

Inciso 8

Generar Back

Generamos un archivo .js con su archivo package.json

En este caso api.js y package.json

Api.js

```
const express = require('express');
const app = express();

app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});

// Ruta para obtener la hora y el nombre
app.get('/info', (req, res) => {
  const nombre = "José De León";
  const hora = new Date().toLocaleTimeString();
  const data = {
    nombre: nombre,
    hora: hora,
    hello: "Hello-World!"
  };
  res.json(data);
});

// Puerto en el que escucha el servidor
const PORT = 3000;

// Iniciar el servidor
app.listen(PORT, () => {
  console.log(`Servidor API escuchando en el puerto ${PORT}`);
});
```

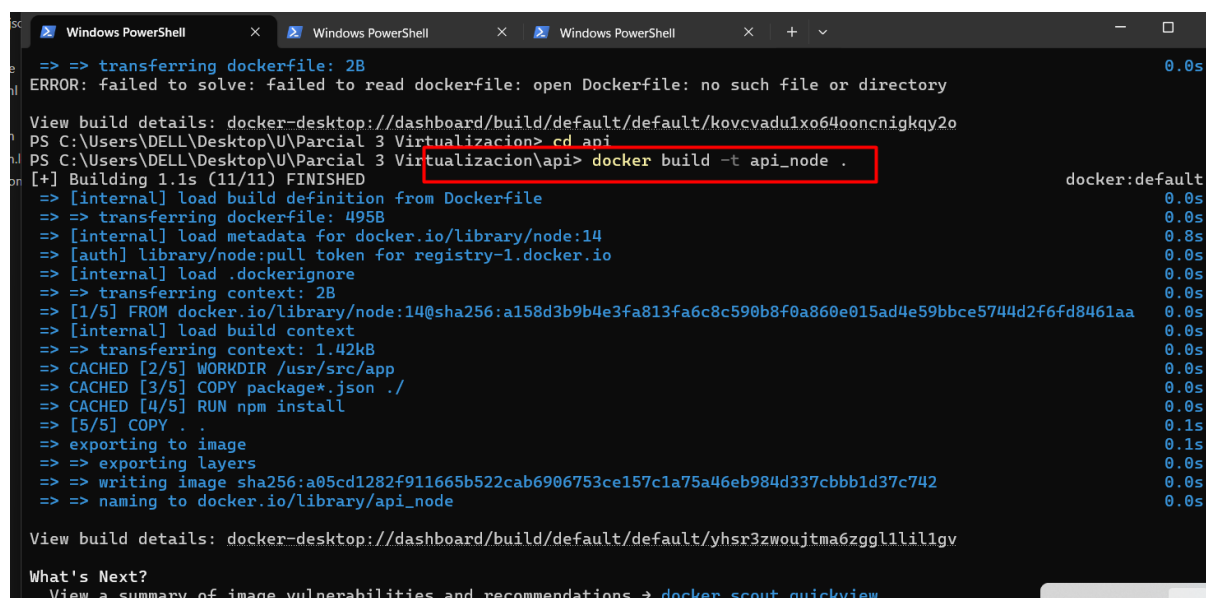
Package.json

```
api > {} package.json > ...
1  {
2    "dependencies": {
3      "express": "^4.17.1"
4    }
5  }
6
```

Ahora generamos un archivo dockerfile para dockerizar el archivo

```
api > Dockerfile
1  # Usar la imagen oficial de Node.js como base
2  FROM node:14
3
4  # Establecer el directorio de trabajo en la imagen
5  WORKDIR /usr/src/app
6
7  # Copiar el package.json y package-lock.json (si existe)
8  COPY package*.json ./
9
10 # Instalar las dependencias
11 RUN npm install
12
13 # Copiar el resto de los archivos de la aplicación
14 COPY . .
15
16 # Exponer el puerto 3000 en el contenedor
17 EXPOSE 3000
18
19 # Comando para iniciar la aplicación
20 CMD ["node", "api.js"]
21
```

Y corremos con los siguientes comandos, con powershell



```
Windows PowerShell
PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion> cd api
PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion\api> docker build -t api_node .

[+] Building 1.1s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 495B
=> [internal] load metadata for docker.io/library/node:14
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
=> [internal] load build context
=> => transferring context: 1.42kB
=> CACHED [2/5] WORKDIR /usr/src/app
=> CACHED [3/5] COPY package*.json ./
=> CACHED [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:a05cd1282f911665b522cab6906753ce157c1a75a46eb984d337cbbb1d37c742
=> => naming to docker.io/library/api_node

View build details: docker-desktop://dashboard/build/default/default/yhsr3zwoujtma6zggl1l1l1gv
What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
```

Y el siguiente para indicar el puerto

José De León
1170419

Virtualización Parcial 3

=> => naming to docker.io/library/api_node

View build details: docker-desktop://dashboard/build/default/default/yhsr3zwoujtma6zggl1lil1gv

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion\api> `docker run -p 3000:3000 api_node`

Servidor API escuchando en el puerto 3000

Generar Front

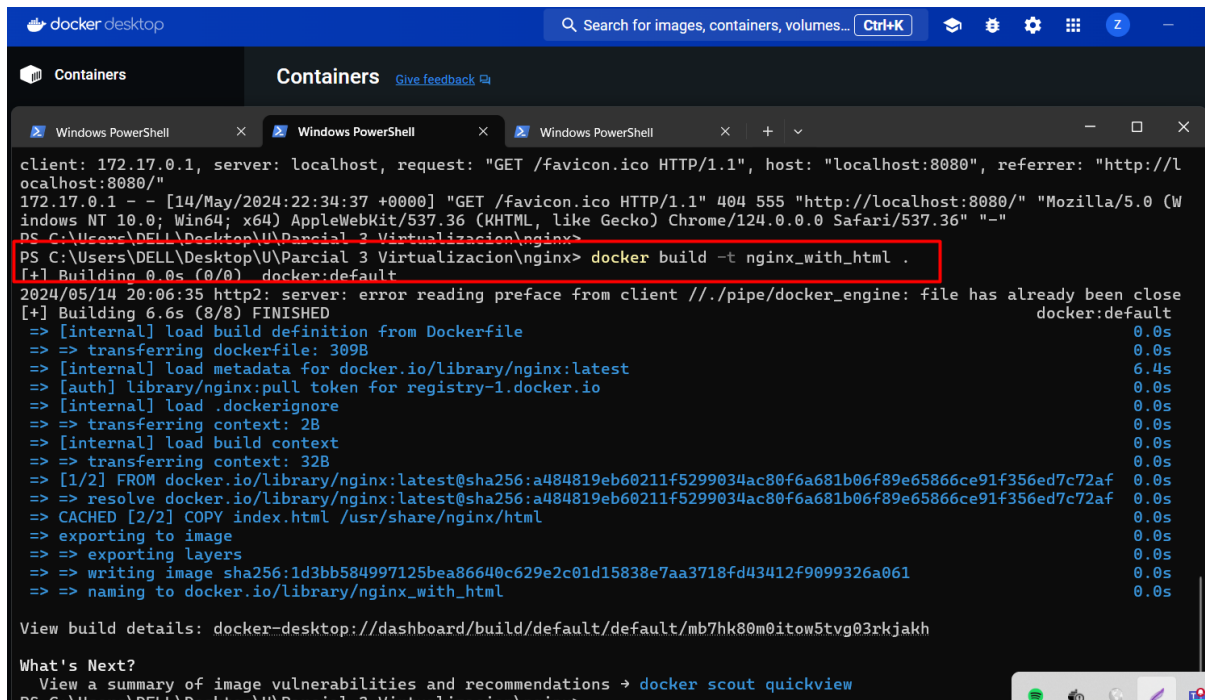
Generamos un archivo HTML, con su apartado de JavaScript, para poder hacer uso del backend (en este caso ya vemos su llamado a la dirección, puerto y ruta).

```
nginx > index.html > ...
1  <!-- index.html -->
2
3  <!DOCTYPE html>
4  <html lang="en">
5  <head>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Consumo de Endpoint</title>
9  </head>
10 <body>
11     <h1>Consumo de Endpoint</h1>
12     <button id="fetchData">Obtener Datos</button>
13     <p id="data"></p>
14
15     <script>
16         document.getElementById('fetchData').addEventListener('click', async () => {
17             const response = await fetch('http://localhost:3000/info');
18             const data = await response.json();
19             document.getElementById('data').innerText = `Nombre: ${data.nombre}, Hora: ${data.hora}, HolaMUND!!!: ${data.hello} `;
20         });
21     </script>
22 </body>
23 </html>
24
```

Para poder dockerizarlo hacemos uso de este archivo “dockerfile”

```
nginx > Dockerfile
1  # Usar la imagen oficial de NGINX como base
2  FROM nginx:latest
3
4  # Copiar el archivo HTML al directorio de contenido estático de NGINX
5  COPY index.html /usr/share/nginx/html
6
7  # Exponer el puerto 80 para que NGINX sea accesible desde fuera del contenedor
8  EXPOSE 80
9
```

Para poder correrlo, uso powershell para facilidad, con estos comandos

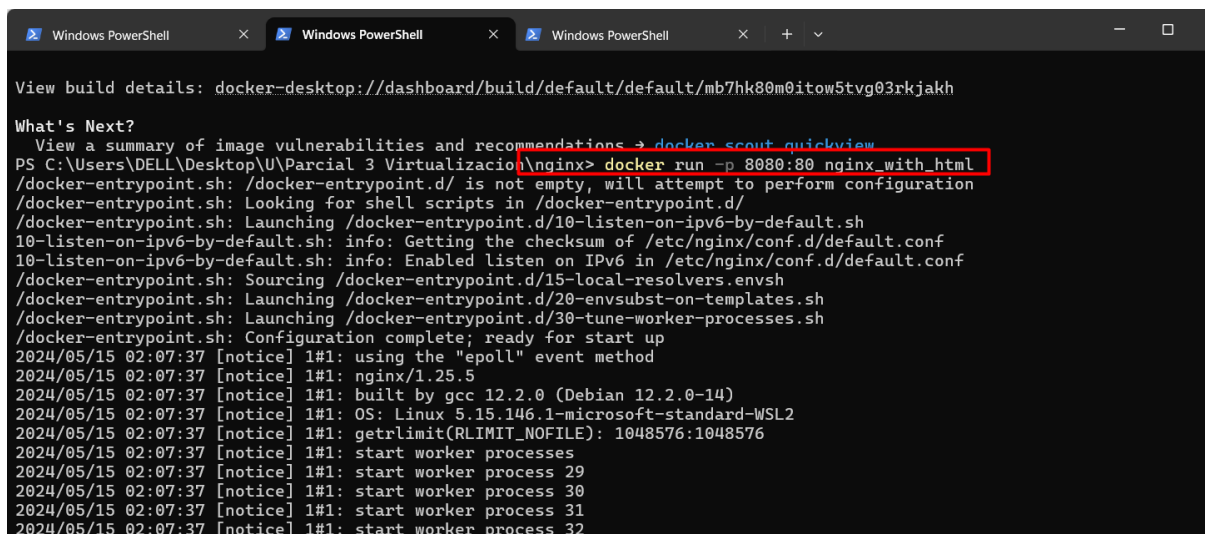


```
client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8080", referer: "http://localhost:8080/"
172.17.0.1 - - [14/May/2024:22:34:37 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion\nginx> docker build -t nginx_with_html .
[+] Building 0.0s (0/0)  docker:default
2024/05/14 20:06:35 http2: server: error reading preface from client //./pipe/docker_engine: file has already been close
[+] Building 6.6s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default
=> => transferring dockerfile: 309B                                              0.0s
=> [internal] load metadata for docker.io/library/nginx:latest                  6.4s
=> [auth] library/nginx:pull token for registry-1.docker.io                    0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 32B                                                0.0s
=> [1/2] FROM docker.io/library/nginx:latest@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af 0.0s
=> => resolve docker.io/library/nginx:latest@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af 0.0s
=> CACHED [2/2] COPY index.html /usr/share/nginx/html                          0.0s
=> exporting to image                                                            0.0s
=> => exporting layers                                                            0.0s
=> => writing image sha256:1d3bb584997125bea86640c629e2c01d15838e7aa3718fd43412f9099326a061 0.0s
=> => naming to docker.io/library/nginx_with_html                             0.0s

View build details: docker-desktop://dashboard/build/default/default/mb7hk80m0itow5tvg03rkjakh

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion\nginx>
```

Y luego este



```
View build details: docker-desktop://dashboard/build/default/default/mb7hk80m0itow5tvg03rkjakh

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS C:\Users\DELL\Desktop\U\Parcial 3 Virtualizacion\nginx> docker run -p 8080:80 nginx_with_html
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/05/15 02:07:37 [notice] 1#1: using the "epoll" event method
2024/05/15 02:07:37 [notice] 1#1: nginx/1.25.5
2024/05/15 02:07:37 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/05/15 02:07:37 [notice] 1#1: OS: Linux 5.15.146.1-microsoft-standard-WSL2
2024/05/15 02:07:37 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/05/15 02:07:37 [notice] 1#1: start worker processes
2024/05/15 02:07:37 [notice] 1#1: start worker process 29
2024/05/15 02:07:37 [notice] 1#1: start worker process 30
2024/05/15 02:07:37 [notice] 1#1: start worker process 31
2024/05/15 02:07:37 [notice] 1#1: start worker process 32
```

Para indicarle puerto.

Para esta al ser no relacional usaremos mongo db con el siguiente comando

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\DELL> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
a8b1c5f80c2d: Pull complete
408f9504c110: Pull complete
03d18b647343: Pull complete
c24f68d81052: Pull complete
1df517147e11: Pull complete
77d5ebe2f2e0: Pull complete
c21b89d414fc: Pull complete
4138c7eb3b71: Pull complete
Digest: sha256:c578cda6630625bee0f7c522120fd89a8ffc2b9d88fa77a6ceb3447cedbd412a
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview mongo
PS C:\Users\DELL>
```

Al tener la imagen, hacemos uso de este para poder levantar la instancia de mongoDB

```
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\DELL> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
a8b1c5f80c2d: Pull complete
408f9504c110: Pull complete
03d18b647343: Pull complete
c24f68d81052: Pull complete
1df517147e11: Pull complete
77d5ebe2f2e0: Pull complete
c21b89d414fc: Pull complete
4138c7eb3b71: Pull complete
Digest: sha256:c578cda6630625bee0f7c522120fd89a8ffc2b9d88fa77a6ceb3447cedbd412a
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview mongo
PS C:\Users\DELL> docker run -d -p 27017-27019:27017-27019 --name mongodb mongo
8f91b91c12d12362d63ab219f0a494c19676c9dc94bf52c82adbaafa17a943a8
PS C:\Users\DELL>
```

Anexos

Ejecución de contenedores.




Container CPU usage ⓘ
0.65% / 800% (8 CPUs available)

Container memory usage ⓘ
103.07MB / 7.51GB

Show charts




⏸

☐ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	 magical_ram 5028d8949299	nginx_with_html	Running	8080:80	0%	10 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	 practical_edt 41ad9479201e	api_node	Running	3000:3000	0%	6 minutes ago	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	 mongodb 8f91b91c12d1	mongo	Running	27017:27017 Show all ports (3)	0.65%	1 minute ago	<div><div></div><div></div><div></div></div>

Ejecución del API (por alguna razon la hora no la da bien)

← → ↻ ⓘ localhost:3000/info

 Gmail  YouTube  Maps

Impresión con formato estilístico ☐

```
{"nombre": "José De León", "hora": "2:23:20 AM", "hello": "Hello-World!"}
```

Front antes de consumo



Front despues de consumo

