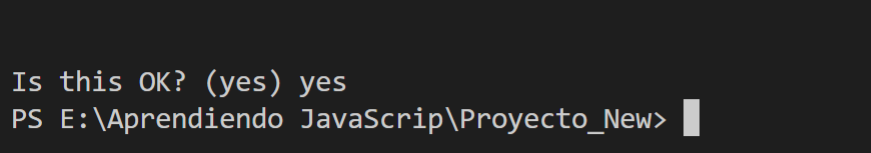


Ejecutar el siguiente comando **npm init**



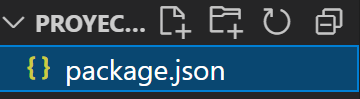




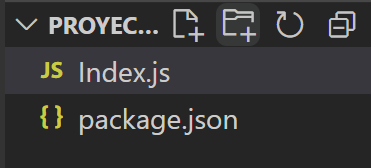


Creamos un archivo principal llamado **Index.js**

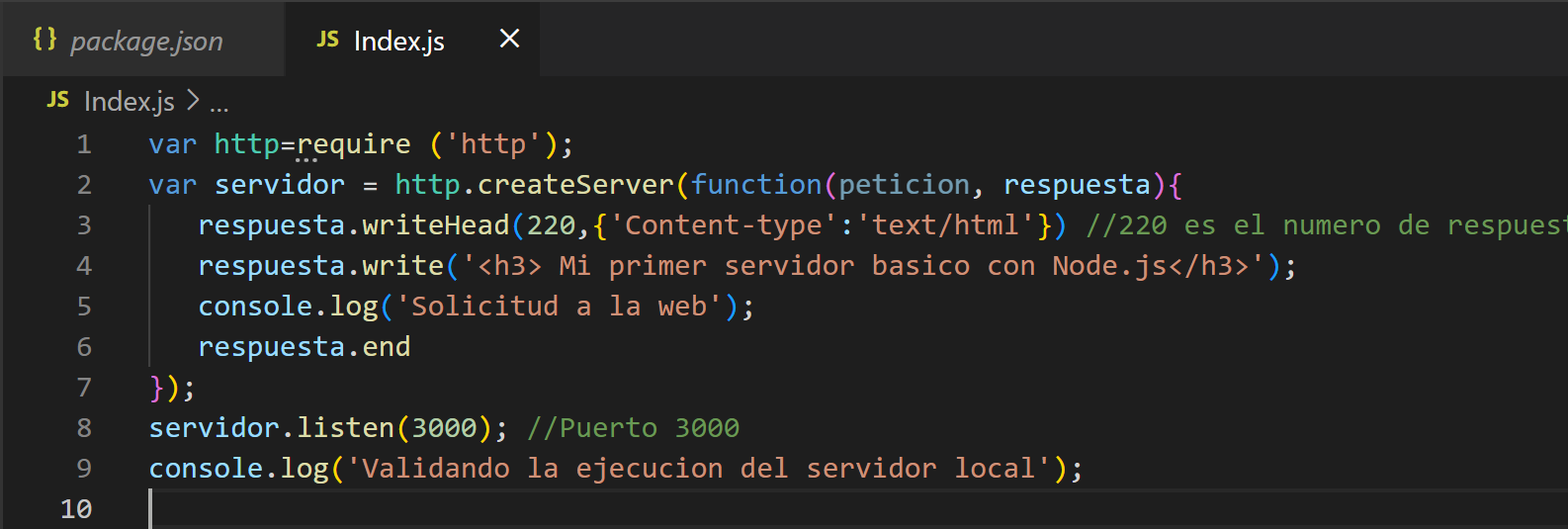
**Antes**



**Despues**

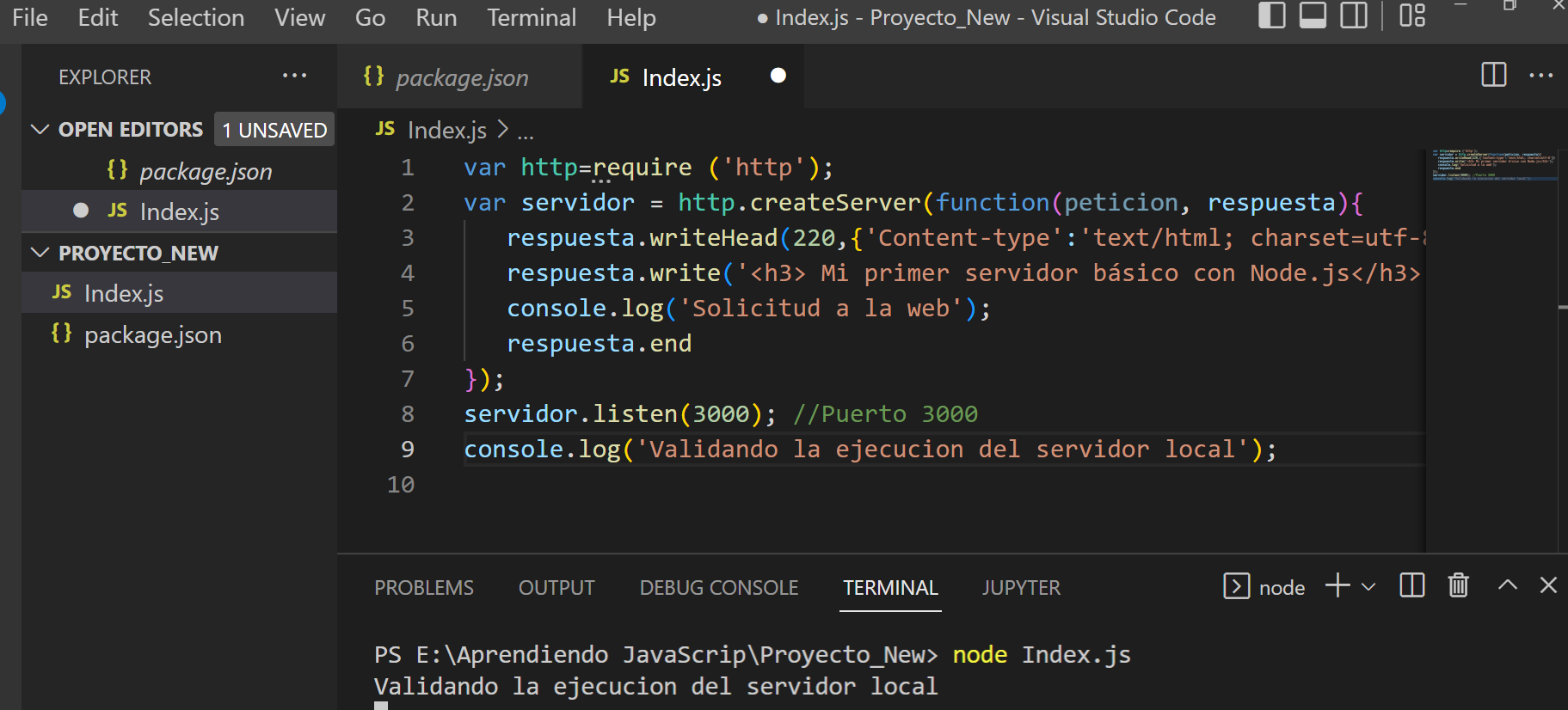


Una vez creado el archivo escribimos la siguiente línea de código:



Guardamos y ejecutamos el código con la siguiente línea de comando :

**node Index.js**



Ahora vamos hacer una petición web en el browser:

Abrimos Google Chrome => escribimos localhost:3000 (puerto)

Resultado:

