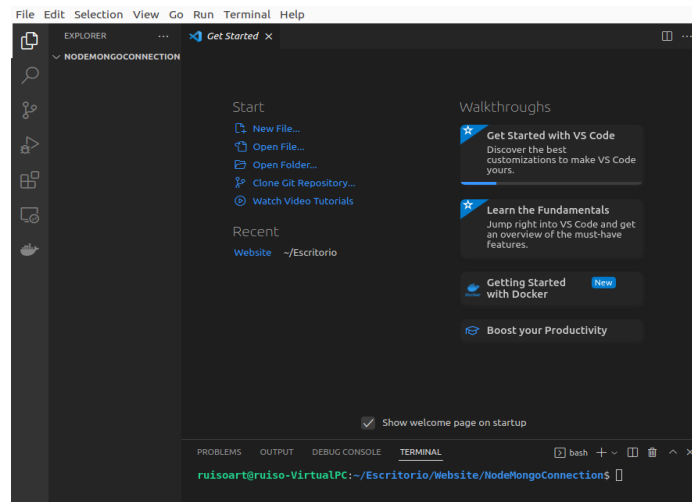
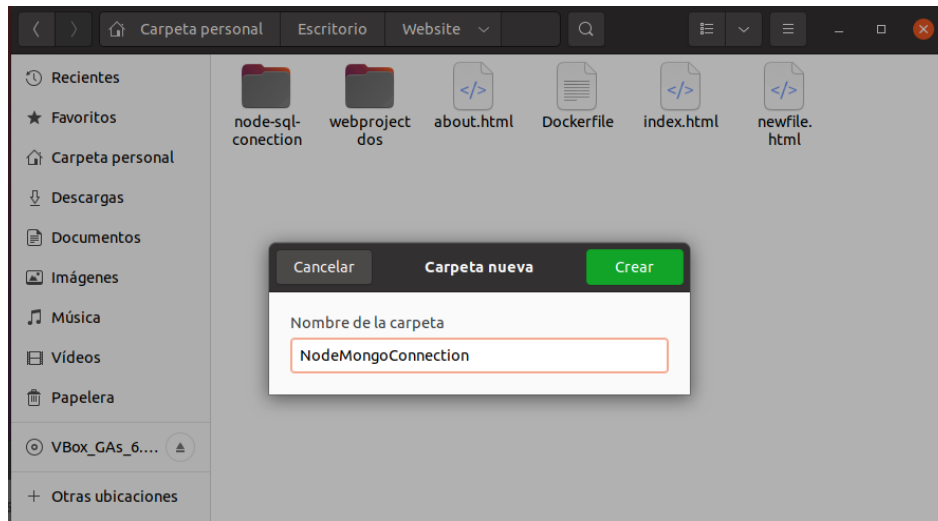


Taller 7: Node Js, Mongo DB y Docker Compose

Luis Felipe Narvaez Gomez. E-mail: luis.narvaez@usantoto.edu.co. Cod:2312660. Facultad de Ingenieria de Sistemas. 2022-1.

Primero hemos de crear una nueva carpeta para esta actividad. Seguidamente abrimos la misma con un editor de codigo como lo es Visual Studio Code y en la mismo abrimos una terminal con la ubicación dada en la misma carpeta.



Primero crearemos un proyecto con NodeJs, para esto en la terminal que tenemos abierta en Visual Studio Code ejecutaremos el comando “npm init”, este inicializara el proyecto creado de paso el archivo “package.json”, el cual contiene los metadatos especificos del proyecto como el Nombre del paquete, la version, la descripcion, los puntos de entrada, comandos de prueba, repositorio git, palabras clave, su autor, y la licencia que tenemos.

```

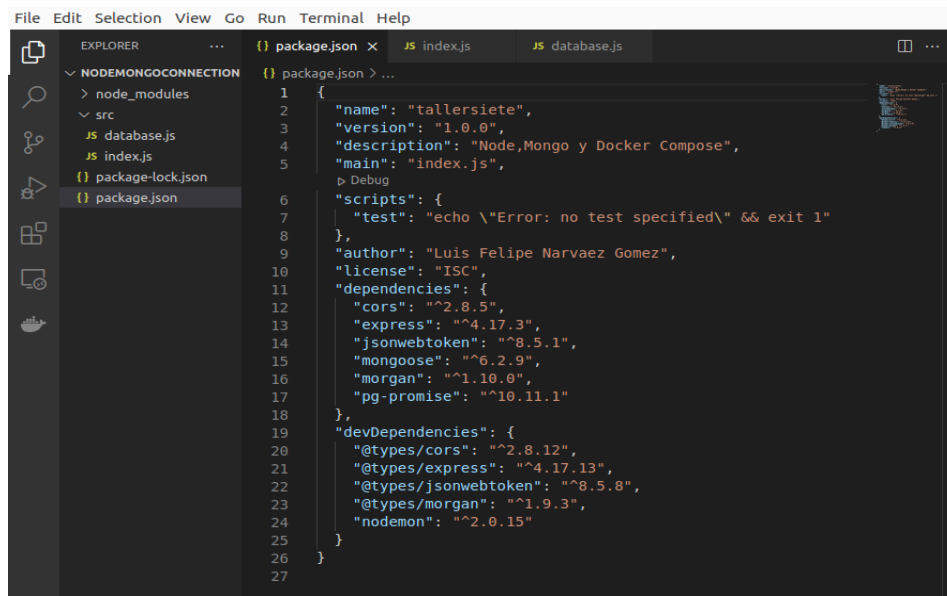
{} package.json > ...
1  {
2    "name": "tallersiete",
3    "version": "1.0.0",
4    "description": "Node,Mongo y Docker Compose",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Luis Felipe Narvaez Gomez",
10   "license": "ISC"
11 }
12

```

Instalaremos cada uno de los diferentes paquetes que podemos llegar a utilizar, en especial paquete de express y mongo db

- npm i cors --> paquete, sistema de bloqueo para permitir ciertos usuarios
- npm i morgan --> paquete, formatea la salida de la terminal, organiza mejor las salidas , entre otras cosas.
- npm i express --> paquetete, maneja el motor V8 para hacer backend
- npm i express mongoose → modulo que sirve como modelo de conexión para poder acceder a mongo
- npm i jsonwebtoken --> paquete, generador de token de autenticacion para seguridad
- npm i pg-promise --> paquete, permite conectar con postgresSQL
- En caso de querer desinstalar un paquete --> npm uninstall "nombrePaquete"
- types --> facilitan las tareas de cada paquete
- npm i @types/cors --save-dev --> types de cors -- en modo desarrollo para que no esten en la produccion
- npm i @types/morgan --save-dev --> types de morgan -- en modo desarrollo para que no esten en la produccion
- npm i @types/express --save-dev --> types de express -- en modo desarrollo para que no esten en la produccion
- npm i @types/jsonwebtoken --save-dev --> types de jsonwebtoken -- en modo desarrollo para que no esten en la produccion
- npm i nodemon --save-dev --> permite montar la aplicacion de manera local

Para instalar cada una de las dependencias, escribiremos y eecutaremos el comando en la terminal de nuestro editor de codigo, esto hara que los mismos se nombren dentro del archivo “package.json” para una proxima reinstalacion con el comando “npm i”, creara el archivo “package-lock.json” y la carpeta “node_modules”.



```
1 {
2   "name": "tallersiete",
3   "version": "1.0.0",
4   "description": "Node,Mongo y Docker Compose",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "Luis Felipe Narvaez Gomez",
10  "license": "ISC",
11  "dependencies": {
12    "cors": "^2.8.5",
13    "express": "^4.17.3",
14    "jsonwebtoken": "^8.5.1",
15    "mongoose": "^6.2.9",
16    "morgan": "^1.10.0",
17    "pg-promise": "^10.11.1"
18  },
19  "devDependencies": {
20    "@types/cors": "^2.8.12",
21    "@types/express": "^4.17.13",
22    "@types/jsonwebtoken": "^8.5.8",
23    "@types/morgan": "^1.9.3",
24    "nodemon": "^2.0.15"
25  }
26 }
```

Ahora dentro de una carpeta “src” podemos tener dos codigos para acceder a una base de datos y montar un servidor, tales como “index.js” y “database.js”.

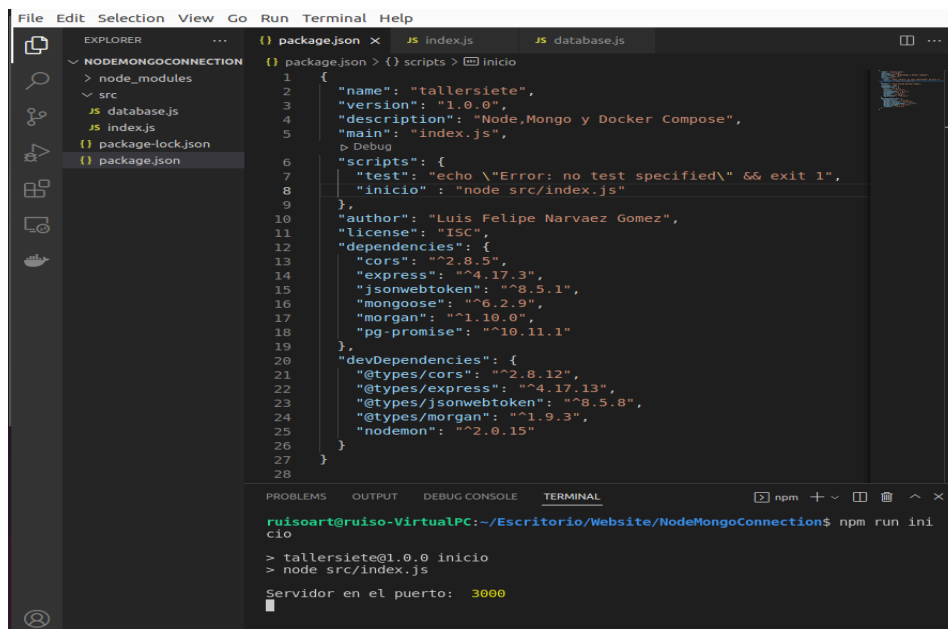


```
1 const express = require('express')
2
3 const app = express();
4
5 app.listen(3000);
6 console.log("Servidor en el puerto: ", 3000);
```

Para ejecutar el servidor dado en el codigo anterior del “index.js” creamos un script en el “package.json” en la seccion de de “scripts”, aquí podemos crear diferentes comandos unicos para nuestro servidor, en este caso uno que inicialice nuestro servicio el cual sera **“inicio” : “node src/index.js”**

Debemos tener en cuenta que los scripts podemos denominarlos como queremos, en este caso he optado por el nombre de “inicio” palabra que utilizare en la consola de comandos para poder ejecutar la tarea que le encomende en el script.

Para ejecutar el comando basta con escribir en la terminal “npm run SCRIPT” en este caso “npm run inicio”; tambien puede llegar a funcionar simplemente “npm SCRIPT”. Al hacer esto ejecutara la tarea “node src/index.js”. Este metodo es muy utilizado cuando queremos simplificar comando muy largos sobre nuestro servicio.

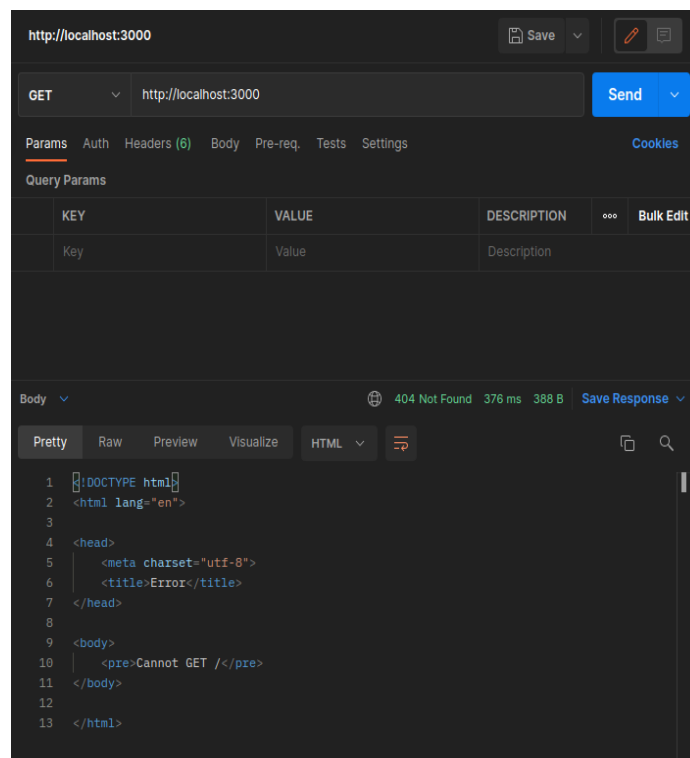


The image shows a Visual Studio Code editor with a project named 'NODEMONGOCONNECTION'. The Explorer sidebar on the left shows the file structure: 'node_modules', 'src' (containing 'database.js', 'index.js', 'package-lock.json', and 'package.json'), and 'package.json'. The main editor displays the content of 'package.json', which includes fields for name, version, description, main, scripts, author, license, dependencies, and devDependencies. The 'scripts' section defines 'test' and 'inicio' commands. The terminal at the bottom shows the execution of 'npm run inicio', which runs 'node src/index.js' and outputs 'Servidor en el puerto: 3000'.


```
1 {
2   "name": "tallersiete",
3   "version": "1.0.0",
4   "description": "Node,Mongo y Docker Compose",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "inicio": "node src/index.js"
9   },
10  "author": "Luis Felipe Narvaez Gomez",
11  "license": "ISC",
12  "dependencies": {
13    "cors": "^2.8.5",
14    "express": "^4.17.3",
15    "jsonwebtoken": "^8.5.1",
16    "mongoose": "^6.2.9",
17    "morgan": "^1.10.0",
18    "pg-promise": "^10.11.1"
19  },
20  "devDependencies": {
21    "@types/cors": "^2.8.12",
22    "@types/express": "^4.17.13",
23    "@types/jsonwebtoken": "^8.5.8",
24    "@types/morgan": "^1.9.3",
25    "nodemon": "^2.0.15"
26  }
27 }
28
```

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ npm run inicio
> tallersiete@1.0.0 inicio
> node src/index.js
Servidor en el puerto: 3000
```


El servicio ya esta corriendo en nuestro espacio local de red, podemos comprobarlo con postman o bien con el navegador.




Para el desarrollo de aplicaciones y servicios podemos utilizar algunas extensiones que nos faciliten el trabajo de escritura de nuestro codigo, tanto en el autocompletado, navegacion de carpetas, syntaxis o simplemente mejor navegacion visual por el arbol de carpetas y archivos. Algunas de estas extensiones recomendadas son:




vscode-icons v11.10.0
VSCode Icons Team | 10,394,494 | ★★★★★
Icons For Visual Studio Code
[Set File Icon Theme](#) [Disable](#) [Uninstall](#) ⚙️




Tabnine AI Autocomplete for
TabNine | 3,411,903 | ★★★★★ (391)
JavaScript, Python, Java, Typescript & all other...
[Disable](#) [Uninstall](#) ⚙️




Prettier - Code formatter v9.1.0
Prettier | 19,852,529 | ★★★★★ (320)
Code formatter using prettier
[Disable](#) [Uninstall](#) ⚙️



Material Icon Theme v4.15.0
Philipp Kief | 11,618,821 | ★★★★★ (245)
Material Design Icons for Visual Studio Code
[Set File Icon Theme](#) [Disable](#) [Uninstall](#) ⚙️

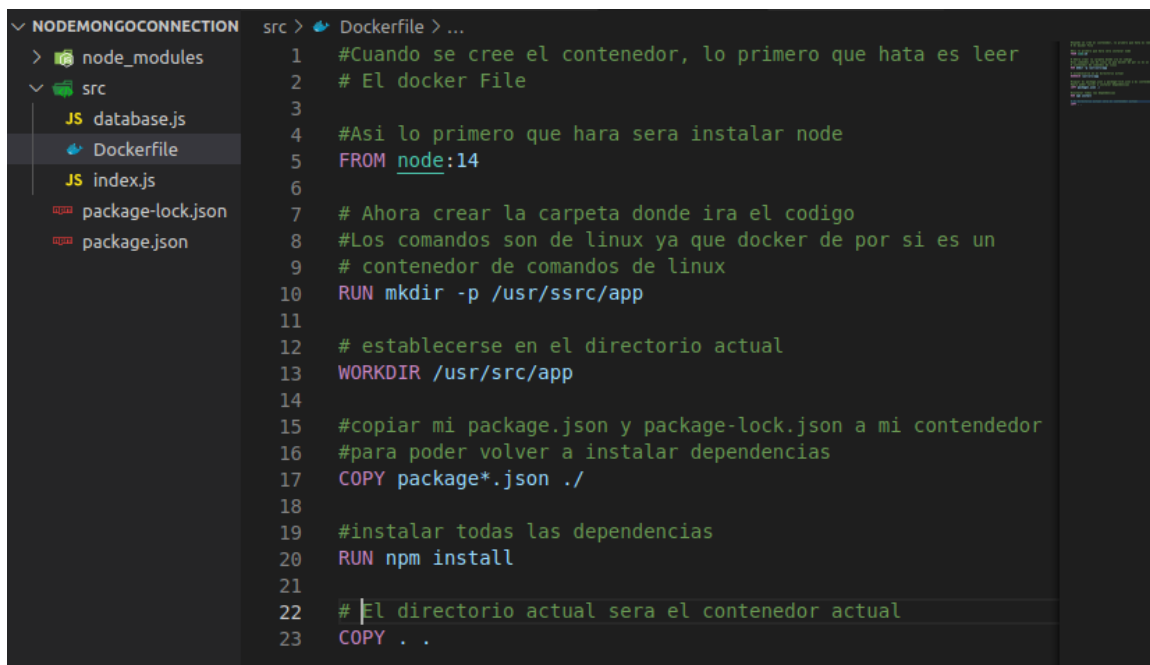


Auto Import v1.5.4
steoates | 2,100,940 | ★★★★★ (106)
Automatically finds, parses and provides code action...
[Install](#) ⚙️



Docker v1.21.0
Microsoft | 14,035,231 | ★★★★★ (68)
Makes it easy to create, manage, and debug containers...
[Disable](#) [Uninstall](#) ⚙️
This extension is enabled globally.

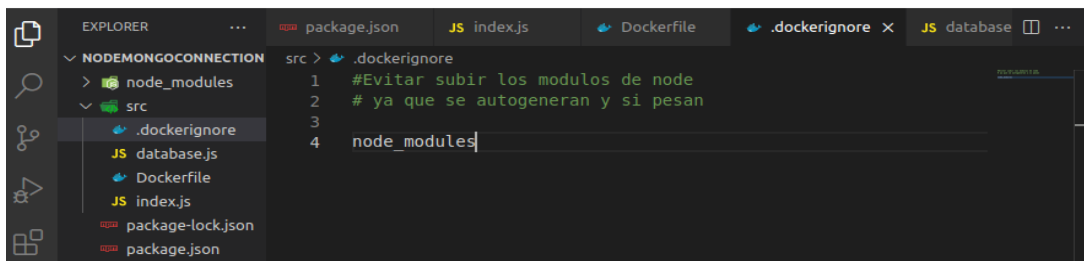
Ahora crearemos el archivo Dockerfile dentro de la carpeta raíz del proyecto, en el se especificara que necesita el proyecto para poder funcionar, asi como la version que utilizaremos de Docker.



```
src > Dockerfile > ...
1 #Cuando se cree el contenedor, lo primero que hata es leer
2 # El docker File
3
4 #Asi lo primero que hara sera instalar node
5 FROM node:14
6
7 # Ahora crear la carpeta donde ira el codigo
8 #Los comandos son de linux ya que docker de por si es un
9 # contenedor de comandos de linux
10 RUN mkdir -p /usr/ssrc/app
11
12 # establecerse en el directorio actual
13 WORKDIR /usr/src/app
14
15 #copiar mi package.json y package-lock.json a mi contenedor
16 #para poder volver a instalar dependencias
17 COPY package*.json ./
18
19 #instalar todas las dependencias
20 RUN npm install
21
22 # El directorio actual sera el contenedor actual
23 COPY . .
```

Ahora bien, tenemos un pequeño problema y es que abra proyectos que utilicen una gran cantidad de modulos de node, por lo que la carptea node_modules, sera cada vez mas pesada como para mover a un repositorio hub, la mejor solucion, no subirla. Esto podemos hacerlo con confianza ya que la carptea que contiene estos modulos de por si es auto generable una vez instalamos esas ependencias con “npm install”.

Para poder evitar que se copieesta carpeta de node_modules o cualquier otro archivo que no queremos que se copie en el hub, crearemos un archivo que se denomine “.dockerignore” (parecito a git ignore en los hub que utilizan git) dentro de la misma direccion raoz del proyecto y lo configuraremos de la siguiente manera:



```
src > .dockerignore
1 #Evitar subir los modulos de node
2 # ya que se autogeneran y si pesan
3
4 node_modules
```

Volvemos ahora a modificar el archivo Dockerfile:

```

1 # El docker File
2
3
4 #Asi lo primero que hara sera instalar node
5 FROM node:14
6
7 # Ahora crear la carpeta donde ira el codigo
8 #Los comandos son de linux ya que docker de por si es un
9 # contenedor de comandos de linux
10 RUN mkdir -p /usr/src/app
11
12 # establecerse en el directorio actual
13 WORKDIR /usr/src/app
14
15 #copiar mi package.json y package-lock.json a mi contenedor
16 #para poder volver a instalar dependencias
17 COPY package*.json ./
18
19 #instalar todas las dependencias
20 RUN npm install
21
22 # El directorio actual sera el contenedor actual
23 COPY . .
24
25 #Se especifica el puerto del contenedor
26 EXPOSE 3000
27
28 #Iniciar los siguientes comandos
29 CMD ["npm", "inicio"]

```

Ahora descargaremos la imagen de node para ejecutar cada uno de los comandos del Dockerfile, para esto utilizamos en la terminal “docker build -t hellonode .”.

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ ls
Dockerfile  node_modules  package.json  package-lock.json  src
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker build -t hellonode .
[sudo] contraseña para ruisoart:
Sending build context to Docker daemon 175.6kB
Step 1/8 : FROM node:14
14: Pulling from library/node
0030cc4ce25c: Pull complete
7ab54d469df6: Pull complete
0c84a1692804: Pull complete
628acdaf8503: Pull complete
cd55abb6ddd3: Pull complete
561384047eed: Pull complete
a8f490a694c4: Pull complete
24154f1a2ff9: Pull complete
f3a6a2d4cf0d: Pull complete
Digest: sha256:c59aa43be5d88cfe9d5e6dd596c380c9fff3eace49c897d4682e66025c53e2c8
Status: Downloaded newer image for node:14
--> 7d03025aad90
Step 2/8 : RUN mkdir -p /usr/src/app
--> Running in ed08d064a67e
Removing intermediate container ed08d064a67e
--> a291ad43721e
Step 3/8 : WORKDIR /usr/src/app
--> Running in 3c9edca8d547
Removing intermediate container 3c9edca8d547
--> fd405acc844a
Step 4/8 : COPY package*.json ./
--> 2443195f210b
Step 5/8 : RUN npm install
--> Running in a93bb5947ff9
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!

> node-mon@2.0.15 postinstall /usr/src/app/node_modules/node-mon
> node bin/postinstall || exit 0

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

npm WARN tallerseite@1.0.0 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 248 packages from 208 contributors and audited 249 packages in 29.787s

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

found 0 vulnerabilities

Removing intermediate container a93bb5947ff9
--> 5f4f3212ae4a
Step 6/8 : COPY . .
--> 6210e1a4fd6e
Step 7/8 : EXPOSE 3000
--> Running in 504b5bffcfb5
Removing intermediate container 504b5bffcfb5
--> 92a9ebf1db9b
Step 8/8 : CMD ["npm", "inicio"]
--> Running in c150aficba95
Removing intermediate container c150aficba95
--> ca3bb24f3fd6
Successfully built ca3bb24f3fd6
Successfully tagged hellonode:latest
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$

```

Una vez hemos hecho lo anterior abremos creado dos imagenes, una llamada “node” y otra llamada “hellonode”. Esto podemos contatarlo con el comando “sudo docker images”

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
hellonode            latest      ca3bb24f3fd6  2 minutes ago  969MB
node                 14         7d03025aad90  15 hours ago   946MB
postgres             latest      5cd1494671e9  2 weeks ago    376MB
dockeronruiso/secondimageruiso  latest      936463c49201  2 weeks ago    142MB
secondimageruiso     latest      936463c49201  2 weeks ago    142MB
wordpress            latest      44a0bf559e55  2 weeks ago    605MB
httpd                 latest      6b8e87fff107  2 weeks ago    144MB
dpage/pgadmin4       latest      4b5bbddb3624  2 weeks ago    340MB
mariadb              latest      f5dd1ac0b00e  2 weeks ago    414MB
mysql                latest      826efd84393b  3 weeks ago    521MB
python               latest      178dcaa62b39  3 weeks ago    917MB
ubuntu               latest      2b4cba85892a  3 weeks ago    72.8MB
nginx                 latest      c919045c4c2b  4 weeks ago    142MB
mysql/mysql-server   latest      434c35b82b08  2 months ago   417MB
alpine               latest      c059bfaa849c  4 months ago    5.59MB
hello-world          latest      feb5d9fea6a5  6 months ago    13.3kB
mariadb/server       10.4       6bc393df66bb  10 months ago  353MB
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$
```

Nota: En esta parte del procedimiento del taller, la imagen que se estaba creando se pifio, por tal motivo creo una segunda imagen que contiene unos ligeros cambios en el package y el dockerfile.

```
package.json > {} scripts > start
1  {
2    "name": "tallersiete",
3    "version": "1.0.0",
4    "description": "Node,Mongo y Docker Compose",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "node src/index.js"
9    },
10   "author": "Luis Felipe Narvaez Gomez",
11   "license": "ISC",
12   "dependencies": {
13     "cors": "^2.8.5",
14     "express": "^4.17.3",
15     "jsonwebtoken": "^8.5.1",
16     "mongoose": "^6.2.9",
17     "morgan": "^1.10.0",
18     "pg-promise": "^10.11.1"
19   },
20   "devDependencies": {
21     "@types/cors": "^2.8.12",
22     "@types/express": "^4.17.13",
23     "@types/jsonwebtoken": "^8.5.8",
24     "@types/morgan": "^1.9.3",
25     "nodemon": "^2.0.15"
26   }
27 }
28
```



```

Dockerfile > ...
1  #Cuando se cree el contenedor, lo primero que hata es leer
2  # El docker File
3
4  #Asi lo primero que hara sera instalar node
5  FROM node:14
6
7  # Ahora crear la carpeta donde ira el codigo
8  #Los comandos son de linux ya que docker de por si es un
9  # contenedor de comandos de linux
10 RUN mkdir -p /usr/src/app
11
12 # establecerse en el directorio actual
13 WORKDIR /usr/src/app
14
15 #copiar mi package.json y package-lock.json a mi contenedor
16 #para poder volver a instalar dependencias
17 COPY package*.json ./
18
19 #instalar todas las dependencias
20 RUN npm i -g
21
22 # El directorio actual sera el contenedor actual
23 COPY . .
24
25 #Se especifica el puerto del contenedor
26 EXPOSE 3000
27
28 #Iniciar los iguientes comandos
29 CMD ["npm", "start"]

```

```

ruisoart@ruiso-VirtualPC:~$ sudo docker images
[sudo] contraseña para ruisoart:
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
hellonode2          latest      5ab6be52ea51     8 minutes ago    971MB
hellonode           latest      16dd38eaa5ad     12 minutes ago   971MB
node                14         7d03025aad90     19 hours ago     946MB
postgres            latest      5cd1494671e9     2 weeks ago      376MB
dockeronruiso/secondimageruiso latest      936463c49201     2 weeks ago      142MB
secondimageruiso    latest      936463c49201     2 weeks ago      142MB
wordpress           latest      44a0bf559e55     2 weeks ago      605MB
httpd               latest      6b8e87fff107     2 weeks ago      144MB
dpage/pgadmin4      latest      4b5bbddb3624     2 weeks ago      340MB
mariadb             latest      f5dd1ac0b00e     2 weeks ago      414MB
mysql               latest      826efd84393b     3 weeks ago      521MB
python              latest      178dcaa62b39     3 weeks ago      917MB
ubuntu              latest      2b4cba85892a     4 weeks ago      72.8MB
nginx               latest      c919045c4c2b     4 weeks ago      142MB
mysql/mysql-server  latest      434c35b82b08     2 months ago     417MB
alpine              latest      c059bfaa849c     4 months ago     5.59MB
hello-world         latest      feb5d9fea6a5     6 months ago     13.3kB
mariadb/server      10.4       6bc393df66bb     10 months ago    353MB
ruisoart@ruiso-VirtualPC:~$

```

Ejecutaremos la imagen que acabamos de crear, con el puerto 3000 pues este el puerto en el que hemos configurado nuestro servidor y a su vez en el puerto 400 el cual sera el puerto encargado de escuchar peticiones.

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ npm start

> tallersiete@1.0.0 start
> node src/index.js

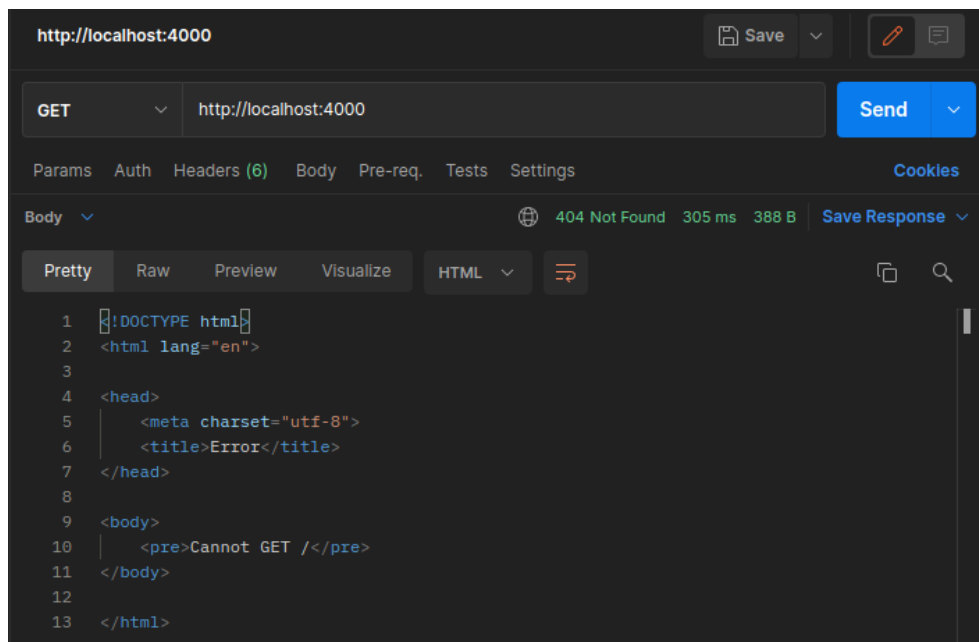
Servidor en el puerto: 3000
^C
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker
build -t hellonode2 .
Sending build context to Docker daemon 175.6kB
Step 1/8 : FROM node:14
--> 7d03025aad90
Step 2/8 : RUN mkdir -p /usr/src/app
--> Using cache
--> 11b7d014a68a
Step 3/8 : WORKDIR /usr/src/app
--> Using cache
--> 31c4abca387b
Step 4/8 : COPY package*.json ./
--> af0399a35b70
Step 5/8 : RUN npm i -g
--> Running in 0f982fa8da7c
+ tallersiete@1.0.0
added 119 packages from 119 contributors in 49.597s
Removing intermediate container 0f982fa8da7c
--> e4e156bc2734
Step 6/8 : COPY . .
--> 5aa87049db3d
Step 7/8 : EXPOSE 3000
--> Running in 853bee9b8df0
Removing intermediate container 853bee9b8df0
--> e903df22a043
Step 8/8 : CMD ["npm", "start"]
--> Running in a89980f76732
Removing intermediate container a89980f76732
--> 5ab6be52ea51
Successfully built 5ab6be52ea51
Successfully tagged hellonode2:latest
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker
run -p 4000:3000 hellonode2

> tallersiete@1.0.0 start /usr/src/app
> node src/index.js

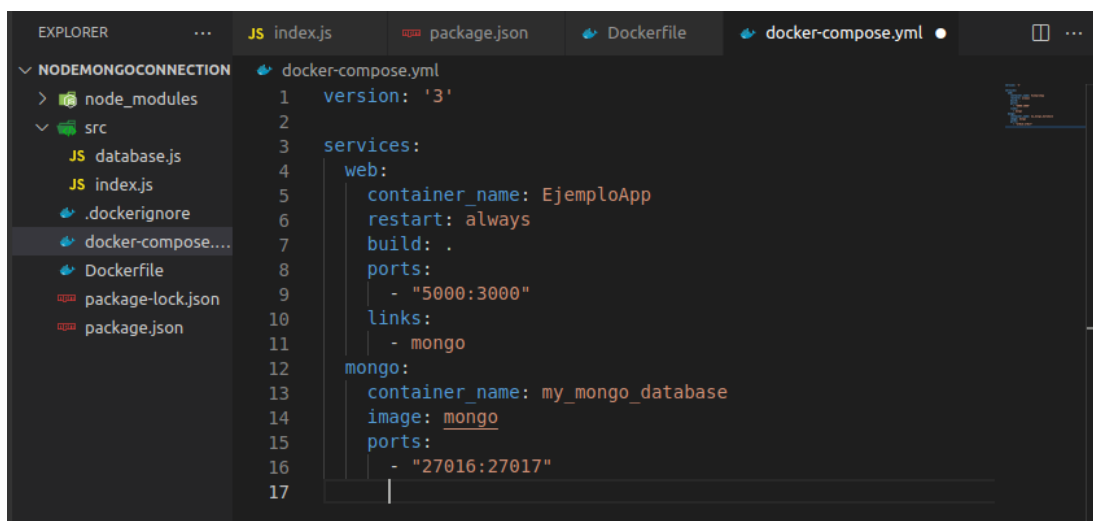
Servidor en el puerto: 3000
□

```

Comprovamos nuevamente que este funcionando con el navegador o bien con Postman.



Paramos el servicio con CTRL+C y creamos el archivo que compondra las distintas imagenes que necesitaremos, este arhivo es docker-compose.yml y estara en la raiz del proyecto con las siguientes instrucciones.



Contruimos el docker-compose con el comando “sudo docker-compose build”.

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

```
src > JS index.js > ...
1  const express = require('express');
2
3  const app = express();
4
5  require('./database');
6
7  app.listen(3000);
8  console.log("Servidor en el puerto: ", 3000);
```

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose build
[sudo] contraseña para ruisoart:
mongo uses an image, skipping
Building web
Step 1/8 : FROM node:14
----> 7d03025aad90
Step 2/8 : RUN mkdir -p /usr/src/app
----> Using cache
----> 11b7d014a68a
Step 3/8 : WORKDIR /usr/src/app
----> Using cache
----> 31c4abca387b
Step 4/8 : COPY package*.json ./
----> 9e7b530a3a25
Step 5/8 : RUN npm i -g
----> Running in 111f9e0f2122
+ tallersiete@1.0.0
added 119 packages from 119 contributors in 47.325s
Removing intermediate container 111f9e0f2122
----> 6f54df873d3c
Step 6/8 : COPY . .
----> a6f65ead500e
Step 7/8 : EXPOSE 3000
----> Running in b88f8fcc98d5
Removing intermediate container b88f8fcc98d5
----> 01a2bff3b354
Step 8/8 : CMD ["npm", "start"]
----> Running in 7b020dc74c0b
Removing intermediate container 7b020dc74c0b
----> 88de402c90fc

Successfully built 88de402c90fc
Successfully tagged nodemongoconnection_web:latest
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$
```

```
ruisoart@ruiso-VirtualPC:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5331a23cc22c	nodemongoconnection_web	"docker-entrypoint.s..."	3 days ago	Up 21 minutes	0.0.0.0:5000->3000/tcp, :::5000->3000/tcp	EjemploApp
be4ac193a191	mongo	"docker-entrypoint.s..."	3 days ago	Up About a minute	0.0.0.0:27016->27017/tcp, :::27016->27017/tcp	my_mongo_database
019100a602d5	hellonode2	"docker-entrypoint.s..."	3 days ago	Exited (0) 3 days ago		busy_heyrovsky
79c61b35961f	hellonode	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		keen_dhawan
1354ca1bdf2a	e7e14b70879f	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		priceless_engelbart
66a0036716be	82d93df916f1	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		mystifying_johnson
774cd34dbc9a	82d93df916f1	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		condescending_austin
d03c9b75d032	82d93df916f1	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		competent_hertz
2f34e1dcfe94	451f54d515af	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		cranky_heisenberg
aea6270c2eab	ca3bb24f3fd6	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		clever_swanson
8fc8a92adb14	ca3bb24f3fd6	"docker-entrypoint.s..."	3 days ago	Exited (126) 3 days ago		reverent_bohr
23a20dba8ced	ca3bb24f3fd6	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		peaceful_wright
c85a8ad6d637	ca3bb24f3fd6	"docker-entrypoint.s..."	3 days ago	Exited (1) 3 days ago		strange_hawking

```
ruisoart@ruiso-VirtualPC:~$ sudo docker start 019
019
ruisoart@ruiso-VirtualPC:~$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5331a23cc22c	nodemongoconnection_web	"docker-entrypoint.s..."	3 days ago	Up 23 minutes	0.0.0.0:5000->3000/tcp, :::5000->3000/tcp	EjemploApp
be4ac193a191	mongo	"docker-entrypoint.s..."	3 days ago	Up 3 minutes	0.0.0.0:27016->27017/tcp, :::27016->27017/tcp	my_mongo_database
019100a602d5	hellonode2	"docker-entrypoint.s..."	3 days ago	Up 53 seconds	0.0.0.0:4000->3000/tcp, :::4000->3000/tcp	busy_heyrovsky

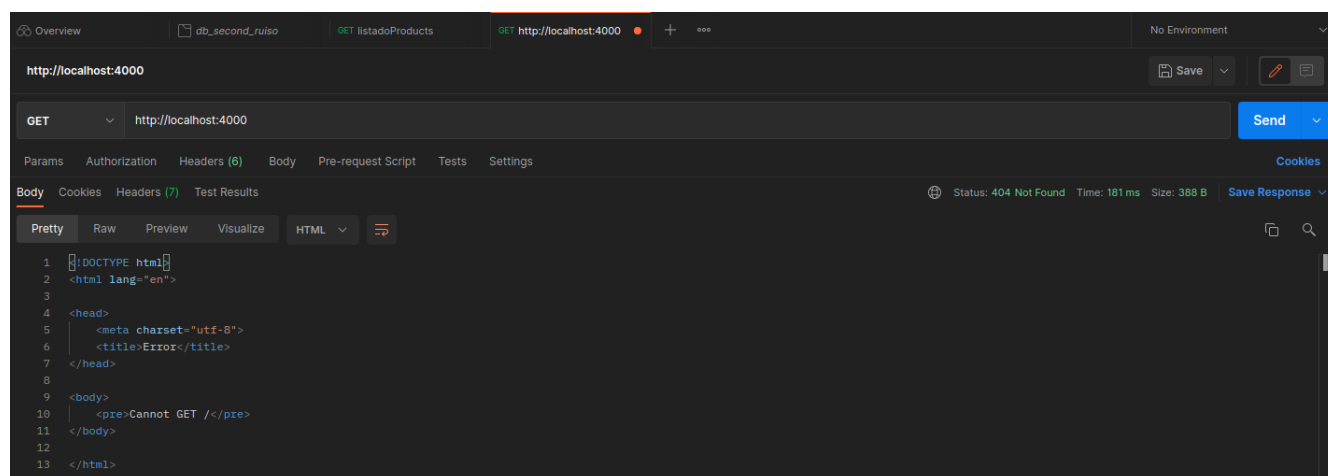
```
ruisoart@ruiso-VirtualPC:~$
```

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose up
Starting my_mongo_database ... done
Starting EjemploApp ... done
Attaching to my_mongo_database, EjemploApp
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:01.405+00:00"},"s":"I", "c":"CONTROL", "id":23
285, "ctx":"-", "msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabl
edProtocols 'none'"}
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:01.429+00:00"},"s":"I", "c":"NETWORK", "id":49
15701, "ctx":"-", "msg":"Initialized wire specification", "attr":{"spec":{"incomingExternalClient":{"m
inWireVersion":0, "maxWireVersion":13}, "incomingInternalClient":{"minWireVersion":0, "maxWireVersion":
13}, "outgoing":{"minWireVersion":0, "maxWireVersion":13}, "isInternalClient":true}}}
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:01.435+00:00"},"s":"W", "c":"ASIO", "id":22
583, "ctx":"-", "msg":"WiredTiger message", "attr":{"message":"[1648758783:468634][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 1 through 2"}}
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:03.468+00:00"},"s":"I", "c":"STORAGE", "id":22
430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr":{"message":"[1648758783:468634][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 2 through 2"}}
EjemploApp |
EjemploApp | > tallersiete@1.0.0 start /usr/src/app
EjemploApp | > node src/index.js
EjemploApp |
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:04.276+00:00"},"s":"I", "c":"STORAGE", "id":22
430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr":{"message":"[1648758784:276091][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_ALL] Main recovery loop: starting at 1/24704 to 2/256"}}
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:04.572+00:00"},"s":"I", "c":"STORAGE", "id":22
430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr":{"message":"[1648758784:572853][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 1 through 2"}}
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:04.747+00:00"},"s":"I", "c":"STORAGE", "id":22
430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr":{"message":"[1648758784:747590][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 2 through 2"}}
EjemploApp | Servidor en el puerto: 3000
my_mongo_database | {"t":{"$date":"2022-03-31T20:33:04.811+00:00"},"s":"I", "c":"STORAGE", "id":22
430, "ctx":"initandlisten", "msg":"WiredTiger message", "attr":{"message":"[1648758784:811030][1:0x7
ff004edbc80], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 1 through 2"}}

```

Podemos probar el servidor que ahora se inicia con el docker compose, abriera en el puerto 4000 tal y como creamos la imagen anteriormente.



Ahora creamos en el proyecto la carpeta de rutas y el archivo de index.routes.js. Paralelamente dentro del archivo docker-compose agregamos el termino “volumes”, el cual permite copiar ambas direcciones del proyecto contenedor y del contenedor al proyecto que estamos realizando.

```

1  version: '3'
2
3  services:
4    web:
5      container_name: EjemploApp
6      restart: always
7      build: .
8      ports:
9        - "5000:3000"
10     links:
11       - mongo
12     volumes:
13       - ./usr/src/app
14
15    mongo:
16      container_name: my_mongo_database
17      image: mongo
18      ports:
19        - "27016:27017"

```

Volvemos a crear una nueva imagen con docker compose build y luego arrancamos con docker compose up.

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose build
[sudo] contraseña para ruisoart:
mongo uses an image, skipping
Building web
Step 1/8 : FROM node:14
--> 7d03025aad90
Step 2/8 : RUN mkdir -p /usr/src/app
--> Using cache
--> 11b7d014a68a
Step 3/8 : WORKDIR /usr/src/app
--> Using cache
--> 31c4abca387b
Step 4/8 : COPY package*.json ./
--> Using cache
--> 9e7b530a3a25
Step 5/8 : RUN npm i -g
--> Using cache
--> 6f54df873d3c
Step 6/8 : COPY . .
--> aa87a9d1411a
Step 7/8 : EXPOSE 3000
--> Running in 14f86c9c62ec
Removing intermediate container 14f86c9c62ec
--> 71a4b12f47e9
Step 8/8 : CMD ["npm", "start"]
--> Running in c42a36b94720
Removing intermediate container c42a36b94720
--> eb0d18e4c38d
Successfully built eb0d18e4c38d
Successfully tagged nodemongoconnection_web:latest
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$

```

Verificamos las imagenes creadas y eliminamos aquellas que sean muy antiguas.


```

ruisoart@ruiso-VirtualPC:~$ sudo docker images
[sudo] contraseña para ruisoart:
REPOSITORY                                TAG      IMAGE ID       CREATED        SIZE
nodemongoconnection_web                   latest   88de402c90fc   3 days ago    971MB
hellonode2                                latest   5ab6be52ea51   3 days ago    971MB
hellonode                                 latest   16dd38eaa5ad   3 days ago    971MB
node                                       14       7d03025aad90   3 days ago    946MB
mongo                                      latest   798d1656acba   2 weeks ago   698MB
postgres                                  latest   5cd1494671e9   2 weeks ago   376MB
dockeronruiso/secondimageruiso           latest   936463c49201   2 weeks ago   142MB
secondimageruiso                         latest   936463c49201   2 weeks ago   142MB
wordpress                                 latest   44a0bf559e55   2 weeks ago   605MB
httpd                                      latest   6b8e87fff107   2 weeks ago   144MB
dpage/pgadmin4                           latest   4b5bbddb3624   2 weeks ago   340MB
mariadb                                   latest   f5dd1ac0b00e   3 weeks ago   414MB
mysql                                      latest   826efd84393b   3 weeks ago   521MB
python                                    latest   178dcaa62b39   3 weeks ago   917MB
ubuntu                                    latest   2b4cba85892a   4 weeks ago   72.8MB
nginx                                      latest   c919045c4c2b   4 weeks ago   142MB
mysql/mysql-server                       latest   434c35b82b08   2 months ago  417MB
alpine                                    latest   c059bfaa849c   4 months ago   5.59MB
hello-world                              latest   feb5d9fea6a5   6 months ago   13.3kB
mariadb/server                           10.4     6bc393df66bb   11 months ago  353MB

```

```

ruisoart@ruiso-VirtualPC:~$ sudo docker rmi hellonode -f
Untagged: hellonode:latest
Deleted: sha256:16dd38eaa5ad65c25be27528043105b27c6a0af968c0facc323518355f76862e
Deleted: sha256:5eb36e11e22942fbd4a11024f34bfba9f7fb1a6d5fed668d0341634d79e00210
Deleted: sha256:bc06b885fef7cd55dab4b0a372c8cffe1754f6e6b4750b5855966c64e69d5d2
Deleted: sha256:d9083ef21170f7e90e497746ec037a371542eae8201d58513ae1dc73f5fa9516
Deleted: sha256:e039d3dac00f3f8d7caafbdd18f5b0334d96679452076ded3f2b79a5f4d488ed
ruisoart@ruiso-VirtualPC:~$

```

```

ruisoart@ruiso-VirtualPC:~$ sudo docker images
REPOSITORY                                TAG      IMAGE ID       CREATED        SIZE
nodemongoconnection_web                   latest   eb0d18e4c38d   4 minutes ago  971MB
<none>                                    <none>    88de402c90fc   3 days ago    971MB
hellonode2                                latest   5ab6be52ea51   3 days ago    971MB
node                                       14       7d03025aad90   3 days ago    946MB
mongo                                      latest   798d1656acba   2 weeks ago   698MB
postgres                                  latest   5cd1494671e9   2 weeks ago   376MB
dockeronruiso/secondimageruiso           latest   936463c49201   2 weeks ago   142MB
secondimageruiso                         latest   936463c49201   2 weeks ago   142MB
wordpress                                 latest   44a0bf559e55   2 weeks ago   605MB
httpd                                      latest   6b8e87fff107   2 weeks ago   144MB
dpage/pgadmin4                           latest   4b5bbddb3624   2 weeks ago   340MB
mariadb                                   latest   f5dd1ac0b00e   3 weeks ago   414MB
mysql                                      latest   826efd84393b   3 weeks ago   521MB
python                                    latest   178dcaa62b39   3 weeks ago   917MB
ubuntu                                    latest   2b4cba85892a   4 weeks ago   72.8MB
nginx                                      latest   c919045c4c2b   4 weeks ago   142MB
mysql/mysql-server                       latest   434c35b82b08   2 months ago  417MB
alpine                                    latest   c059bfaa849c   4 months ago   5.59MB
hello-world                              latest   feb5d9fea6a5   6 months ago   13.3kB
mariadb/server                           10.4     6bc393df66bb   11 months ago  353MB
ruisoart@ruiso-VirtualPC:~$

```

Inicamos nuevamente sudo docker-compose up, esto deberia iniciar cada una de las diferentes lineas escritas en el archivo compose.

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose up
[sudo] contraseña para ruisoart:
Starting my_mongo_database ... done
Recreating EjemploApp ... done
Attaching to my_mongo_database, EjemploApp
my_mongo_database | {"t":{"$date":"2022-04-04T00:57:25.520+00:00"},"s":"I", "c":"CONTROL", "id":23
285, "ctx":"-","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabl
edProtocols 'none'"}

```

```

ruisoart@ruiso-VirtualPC:~$ sudo docker ps
[sudo] contraseña para ruisoart:
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                               NAMES
4e1e2d164fe7   nodemongoconnection_w "docker-entrypoint.s..." About a minute Up About a minute 0.0.0.0:5000->3000/tcp, :::5000->3000/tcp EjemploApp
be4ac193a191   mongo                 "docker-entrypoint.s..." 3 days ago    Up About a minute 0.0.0.0:27016->27017/tcp, :::27016->27017/tcp my_mongo_database
019100a602d5   hellonode2            "docker-entrypoint.s..." 3 days ago    Up 48 minutes   0.0.0.0:4000->3000/tcp, :::4000->3000/tcp busy_heyrovsky

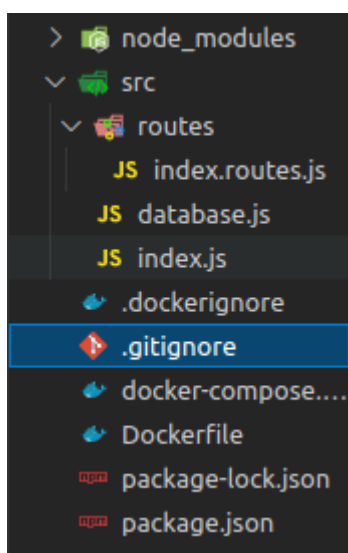
```

Abrimos una nueva terminal para no afectar el docker-compose que ya hemos iniciado, en ella hacemos a la parte de bash para poder generar desde allí la creación del archivo “gitignore”, en caso de que no suceda siempre podemos hacerlo manualmente, pues el mismo se crea vacío (es recomendable poner la misma instrucción que “dockerignore”).

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker exec -it EjemploApp bash
root@4e1e2d164fe7:/usr/src/app# ls -la
.      .dockerignore  docker-compose.yml  package-lock.json  src
..     Dockerfile     node_modules        package.json
root@4e1e2d164fe7:/usr/src/app#

```



En una nueva consola de comandos, pero con la misma ruta de origen del proyecto ejecutamos la siguiente instrucción para poder obtener el paquete que facilite el reinicio del servidor de forma automática (lo que hace este paquete es detectar cualquier cambio en los archivos js y automáticamente recargar el servicio con los nuevos cambios).

```

ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ npm i nodemon -D
up to date, audited 250 packages in 5s
22 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

```

Una vez tenemos este paquete lo configuramos dentro de nuestro archivo package.json para que así se reinicie el servicio con todo y el “nodemon”.


```

package.json > {} scripts > dev
1  {
2    "name": "tallersiete",
3    "version": "1.0.0",
4    "description": "Node,Mongo y Docker Compose",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1",
8      "start": "node src/index.js",
9      "dev": "nodemon src/index.js"
10   },
11   "author": "Luis Felipe Narvaez Gomez",
12   "license": "ISC",
13   "dependencies": {
14     "cors": "^2.8.5",
15     "express": "^4.17.3",
16     "jsonwebtoken": "^8.5.1",
17     "mongoose": "^6.2.9",
18     "morgan": "^1.10.0",
19     "pg-promise": "^10.11.1"
20   },
21   "devDependencies": {
22     "@types/cors": "^2.8.12",
23     "@types/express": "^4.17.13",
24     "@types/jsonwebtoken": "^8.5.8",
25     "@types/morgan": "^1.9.3",
26     "nodemon": "^2.0.15"
27   }
28 }
29

```

Ya que hemos creado un nuevo script, hay que nombrarlo dentro del Docker File y volver a repetir la construccion y arranque del servidor.

```

Dockerfile > ...
1  #Cuando se cree el contenedor, lo primero que hata es leer
2  # El docker File
3
4  #Asi lo primero que hara sera instalar node
5  FROM node:14
6
7  # Ahora crear la carpeta donde ira el codigo
8  #Los comandos son de linux ya que docker de por si es un
9  # contenedor de comandos de linux
10 RUN mkdir -p /usr/src/app
11
12 # establecerse en el directorio actual
13 WORKDIR /usr/src/app
14
15 #copiar mi package.json y package-lock.json a mi contenedor
16 #para poder volver a instalar dependencias
17 COPY package*.json ./
18
19 #instalar todas las dependencias
20 RUN npm i -g
21
22 # El directorio actual sera el contenedor actual
23 COPY . .
24
25 #Se especifica el puerto del contenedor
26 EXPOSE 3000
27
28 #Iniciar los siguientes comandos
29 CMD ["npm", "start", "dev"]

```

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose stop
[sudo] contraseña para ruisoart:
Stopping EjemploApp      ... done
Stopping my_mongo_database ... done
```

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose build
mongo uses an image, skipping
Building web
Step 1/8 : FROM node:14
--> 7d03025aad90
Step 2/8 : RUN mkdir -p /usr/src/app
--> Using cache
--> 11b7d014a68a
Step 3/8 : WORKDIR /usr/src/app
--> Using cache
--> 31c4abca387b
Step 4/8 : COPY package*.json ./
--> 00692f2324cc
Step 5/8 : RUN npm i -g
--> Running in 28c65271b381
+ tallersiete@1.0.0
added 119 packages from 119 contributors in 15.876s
Removing intermediate container 28c65271b381
--> ee0bd9b9b289
Step 6/8 : COPY . .
--> a7b6e6c89710
Step 7/8 : EXPOSE 3000
--> Running in 883d26806a0b
Removing intermediate container 883d26806a0b
--> bce528c4c813
Step 8/8 : CMD ["npm", "start", "dev"]
--> Running in 8366f442e04e
Removing intermediate container 8366f442e04e
--> d1e0f97521f0

Successfully built d1e0f97521f0
Successfully tagged nodemongoconnection_web:latest
```

```
ruisoart@ruiso-VirtualPC:~/Escritorio/Website/NodeMongoConnection$ sudo docker-compose up
Starting my_mongo_database ... done
Recreating EjemploApp      ... done
Attaching to my_mongo_database, EjemploApp
my_mongo_database | {"t":{"$date":"2022-04-04T01:36:21.079Z"},"s":"I", "c":"NETWORK", "id":4915701,
"ctx":"-", "msg":"Initialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVer
sion":0,"maxWireVersion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoi
```

Nota: Cada que hacemos estos cambios en la recreacion de la imagen y el arranque del servidor, podemos confirmar que este funcionando correctamente con un navegador web o con el Software de Postman, la respuesta o peticion GET que antes recibiamos no debe haber cambiado.

Ahora podemos modificar el archivo de ruta y el archivo index, cambios que seran detectados por “nodemon” y que reiniciara el servidor automaticamente aplicando las nuevas rutas.

```
src > routes > JS index.routes.js > ...
1  const {Router} = require('express');
2  const router = Router();
3
4  router.get('/', (req,res)=>{
5    res.send('Hellooooooooo')
6  });
7
8  module.exports = router;
```

```
src > JS index.js > ...
1  const express = require('express');
2
3  const app = express();
4
5  require('./database');
6
7  app.use(require('./routes/index.routes'));
8
9  app.listen(3000);
10 console.log("Servidor en el puerto: ", 3000);
```

Con estos cambios podemos rectificar que ahora funione el mensaje que mandamos, en caso de que los cambios no se detecten automaticamente, podemos rehacer los pasos:

sudo docker-compose stop
sudo docker-compose build
sudo docker-compose up

The screenshot shows a web browser interface with the address bar set to `http://localhost:5000`. Below the address bar, the HTTP method is set to `GET` and the URL is `http://localhost:5000`. The `Send` button is visible. The response is displayed in the `Body` tab, showing a `200 OK` status, a response time of `45 ms`, and a response size of `238 B`. The response body is `Hellooooooooo`.