Enlaces:

https://nodejs.org/es/

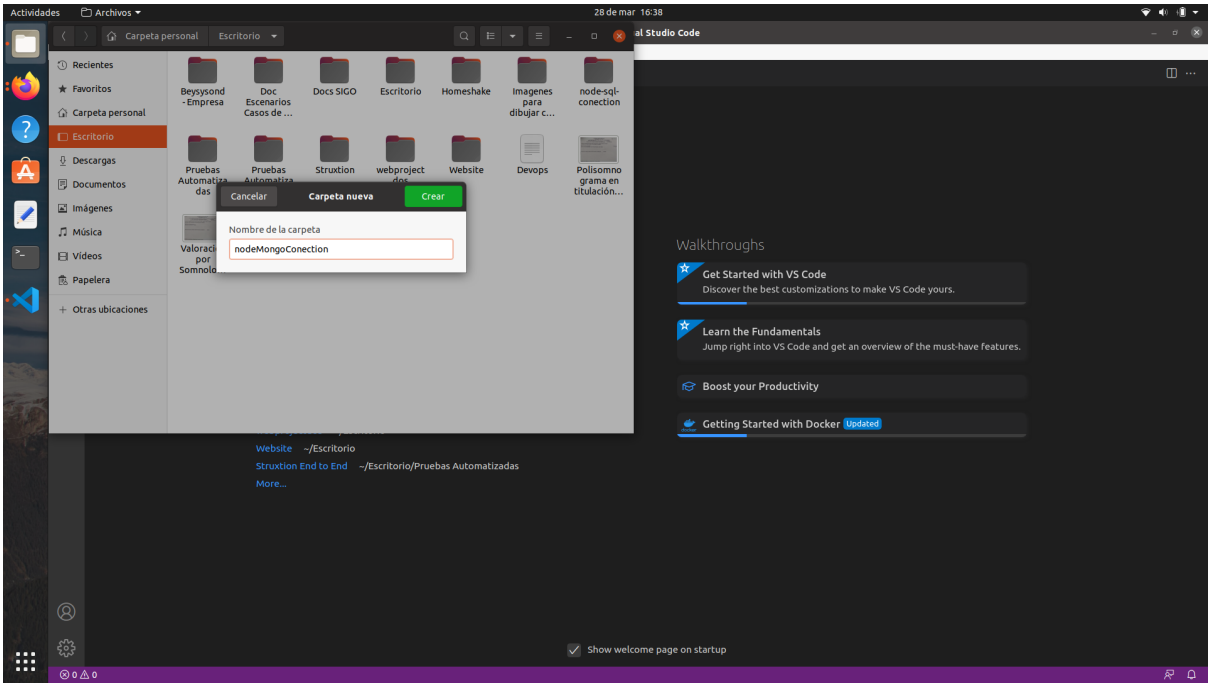
Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

https://www.mongodb.com/es

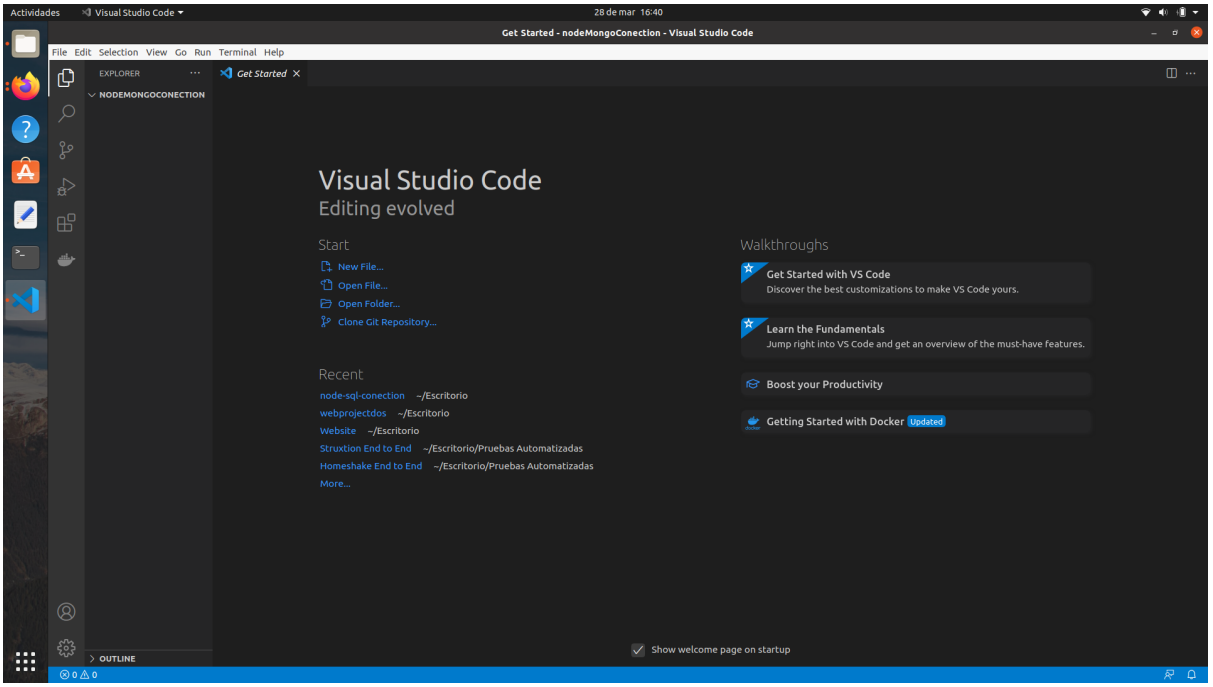
MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

Node, Mongo & Docker Compose

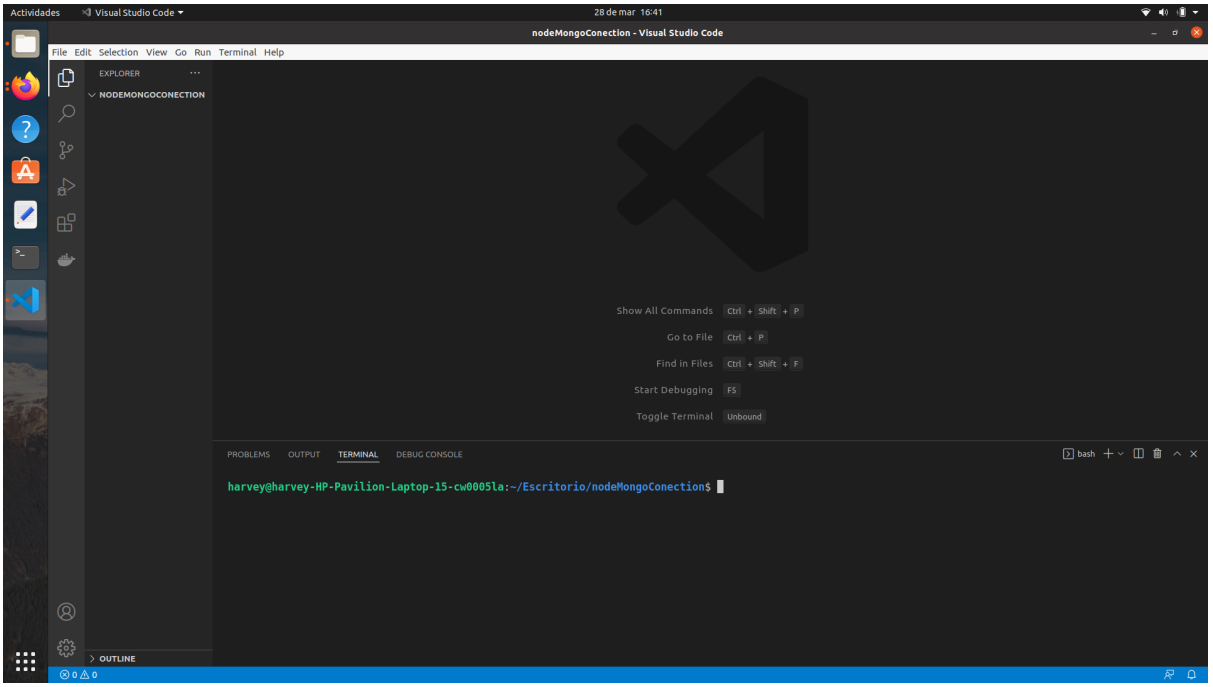
1. Crear una nueva carpeta para el proyecto:



2. Abrir la carpeta con Visual Studio Code:



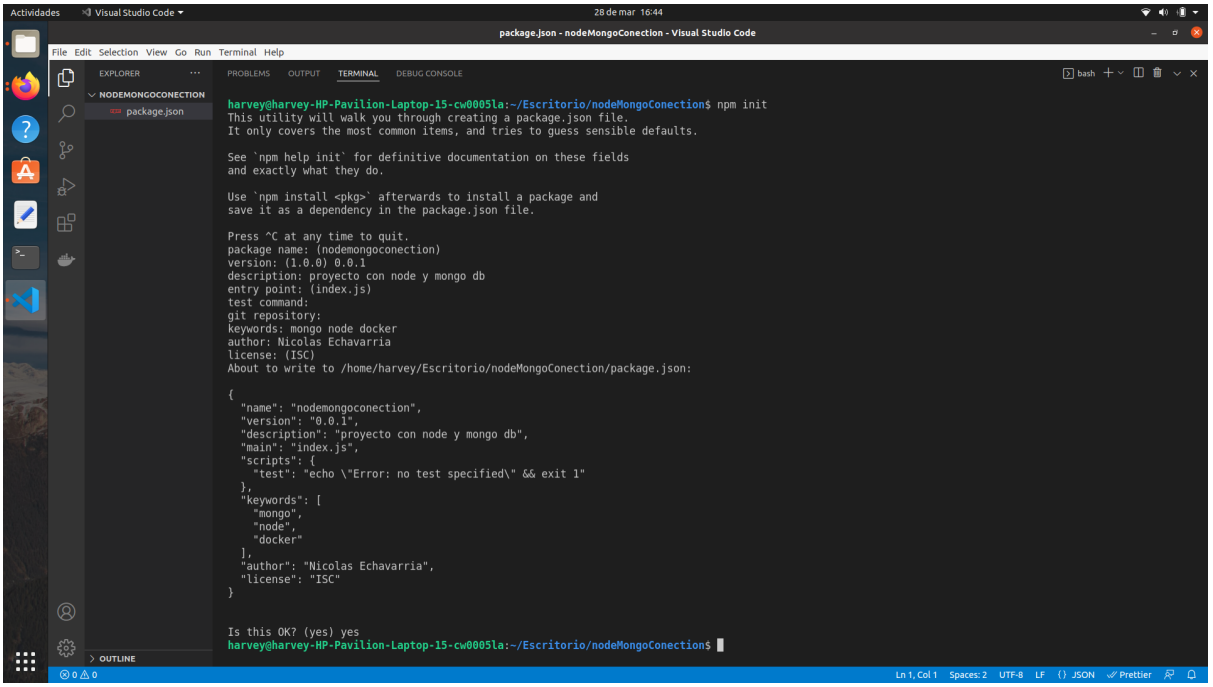
3. Abrir una nueva terminal:

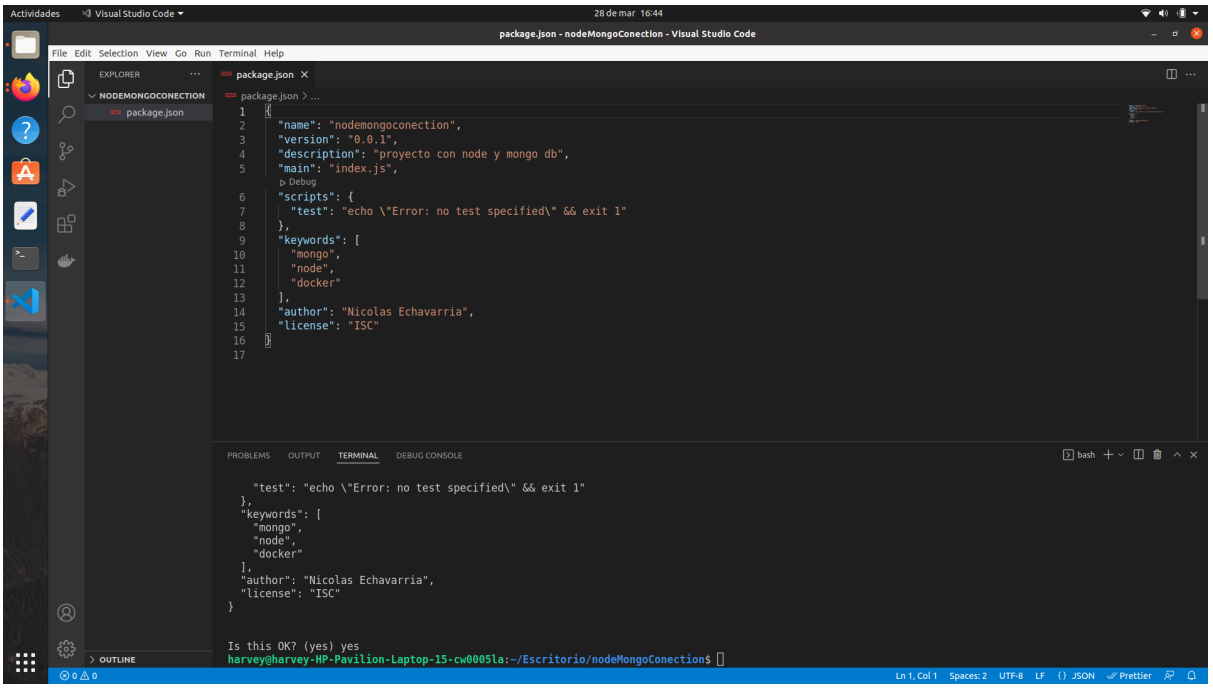


4. Creación del proyecto de node: para ello abrimos una nueva terminal de VisualStudio Code y ejecutamos el comando npm init con el fin de inicializar el proyecto, creando así un archivo llamado **package.json**. Dentro de ese archivo, encontrarás metadatos específicos para el proyecto.

   Los datos que pedirá son:

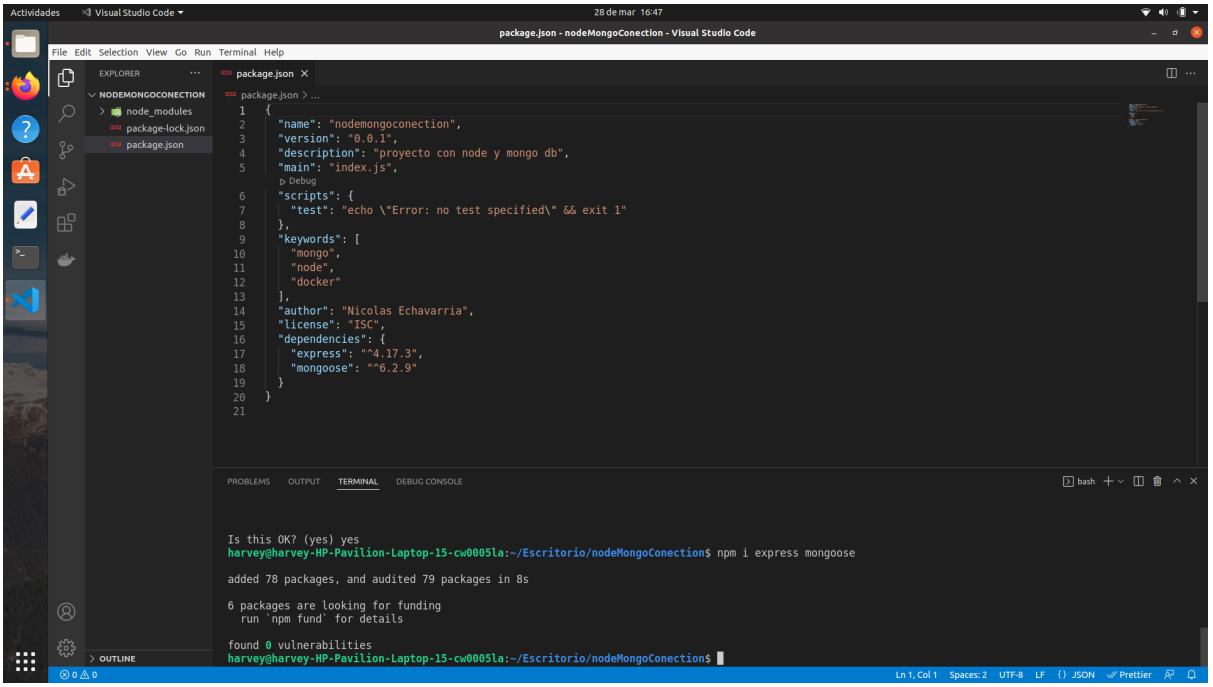
   a) El nombre del paquete
   b) Versión
   c) Descripción
   d) El punto de entrada
   e) Comandos de prueba
   f) Repositorio de git
   g) Palabras Clave
   h) Autor
   i) Licencia



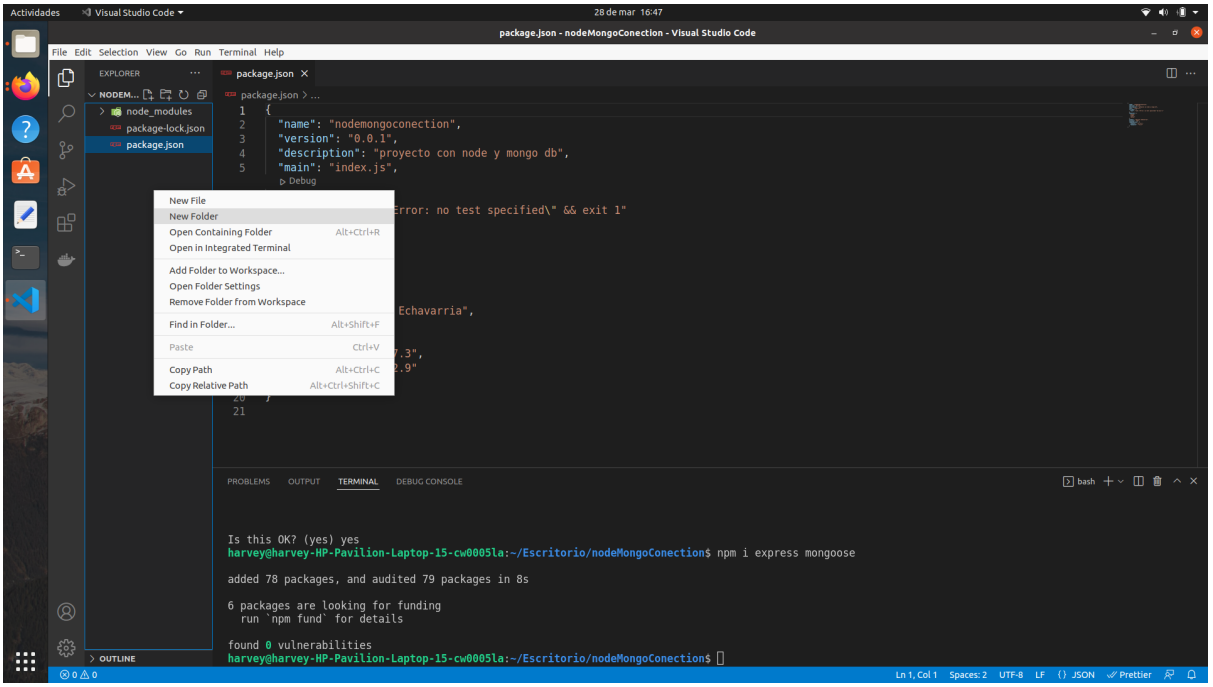


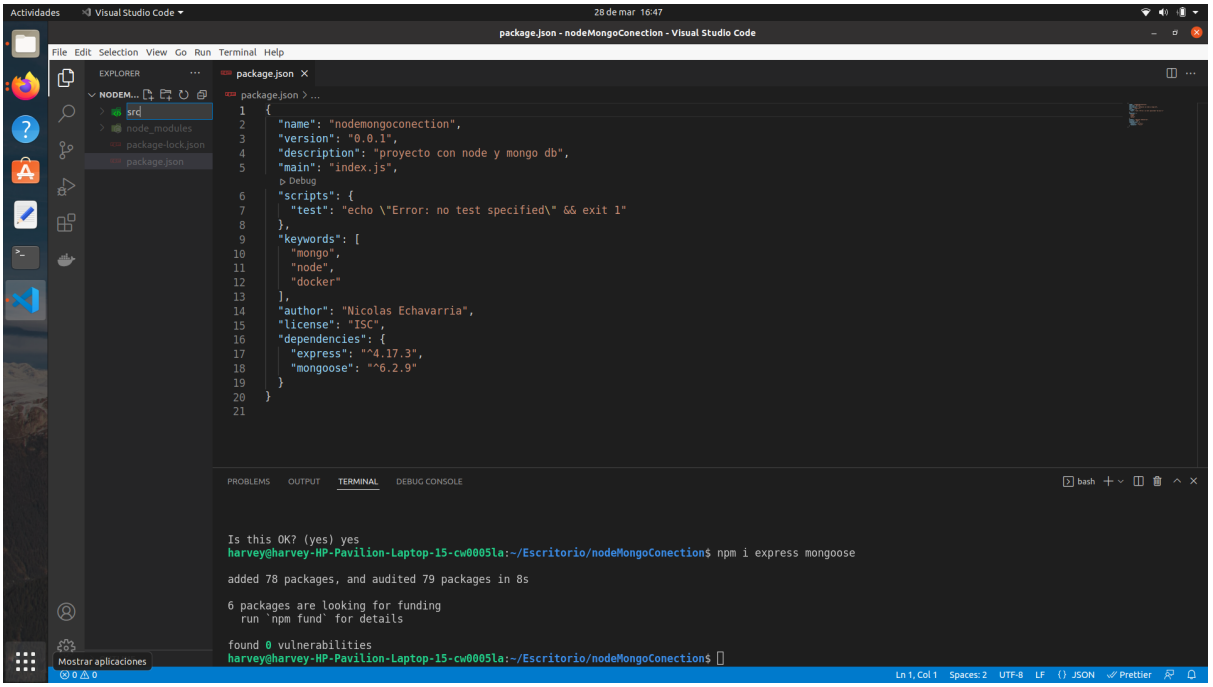
5. Instalar los paquetes de express y mongo con el siguiente comando: npm i express mongoose, mongoose es un módulo que sirve como modelo de conexión para poder acceder a mongo.

6. Una vez instalados, podemos observar que se crea una carpeta de node_modules y los archivos: package.json y package-lock.json.

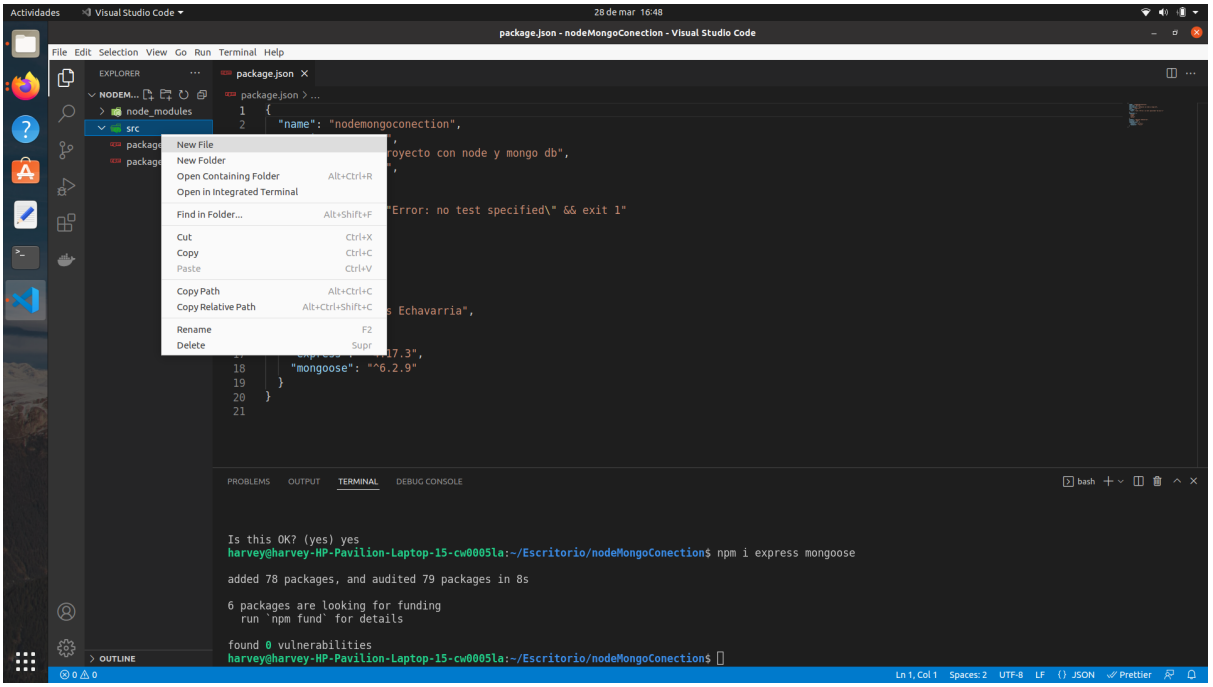


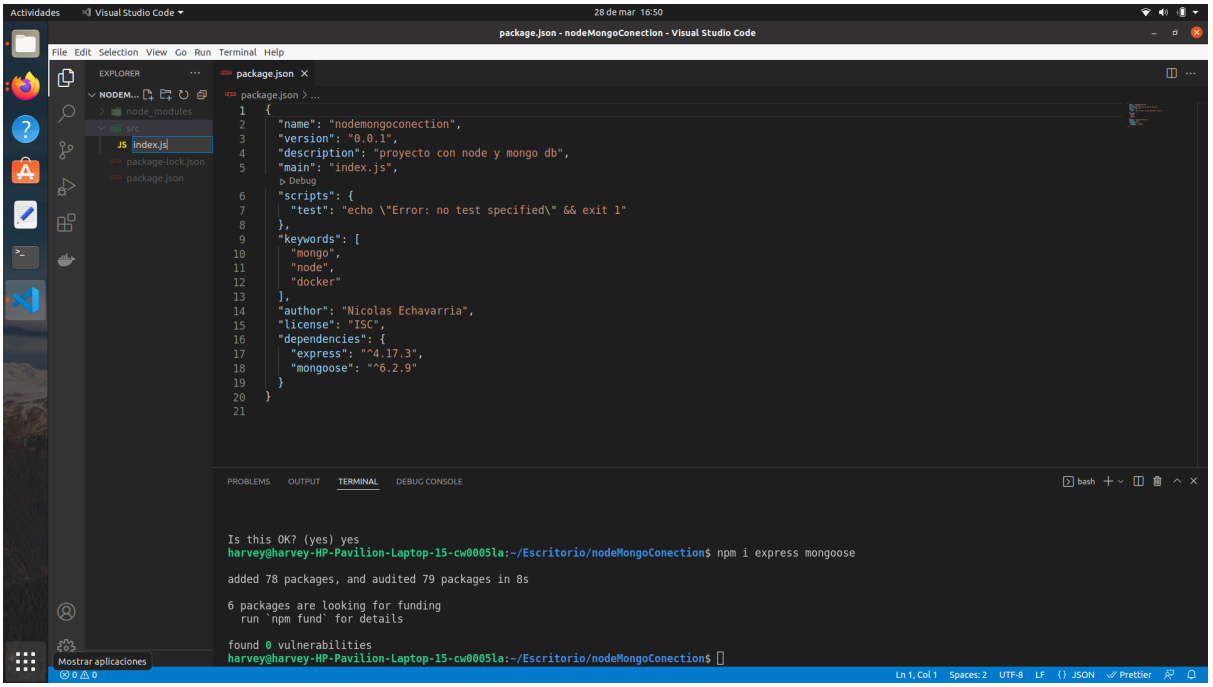
7. Crear una carpeta nueva src para almacenar nuestro código.

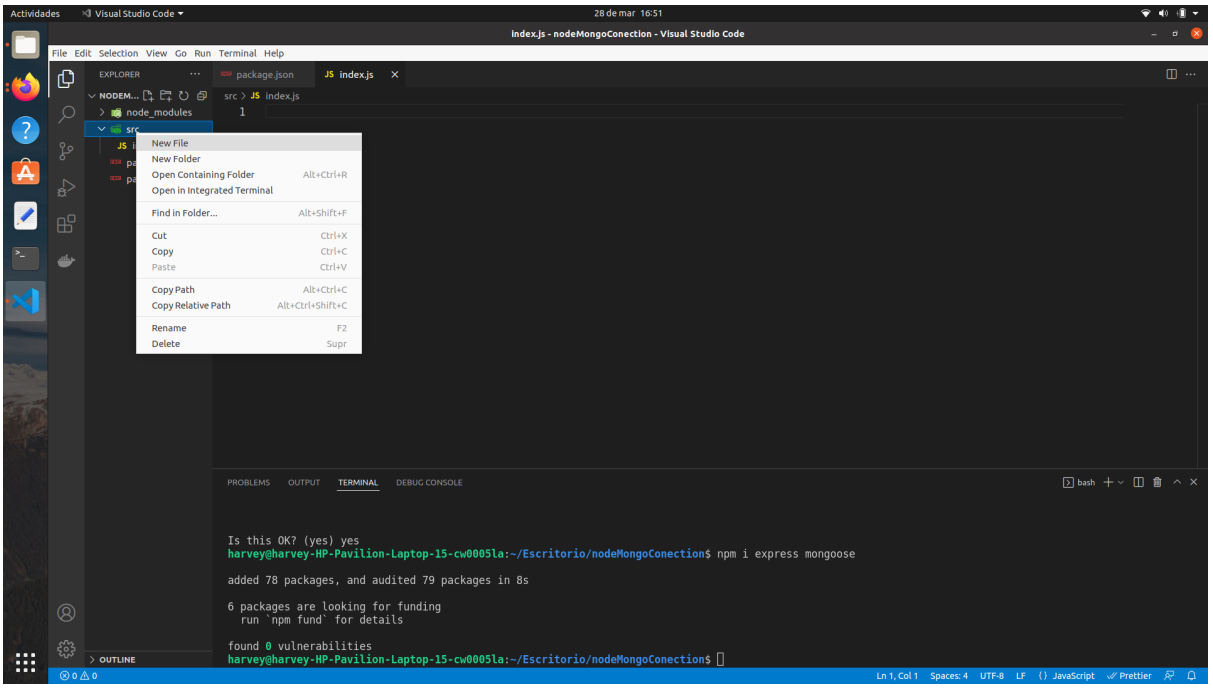


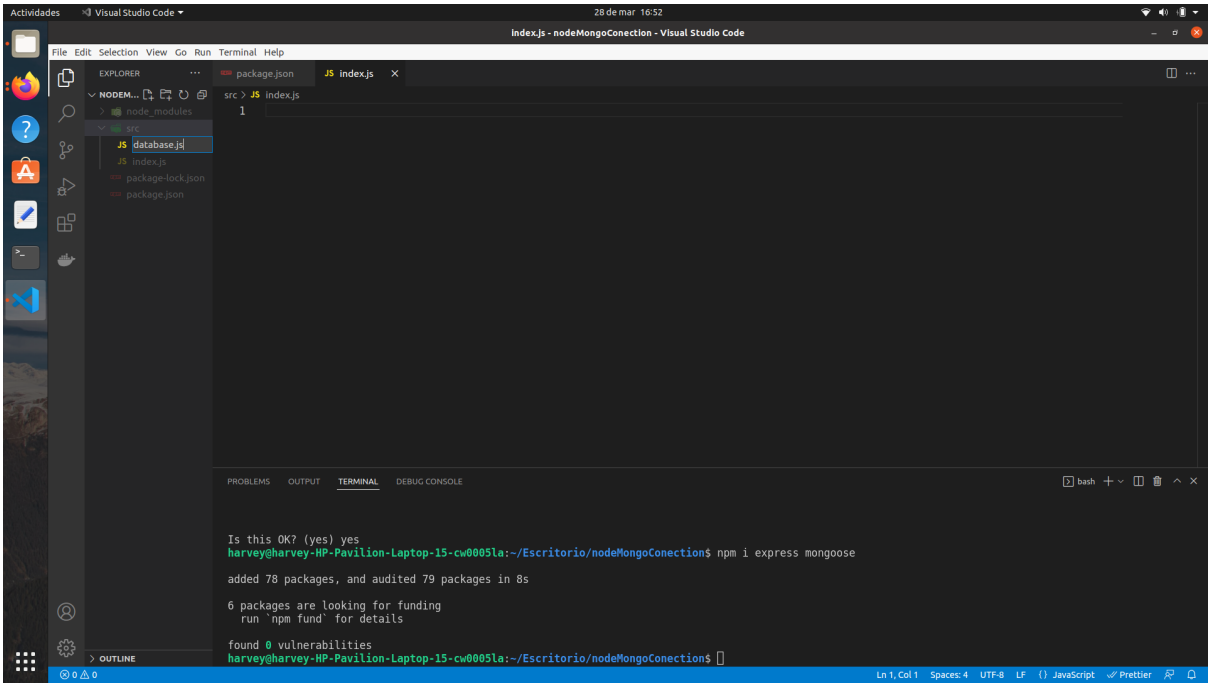


8. Dentro de la carpeta src crear un archivo index.js para crear el servidor y database.js para conectarnos a la base de datos.

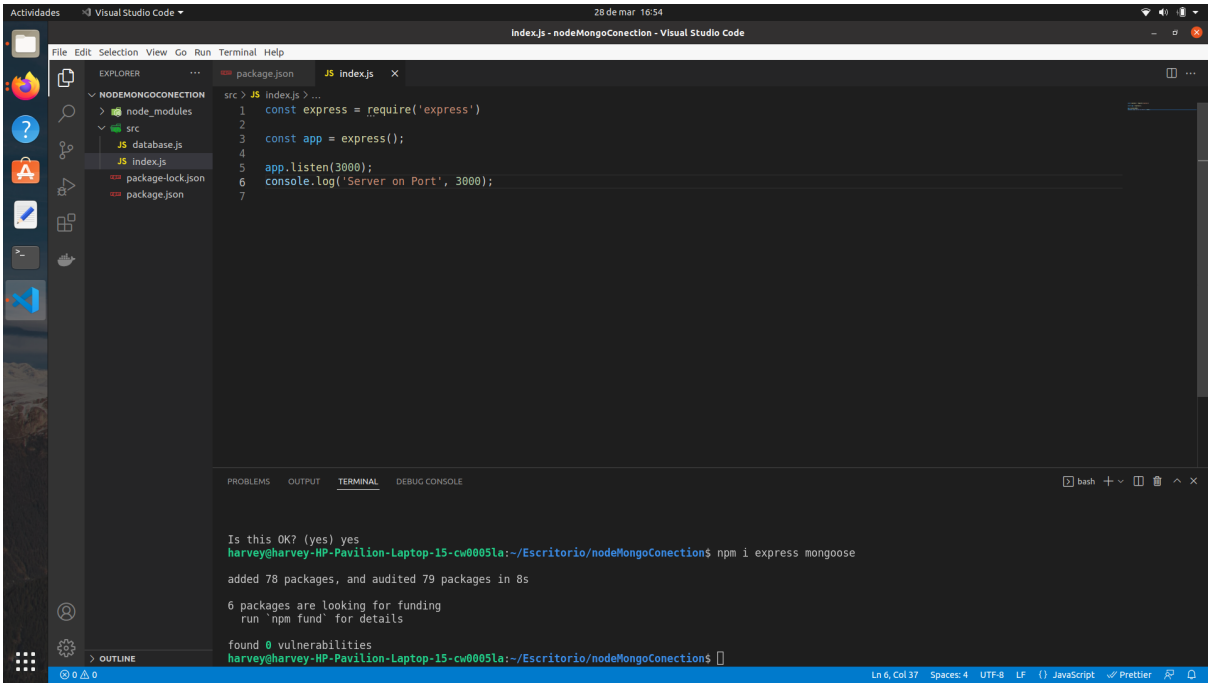



Actividades
Visual Studio Code
28 de mar 16:50
package.json - nodeMongoConection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NODEM...
node_modules
src
index.js
package-lock.json
package.json
package.json
{
"name": "nodemongoconection",
"version": "0.0.1",
"description": "proyecto con node y mongo db",
"main": "index.js",
Debug
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [
"mongo",
"node",
"docker"
],
"author": "Nicolas Echavarria",
"license": "ISC",
"dependencies": {
"express": "^4.17.3",
"mongoose": "^6.2.9"
}
}
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm i express mongoose
added 78 packages, and audited 79 packages in 8s
6 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$
Mostrar aplicaciones
Ln 1, Col 1
Spaces: 2
UTF-8
LF
{} JSON
Prettier

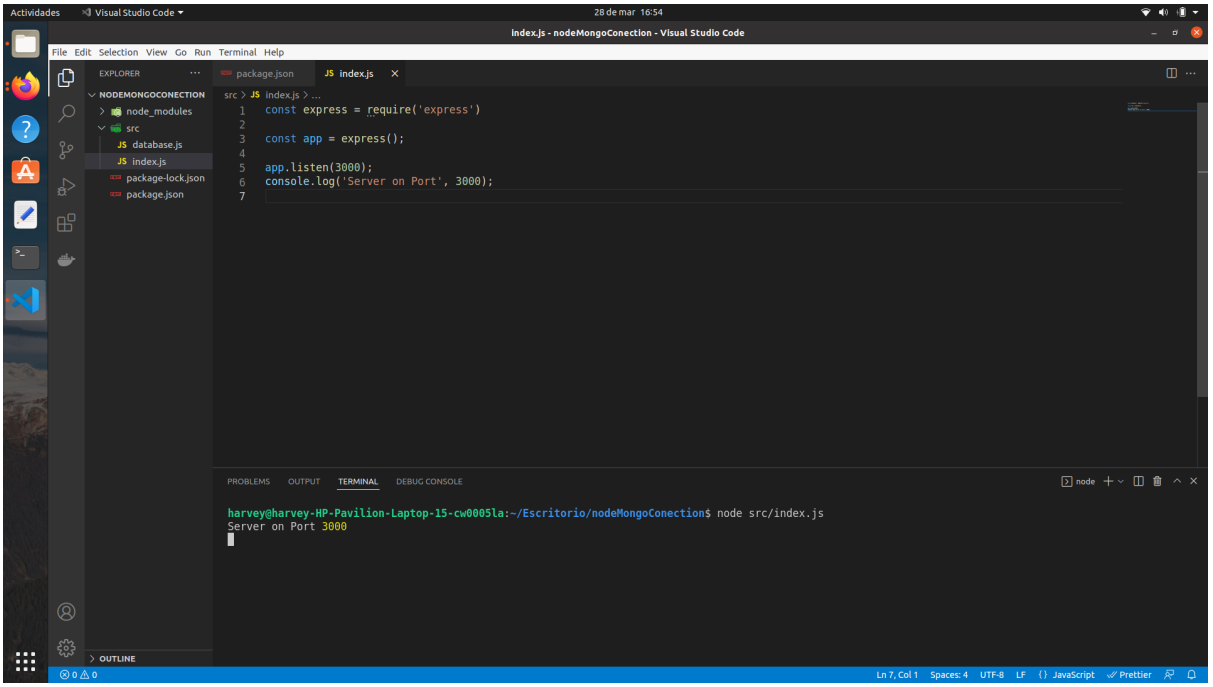

Actividades
Visual Studio Code
28 de mar 16:51
index.js - nodeMongoConection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
package.json
index.js
src > JS index.js
NODEM...
node_modules
src
New File
New Folder
Open Containing Folder Alt+Ctrl+R
Open in Integrated Terminal
Find in Folder... Alt+Shift+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Alt+Ctrl+C
Copy Relative Path Alt+Ctrl+Shift+C
Rename F2
Delete Supr
OUTLINE
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm i express mongoose
added 78 packages, and audited 79 packages in 8s
6 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$
Ln 1, Col 1
Spaces: 4
UTF-8
LF
{} JavaScript
Prettier



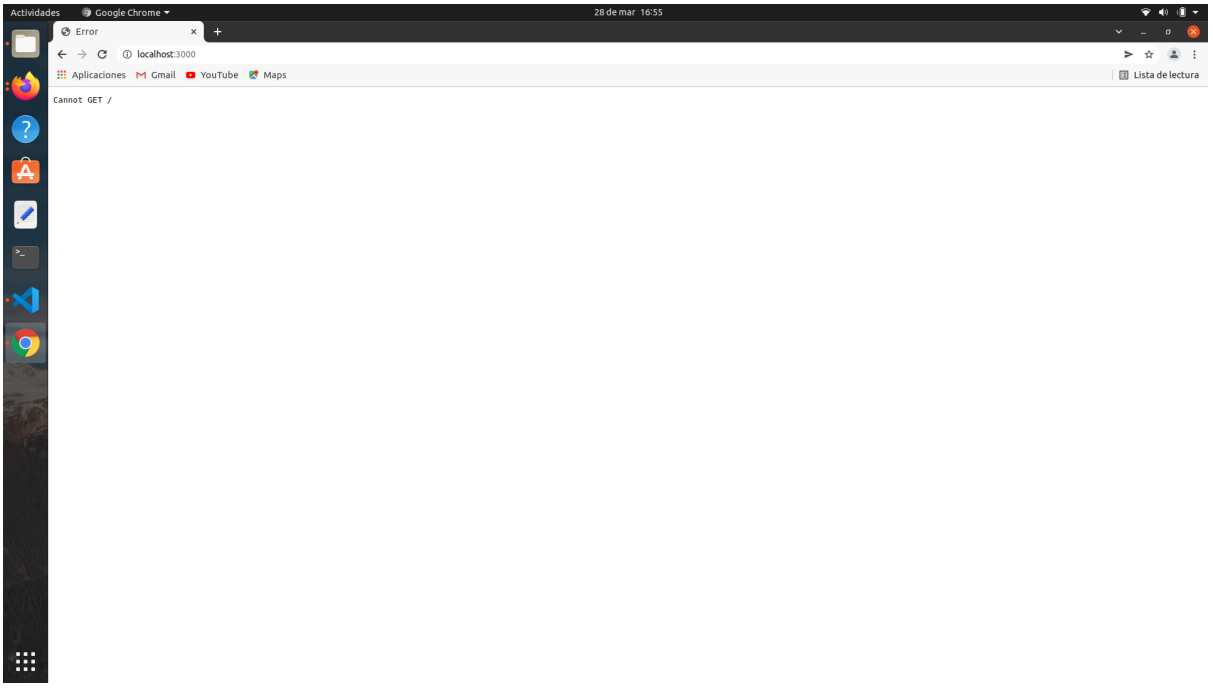
9. Configurar el archivo index.js de la siguiente manera:



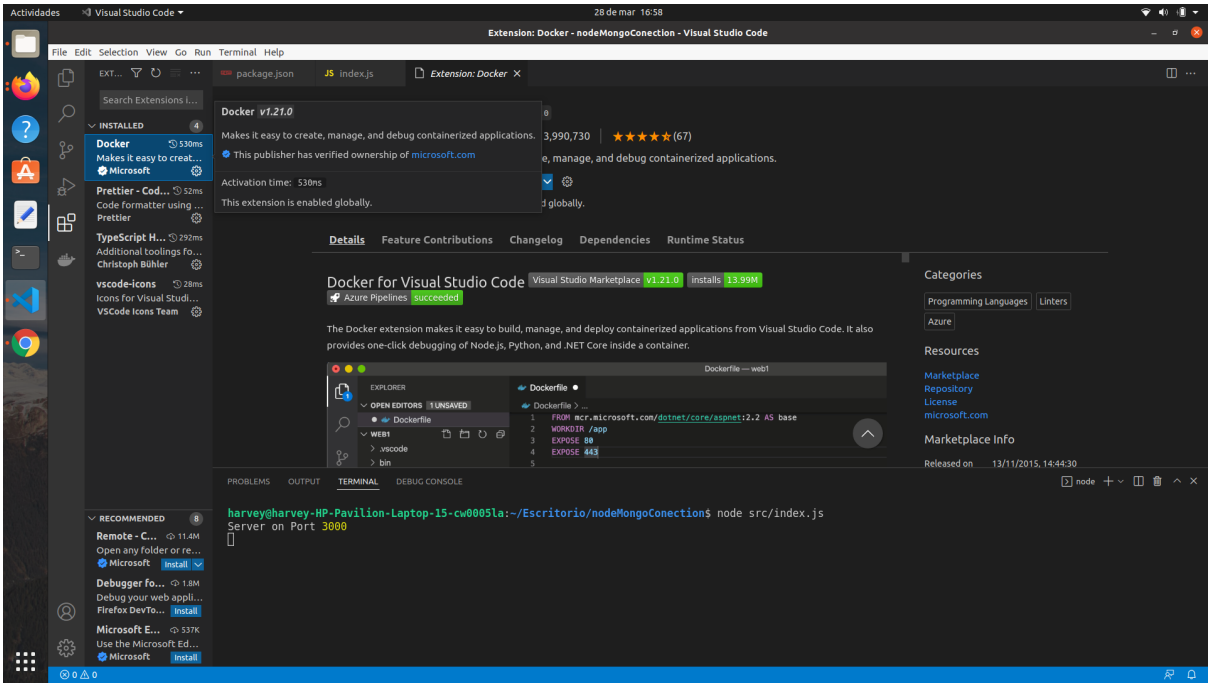
10. Ejecutamos el archivo index.js mediante el comando: node src/index.js



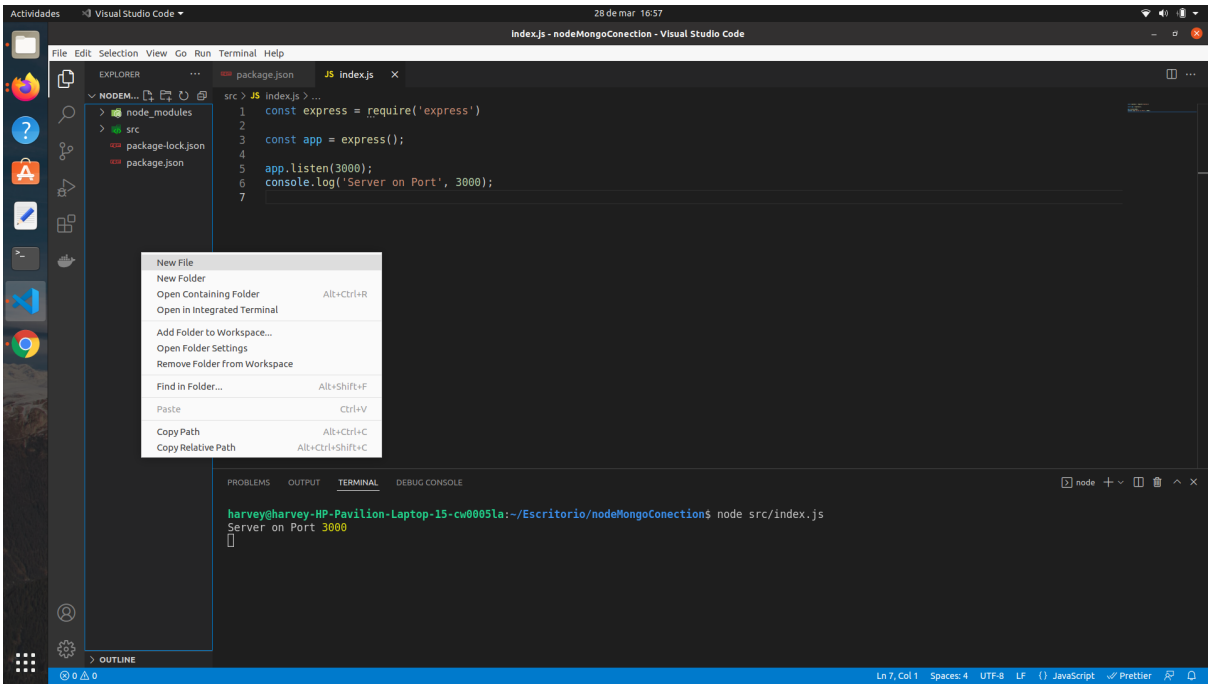
11. Verificamos en el navegador: (El servidor se encuentra funcionando pero no hemos respondido nada)

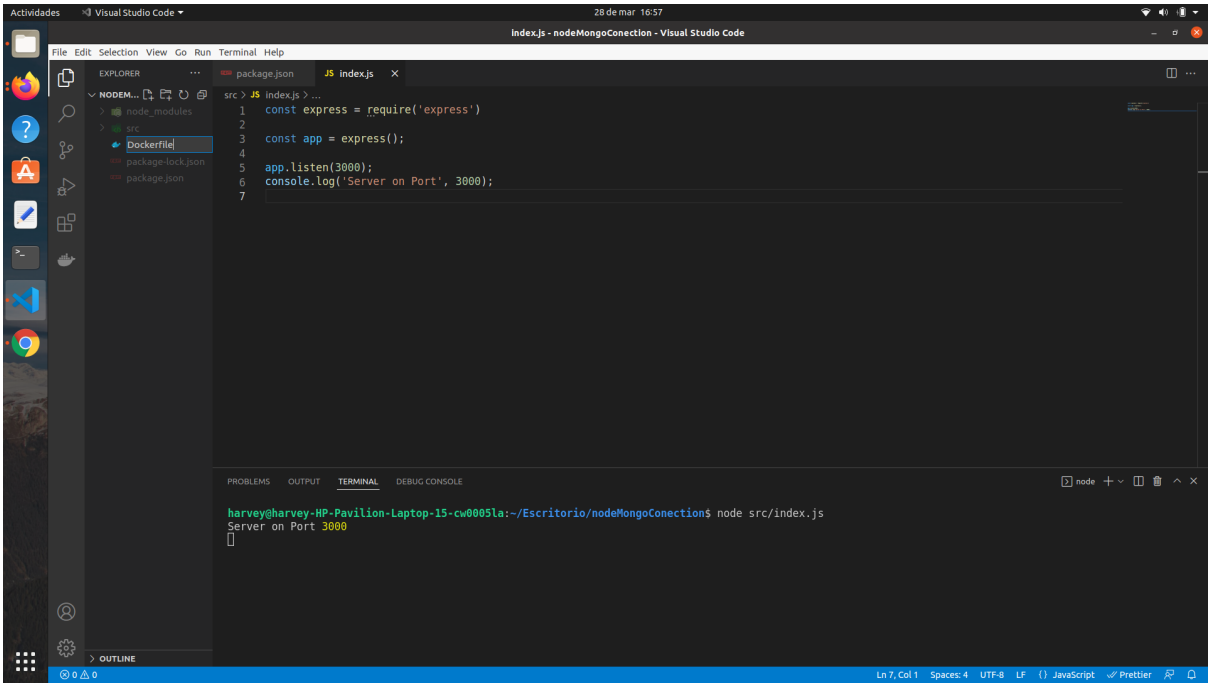


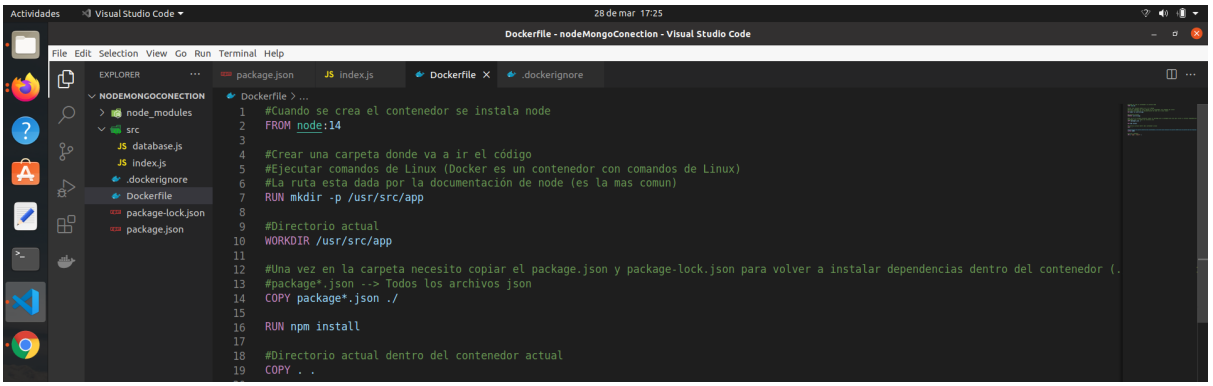
12. Podemos agregar la extensión de visual studio code de docker para tener ayudas en la sintaxis:



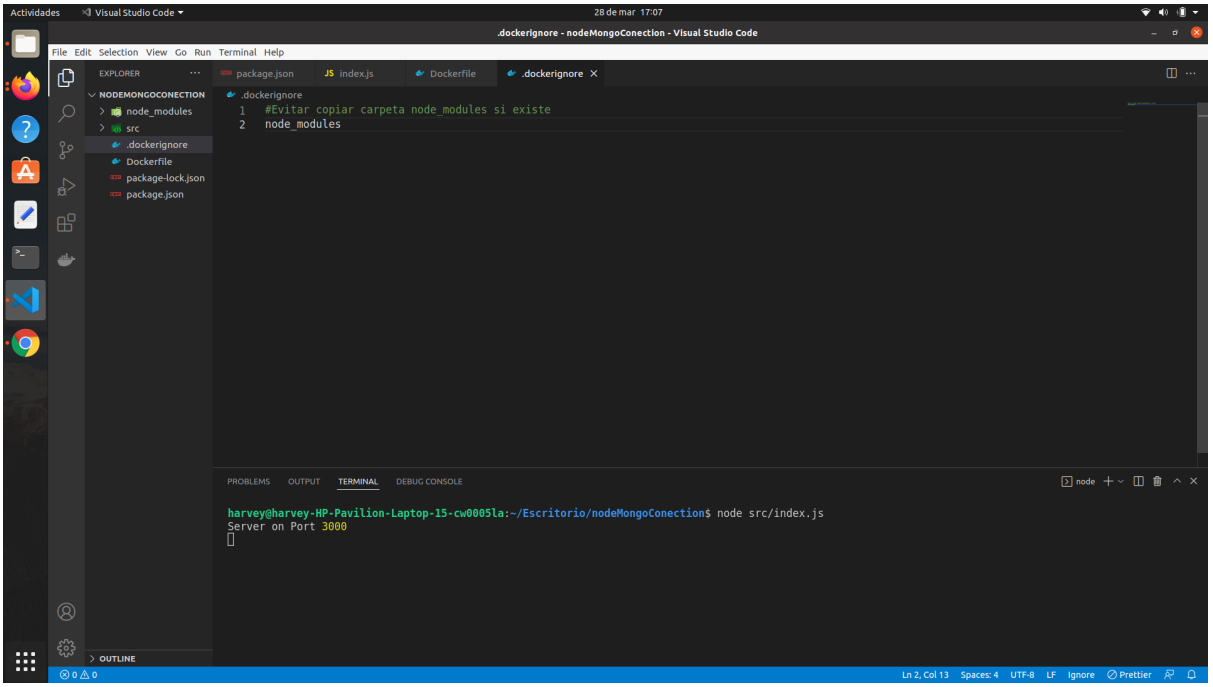
13. Creamos el archivo Dockerfile dentro de la carpeta raíz del proyecto para especificar qué necesita el proyecto para funcionar y lo configuramos de la siguiente manera: (recuerda que las versiones de la imagen la podemos encontrar en la sección de documentación de la imagen en docker hub)



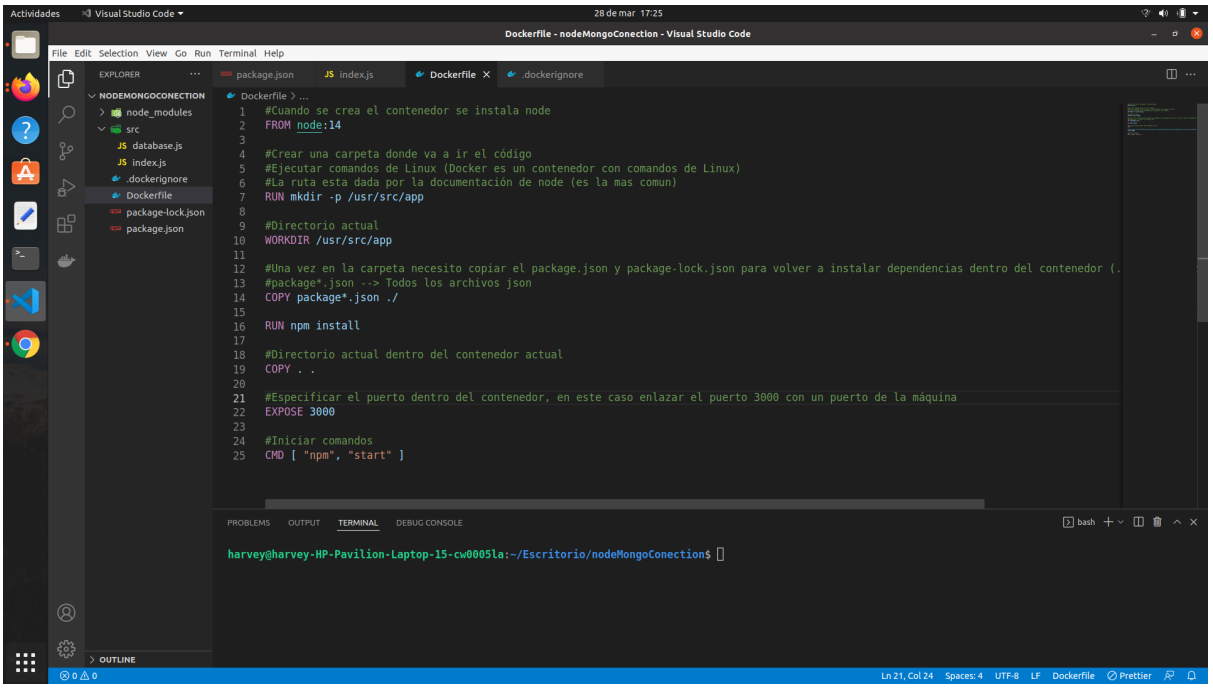




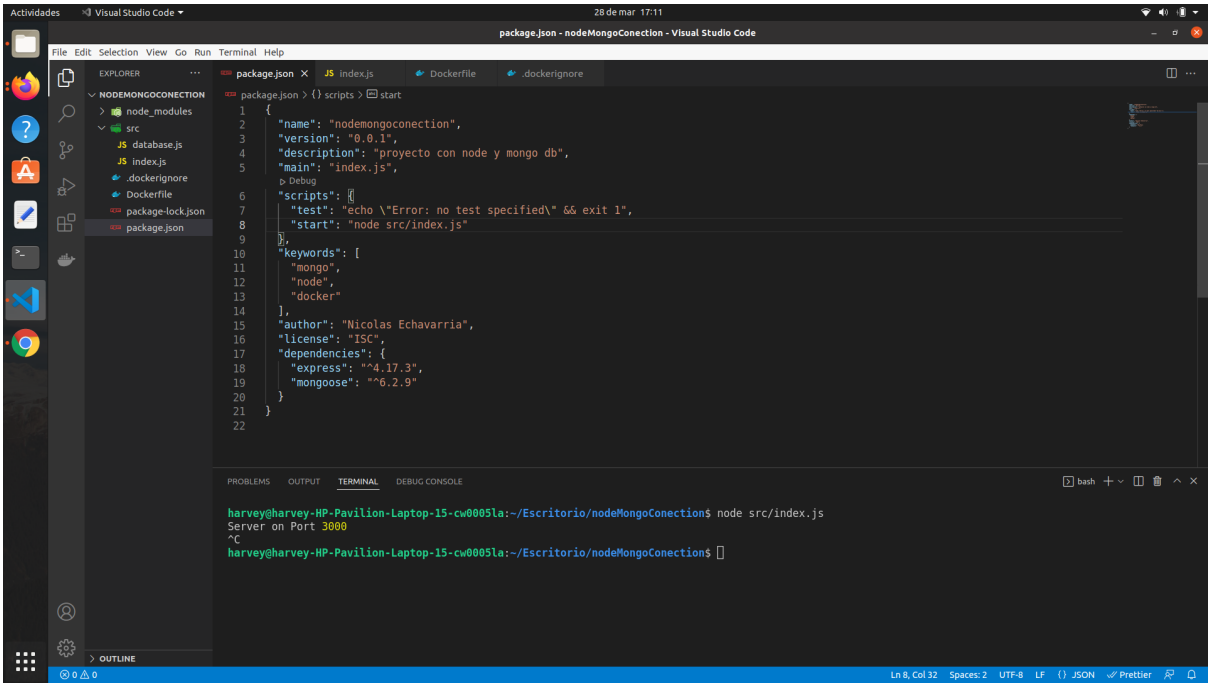
14. Para evitar que se copie la carpeta node_modules o cualquier otro archivo que no queramos creamos un archivo llamado .dockerignore dentro de la carpeta raíz del proyecto y lo configuramos de la siguiente manera: (este archivo tiene un funcionamiento similar al de gitignore)



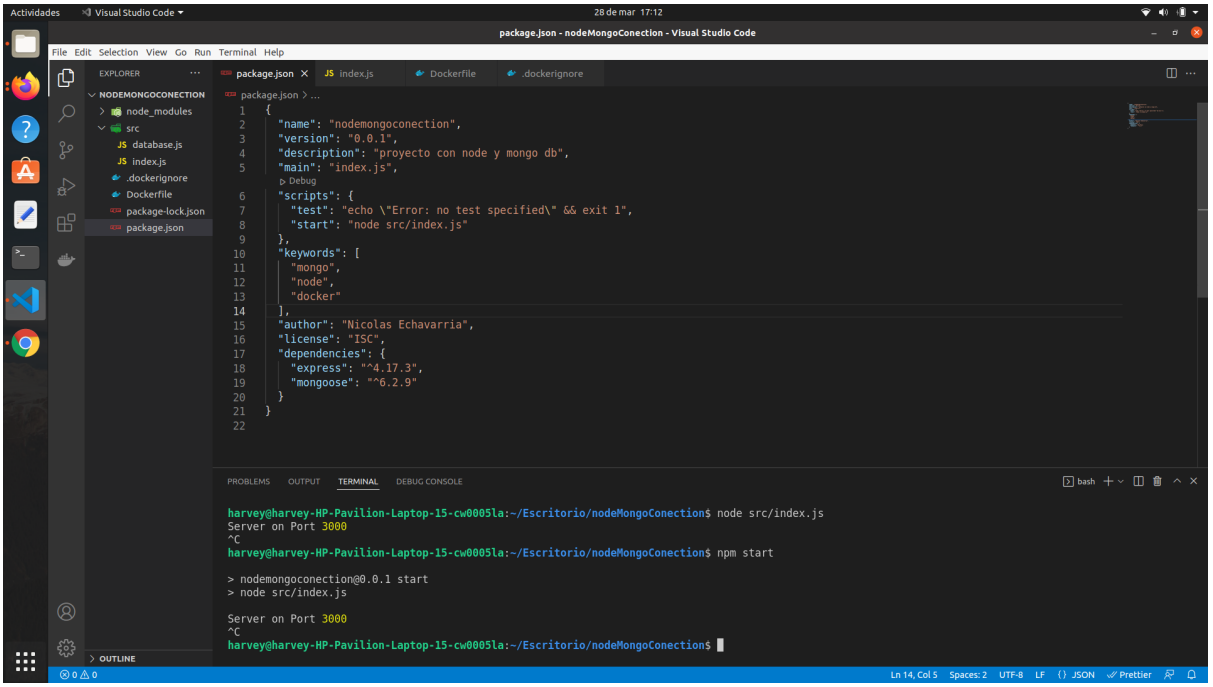
15. Realizamos la siguiente modificación en el archivo Dockerfile:



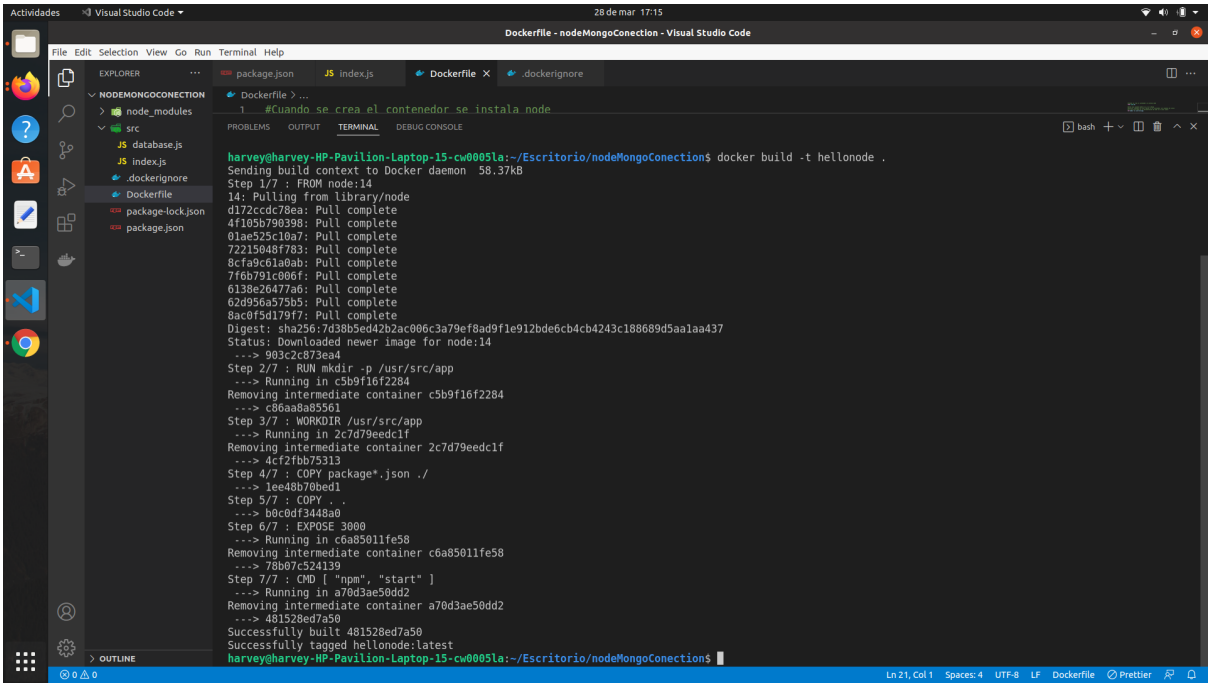
16. Modificamos el archivo package.json de manera tal que creemos una nueva línea que permita generar un script que ejecute el proyecto (línea 8):

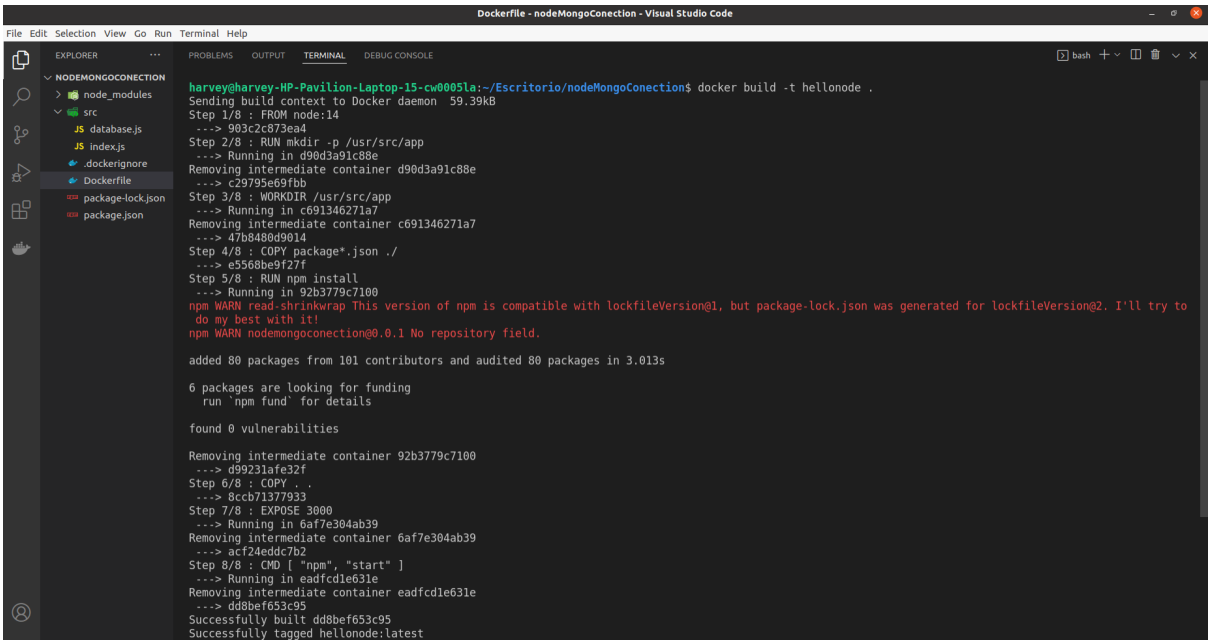


17. Vamos a la terminal y cancelamos el proceso con control + c, luego ejecutamos el comando npm start para iniciar nuevamente la ejecución del proyecto:

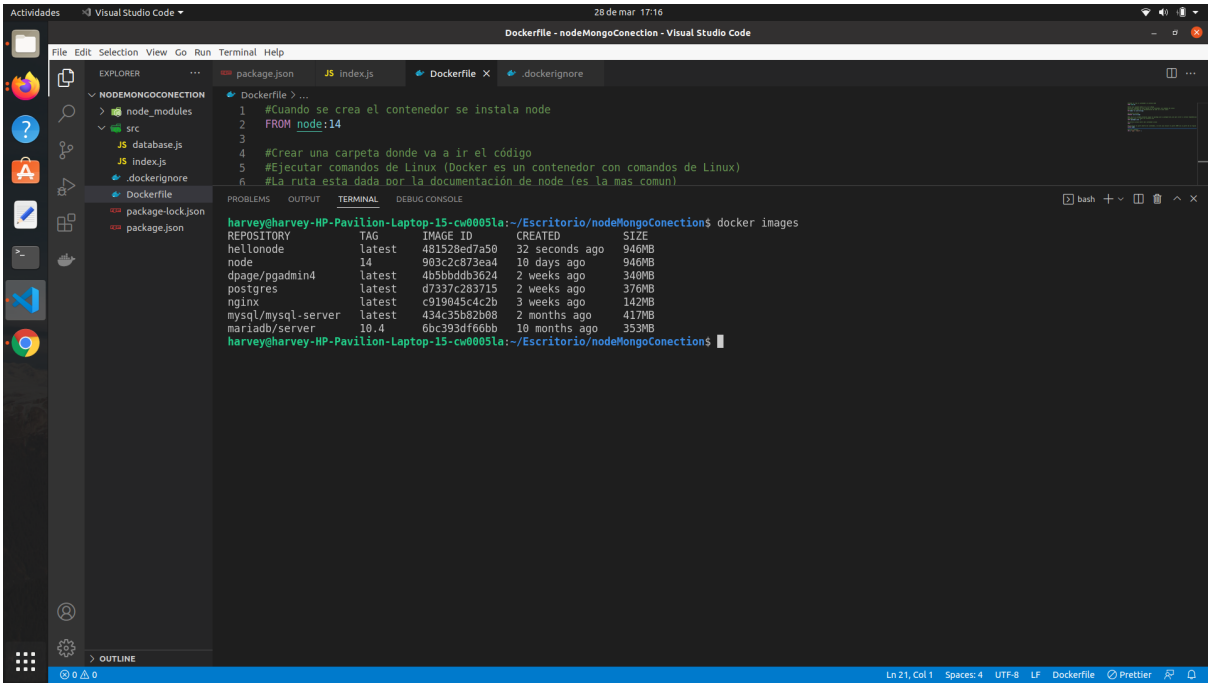


18. Volvemos a parar el proceso con control + c y ejecutamos el comando: docker build -t hellonode . (Descargar imagen de node y empieza a ejecutar los comandos uno a uno)

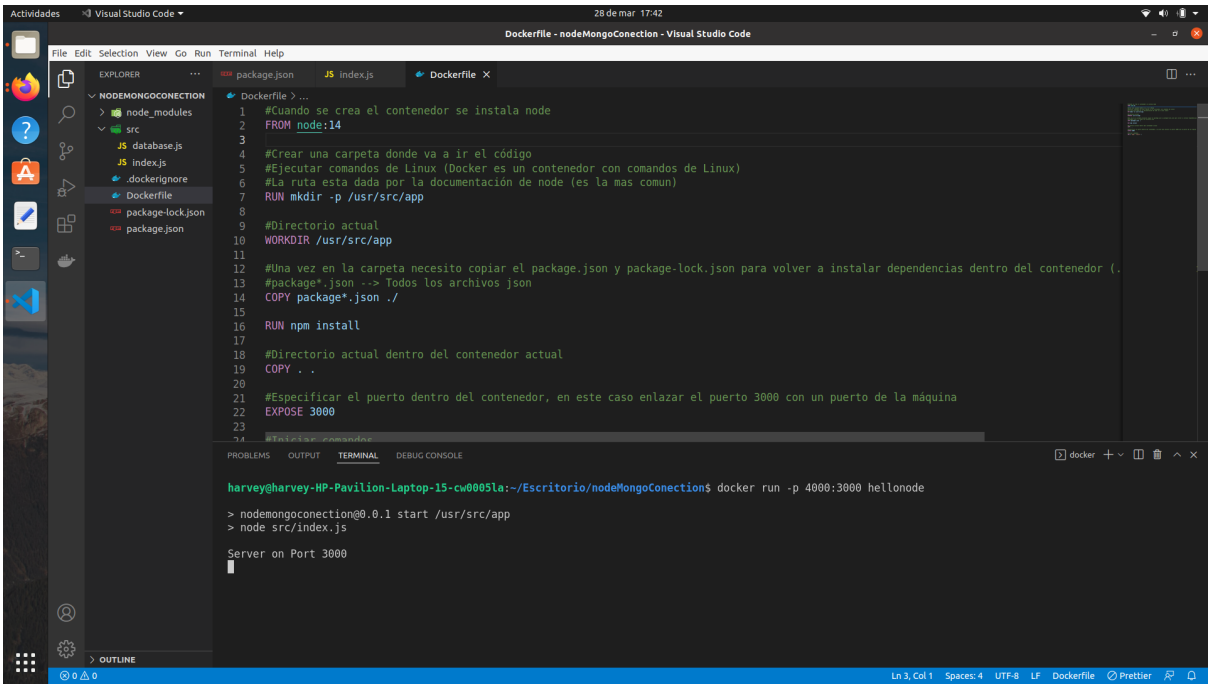




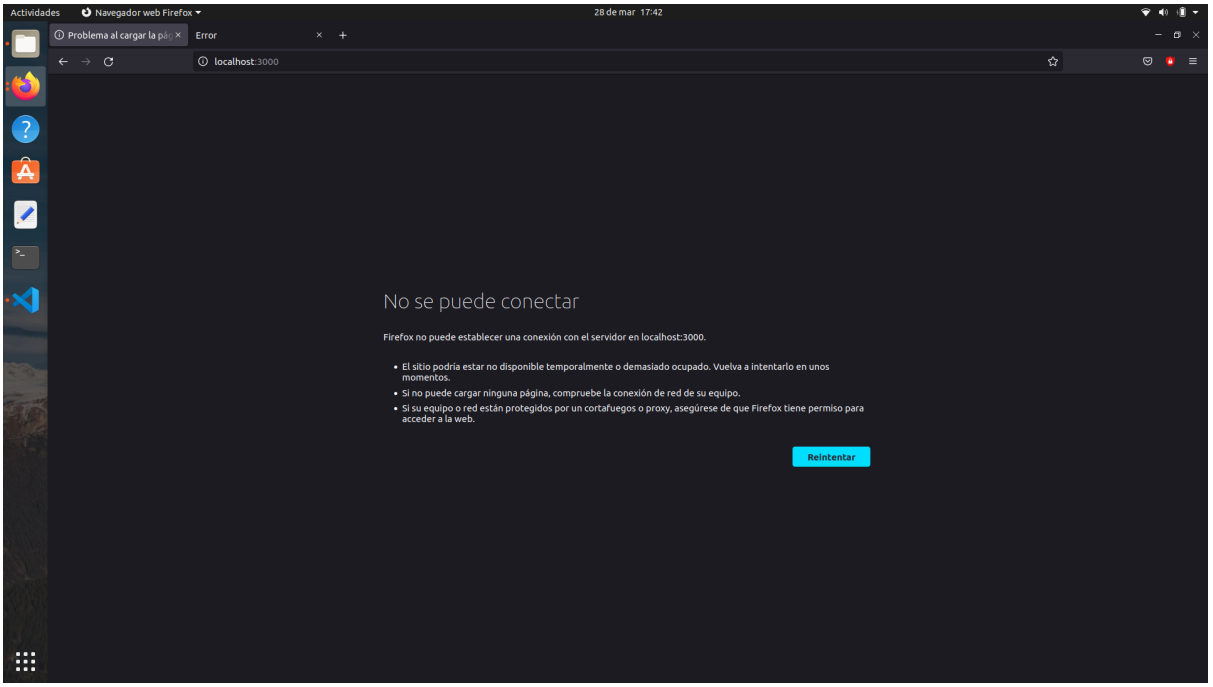
19. Ejecutamos el comando docker images y vemos que tenemos una imagen llamada node y otra llamada hellonode, la primera hace referencia a la imagen que se descargo y la segunda hace referencia a la imagen que creamos.

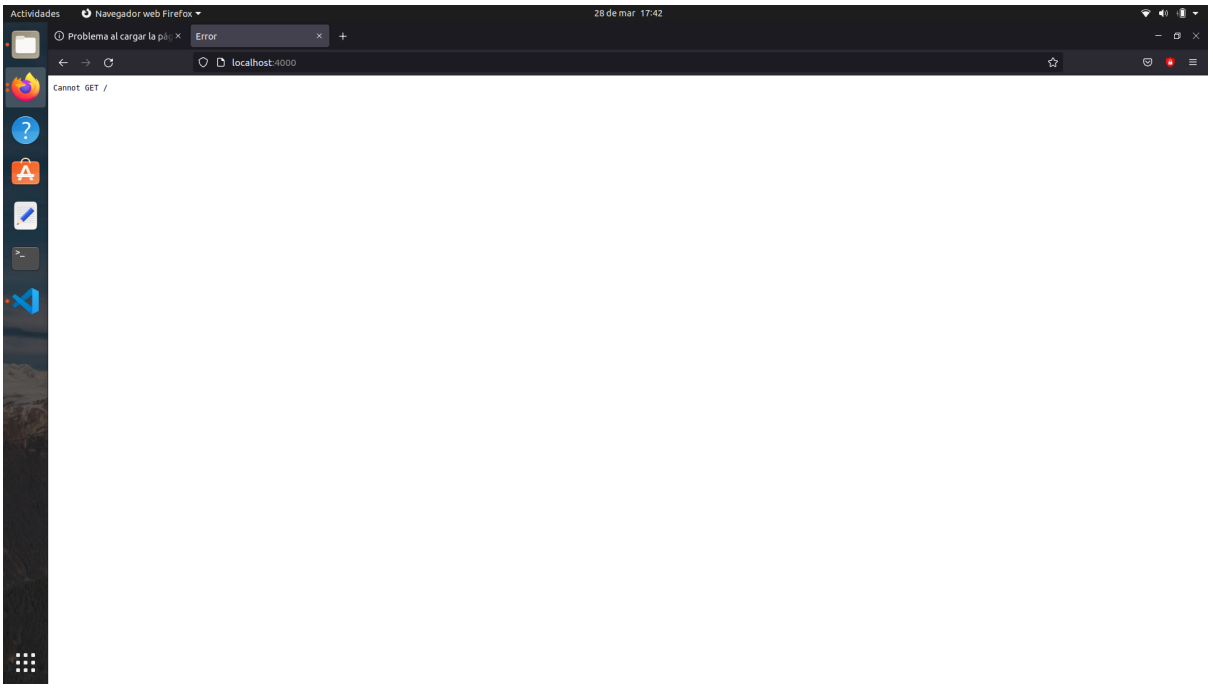


20. Ejecutamos el siguiente comando para correr la imagen que acabamos de crear: docker run -p 4000:3000 hellonode (el puerto es 3000 por que es el que hemos configurado para nuestro servidor y el puerto 4000 es por el que escucha nuestro computador):

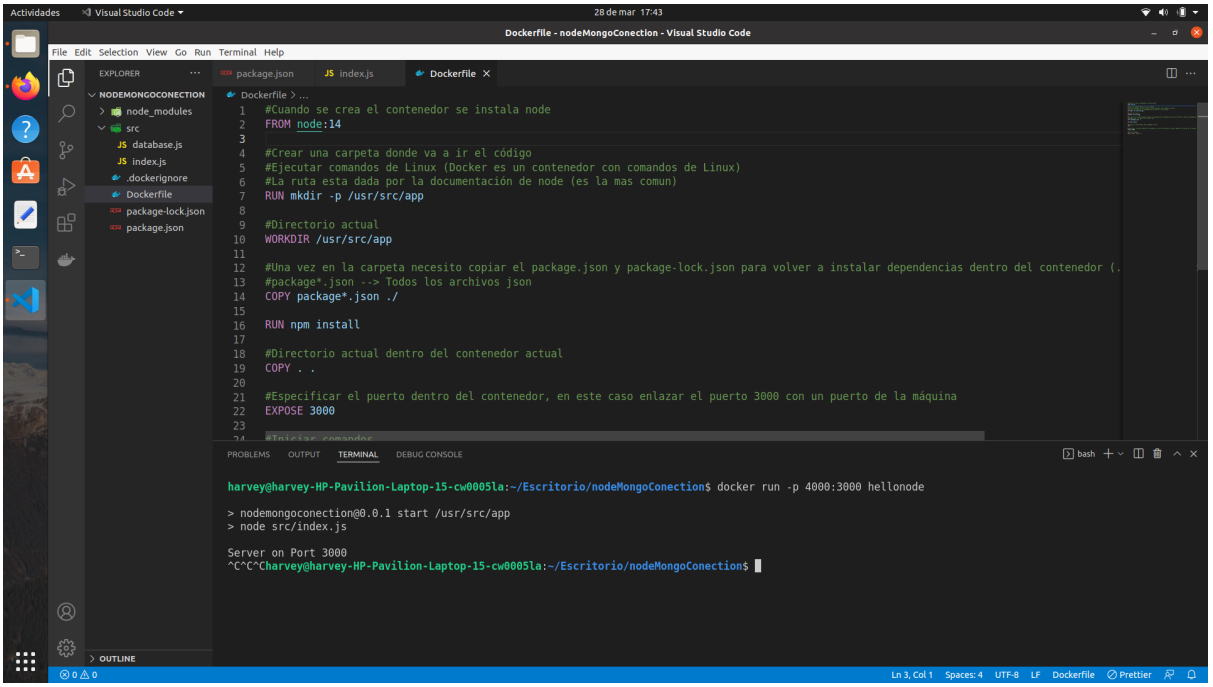


21. Validamos en el navegador:

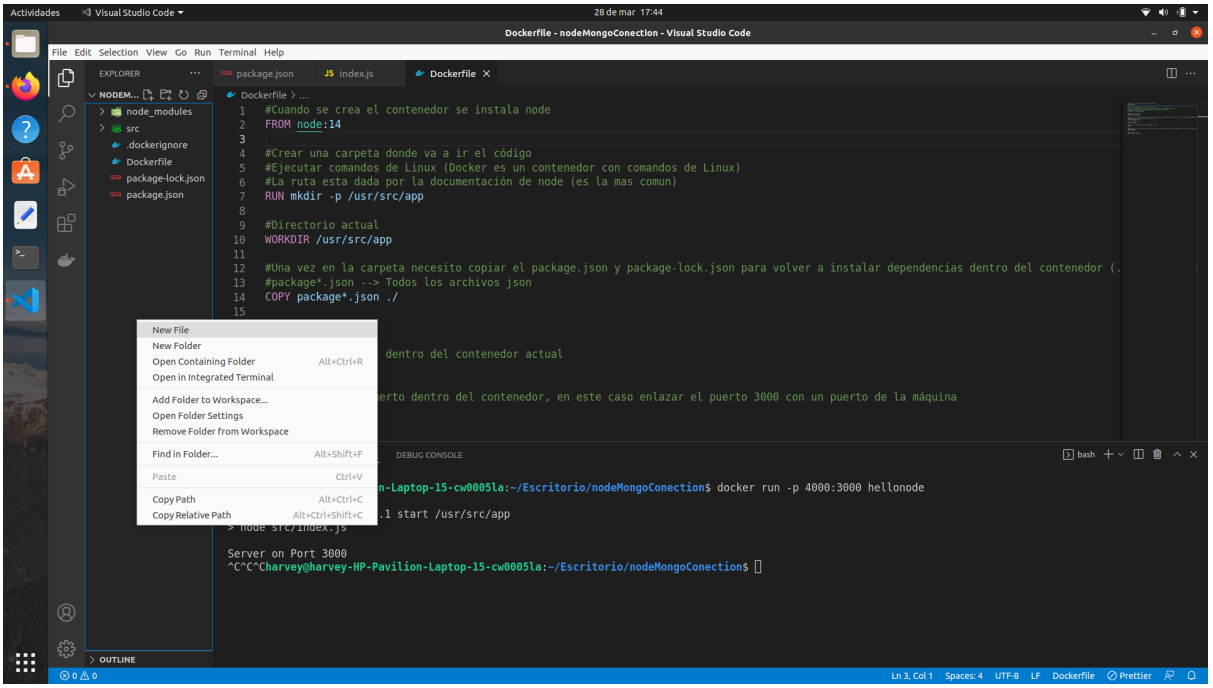


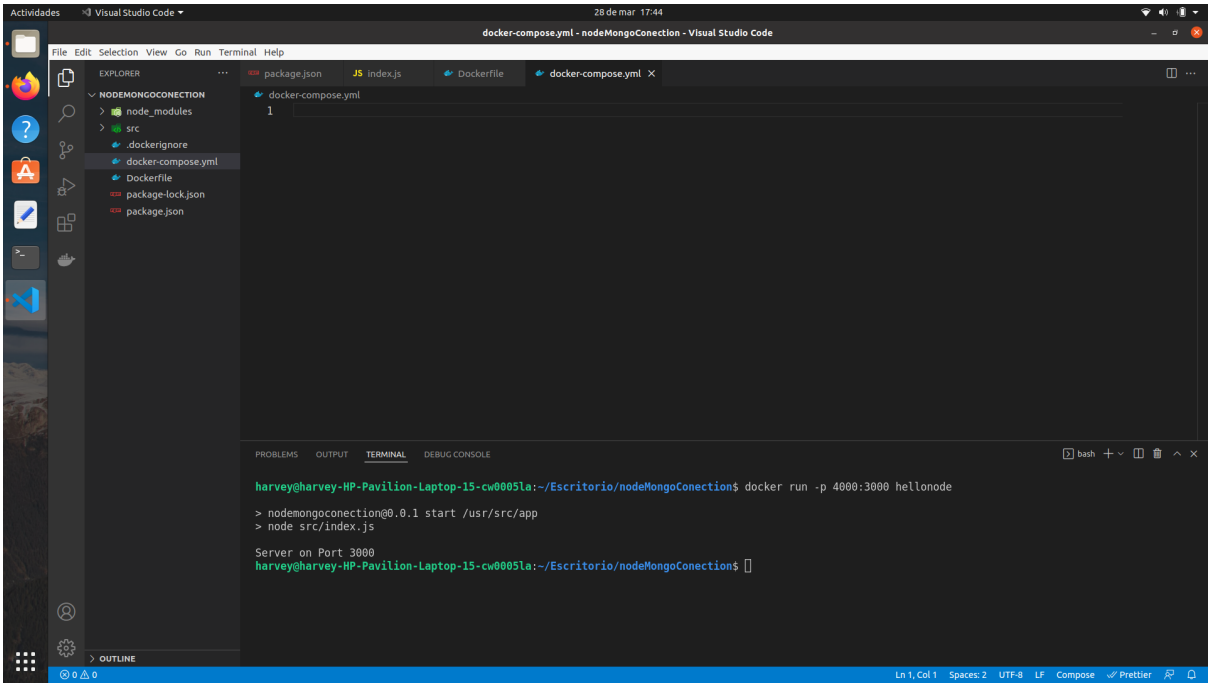


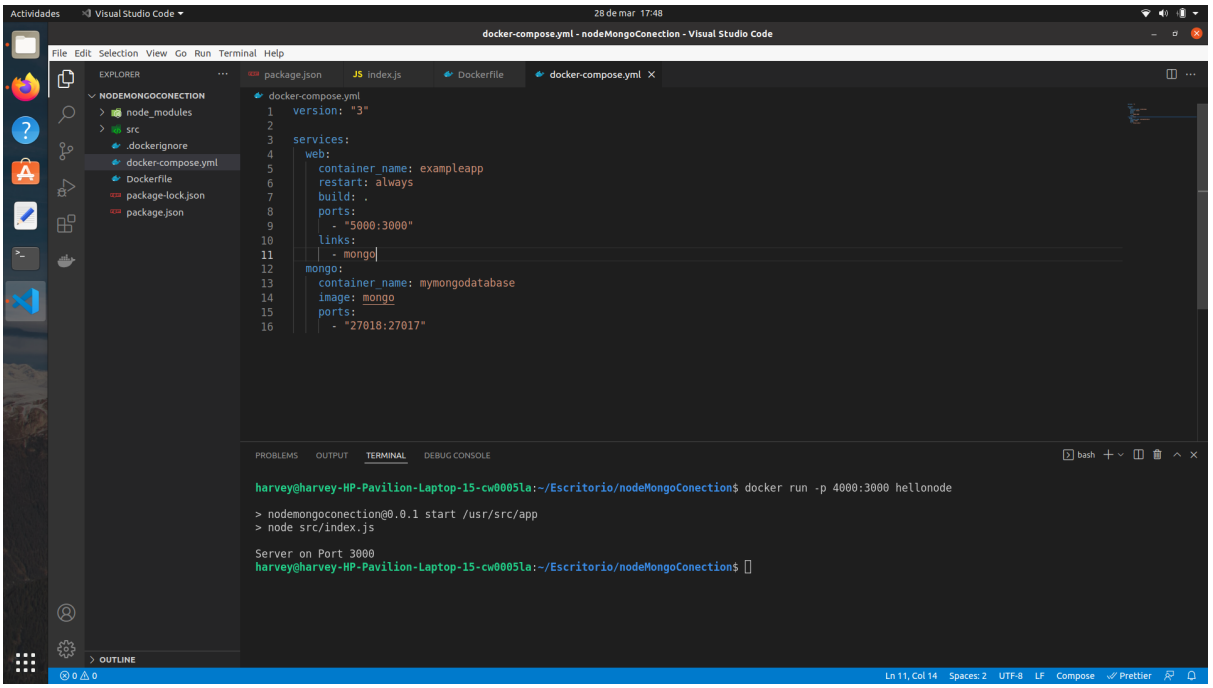
22. Cancelamos el proceso con control + c



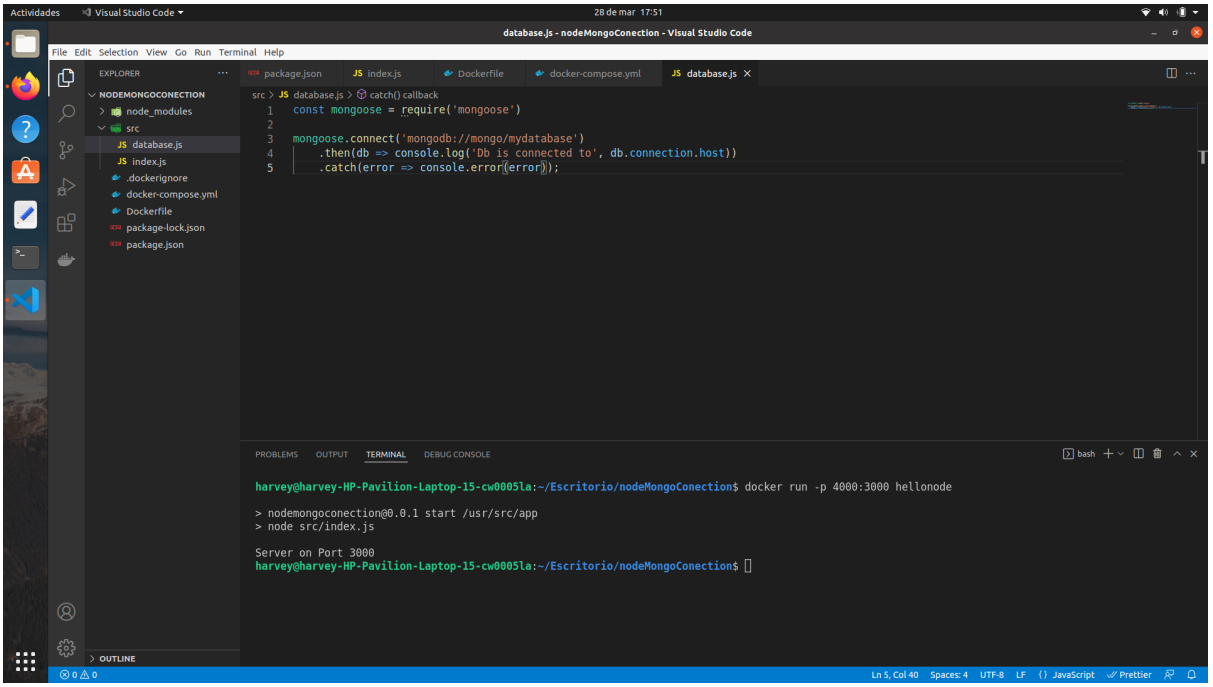
23. Ahora necesitamos un archivo que nos permita componer distintas imágenes, para ello creamos el archivo docker-compose.yml dentro de la carpeta raíz del proyecto, y lo configuramos de la siguiente manera:



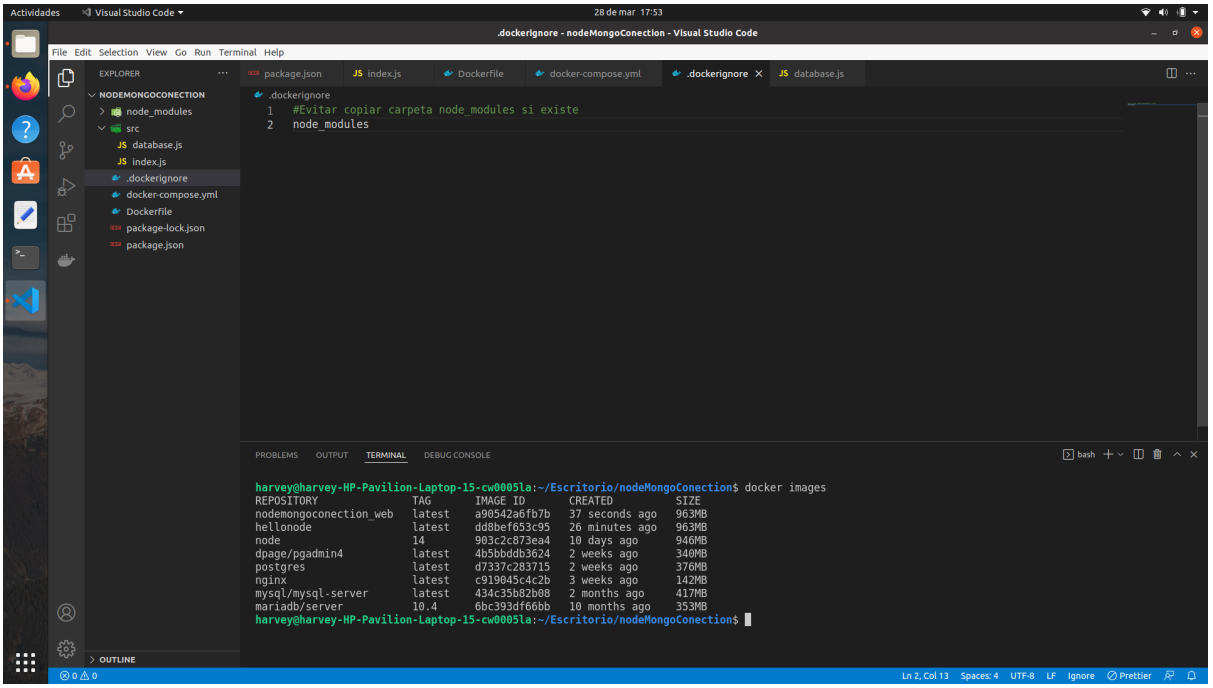




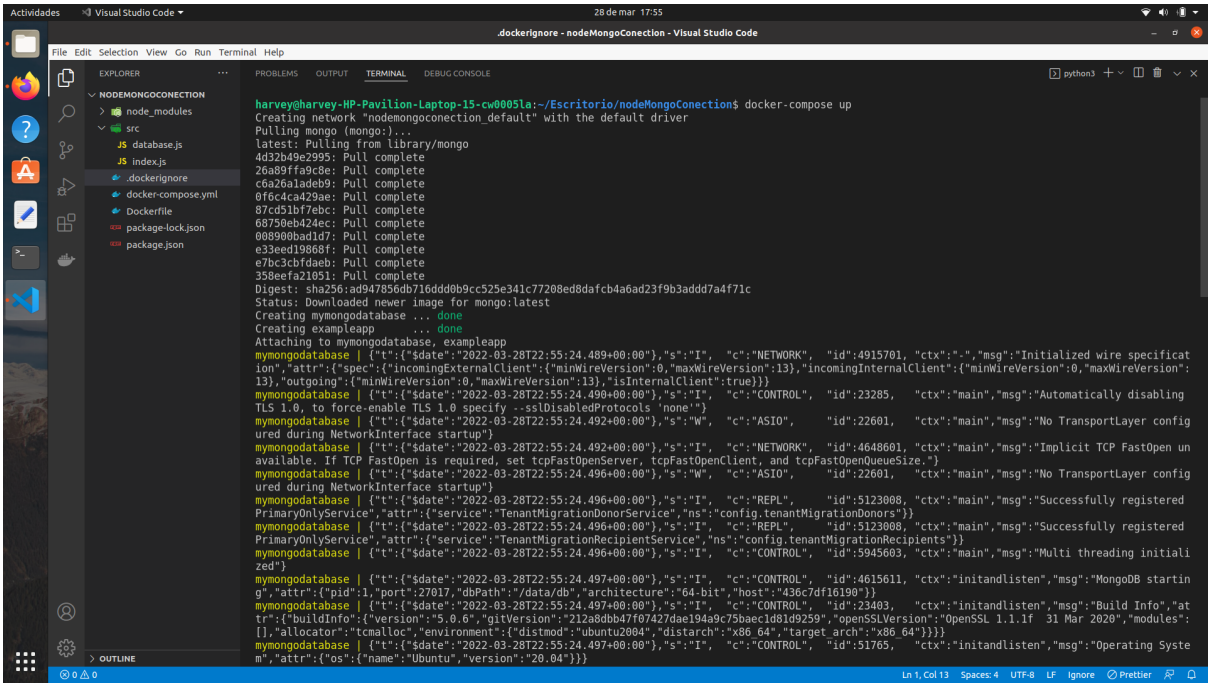
24. Configurar el archivo database.js para crear la conexión:



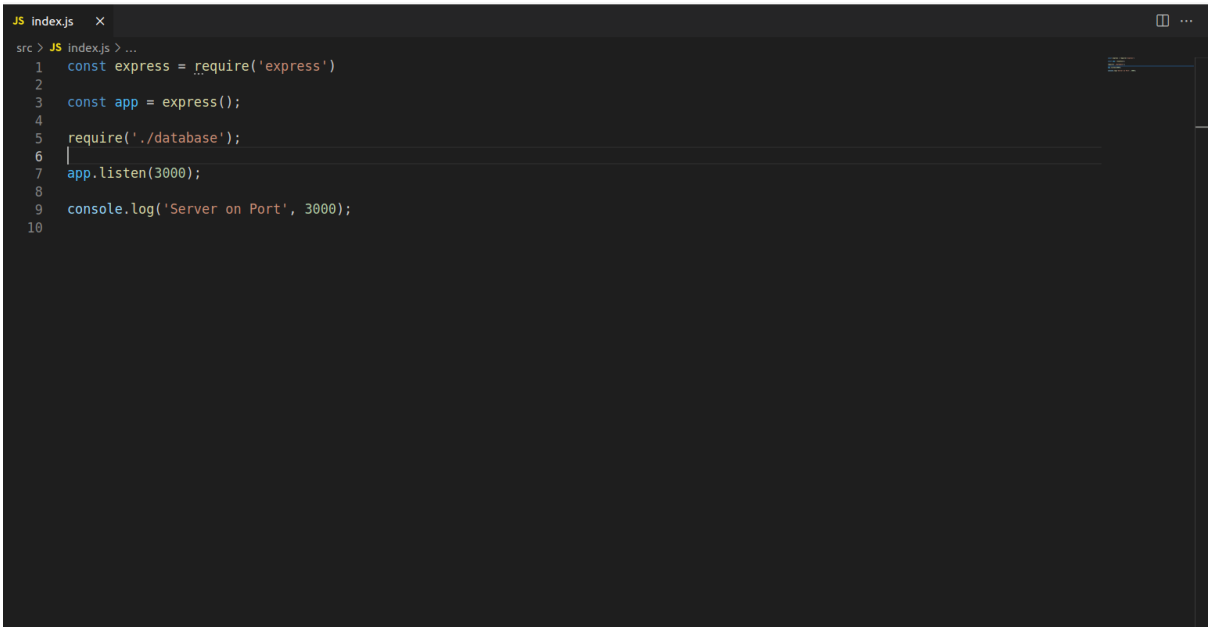
25. Ejecutamos el comando docker-compose build
26. Ejecutamos docker images y vemos que se ha creado una imagen node-docker-mongo_web:



27. Para probarlo ejecutamos el comando: docker-compose up para iniciar los servicios



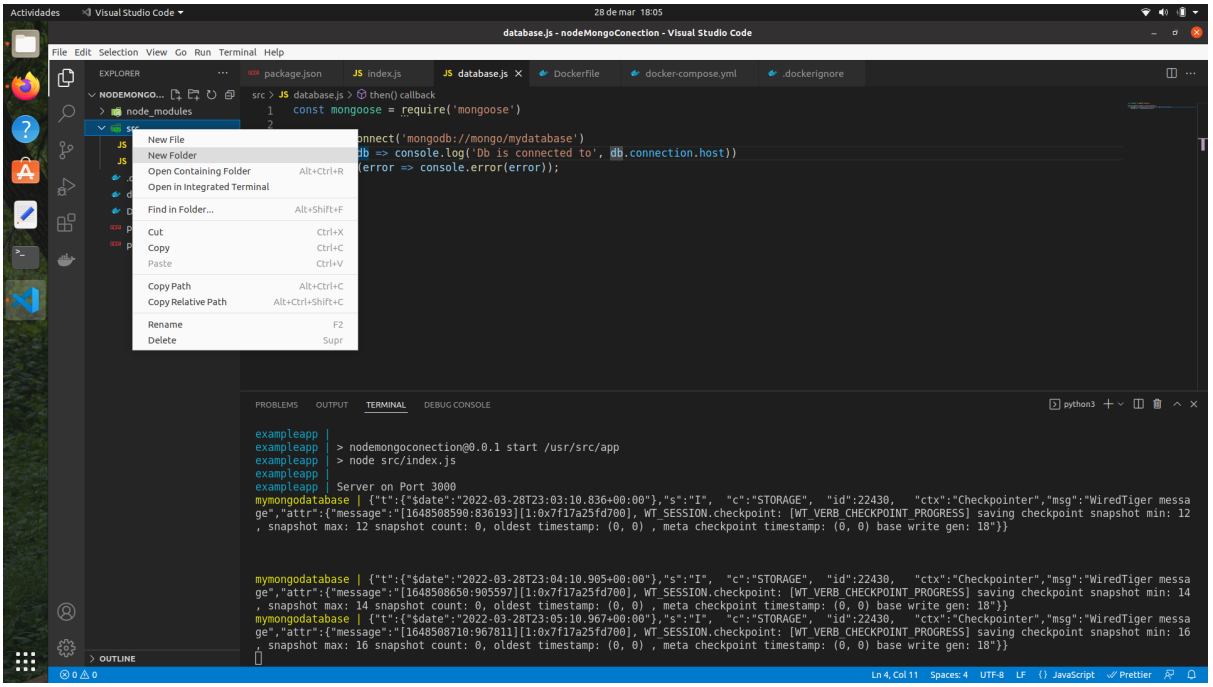
28. Para ver la cadena de conexión modificamos de la siguiente manera el archivo index.js:

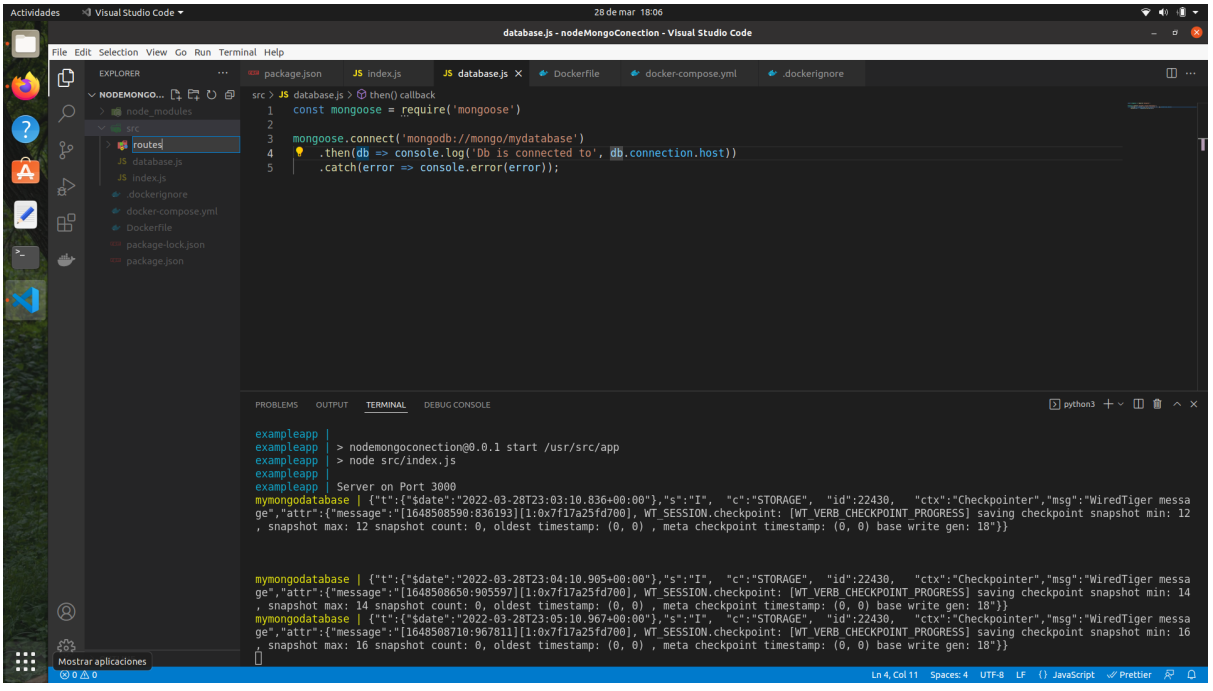


29. Escribimos control + c para detener el servicio y volvemos a ejecutar docker build y docker-compose up para volver a iniciar el servicio observando que ya tenemos el mensaje de que estamos conectados a la base de datos de mongo.

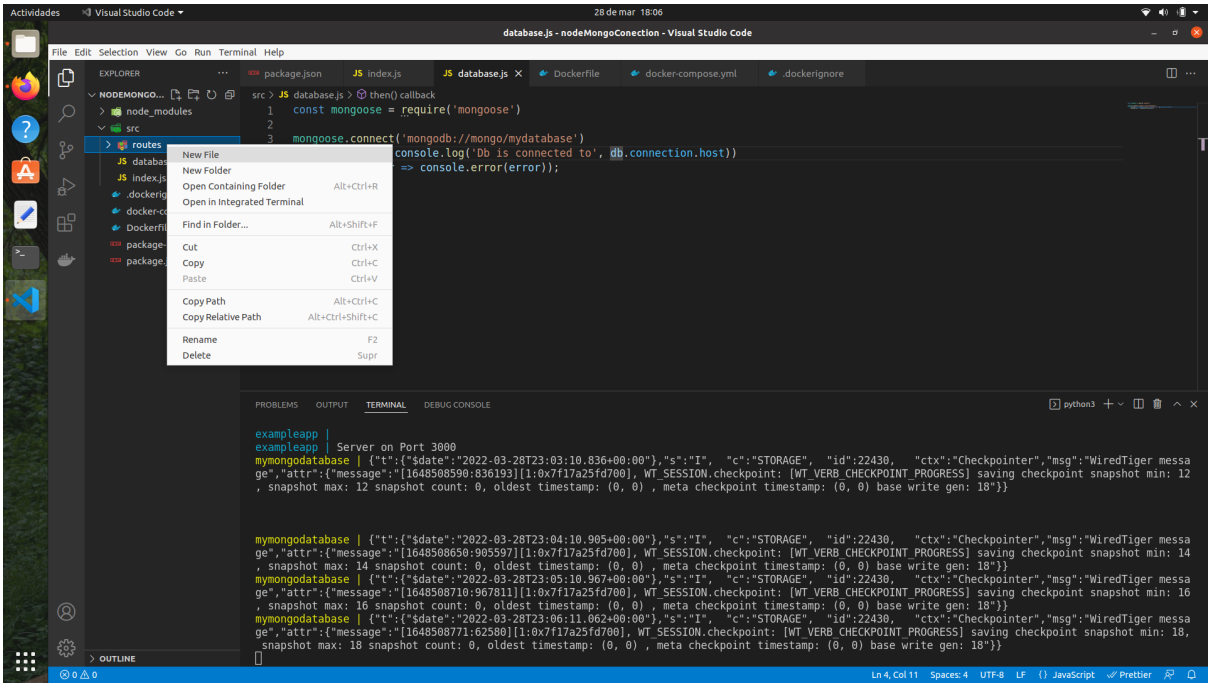
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

g for connections","attr":{"port":27017,"ssl":"off"}}
exampleapp |
exampleapp | > nodemongoconection@0.0.1 start /usr/src/app
exampleapp | > node src/index.js "dev"
exampleapp |
exampleapp | Server on Port 3000
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:35.376+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg":"Connec
tion accepted","attr":{"remote":"172.19.0.3:60714","uuid":"869a1343-be6b-4200-bda4-1c56a7844c4e","connectionId":1,"connectionCount":1}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:35.387+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn1","msg":"client me
tadata","attr":{"remote":"172.19.0.3:60714","client":"conn1","doc":{"driver":{"name":"nodejs|Mongoose","version":"4.3.1"},"os":{"type":
"Linux","name":"linux","architecture":"x64","version":"5.4.0-105-generic"},"platform":"Node.js v14.19.1, LE (unified)","version":"4.3.1
|6.2.9"}}}
exampleapp | Db is connected to mongo
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:45.907+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg":"Connec
tion accepted","attr":{"remote":"172.19.0.3:60716","uuid":"66b81404-8806-4254-b592-bc64e932c086","connectionId":2,"connectionCount":2}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:45.909+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn2","msg":"client me
```

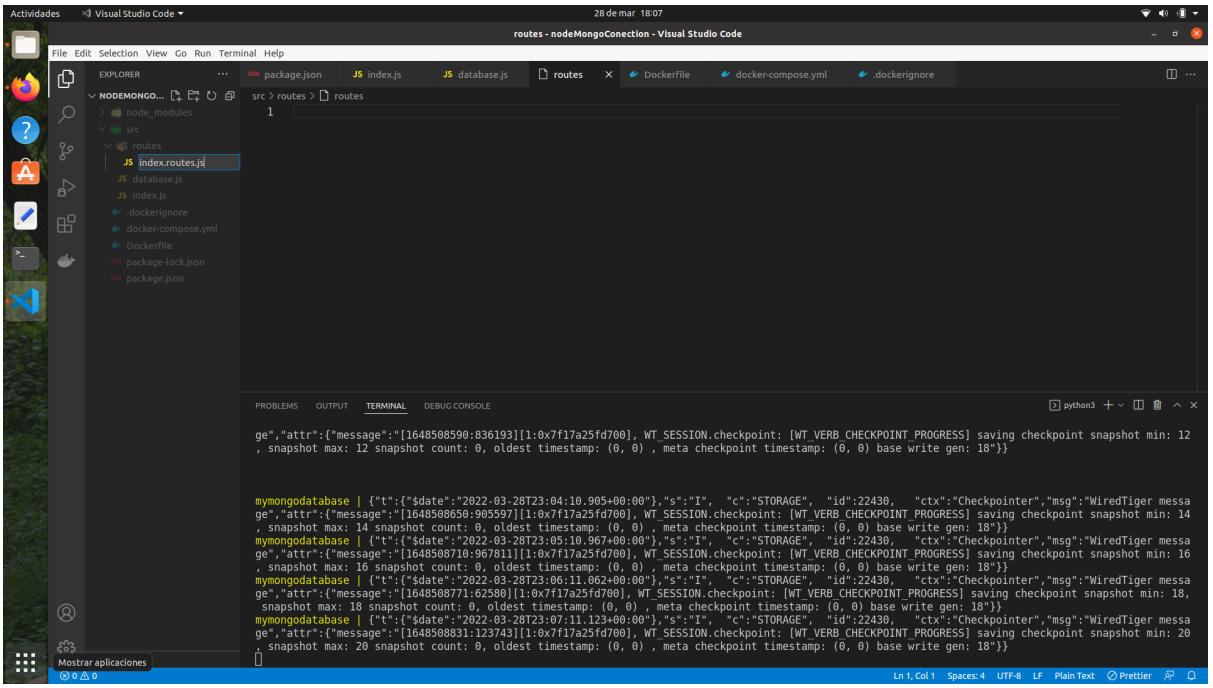
30. Para agregar las rutas dentro de la carpeta src creamos una nueva carpeta llamada routes:



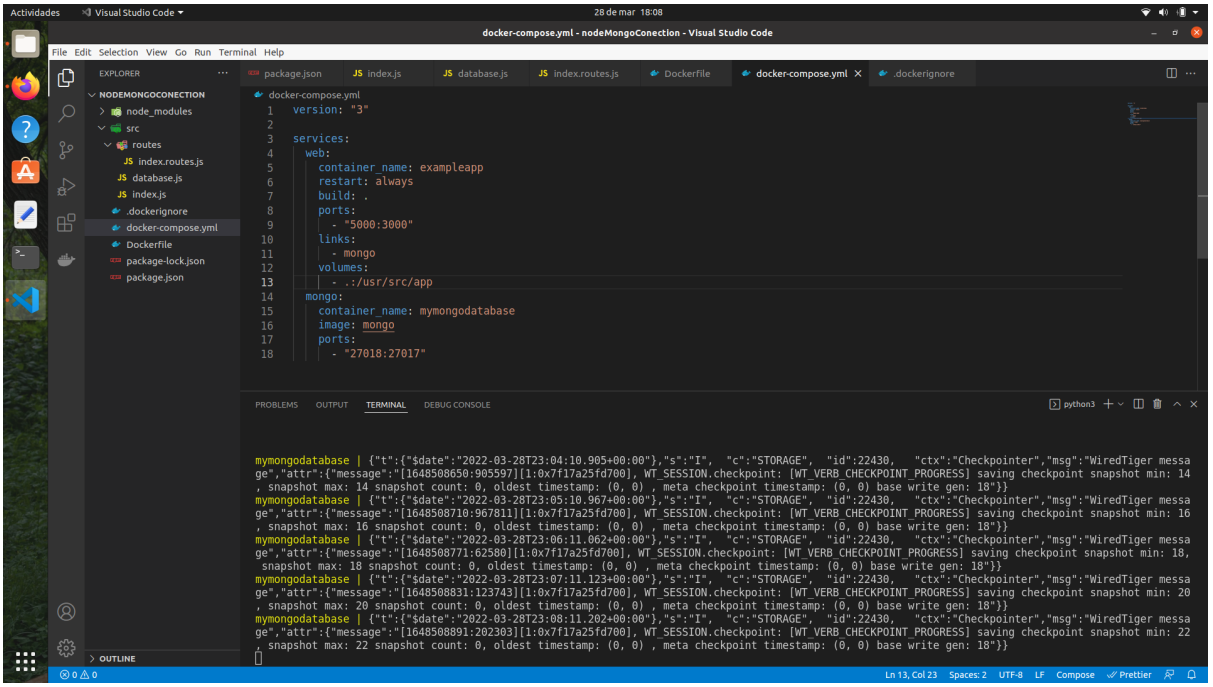


31. Dentro de la carpeta routes creamos un archivo llamado index.routes.js:

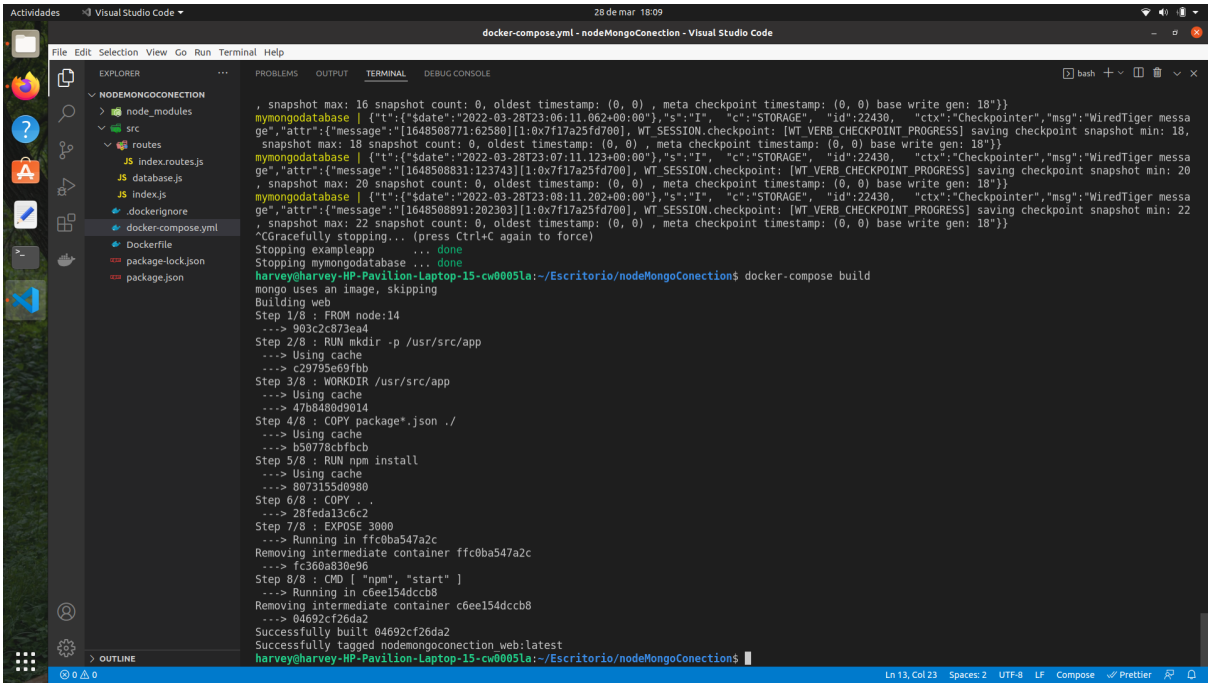


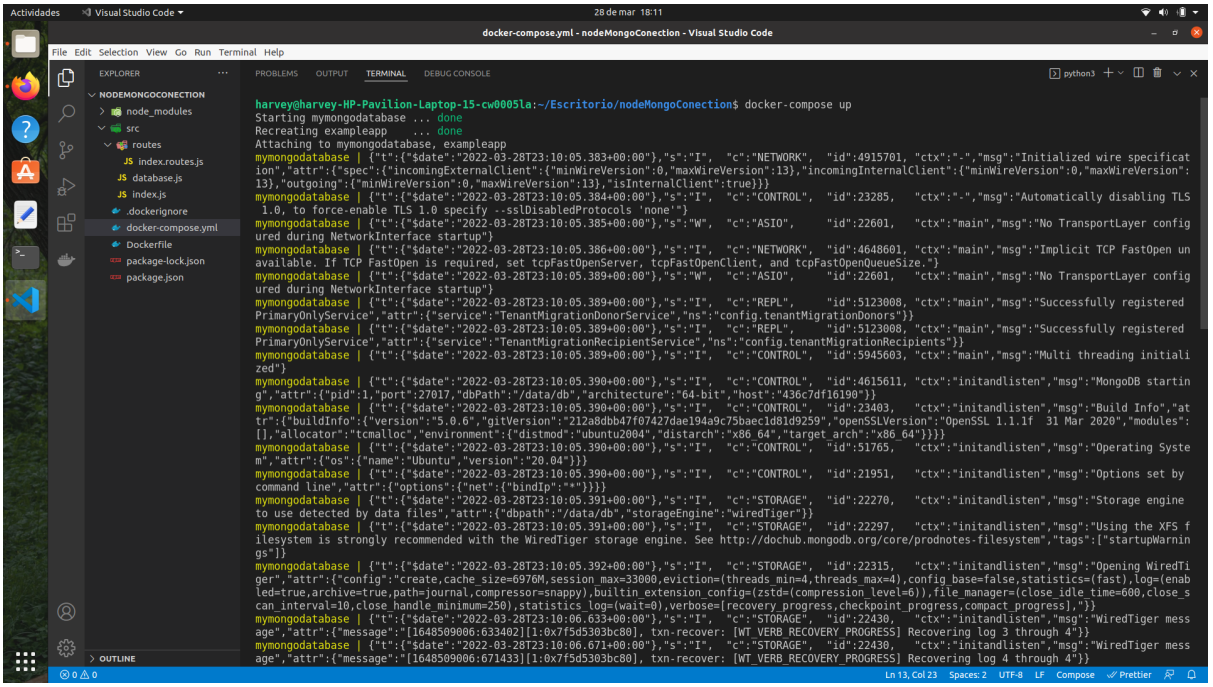


32. Volvemos a dar control + c para detener el proceso
33. Dentro del archivo docker-compose agregamos el término volumes que permite copiar en ambas direcciones (del proyecto al contenedor y del contenedor al proyecto) y modificamos de la siguiente manera:

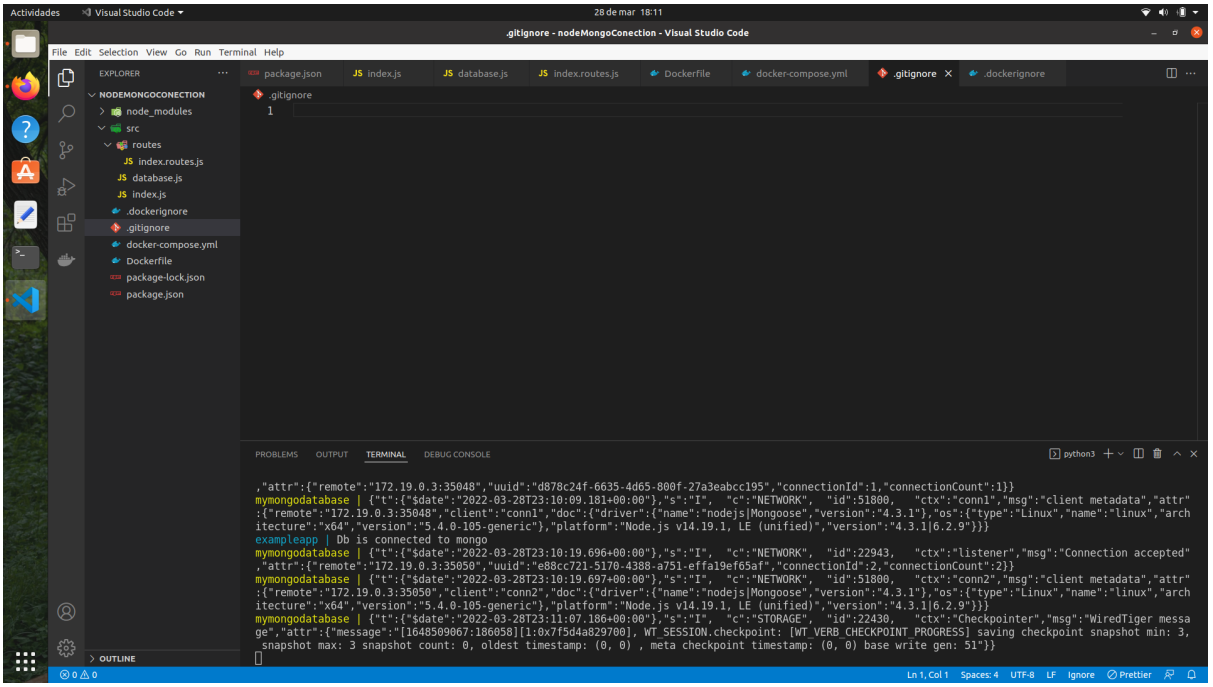


34. Ejecutamos docker-compose build y luego docker-compose up

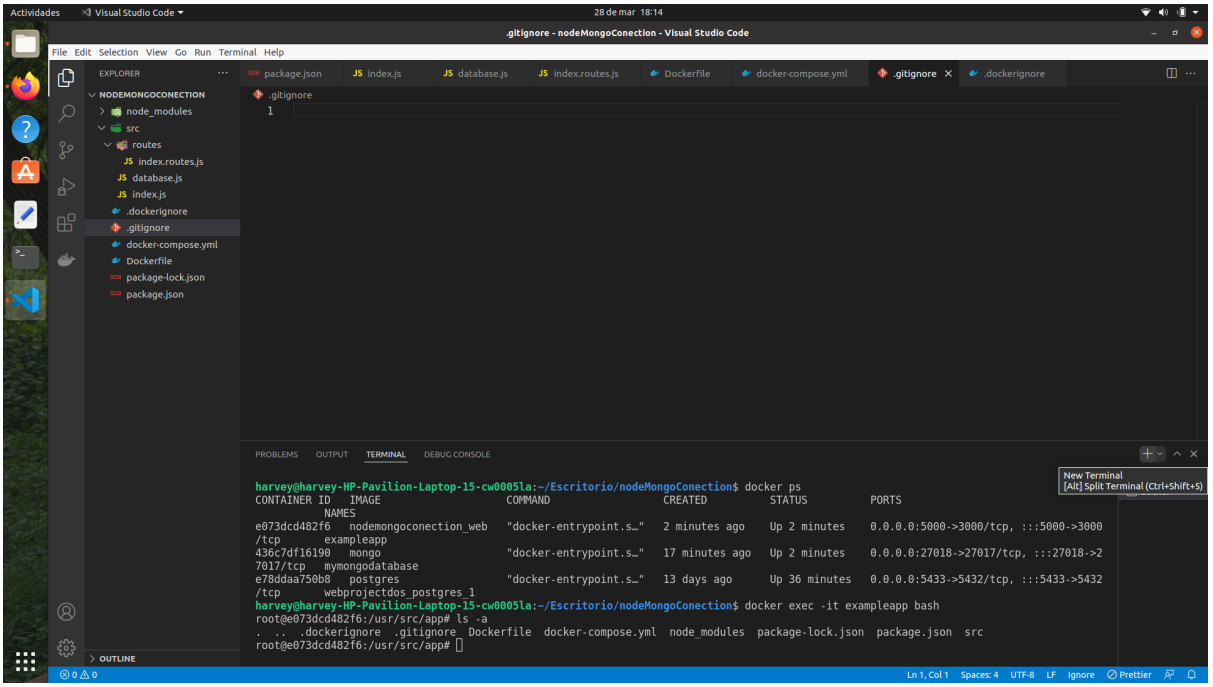


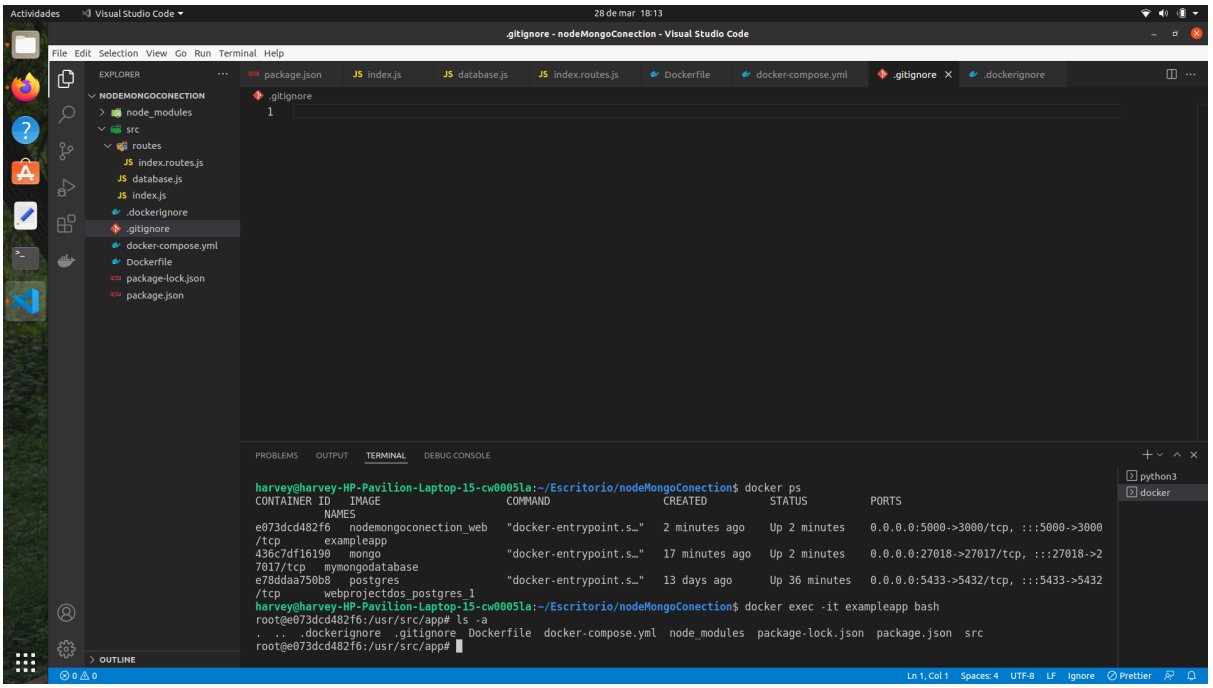


35. Para validar que el paso 33 funciona creamos un nuevo archivo llamado .gitignore dentro de la carpeta raíz del proyecto y agregamos la siguiente línea:

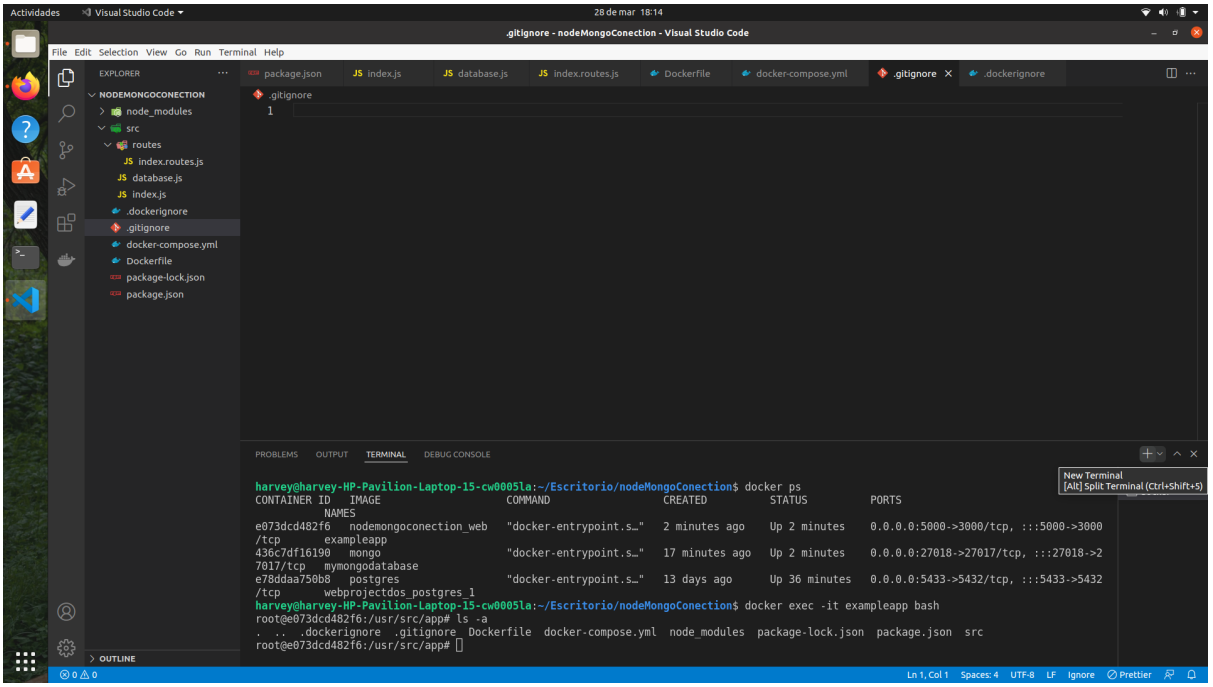


36. Abrimos una nueva pestaña de la terminal y ejecutamos docker ps

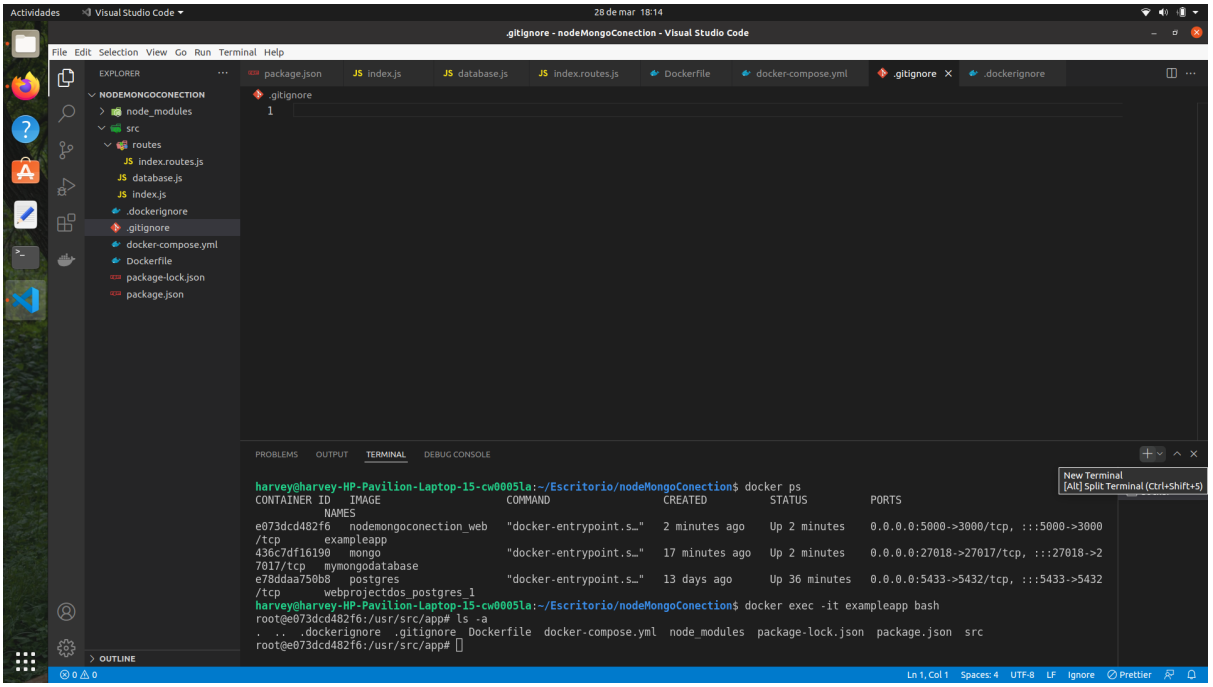




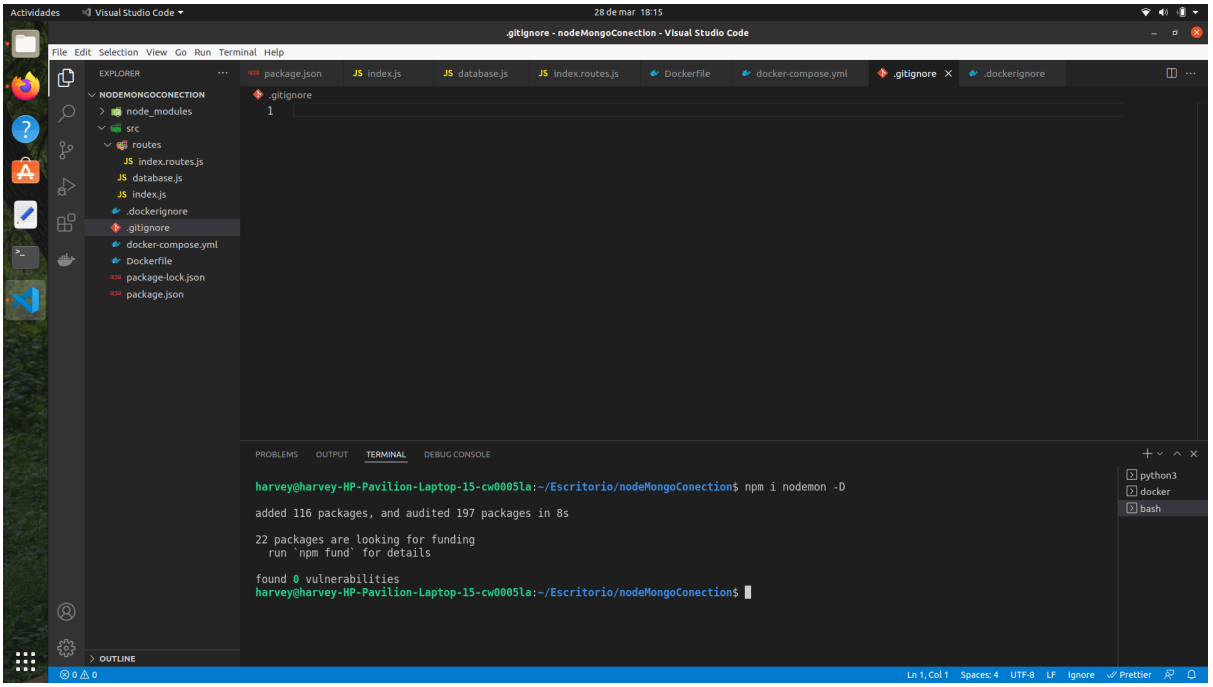
37. Nos conectamos por a la carpeta por medio del comando docker exec -it exampleapp bash



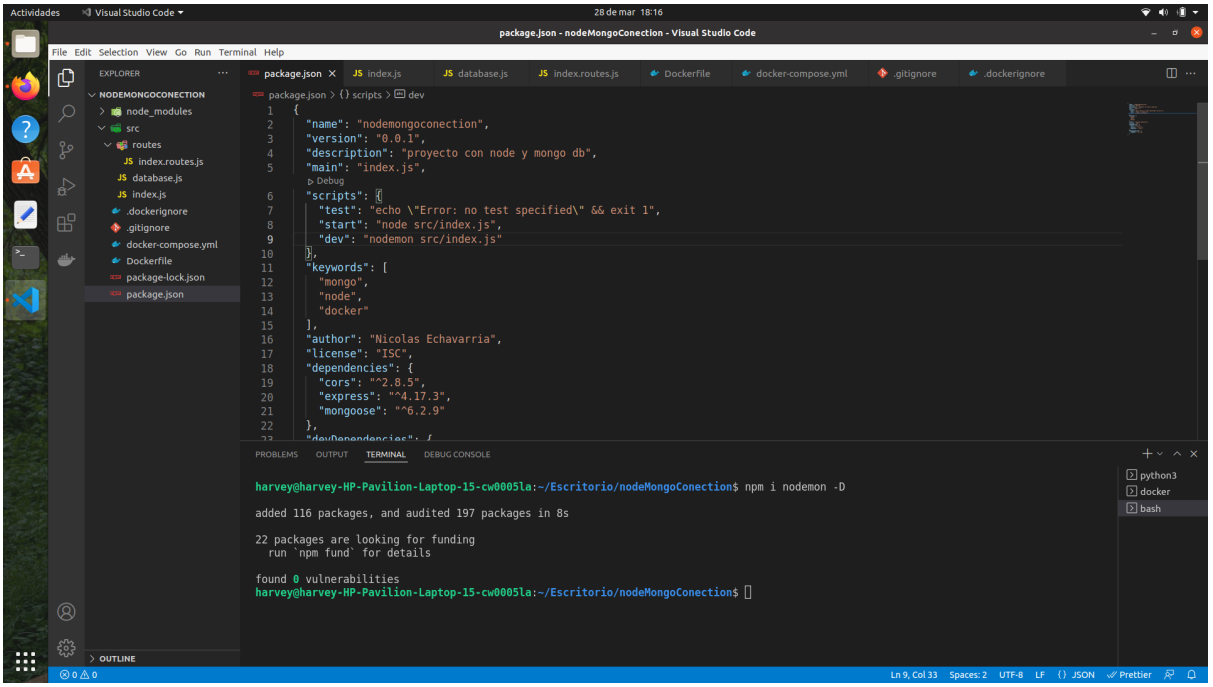
38. Luego escribimos ls -a y damos enter para ver el archivo .gitignore creado



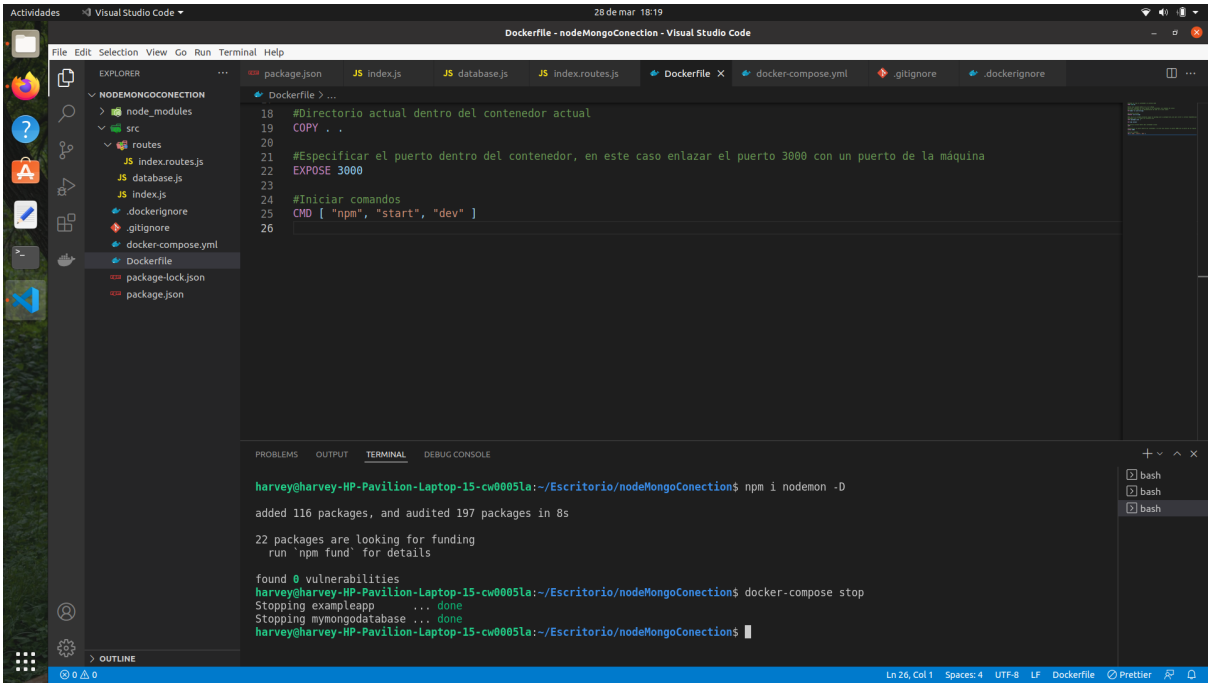
39. Abrimos una nueva consola y ejecutamos: npm i nodemon -D para poder poder obtener el paquete que facilita reiniciar el servidor automáticamente



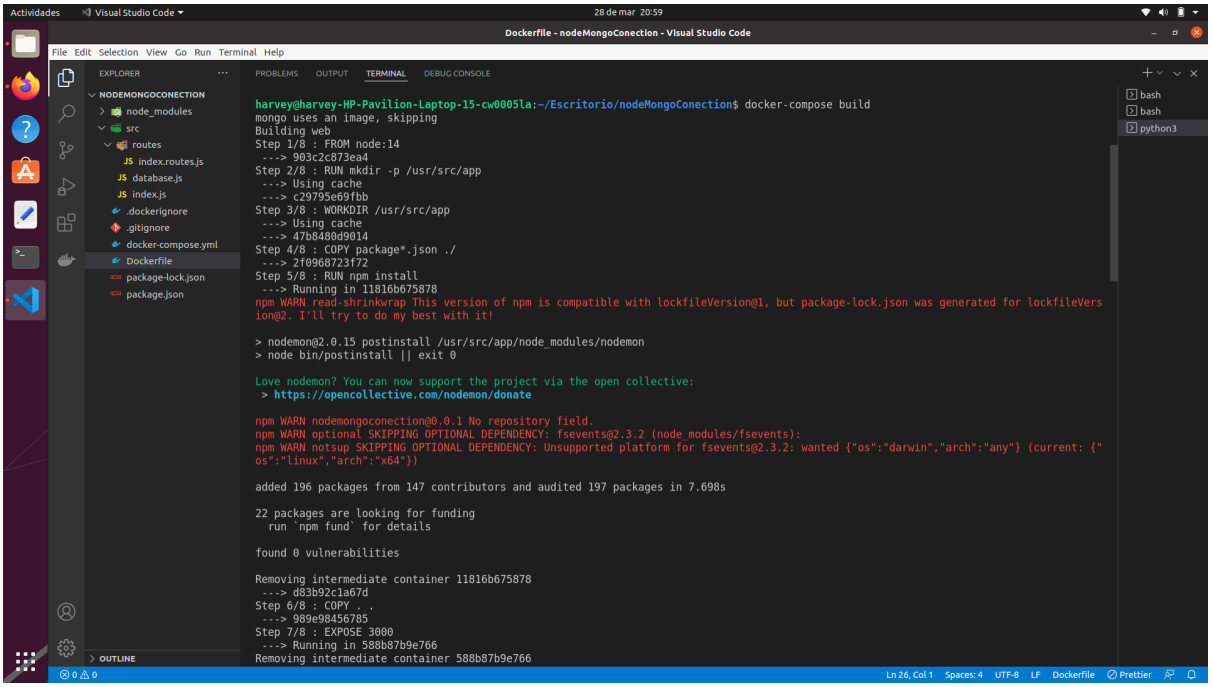
40. Luego abrimos el archivo package.json y agregamos un nuevo script para hacer que se inicie el contenedor con nodemon:



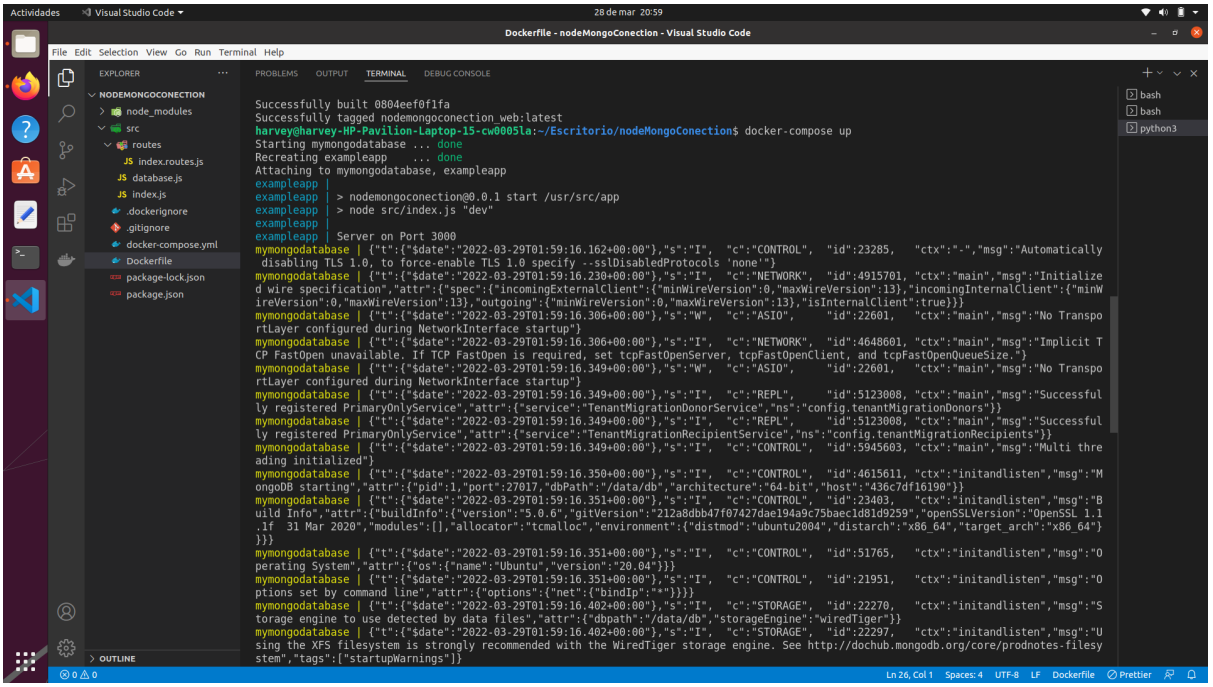
41. Realizamos el siguiente cambio en el archivo Dockerfile para que inicie con nodemon:



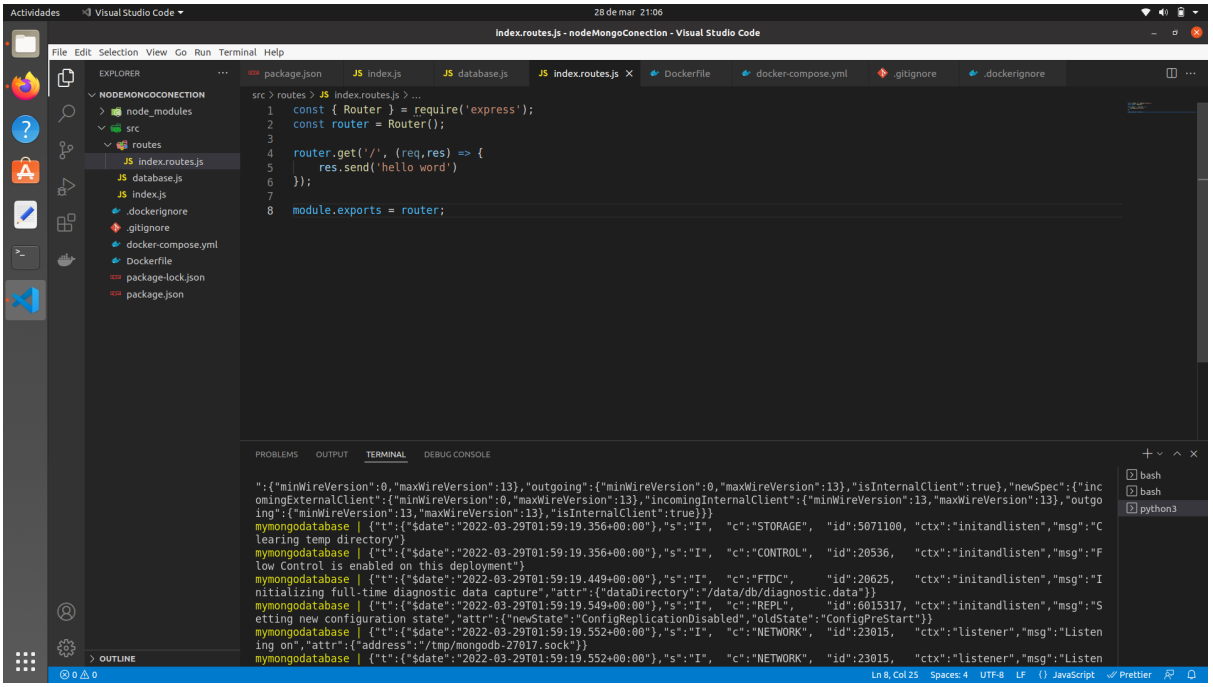
42. Presionamos control + c para parar la ejecución
43. Ejecutamos docker-compose build para volver a generar la imagen principal



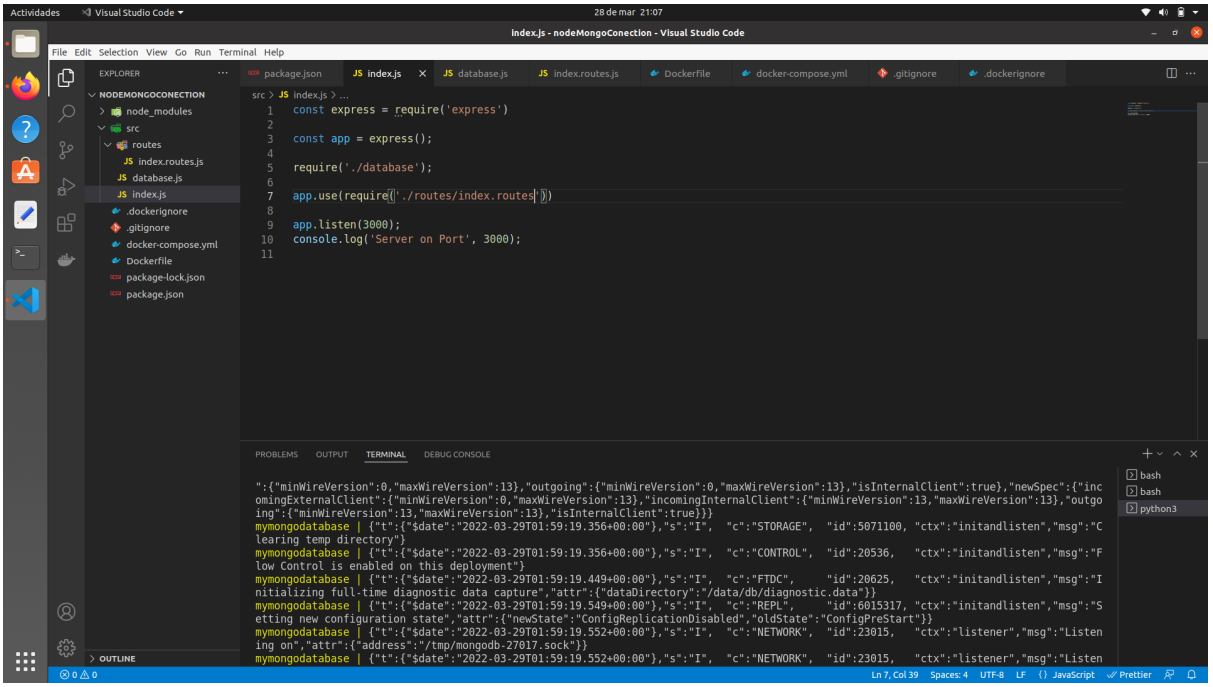
44. Ejecutamos docker-compose up



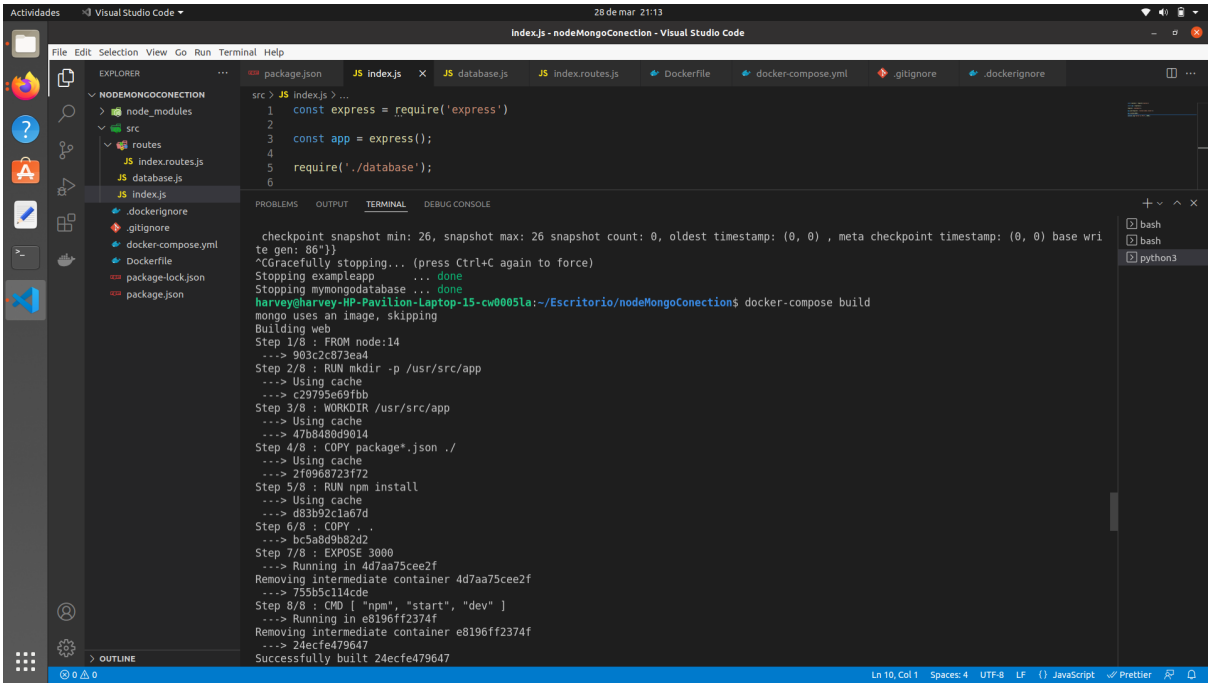
45. Abrimos el archivo index.routes.js y realizamos los siguientes cambios:

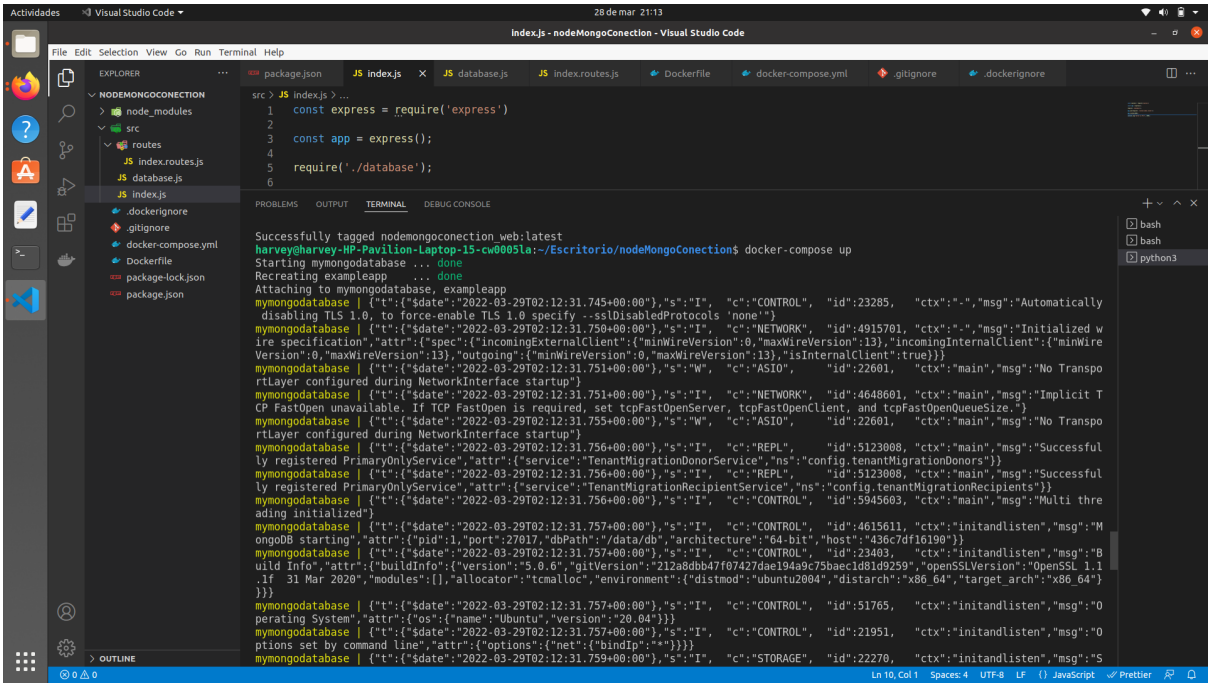


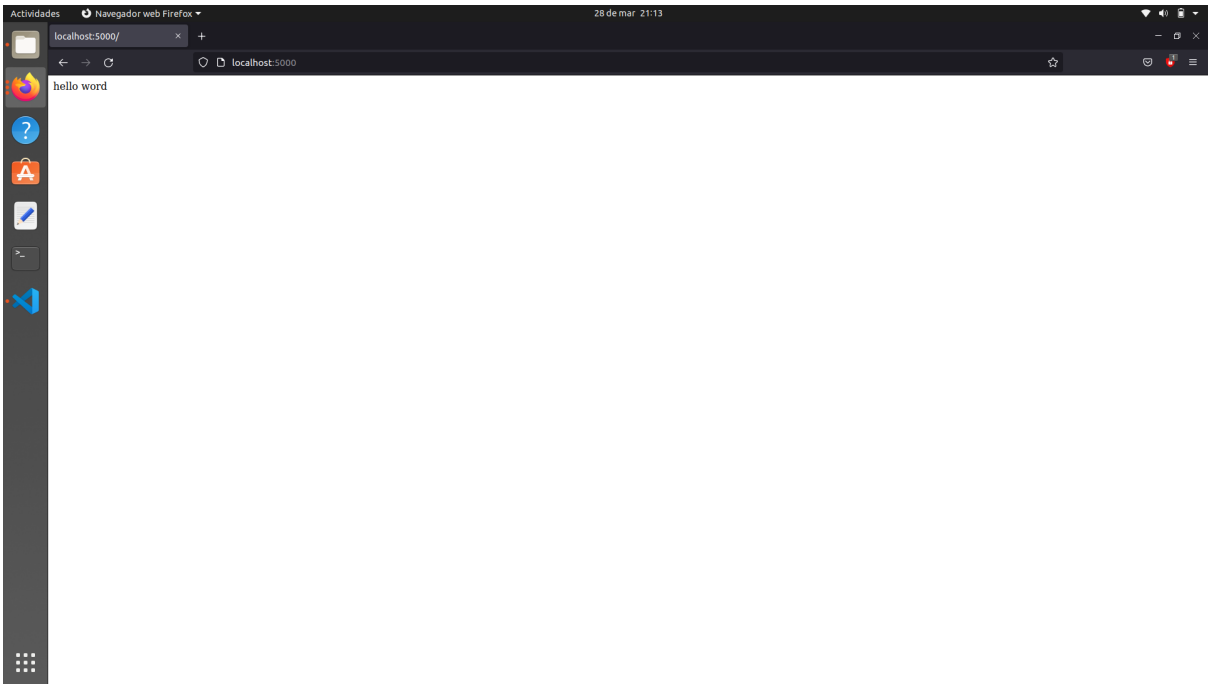
46. Abrimos el archivo index.js y realizamos los siguientes cambios:



47. Ejecutamos el comando docker-compose build, docker-compose up y verificamos en el navegador:




Actividades
Visual Studio Code
28 de mar 21:13
index.js - nodeMongoConection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NODEMONGOCONECTION
node_modules
src
routes
index.routes.js
database.js
index.js
.dockerignore
.gitignore
docker-compose.yml
Dockerfile
package-lock.json
package.json
OUTLINE
package.json
index.js
database.js
index.routes.js
DockerFile
docker-compose.yml
.gitignore
.dockerignore
src > index.js > ...
1 const express = require('express')
2
3 const app = express();
4
5 require('./database');
6
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
bash
bash
python3
Successfully tagged nodemongoconection web:latest
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker-compose up
Starting mymongodatabase ... done
Recreating exampleapp ... done
Attaching to mymongodatabase, exampleapp
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.745+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"-","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.750+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"-","msg":"Initialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true}}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.751+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.751+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.755+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.756+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","ns":"config.tenantMigrationDonors"}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.756+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationRecipientService","ns":"config.tenantMigrationRecipients"}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.756+00:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"main","msg":"Multi threading initialized"}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.757+00:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":1,"port":27017,"dbPath":"/data/db","architecture":"64-bit","host":"436c7df16190"}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.757+00:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"5.0.6","gitVersion":"212a8dbb47f07427dae194a9c75baec1d81d9259","openSSLVersion":"OpenSSL 1.1.1f 31 Mar 2020","modules":[],"allocator":"tcmalloc","environment":{"distmod":"ubuntu2004","distarch":"x86_64","target_arch":"x86_64"}}}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.757+00:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Ubuntu","version":"20.04"}}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.757+00:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{"net":{"bindIp":"*"}}}}
mymongodatabase | {"t":{"$date":"2022-03-29T02:12:31.759+00:00"},"s":"I", "c":"STORAGE", "id":22270, "ctx":"initandlisten","msg":"S
0 0
Ln 10, Col 1 Spaces: 4 UTF-8 LF {} JavaScript Prettier


Actividades
Navegador web Firefox
28 de mar 21:13
localhost:5000/
localhost:5000
hello word