

Enlaces:

<https://nodejs.org/es/>

Node.js® es un entorno de ejecución para JavaScript construido con [V8, motor de JavaScript de Chrome](#).



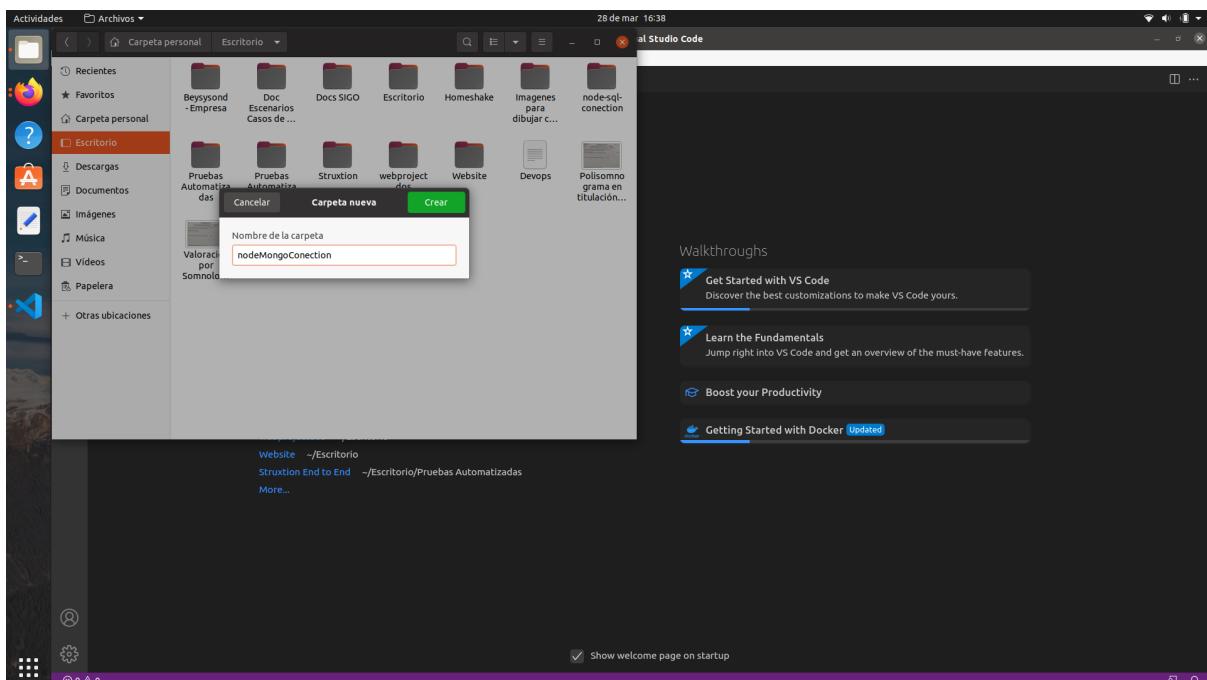
<https://www.mongodb.com/es>

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

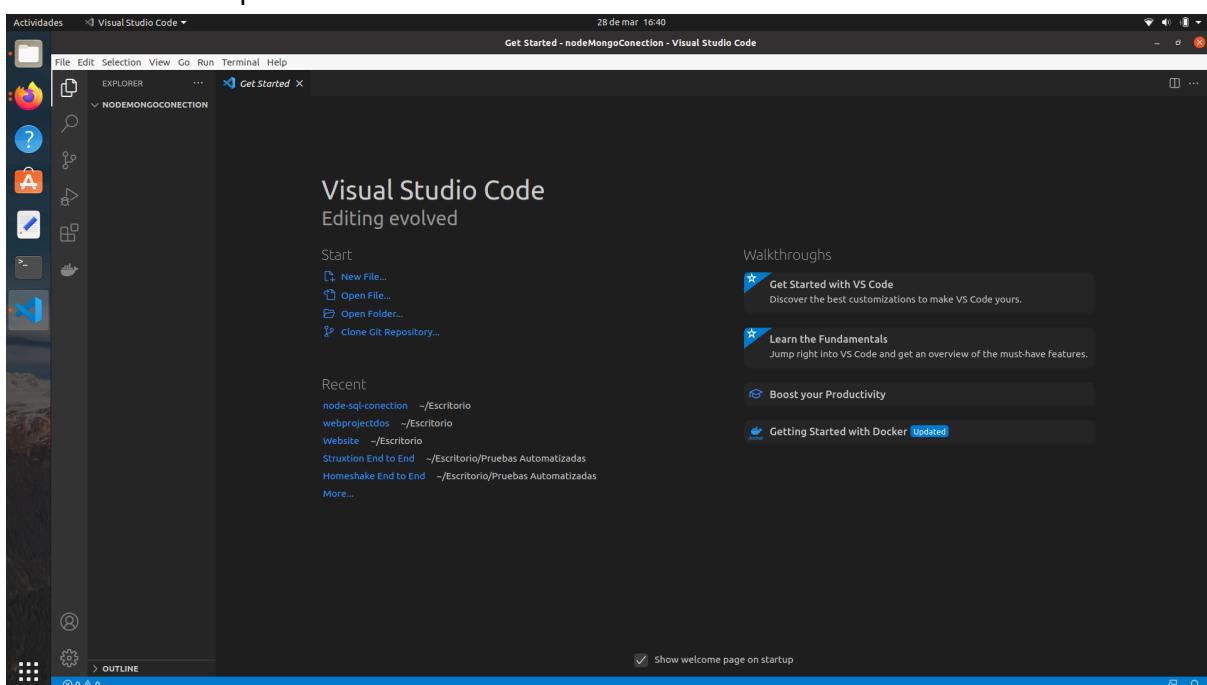


Node, Mongo & Docker Compose

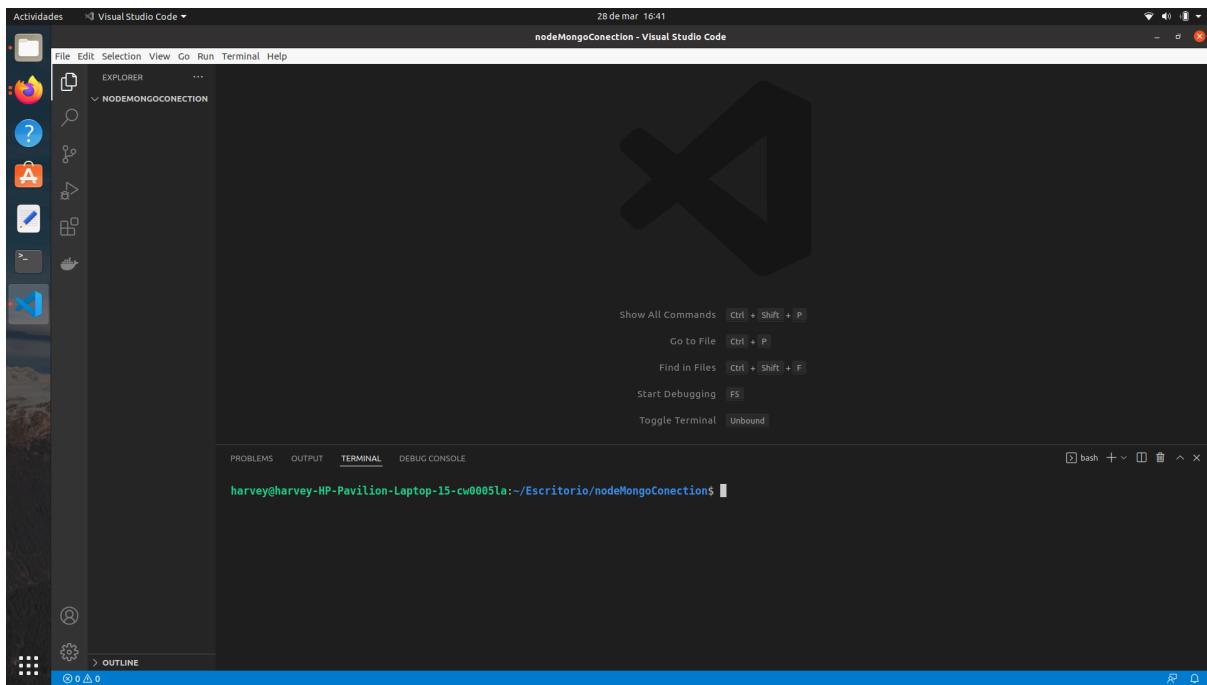
1. Crear una nueva carpeta para el proyecto:



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

```

28 de mar 16:44
package.json - nodeMongoConection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Press ^C at any time to quit.
package name: (nodemongoconection)
version: (1.0.0) 0.0.1
description: proyecto con node y mongo db
entry point: (index.js)
test command:
git repository:
keywords: mongo node docker
author: Nicolas Echavarria
license: (ISC)
About to write to /home/harvey/Escritorio/nodeMongoConection/package.json:

{
  "name": "nodemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"$Error: no test specified\\\" && exit 1"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC"
}

Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

```

Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JSON ✓ Prettier R Q


```

28 de mar 16:44
package.json - nodeMongoConection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

```

Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JSON ✓ Prettier R Q

- 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.

```

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

```

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

```
28 de mar 16:47
package.json - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json X
NODEMONGOCONNECTION
  node_modules
    package-lock.json
    package.json
1 {
  "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
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ 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/nodeMongoConnection$
```

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

```
28 de mar 16:47
package.json - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json X
NODEMONGOCONNECTION
  node_modules
    package-lock.json
    package.json
1 {
  "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"
  }
}
New File Error: no test specified\\\" && exit 1"
New Folder
Open Containing Folder Alt+Ctrl+R
Open in Integrated Terminal
Add Folder to Workspace...
Open Folder Settings
Remove Folder from Workspace
Find in Folder... Alt+Shift+F
Paste Ctrl+V
Copy Path Alt+Ctrl+C
Copy Relative Path Alt+Ctrl+Shift+C
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ 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/nodeMongoConnection$
```

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'NODEN...'. Inside it, there are 'srcl', 'node_modules', 'package-lock.json', and 'package.json'. The 'package.json' file is selected and open in the main editor area. The terminal at the bottom shows the command 'npm i express mongoose' being run, along with its output: 'added 78 packages, and audited 79 packages in 8s'. The status bar at the bottom right indicates 'Ln 1, Col 1 Spaces: 2 UTF-8 LF () JSON ✓ Prettier'.

```
28 de mar 16:47
package.json - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER
NODEN...
srcl
node_modules
package-lock.json
package.json

1 {
  "name": "nodemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Warning: no test specified\\\" && exit 1"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3",
    "mongoose": "^6.2.9"
  }
}
21

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ 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/nodeMongoConnection$ 
Ln 1, Col 1 Spaces: 2 UTF-8 LF () JSON ✓ Prettier
```

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.

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'NODEN...'. Inside it, there are 'srcl', 'node_modules', 'package.json', and 'src'. A context menu is open over the 'src' folder, listing options like 'New File', 'New Folder', 'Open Containing Folder', 'Open in Integrated Terminal', 'Find in Folder...', 'Cut', 'Copy', 'Paste', 'Copy Path', 'Copy Relative Path', 'Rename', and 'Delete'. The 'package.json' file is selected and open in the main editor area. The terminal at the bottom shows the command 'npm i express mongoose' being run, along with its output: 'added 78 packages, and audited 79 packages in 8s'. The status bar at the bottom right indicates 'Ln 1, Col 1 Spaces: 2 UTF-8 LF () JSON ✓ Prettier'.

```
28 de mar 16:48
package.json - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER
NODEN...
srcl
node_modules
package.json
src

1 {
  "name": "nodemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Warning: no test specified\\\" && exit 1"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3",
    "mongoose": "^6.2.9"
  }
}
21

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ 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/nodeMongoConnection$ 
Ln 1, Col 1 Spaces: 2 UTF-8 LF () JSON ✓ Prettier
```

Actividades ➔ Visual Studio Code ➔ package.json - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ➔ NODEN... ➔ node_modules ➔ src ➔ index.js ➔ package-lock.json ➔ package.json

```
package.json > ...
```

```
1 {  
2   "name": "nodemongoconection",  
3   "version": "0.0.1",  
4   "description": "proyecto con node y mongo db",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\" Error: no test specified\\\" && exit 1"  
8   },  
9   "keywords": [  
10     "mongo",  
11     "node",  
12     "docker"  
13   ],  
14   "author": "Nicolas Echavarria",  
15   "license": "ISC",  
16   "dependencies": {  
17     "express": "^4.17.3",  
18     "mongoose": "^6.2.9"  
19   }  
20 }  
21
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection\$ 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/nodeMongoConnection\$

Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JSON ✓ Prettier R Q

Actividades ➔ Visual Studio Code ➔ index.js - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ➔ NODEN... ➔ node_modules ➔ src ➔ index.js ➔ package.json

```
index.js > ...
```

src ➔ index.js

1

src ➔ index.js

New File New Folder Open Containing Folder Open in Integrated Terminal

Find in Folder... Cut Copy Paste

Copy Path Alt+Ctrl+C Copy Relative Path Alt+Ctrl+Shift+C

Rename F2 Delete Supr

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Is this OK? (yes) yes
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection\$ 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/nodeMongoConnection\$

Ln 1, Col 1 Spaces: 4 UTF-8 LF {} JavaScript ✓ Prettier R Q

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the following command and its execution:

```
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$
```

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

The screenshot shows the Visual Studio Code interface with the code editor tab selected. The code editor displays the following configuration for the index.js file:

```
const express = require('express');
const app = express();
app.listen(3000);
console.log('Server on Port', 3000);
```

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

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure named 'NODEMONGOCONNECTION'. Inside 'src', there are files 'database.js' and 'index.js'. The 'index.js' file is open in the main editor area, showing the following code:

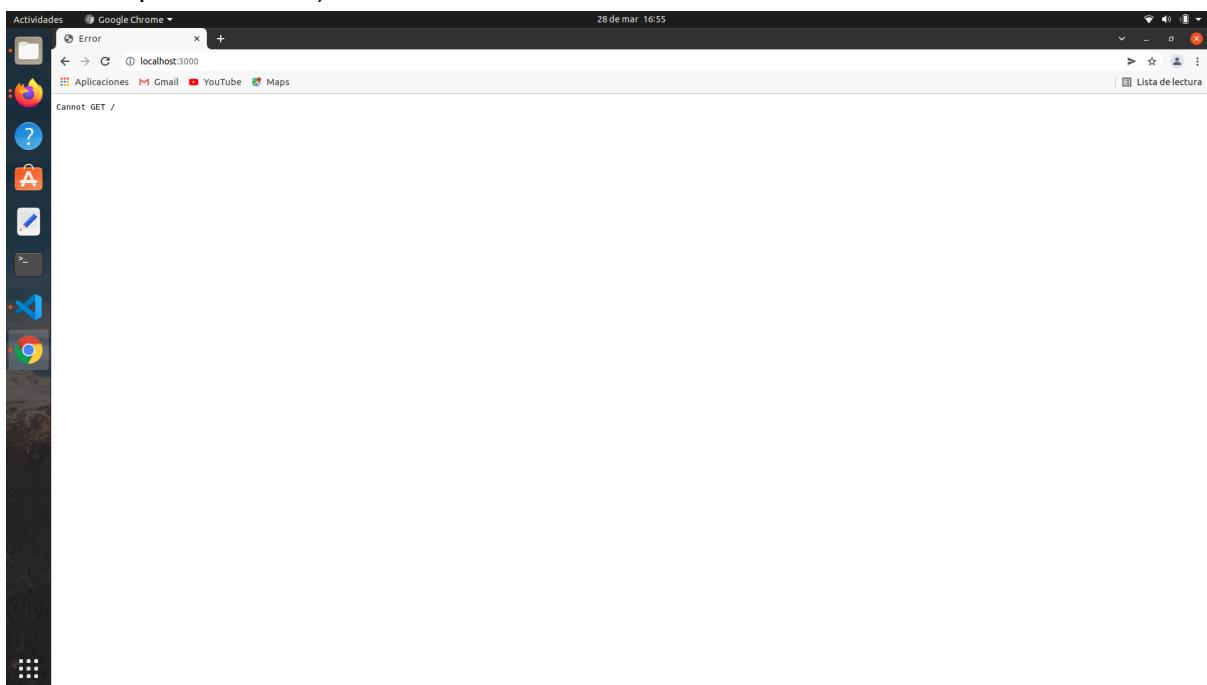
```
const express = require('express');
const app = express();
app.listen(3000);
console.log('Server on Port 3000');
```

Below the editor, the 'TERMINAL' tab is selected, showing the command line output:

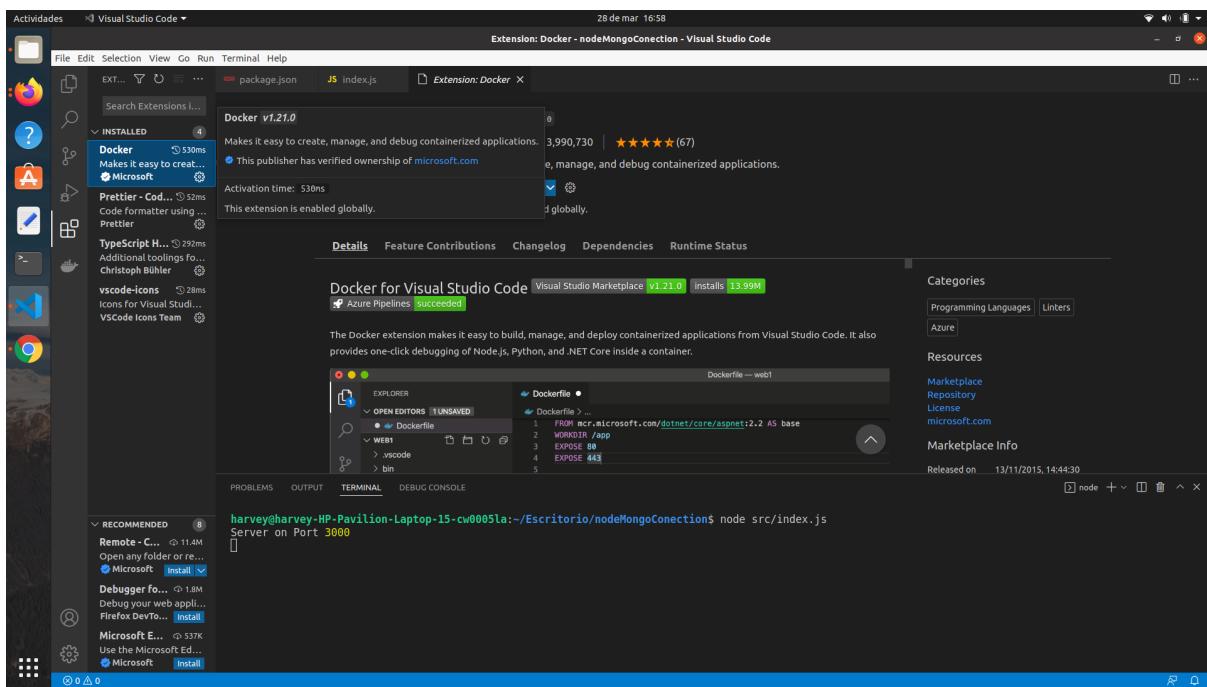
```
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ node src/index.js
Server on Port 3000
```

The status bar at the bottom indicates the file is a JavaScript file ('JavaScript') and shows line 7, column 1.

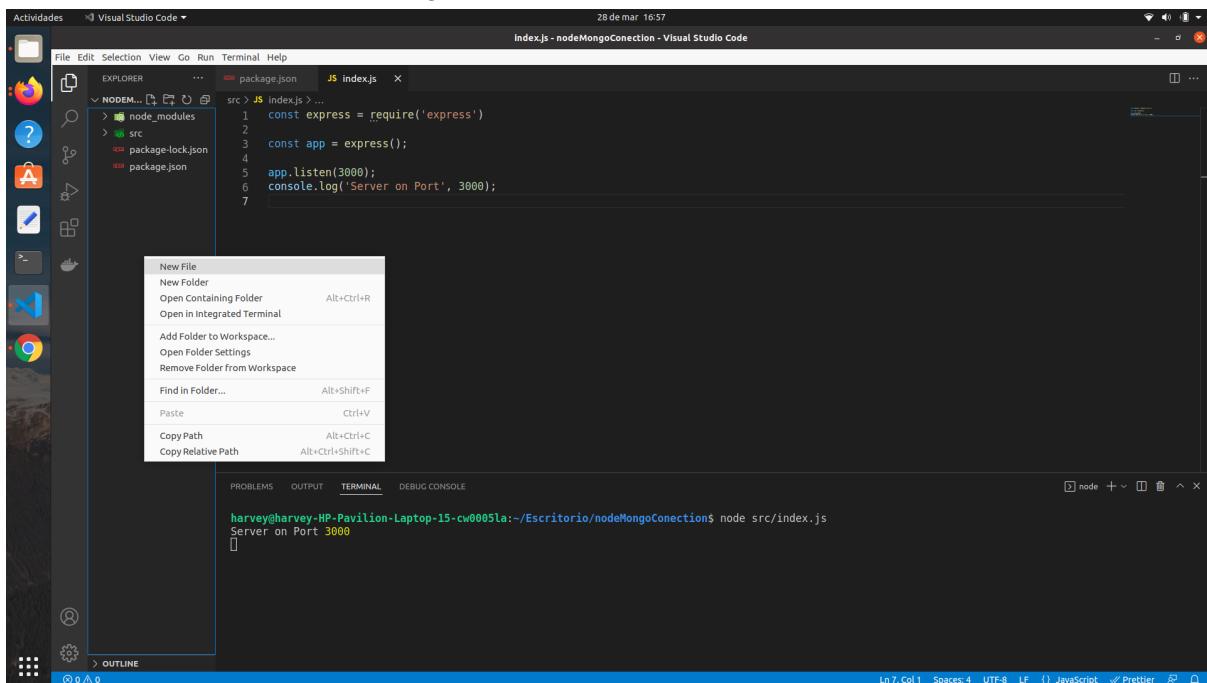
11. Verificamos en el navegador: (El servidor se encuentra funcionando pero no hemos respondido nada)



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)



The screenshot shows two instances of Visual Studio Code side-by-side. The left instance is titled 'Index.js - nodeMongoConnection - Visual Studio Code' and displays the following code:

```

const express = require('express');
const app = express();
app.listen(3000);
console.log('Server on Port', 3000);

```

The right instance is titled 'Dockerfile - nodeMongoConnection - Visual Studio Code' and displays the following Dockerfile:

```

#Cuando se crea el contenedor se instala node
FROM node:14

#Crear una carpeta donde va a ir el código
#Ejecutar comandos de Linux (Docker es un contenedor con comandos de Linux)
#La ruta esta dada por la documentación de node (es la mas común)
RUN mkdir -p /usr/src/app

#Directorio actual
WORKDIR /usr/src/app

#Una vez en la carpeta necesito copiar el package.json y package-lock.json para volver a instalar dependencias dentro del contenedor .
#package*.json -> Todos los archivos json
COPY package*.json .

RUN npm install

#Directorio actual dentro del contenedor actual

```

- 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`)

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure named "NODEMONGOCONNECTION". Inside "src", there are files: ".dockerignore", "Dockerfile", "package-lock.json", and "package.json".
- Code Editor:** Displays the content of the ".dockerignore" file:

```
1 #Evitar copiar carpeta node_modules si existe
2 node_modules
```
- Terminal:** Shows the command "node src/index.js" being run, with the output "Server on Port 3000".
- Status Bar:** Shows the file is 2 lines long, 13 columns wide, and uses UTF-8 encoding.

15. Realizamos la siguiente modificación en el archivo Dockerfile:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure named "NODEMONGOCONNECTION". Inside "src", there are files: "database.js", "index.js", ".dockerignore", and "Dockerfile".
- Code Editor:** Displays the content of the "Dockerfile":

```
1 #Cuando se crea el contenedor se instala node
2 FROM node:14
3
4 #Crear una carpeta donde va a ir el código
5 #Ejecutar comandos de Linux (Docker es un contenedor con comandos de Linux)
6 #La ruta esta dada por la documentación de node (es la mas común)
7 RUN mkdir -p /usr/src/app
8
9 #Directorio actual
10 WORKDIR /usr/src/app
11
12 #Una vez en la carpeta necesito copiar el package.json y package-lock.json para volver a instalar dependencias dentro del contenedor .
13 #package*.json --> Todos los archivos json
14 COPY package*.json .
15
16 RUN npm install
17
18 #Directorio actual dentro del contenedor actual
19 COPY .
20
21 #Especificar el puerto dentro del contenedor, en este caso enlazar el puerto 3000 con un puerto de la máquina
22 EXPOSE 3000
23
24 #Iniciar comandos
25 CMD [ "npm", "start" ]
```
- Terminal:** Shows the command "node src/index.js" being run.
- Status Bar:** Shows the file is 21 lines long, 24 columns wide, and uses UTF-8 encoding.

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

```

{
  "name": "nudemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"$Error: no test specified\\\" && exit 1",
    "start": "node src/index.js"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3",
    "mongoose": "^6.2.9"
  }
}

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection\$ node src/index.js
Server on Port 3000
^C
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection\$

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:

```

{
  "name": "nudemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"$Error: no test specified\\\" && exit 1",
    "start": "node src/index.js"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3",
    "mongoose": "^6.2.9"
  }
}

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection\$ node src/index.js
Server on Port 3000
^C
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection\$ npm start
> nudemongoconection@0.0.1 start
> node src/index.js
Server on Port 3000
^C
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection\$

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)

```

28 de mar. 17:15
Dockerfile - nodeMongoConection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js Dockerfile .dockerignore
NODEMONGOCONNECTION
src database.js index.js Dockerfile Dockerfile
package-lock.json package.json

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker build -t hellonode .
Sending build context to Docker daemon 58.37kB
Step 1/7 : FROM node:14
14: Pulling from library/node
d172cccd78ea: Pull complete
4f10b5790398: Pull complete
01ae525c10a7: Pull complete
722150487783: Pull complete
8cfac9c61a0a: Pull complete
7f6b791c006f: Pull complete
6138e26477a6: Pull complete
62d956a575b5: Pull complete
8ac0f51797f1: Pull complete
Digest: sha256:7d30b5ed42b2a006c3a79ef8ad9f1e912bde6cb4243c188689d5aalaa437
Status: Downloaded newer image for node:14
...
Step 2/7 : RUN mkdir -p /usr/src/app
--> Running in c5b9f16f2284
Removing intermediate container c5b9f16f2284
--> c86aa8a85561
Step 3/7 : WORKDIR /usr/src/app
--> Running in 2c7d79eedc1f
Removing intermediate container 2c7d79eedc1f
--> 4cf2fbb75313
Step 4/7 : COPY package*.json ../
--> lee48b70bed1
Step 5/7 : COPY .
--> bdc0df3448a0
Step 6/7 : EXPOSE 3000
--> Running in c6a85011fe58
Removing intermediate container c6a85011fe58
--> 78007c524130
Step 7/7 : CMD [ "npm", "start" ]
--> Running in a70d3ae50dd2
Removing intermediate container a70d3ae50dd2
--> 481528ed7a50
Successfully built 481528ed7a50
Successfully tagged hellonode:latest
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

Ln 21, Col 1 Spaces: 4 UTF-8 LF Dockerfile Prettier
```



```

File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
NODEMONGOCONNECTION
src database.js index.js Dockerfile Dockerfile
package-lock.json package.json

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker build -t hellonode .
Sending build context to Docker daemon 59.39kB
Step 1/8 : FROM node:14
14: Pulling from library/node
...
Step 2/8 : RUN mkdir -p /usr/src/app
--> Running in d90d3a91c88e
Removing intermediate container d90d3a91c88e
--> C29795e69fb
Step 3/8 : WORKDIR /usr/src/app
--> Running in c691346271a7
Removing intermediate container c691346271a7
--> 47b480d9014
Step 4/8 : COPY package*.json ../
--> e5568be9f27f
Step 5/8 : RUN npm install
--> Running in 92b3779c7100
npm WARN read-shrinkwrap This version of npm is compatible with lockfileVersion@1, but package-lock.json was generated for lockfileVersion@2. I'll try to do my best with it!
npm WARN nodemongoconection@0.0.1 No repository field.

added 80 packages from 101 contributors and audited 80 packages in 3.013s
6 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

Removing intermediate container 92b3779c7100
--> d99231afe32f
Step 6/8 : COPY .
--> 8ccb71377933
Step 7/8 : EXPOSE 3000
--> Running in 6af7e304ab39
Removing intermediate container 6af7e304ab39
--> acf24edd7c62
Step 8/8 : CMD [ "npm", "start" ]
--> Running in eadfcde631e
Removing intermediate container eadfcde631e
--> dd8bef653c95
Successfully built dd8bef653c95
Successfully tagged hellonode:latest
```

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.

```

28 de mar. 17:16
Dockerfile - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js Dockerfile .dockerignore
NODEMONGOCONNECTION
  node_modules
  src
    database.js
    index.js
    dockerignore
    Dockerfile
      package-lock.json
      package.json

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hellonode latest 481528ed7a50 32 seconds ago 946MB
node 14 983c2b73ea4 10 days ago 946MB
dpage/pgadmin4 latest 4b5bbdd3624 2 weeks ago 340MB
postgres latest d7337c285715 2 weeks ago 376MB
nginx latest c9198454c2b 3 weeks ago 142MB
mysql/mysql-server latest 434c35b82b68 2 months ago 417MB
mariadb/server 10.4 6bc393d66bb 10 months ago 353MB
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$
```

Ln 21, Col 1 Spaces: 4 UTF-8 LF Dockerfile ⚙ Prettier ⚙

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):

```

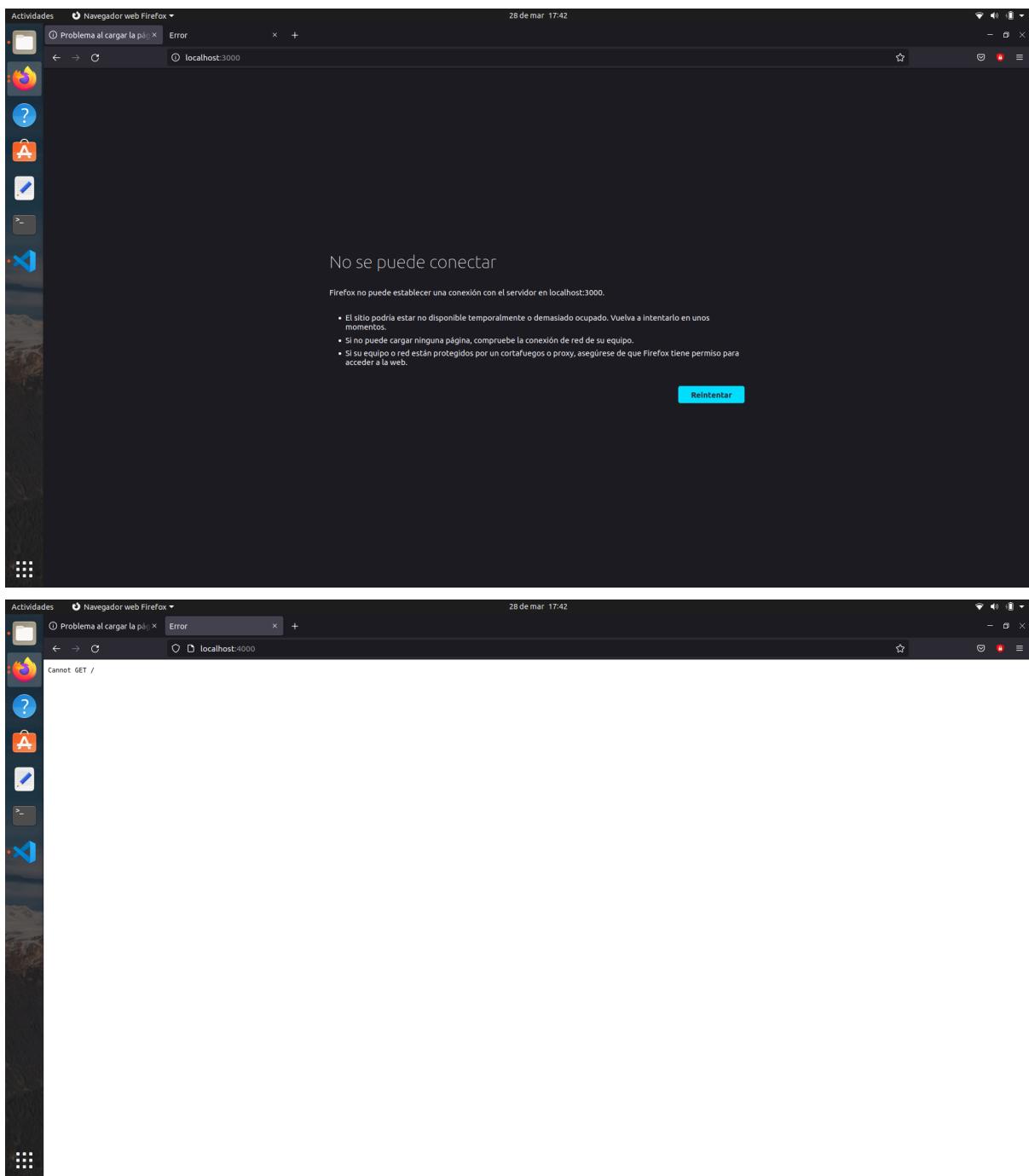
28 de mar. 17:42
Dockerfile - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js Dockerfile
NODEMONGOCONNECTION
  node_modules
  src
    database.js
    index.js
    dockerignore
    Dockerfile
      package-lock.json
      package.json

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker run -p 4000:3000 hellonode
> nodemongocnection@0.0.1 start /usr/src/app
> node src/index.js
Server on Port 3000
|
```

Ln 3, Col 1 Spaces: 4 UTF-8 LF Dockerfile ⚙ Prettier ⚙

21. Validamos en el navegador:



22. Cancelamos el proceso con control + c

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure with a root folder named "NODEMONGOCONNECTION". Inside it are "node_modules", "src" (containing "database.js", "index.js", "dockerignore", and "Dockerfile"), "package-lock.json", and "package.json".
- Dockerfile Content:**

```

FROM node:14
#Crear una carpeta donde va a ir el código
#Ejecutar comandos de Linux (Docker es un contenedor con comandos de Linux)
#La ruta esta dada por la documentación de node (es la mas común)
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
#Una vez en la carpeta necesito copiar el package.json y package-lock.json para volver a instalar dependencias dentro del contenedor .
#package*.json -> Todos los archivos json
COPY package*.json .
RUN npm install
#Directorio actual dentro del contenedor actual
COPY .
#Especificar el puerto dentro del contenedor, en este caso enlazar el puerto 3000 con un puerto de la máquina
EXPOSE 3000
    
```
- Terminal Output:**

```

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker run -p 4000:3000 hellonode
> nodemongoconection@0.0.1 start /usr/src/app
> node src/index.js

Server on Port 3000
^C^C^C harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 
    
```

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

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure with a root folder named "NODEMONGOCONNECTION". Inside it are "node_modules", "src" (containing "database.js", "index.js", "dockerignore", and "Dockerfile"), "package-lock.json", and "package.json".
- Dockerfile Content:**

```

FROM node:14
#Crear una carpeta donde va a ir el código
#Ejecutar comandos de Linux (Docker es un contenedor con comandos de Linux)
#La ruta esta dada por la documentación de node (es la mas común)
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
#Una vez en la carpeta necesito copiar el package.json y package-lock.json para volver a instalar dependencias dentro del contenedor .
#package*.json -> Todos los archivos json
COPY package*.json .
RUN npm install
#Directorio actual dentro del contenedor actual
COPY .
#Especificar el puerto dentro del contenedor, en este caso enlazar el puerto 3000 con un puerto de la máquina
EXPOSE 3000
    
```
- Context Menu (Open In Integrated Terminal):** A context menu is open over the Dockerfile, with the following options visible:
 - New File
 - New Folder
 - Open Containing Folder
 - Open in Integrated Terminal
 - Add Folder to Workspace...
 - Open Folder Settings
 - Remove Folder from Workspace
 - Find in Folder...
 - Paste
 - Copy Path
 - Copy Relative Path
- Terminal Output:**

```

n-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker run -p 4000:3000 hellonode
> nodemongoconection@0.0.1 start /usr/src/app
> node src/index.js

Server on Port 3000
^C^C^C harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 
    
```

The screenshot displays two terminal sessions in Visual Studio Code. Both sessions show a command-line interface for Docker operations.

Top Terminal Session:

```
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker run -p 4000:3000 hellonode
> nodemongoconection@0.0.1 start /usr/src/app
> node src/index.js
Server on Port 3000
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$
```

Bottom Terminal Session:

```
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker run -p 4000:3000 hellonode
> nodemongoconection@0.0.1 start /usr/src/app
> node src/index.js
Server on Port 3000
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$
```

Code Editor Content (Bottom Window):

```
version: "3"
services:
  web:
    container_name: exampleapp
    restart: always
    build:
      .
    ports:
      - "5000:3000"
    links:
      - mongo
  mongo:
    container_name: mymongodatabase
    image: mongo
    ports:
      - "27018:27017"
```

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

The screenshot shows a Visual Studio Code interface. The Explorer sidebar on the left displays a project structure for 'NODEMONGOCONNECTION' with folders like '.node_modules', 'src', and files such as 'database.js', 'index.js', 'Dockerfile', 'docker-compose.yml', 'package-lock.json', and 'package.json'. The main editor area shows the content of 'database.js':

```
src > JS database.js > catch() callback
1 const mongoose = require('mongoose')
2
3 mongoose.connect('mongodb://mongo/mydatabase')
4   .then(db => console.log('Db is connected to', db.connection.host))
5   .catch(error => console.error(error));
```

The terminal at the bottom shows the command being run:

```
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker run -p 4000:3000 hellonode
> nodemongoconection@0.0.1 start /usr/src/app
> node src/index.js
Server on Port 3000
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$
```

25. Ejecutamos el comando docker-compose build

26. Ejecutamos docker images y vemos que se ha creado una imagen node-docker-mongo_web:

The screenshot shows a Visual Studio Code interface. The Explorer sidebar on the left displays a project structure for 'NODEMONGOCONNECTION' with files like '.dockerignore', 'database.js', 'index.js', 'Dockerfile', 'docker-compose.yml', 'package-lock.json', and 'package.json'. The main editor area shows the content of '.dockerignore':

```
.dockerignore
1 #Evitar copiar carpeta node_modules si existe
2 node_modules
```

The terminal at the bottom shows the command being run:

```
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
nodemongoconection_web  latest   a99542a6fb7b  37 seconds ago  963MB
hellonode           latest   ddd8eaf653c9  26 minutes ago  963MB
node                14      993c2c873ea4  10 days ago   946MB
dpage/pgadmin4       latest   4b5bbddd3624  2 weeks ago   340MB
postgres            latest   d7337c283715  2 weeks ago   376MB
nginx               latest   c919845c4c2b  3 weeks ago   142MB
mysql/mysql-server   latest   434c35b82bb8  2 months ago  417MB
mariadb/server       10.4    6bc393df66bb  10 months ago 353MB
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$
```

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

```

28 de mar. 17:55
dockerignore - nodeMongoConnection - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
NODEMONGOCONNECTION
  > JS database.js
  > JS index.js
  > dockerignore
  > docker-compose.yml
  > Dockerfile
  > package-lock.json
  > package.json
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker-compose up
Creating network "nodemongoconnection_default" with the default driver
Pulling mongo (mongo:...).
latest: Pulling from library/mongo
4d32b49e2995: Pull complete
c6a9f9fa5cb8: Pull complete
c6a26a1adeb9: Pull complete
0f6c4c4a29a8: Pull complete
87cd51bf7ebc: Pull complete
68750e0424ec: Pull complete
0089000bad1d7: Pull complete
e33eed19868f: Pull complete
e7bc3c3cfdaeb: Pull complete
358effa21051: Pull complete
Digest: sha256:ad947856d0716ddd8b9cc525e341c77208ed8dafcb4a6ad23f9b3add7a4f71c
Status: Downloaded newer image for mongo:latest
Creating mongo... done
Creating exampleapp... done
Attaching to mongo,mongo:main
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.489+00:00"}, "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "", "msg": "Initialized wire specification", "attr": {"spec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}, "isInternalClient": true}}}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.490+00:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "main", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.492+00:00"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer config used during NetworkInterface startup"}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.492+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."}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.496+00:00"}, "s": "I", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer config used during NetworkInterface startup"}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.496+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationonorService", "ns": "config.tenantMigrationOnBoros"}}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.496+00:00"}, "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "ns": "config.tenantMigrationRecipients"}}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.496+00:00"}, "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "main", "msg": "Multi threading initialized"}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.497+00:00"}, "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting up", "attr": {"pid": 27017, "port": "27017", "dbPath": "/data/db", "architecture": "64-bit", "host": "430c7df16190"}}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.497+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"}]}}
mongo|mongodatabase | {"t": {"$date": "2022-03-28T22:55:24.497+00:00"}, "s": "I", "c": "CONTROL", "id": 51765, "ctx": "initandlisten", "msg": "Operating System", "attr": {"os": {"name": "Ubuntu", "version": "20.04"}}}

```

Ln 1, Col 13 Spaces: 4 UTF-8 LF Ignore ⌂ Prettier ⌂

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

```

JS index.js ×
src > JS indexjs > ...
1 const express = require('express')
2
3 const app = express();
4
5 require('./database');
6
7 app.listen(3000);
8
9 console.log('Server on Port', 3000);
10

```

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 | > nodemongoconection@0.0.1 start /usr/src/app
exampleapp | > node src/index.js "dev"
exampleapp |
exampleapp | Server on Port 3000
mymongodatabase | {"t": {"$date": "2022-03-29T02:12:35.376+00:00"}, "s": "I", "c": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accepted", "attr": {"remote": "172.19.0.3:60714", "uuid": "869a1343-be6b-4200-bda4-1c56a7844c4e", "connectionId": 1, "connectionCount": 1}}
mymongodatabase | {"t": {"$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": {"$date": "2022-03-29T02:12:45.907+00:00"}, "s": "I", "c": "NETWORK", "id": 22943, "ctx": "listener", "msg": "Connection accepted", "attr": {"remote": "172.19.0.3:60716", "uuid": "66b81404-8806-4254-b592-bc64e932c086", "connectionId": 2, "connectionCount": 2}}
mymongodatabase | {"t": {"$date": "2022-03-29T02:12:45.909+00:00"}, "s": "I", "c": "NETWORK", "id": 51800, "ctx": "conn2", "msg": "client me tadata", "attr": {"remote": "172.19.0.3:60716", "client": "conn2", "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]"}}

```

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

The screenshot shows the Visual Studio Code interface with two tabs open: 'database.js - nodeMongoConnection' and 'index.js'. The 'database.js' tab contains code for connecting to MongoDB and logging the connection status. The 'index.js' tab contains the main application logic. The terminal at the bottom shows the application starting and connecting to the database.

```

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js JS database.js Dockerfile docker-compose.yml dockerignore
NODEMONGO... src > JS database.js (then) callback
node_modules
JS New File Alt+Shift+N
JS New Folder Alt+Shift+F
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
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
exampleapp | > nodemongoconection@0.0.1 start /usr/src/app
exampleapp | > node src/index.js
exampleapp | Server on Port 3000
mymongodatabase | {"t": {"$date": "2022-03-28T23:03:10.836+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508590:836193][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 12 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t": {"$date": "2022-03-28T23:04:10.905+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508650:905597][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 14 , snapshot max: 14 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t": {"$date": "2022-03-28T23:05:10.967+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508710:967811][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 16 , snapshot max: 16 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}

```


The screenshot shows the Visual Studio Code interface with two tabs open: 'database.js - nodeMongoConnection' and 'index.js'. The 'database.js' tab contains code for connecting to MongoDB and logging the connection status. The 'index.js' tab contains the main application logic. The terminal at the bottom shows the application starting and connecting to the database.

```

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js JS database.js Dockerfile docker-compose.yml dockerignore
NODEMONGO... src > JS database.js (then) callback
node_modules
src > JS routes database.js index.js Dockerfile docker-compose.yml dockerignore
routes
JS routes
JS database.js
JS index.js
Dockerfile
docker-compose.yml
Dockerfile
package-lock.json
package.json
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
exampleapp | > nodemongoconection@0.0.1 start /usr/src/app
exampleapp | > node src/index.js
exampleapp | Server on Port 3000
mymongodatabase | {"t": {"$date": "2022-03-28T23:03:10.836+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508590:836193][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 12 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t": {"$date": "2022-03-28T23:04:10.905+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508650:905597][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 14 , snapshot max: 14 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t": {"$date": "2022-03-28T23:05:10.967+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpointer", "msg": "WiredTiger message", "attr": {"message": "[1648508710:967811][1:0x7f17a25fd700]", "WT_SESSION.checkpoint": [WT_Verb::CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 16 , snapshot max: 16 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}

```

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

The screenshot shows two instances of Visual Studio Code side-by-side, both displaying the same file: `database.js`. The file contains code for connecting to a MongoDB database and logging the connection status.

```
const mongoose = require('mongoose')
mongoose.connect('mongodb://mongo/mydatabase')
  .then(() => console.log('Db is connected to', db.connection.host))
  .catch(error => console.error(error));
```

The terminal below the code shows multiple log entries from the MongoDB server, indicating checkpoints and snapshot activity. The logs are identical in both instances.

```
examleapp | Server on Port 2000
mongodatabase | {"t":{"$date":"2022-03-28T23:03:10.836+00:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1648508350:836193][1:0x7f17a25fd700], WT SESSION.checkpoint: [WT VERB CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 12 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mongodatabase | {"t":{"$date":"2022-03-28T23:05:10.967+00:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1648508370:967811][1:0x7f17a25fd700], WT SESSION.checkpoint: [WT VERB CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 12 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mongodatabase | {"t":{"$date":"2022-03-28T23:06:11.062+00:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1648508371:62580][1:0x7f17a25fd700], WT SESSION.checkpoint: [WT VERB CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 12 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mongodatabase | {"t":{"$date":"2022-03-28T23:07:11.123+00:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1648508381:123781][1:0x7f17a25fd700], WT SESSION.checkpoint: [WT VERB CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 12 , snapshot max: 20 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
```

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:

```

version: '3'
services:
  web:
    container_name: exampleapp
    restart: always
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5000:3000"
    links:
      - mongo
    volumes:
      - ./usr/src/app
  mongo:
    container_name: mymongodatabase
    image: mongo
    ports:
      - "27018:27017"

```

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

```

, snapshot max: 16 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18")
mymongodatabase | {"t":{"$date":"2022-03-28T23:04:10.905+00:00"},"s":1,"c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1649508650:905597][1:0x7f17a25fd700] WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 14, snapshot max: 14 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t":{"$date":"2022-03-28T23:05:10.967+00:00"},"s":1,"c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1649508710:967811][1:0x7f17a25fd700] WT SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 16, snapshot max: 16 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t":{"$date":"2022-03-28T23:06:11.062+00:00"},"s":1,"c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1649508771:625801[1:0x7f17a25fd700] WT SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 18, snapshot max: 18 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t":{"$date":"2022-03-28T23:07:11.123+00:00"},"s":1,"c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1649508811:202303[1:0x7f17a25fd700] WT SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 20, snapshot max: 20 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}
mymongodatabase | {"t":{"$date":"2022-03-28T23:08:11.202+00:00"},"s":1,"c":"STORAGE", "id":22430, "ctx":"Checkpoint","msg":"WiredTiger message","attr":{"message":"[1649508891:202303[1:0x7f17a25fd700] WT SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 22, snapshot max: 22 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 18"}}

Starting exampleapp ... done
Stopping mymongodatabase ... done
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker-compose up
mongo uses an image, skipping
Building web
Step 1/8 : FROM node:14
--> 903cc2873ea4
Step 2/8 : RUN mkdir -p /usr/src/app
--> c20795e69fb0
Step 3/8 : WORKDIR /usr/src/app
--> Using cache
--> 47b8480d9014
Step 4/8 : COPY package*.json ./
--> Using cache
--> b50778cbfbcb
Step 5/8 : RUN npm install
--> Using cache
--> b073155d0980
Step 6/8 : COPY . .
--> 28fedaa13c6c2
Step 7/8 : EXPOSE 3000
--> Running in ffc0ba547a2c
Removing intermediate container ffc0ba547a2c
--> fc366930e909
Step 0/1 : CMD ["npm", "start" ]
--> Running in c6ee154dccb8
Removing intermediate container c6ee154dccb8
--> 04692cf26da2
Successfully built 04692cf26da2
Successfully tagged nodemongoconnection web:latest
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ 
```

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:

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** .gitignore - nodeMongoConnection - Visual Studio Code
- File Explorer:** Shows a project structure under 'NODEMONGOCONNECTION'. The 'src' folder contains 'routes', 'index.routes.js', 'database.js', and 'Index.js'. The root directory contains '.gitignore', 'docker-compose.yml', 'Dockerfile', 'package-lock.json', and 'package.json'.
- Terminal:** Displays the following log output from the 'mongod' command:

```
,"attr":{"remote":172.19.0.3:35048,"uid":"d878c24f-6635-4d65-800f-27a3eabcc195","connectionId":1,"connectionCount":1}  
mymongodatabase | {"t":("sdate":2022-03-28T23:10:09.181+00:00"),"s":"I","c":"NETWORK","id":51800,"ctx":{"conn1","msg":"client metadata","attr":{"remote":172.19.0.3:35048,"client":"conn1","doc":{"driver":{"name":"nodejsMongoose","version":4.3.1}},"os":{"type":"Linux","name":"Linux","architecture":"x64","version":5.4.105-generic},"platform":{"Node.js v14.19.1, LE (unified)","version":4.3.16.2.9}}}  
exampleapp | DB is connected to mongo  
mymongodatabase | {"t":("sdate":2022-03-28T23:10:19.696+00:00),"s":"I","c":"NETWORK","id":22943,"ctx":{"listener","msg":"Connection accepted","attr":{"remote":172.19.0.3:35048,"client":null,"doc":{"driver":{"name":"nodejsMongoose","version":4.3.1}},"os":{"type":"Linux","name":"Linux","architecture":"x64","version":5.4.105-generic}}}  
mymongodatabase | [{"t":("sdate":2022-03-28T23:10:19.697+00:00),"s":"I","c":"NETWORK","id":51800,"ctx":{"conn2","msg":"client metadata","attr":{"remote":172.19.0.3:35050,"client":"conn2","doc":{"driver":{"name":"nodejsMongoose","version":4.3.1}},"os":{"type":"Linux","name":"Linux","architecture":"x64","version":5.4.105-generic}}}  
mymongodatabase | {"t":("sdate":2022-03-28T23:11:07.186+00:00),"s":"I","c":"STORAGE","id":22430,"ctx":{"checkpoint1,"msg":"WiredTiger message","attr":{"message":"[1648509667:186058][1:6x7f5d4a829700], WT_SESSION_checkpoint:[WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 3, snapshot max: 3 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 51"}}
```
- Status Bar:** L1 Col 1 Spaces: 4 UTF-8 LF Ignore ⌘ Prettier ⌘ D

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

```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e073dc482f6 nodemongoconection_web "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:5000->3000
/tcp exampleapp
436c7df16190 mongo "docker-entrypoint.s..." 17 minutes ago Up 2 minutes 0.0.0.0:27018->27017/tcp, ::1:27018->2
7017/tcp mymongodatabase
e78ddaa750b8 postgres "docker-entrypoint.s..." 13 days ago Up 36 minutes 0.0.0.0:5433->5432/tcp, ::1:5433->5432
/tcp webprojectdos_postgres_1
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker exec -it exampleapp bash
root@e073dc4d482f6:/usr/src/app# ls -a
. .dockerignore .gitignore Dockerfile docker-compose.yml node_modules package-lock.json package.json src
root@e073dc4d482f6:/usr/src/app#

```



```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e073dc482f6 nodemongoconection_web "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:5000->3000
/tcp exampleapp
436c7df16190 mongo "docker-entrypoint.s..." 17 minutes ago Up 2 minutes 0.0.0.0:27018->27017/tcp, ::1:27018->2
7017/tcp mymongodatabase
e78ddaa750b8 postgres "docker-entrypoint.s..." 13 days ago Up 36 minutes 0.0.0.0:5433->5432/tcp, ::1:5433->5432
/tcp webprojectdos_postgres_1
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker exec -it exampleapp bash
root@e073dc4d482f6:/usr/src/app# ls -a
. .dockerignore .gitignore Dockerfile docker-compose.yml node_modules package-lock.json package.json src
root@e073dc4d482f6:/usr/src/app#

```

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

```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e073dc482f6 nodemongoconection_web "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:5000->3000/tcp, :::5000->3000
/tcp exampleapp
436c7df16190 mongo
7017/tcp mymongodatabase
e78ddaa750b8 postgres
/tcp webprojectdos_postgres_1
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker exec -it exampleapp bash
root@e073dc4d482f6:/usr/src/app# ls -a
. .dockerignore .gitignore Dockerfile docker-compose.yml node_modules package-lock.json package.json src
root@e073dc4d482f6:/usr/src/app# 

```

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

```

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e073dc482f6 nodemongoconection_web "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:5000->3000/tcp, :::5000->3000
/tcp exampleapp
436c7df16190 mongo
7017/tcp mymongodatabase
e78ddaa750b8 postgres
/tcp webprojectdos_postgres_1
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker exec -it exampleapp bash
root@e073dc4d482f6:/usr/src/app# ls -a
. .dockerignore .gitignore Dockerfile docker-compose.yml node_modules package-lock.json package.json src
root@e073dc4d482f6:/usr/src/app# 

```

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

```

28 de mar 18:15
gltignore • nodeMongoConection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json JS index.js JS database.js JS index.routes.js Dockerfile docker-compose.yml .gitignore .dockerignore
NODEMONGOCONNECTION
  node_modules
  src
    routes
      index.routes.js
      database.js
      index.js
      dockerignore
      .gitignore
      docker-compose.yml
      Dockerfile
      package-lock.json
      package.json
  .gitignore
  Dockerfile
  package.json

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm i nodemon -D
added 116 packages, and audited 197 packages in 8s
22 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

```

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

```

28 de mar 18:16
package.json - nodeMongoConection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER package.json X JS index.js JS database.js JS index.routes.js Dockerfile docker-compose.yml .gitignore .dockerignore
NODEMONGOCONNECTION
  node_modules
  src
    routes
      index.routes.js
      database.js
      index.js
      dockerignore
      .gitignore
      docker-compose.yml
      Dockerfile
      package-lock.json
      package.json
  package.json

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm i nodemon -D
added 116 packages, and audited 197 packages in 8s
22 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

```

```

{
  "name": "nodemongoconection",
  "version": "0.0.1",
  "description": "proyecto con node y mongo db",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node src/index.js",
    "dev": "nodemon src/index.js"
  },
  "keywords": [
    "mongo",
    "node",
    "docker"
  ],
  "author": "Nicolas Echavarria",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.3",
    "mongoose": "^6.2.9"
  }
}

```

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

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with files like package.json, index.js, database.js, index.routes.js, Dockerfile, docker-compose.yml, .gitignore, and .dockerignore.
- Dockerfile Content:**

```

18 #Directorio actual dentro del contenedor actual
19 COPY .
20
21 #Especificar el puerto dentro del contenedor, en este caso enlazar el puerto 3000 con un puerto de la máquina
22 EXPOSE 3000
23
24 #Iniciar comandos
25 CMD [ "npm", "start", "dev" ]

```
- Terminal Output:**

```

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ npm i nodemon -D
added 116 packages, and audited 197 packages in 8s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker-compose stop
Stopping exampleapp ... done
Stopping mymongodatabase ... done
harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ 

```
- Bottom Status Bar:** ShowsLn 26, Col 1 Spaces: 4 UTF-8 LF Dockerfile Prettier

42. Presionamos control + c para parar la ejecución

43. Ejecutamos docker-compose build para volver a generar la imagen principal

The screenshot shows the Visual Studio Code interface with the following details:

- Terminal Output:**

```

harvey@harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConection$ docker-compose build
mongo uses an image, skipping
Building web
Step 1/8 : FROM node:14
--> 903c2c873ea4
Step 2/8 : RUN mkdir -p /usr/src/app
--> Using cache
--> c29795e69fb
Step 3/8 : WORKDIR /usr/src/app
--> Using cache
--> 47b8480d9014
Step 4/8 : COPY package*.json .
--> 2f9968723f72
Step 5/8 : RUN npm install
--> Running in 11816b675878
npm WARN read-shrinkwrap This version of npm is compatible with lockfileVersion@1, but package-lock.json was generated for lockfileVersion@2. I'll try to do my best with it!
> nodemon@2.0.15 postinstall /usr/src/app/node_modules/nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
> https://opencollective.com/nodemon/donate

npm WARN nodemongoconeccion@0.1 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
added 196 packages from 147 contributors and audited 197 packages in 7.698s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Removing intermediate container 11816b675878
--> d83b92c1a67d
Step 6/8 : COPY .
--> 898e98456785
Step 7/8 : EXPOSE 3000
--> Running in 588bb87b9e766
Removing intermediate container 588bb87b9e766

```
- Bottom Status Bar:** ShowsLn 26, Col 1 Spaces: 4 UTF-8 LF Dockerfile Prettier

44. Ejecutamos docker-compose up

```

28 de mar 20:59
Dockerfile - nodeMongoConnection - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
NODEMONGOCONNECTION
  node_modules
  src
    routes
      index.routes.js
      database.js
      index.js
      dockerignore
      .gitignore
      docker-compose.yml
      package-lock.json
      Dockerfile
      package.json

TERMINAL
2022-03-29T01:59:16.162+00:00] "s": "I", "c": "CONTROL", "id": 23285, "ctx": "-", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.3 specify --sslDisabledProtocols 'none'"}
[2022-03-29T01:59:16.230+00:00] "s": "I", "c": "NETWORK", "id": 4915701, "ctx": "main", "msg": "Initialize d wire specification", "attr": {"spec": ("minWireVersion": 0, "maxWireVersion": 13), "incomingInternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}, "isInternalClient": true}}
[2022-03-29T01:59:16.306+00:00] "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No TransportLayer configured during NetworkInterface startup"
[2022-03-29T01:59:16.306+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."
[2022-03-29T01:59:16.349+00:00] "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfuly registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "ns": "config.tenantMigrationDonors"}
[2022-03-29T01:59:16.349+00:00] "s": "I", "c": "REPL", "id": 5123008, "ctx": "main", "msg": "Successfuly registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "ns": "config.tenantMigrationRecipients"}
[2022-03-29T01:59:16.349+00:00] "s": "I", "c": "CONTROL", "id": 5945603, "ctx": "main", "msg": "Multi threading initialized"
[2022-03-29T01:59:16.350+00:00] "s": "I", "c": "CONTROL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 1, "dbPath": "/data/db", "architecture": "64-bit", "host": "436cd7f16190"}}
[2022-03-29T01:59:16.351+00:00] "s": "I", "c": "CONTROL", "id": 23463, "ctx": "initandlisten", "msg": "Build Info", "attr": {"version": "2022-03-29T01:59:16.351+00:00", "gitVersion": "212a0db47f042d1e1949c7baec1d109259", "opensslVersion": "OpenSSL 1.1.1l 31 Mar 2020", "modules": {}, "allocator": "tcmalloc", "environment": {"distmod": "ubuntu2004", "distarch": "x86_64", "target_arch": "x86_64"}}
[2022-03-29T01:59:16.351+00:00] "s": "I", "c": "CONTROL", "id": 51765, "ctx": "initandlisten", "msg": "Operating System", "attr": {"os": "Ubuntu", "version": "20.04"}}
[2022-03-29T01:59:16.351+00:00] "s": "I", "c": "CONTROL", "id": 21951, "ctx": "initandlisten", "msg": "Options set by command line", "attr": {"options": {"net": ["bindip"]}}
[2022-03-29T01:59:16.402+00:00] "s": "I", "c": "STORAGE", "id": 22270, "ctx": "initandlisten", "msg": "Storage engine to use detected by data files", "attr": {"dpPath": "/data/db", "storageEngine": "wiredTiger"}
[2022-03-29T01:59:16.402+00:00] "s": "I", "c": "STORAGE", "id": 22297, "ctx": "initandlisten", "msg": "Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filestem", "tags": ["startupWarnings"]}

```

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

```

src > routes > index.routes.js ...
File Edit Selection View Go Run Terminal Help
Index.routes.js - nodeMongoConnection - Visual Studio Code
src > routes > index.routes.js ...
  1 const { Router } = require('express');
  2 const router = Router();
  3
  4 router.get('/', (req, res) => {
  5   res.send('hello word')
  6 });
  7
  8 module.exports = router;

```

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
": {"minWireVersion": 0, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 0, "maxWireVersion": 13}, "isInternalClient": true}, "newSpec": {"incomingExternalClient": {"minWireVersion": 0, "maxWireVersion": 13}, "incomingInternalClient": {"minWireVersion": 13, "maxWireVersion": 13}, "outgoing": {"minWireVersion": 13, "maxWireVersion": 13}, "isInternalClient": true}}
[2022-03-29T01:59:19.356+00:00] "s": "I", "c": "STORAGE", "id": 5071100, "ctx": "initandlisten", "msg": "Clearing temp directory"
[2022-03-29T01:59:19.356+00:00] "s": "I", "c": "CONTROL", "id": 20536, "ctx": "initandlisten", "msg": "File Configuration is enabled on this deployment"
[2022-03-29T01:59:19.449+00:00] "s": "I", "c": "FTDC", "id": 20625, "ctx": "initandlisten", "msg": "Initializing full-time diagnostic data capture", "attr": {"dataDirectory": "/data/db/diagnostic.data"}}
[2022-03-29T01:59:19.549+00:00] "s": "I", "c": "REPL", "id": 6015317, "ctx": "initandlisten", "msg": "Setting new configuration state", "attr": {"newState": "ConfigReplicationDisabled", "oldState": "ConfigPreStart"}
[2022-03-29T01:59:19.552+00:00] "s": "I", "c": "NETWORK", "id": 23015, "ctx": "listener", "msg": "Listening on", "attr": {"address": "/tmp/mongodb-27017.sock"}
[2022-03-29T01:59:19.552+00:00] "s": "I", "c": "NETWORK", "id": 23015, "ctx": "listener", "msg": "Listened on port 27017"

```

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

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure for 'NODEMONGOCONNECTION' with files like package.json, index.js, database.js, index.routes.js, Dockerfile, docker-compose.yml, .gitignore, .dockerignore, Dockerfile, package-lock.json, and package.json. The code editor window shows the 'index.js' file with the following content:

```

const express = require('express')
const app = express()
require('./database')
app.use(require('./routes/index.routes'))
app.listen(3000)
console.log('Server on Port', 3000)

```

The terminal window on the right shows the command 'nodeMongoConnection' running and outputting logs related to MongoDB connection and initialization. The status bar at the bottom indicates the file is 'index.js - nodeMongoConnection - Visual Studio Code'.

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

The screenshot shows the Visual Studio Code interface again. The terminal window on the right shows the command 'harvey@harvey-HP-Pavilion-15-cw0005la:~/Escritorio/nodeMongoConnection\$ docker-compose build' being run. The output shows the build process for the 'nodeMongoConnection' service, which includes creating a Docker image, building the image, and running the container. The status bar at the bottom indicates the file is 'index.js - nodeMongoConnection - Visual Studio Code'.

The screenshot displays a dual-monitor setup. The left monitor shows a terminal window in Visual Studio Code with the title "index.js - nodeMongoConnection - Visual Studio Code". It contains the following command-line session:

```
28 de mar 21:13
File Edit Selection View Go Run Terminal Help
src > JS index.js x JS database.js x JS index.routes.js x Dockerfile x docker-compose.yml x .gitignore x .dockerngore
EXPLORER
NODEMONGOCONNECTION
  > node_modules
    > src
      > routes
        JS index.routes.js
        JS database.js
      JS index.js
    Dockerfile
    .dockerngore
    .gitignore
    docker-compose.yml
    Dockerfile
    package-lock.json
  package.json

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Successfully tagged nodemongoconection_web:latest
harvey@Harvey-HP-Pavilion-Laptop-15-cw0005la:~/Escritorio/nodeMongoConnection$ docker-compose up
Starting mongoDatabase ... done
Recreating exampleapp ... done
Attaching to mongoDatabase, exampleapp
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.745+00:00"}, "s": "I", "c": "CONTROL", "id": 23285, "ctx": "-", "msg": "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongoDatabase | {"t": {"$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}, "isInternalClient": true}}}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.751+00:00"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No Transpo rtLayer configured during NetworkInterface startup"}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.751+00:00"}, "s": "I", "c": "NETWORK", "id": 4048601, "ctx": "main", "msg": "Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.755+00:00"}, "s": "W", "c": "ASIO", "id": 22601, "ctx": "main", "msg": "No Transpo rtLayer configured during NetworkInterface startup"}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.756+00:00"}, "s": "I", "c": "REPL", "id": 5123068, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationDonorService", "ns": "config.tenantMigrationDonors"}}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.756+00:00"}, "s": "I", "c": "REPL", "id": 5123068, "ctx": "main", "msg": "Successfully registered PrimaryOnlyService", "attr": {"service": "TenantMigrationRecipientService", "ns": "config.tenantMigrationRecipients"}}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.756+00:00"}, "s": "I", "c": "CONTROL", "id": 5945663, "ctx": "main", "msg": "Multi threading initialized"}
mongoDatabase | {"t": {"$date": "2022-03-29T02:12:31.756+00:00"}, "s": "I", "c": "REPL", "id": 4615611, "ctx": "initandlisten", "msg": "MongoDB starting", "attr": {"pid": 1, "port": 27017, "dbPath": "/data/db", "architecture": "64-bit", "host": "436c7df1699"}}
```

The right monitor shows a Firefox browser window with the URL "localhost:5000/" loaded, displaying the text "hello word".