

Desarrollo Web Integral

Fatima Natalia Ruiz Rivera

1220100350

GDGS0391

Practica 2

DOCENTE:

Gabriel Barrón Rodriguez

14/julio/2023

INDICE

CONTENIDO

Crear un proyecto de angular	2
Se crea el proyecto de angular	3
Crear mi propia aplicación en local.....	4
Agregar el archivo DOCKER	4
Crear la imagen	4
verificar las imágenes creadas	5
Ejecutar la imagen y exponerla en el puerto 8080	6
Mostrar un listado de los contenedores en ejecución	7

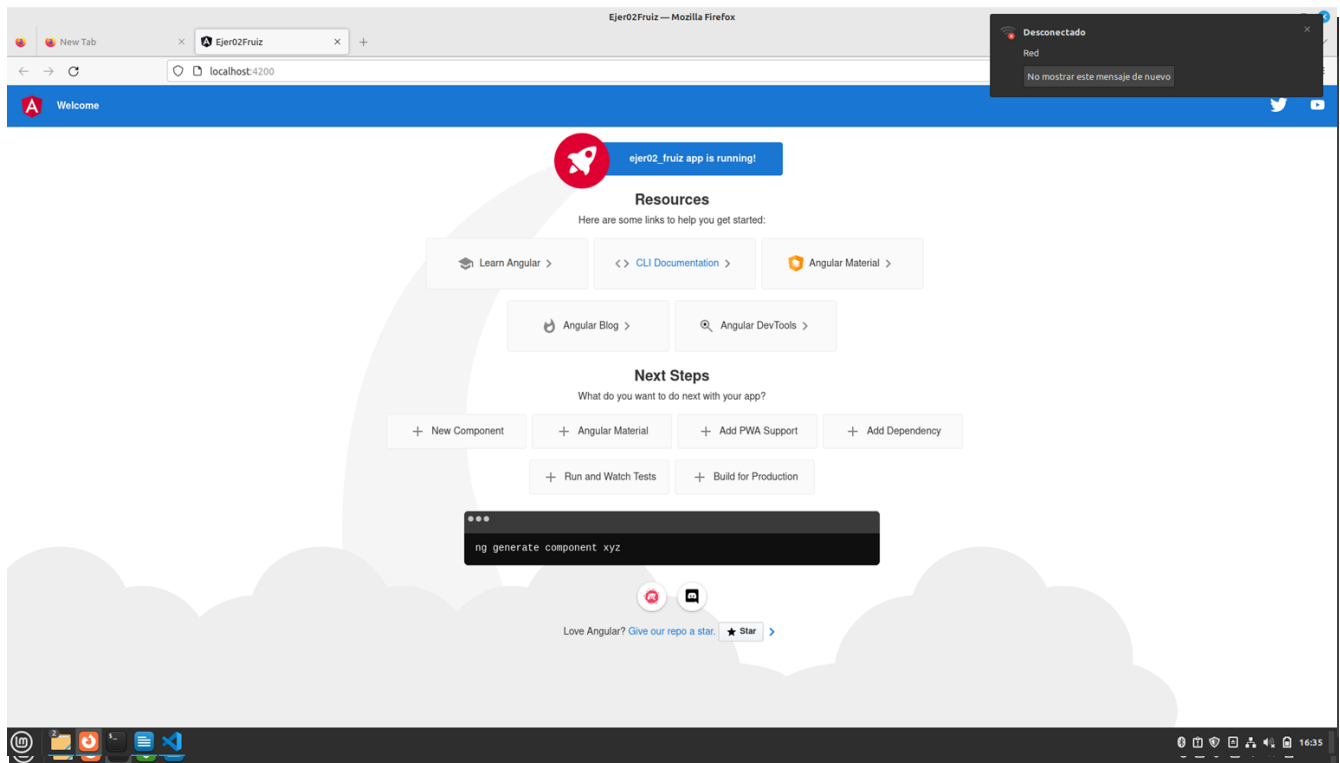
Crear un proyecto de angular

```
fatima@fatima-LP14123: ~/Escritorio
fatima@fatima-LP14123:~$ cd Escritorio
fatima@fatima-LP14123:~/Escritorio$ ng new ejer02_frui2 --style=scss --routing
```

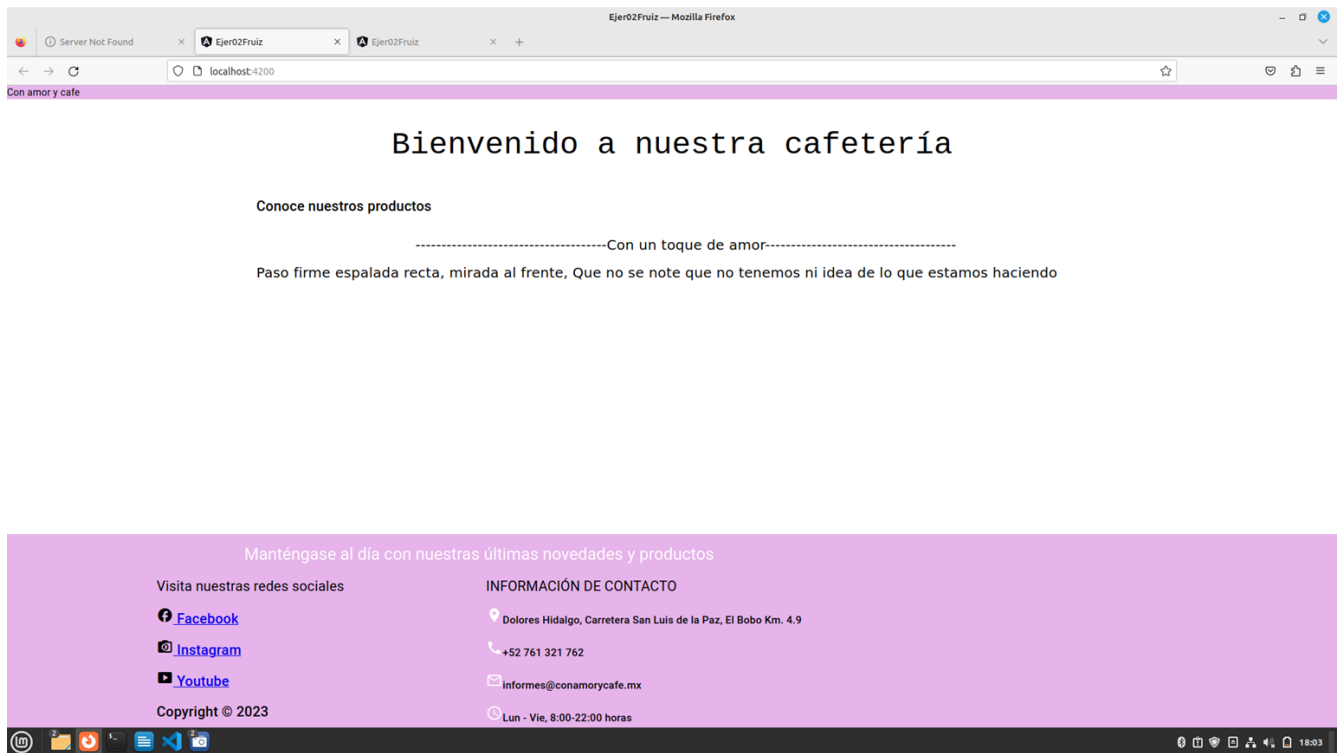
```
fatima@fatima-LP14123:~/Escritorio$ ng new ejer02_frui2 --style=scss --routing
CREATE ejer02_frui2/README.md (1065 bytes)
CREATE ejer02_frui2/.editorconfig (274 bytes)
CREATE ejer02_frui2/.gitignore (548 bytes)
CREATE ejer02_frui2/angular.json (2984 bytes)
CREATE ejer02_frui2/package.json (1043 bytes)
CREATE ejer02_frui2/tsconfig.json (901 bytes)
CREATE ejer02_frui2/tsconfig.app.json (263 bytes)
CREATE ejer02_frui2/tsconfig.spec.json (273 bytes)
CREATE ejer02_frui2/.vscode/extensions.json (139 bytes)
CREATE ejer02_frui2/.vscode/launch.json (478 bytes)
CREATE ejer02_frui2/.vscode/tasks.json (938 bytes)
CREATE ejer02_frui2/src/main.ts (214 bytes)
CREATE ejer02_frui2/src/favicon.ico (949 bytes)
CREATE ejer02_frui2/src/index.html (297 bytes)
CREATE ejer02_frui2/src/styles.scss (80 bytes)
CREATE ejer02_frui2/src/app/app-routing.module.ts (245 bytes)
CREATE ejer02_frui2/src/app/app.module.ts (392 bytes)
CREATE ejer02_frui2/src/app/app.component.scss (0 bytes)
CREATE ejer02_frui2/src/app/app.component.html (23115 bytes)
CREATE ejer02_frui2/src/app/app.component.spec.ts (1009 bytes)
CREATE ejer02_frui2/src/app/app.component.ts (217 bytes)
CREATE ejer02_frui2/src/assets/.gitkeep (0 bytes)
Installing packages (npm)...
```

UNIDAD 1 Herramientas para la administración de proyectos

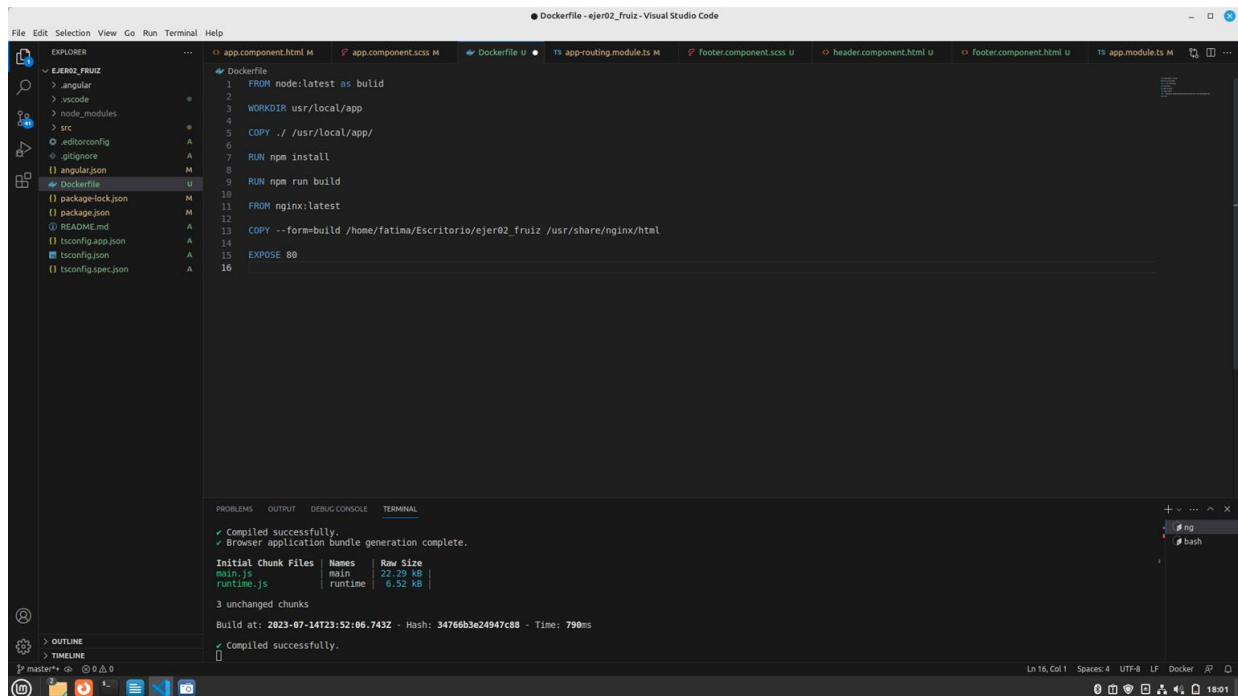
Se crea el proyecto de angular



Crear mi propia aplicación en local

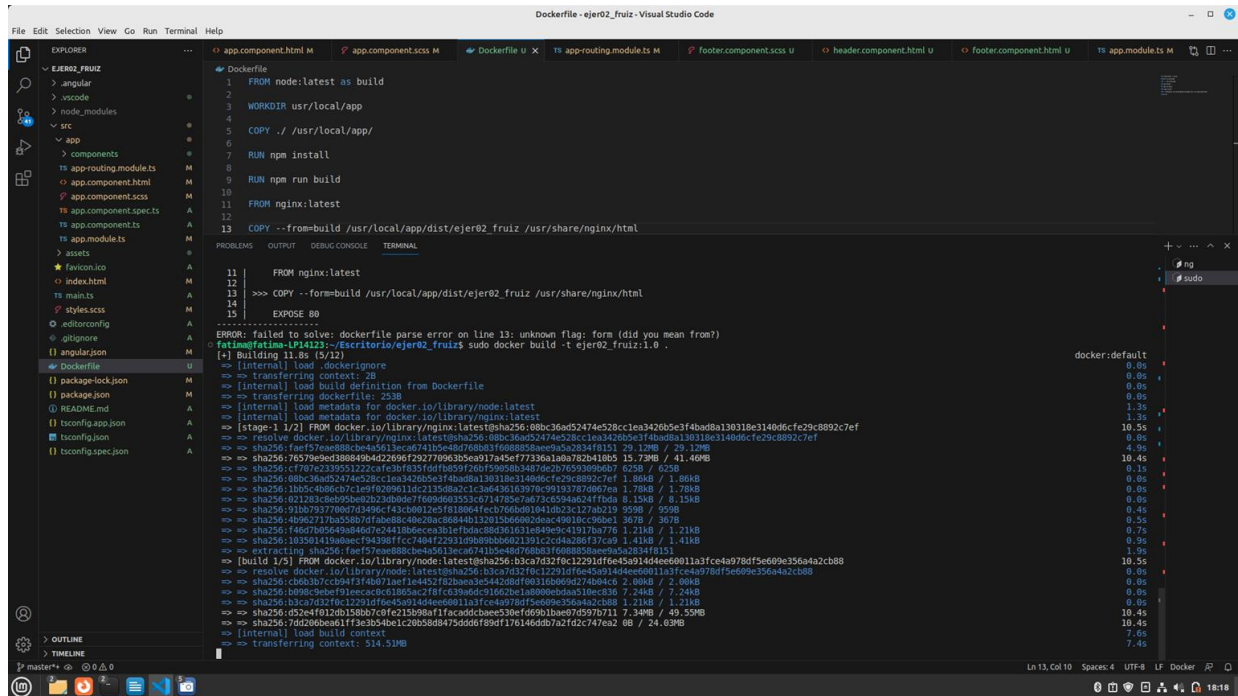


Agregar el archivo DOCKER



Crear la imagen

UNIDAD 1 Herramientas para la administración de proyectos



The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
1 FROM node:latest as build
2
3 WORKDIR /usr/local/app
4
5 COPY ./ /usr/local/app/
6
7 RUN npm install
8
9 RUN npm run build
10
11 FROM nginx:latest
12
13 COPY --from=build /usr/local/app/dist/ejer02_frui2 /usr/share/nginx/html
```

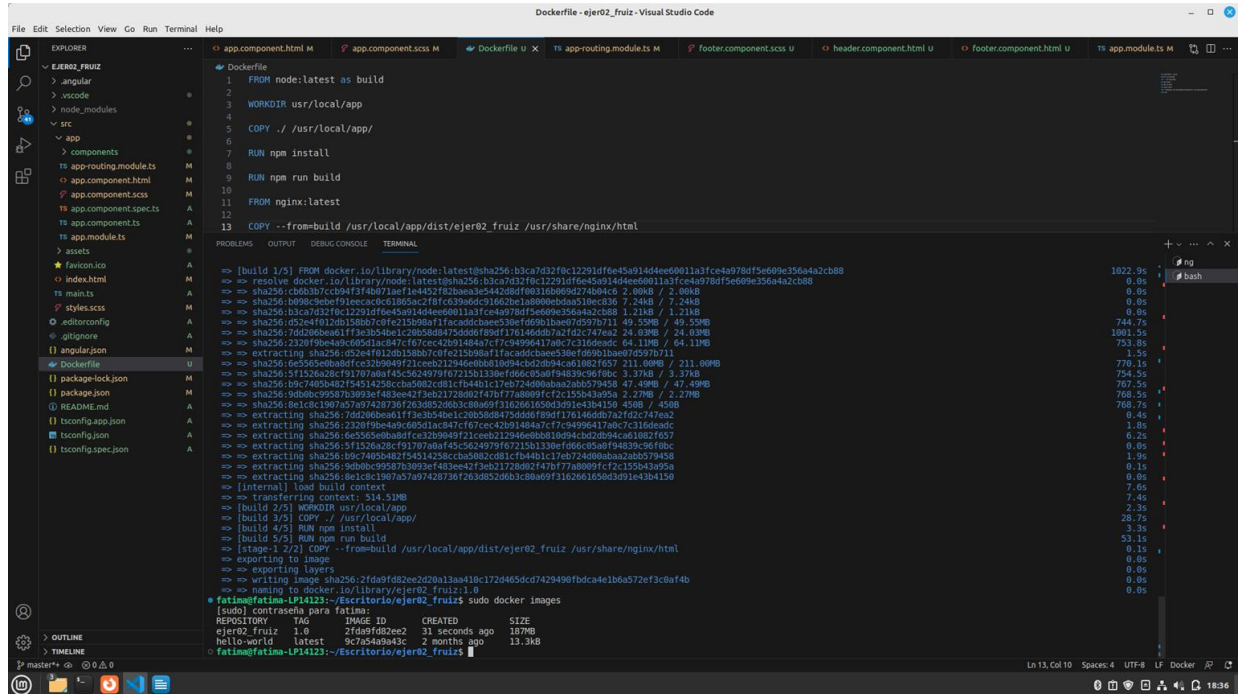
The terminal output shows the build process for the Docker image. It starts with the command `sudo docker build -t ejer02_frui2:1.0 .` and displays the progress of the build, including the installation of dependencies and the copying of files to the final image.

```
[+] Building 11.5s (0/12)
=> [internal] load .dockerignore
=> transferring context: 2B
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 253B
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load metadata for docker.io/library/nginx:latest
=> [stage 1/2] FROM docker.io/library/nginx:latest@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e314d0d6fe29c8092c7ef
=> resolve docker.io/library/nginx:latest@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e314d0d6fe29c8092c7ef
=> sha256:f4e757a0e0a0b0e4501bc0d74105e4d0f08a031f088080a0e45a30341f0131 29.12MB / 29.12MB
=> sha256:76579e9ed388849b4d22696f292770963b5ea917a45ef77336a1a0a782b410b5 15.73MB / 41.46MB
=> sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e314d0d6fe29c8092c7ef 1.86kB / 1.86kB
=> sha256:1b05c4b86cb7c1ef0209611dc213588a2c1c386436163978c991337870867ea 1.78kB / 1.78kB
=> sha256:021183c4e0950e0222b0d0e7f609d603533c771702e7a072c5094e24f0b 0.13kB / 0.13kB
=> sha256:91bb7937780d7d3496cf43cb0012e5f818864fecb766d01041db23c127ab219 959B / 959B
=> sha256:409d2717b55b07dfab88c40c20ac0844b12015666020ca49010c90b0e1 367B / 367B
=> sha256:f467b07569a8467c241180eca3b1ef6a08036311e490e9c410173b776 1.21kB / 1.21kB
=> sha256:103501419a0ecf94398ffcc7404f22931d9889b0b0821391c2c04286f37ca9 1.41kB / 1.41kB
=> extracting sha256:f4e757a0e0a0b0e4501bc0d74105e4d0f08a031f088080a0e45a30341f0131
=> [build 1/3] FROM docker.io/library/node:latest@sha256:b3ca7d32f8c12291d6f6e45a9144ee0011a3fce4a978df5e609e356a4a2cb88
=> resolve docker.io/library/node:latest@sha256:b3ca7d32f8c12291d6f6e45a9144ee0011a3fce4a978df5e609e356a4a2cb88
=> sha256:c083b7cc9413f40871ef4452f02b0a3d54330d4f061104099274004c0 2.00kB / 2.00kB
=> sha256:b096c9c0e0f1e0c0d01805ac210fc039addc916a2be1a00e0b0a510ec830 2.24kB / 2.24kB
=> sha256:b3ca7d32f8c12291d6f6e45a9144ee0011a3fce4a978df5e609e356a4a2cb88 1.21kB / 1.21kB
=> sha256:b524af012db130807c0f212b08a1f4ad0c0e530ef609b1a0e97d5970711 2.34MB / 49.59MB
=> sha256:7d52060ea01ff3e3b54b0c120b58d8475d506f89d176146ddb7a2f2c747ea2 0B / 24.03MB
=> [internal] load build context
=> transferring context: 314.53MB
```

verificar las imágenes creadas

sudo docker images

UNIDAD 1 Herramientas para la administración de proyectos



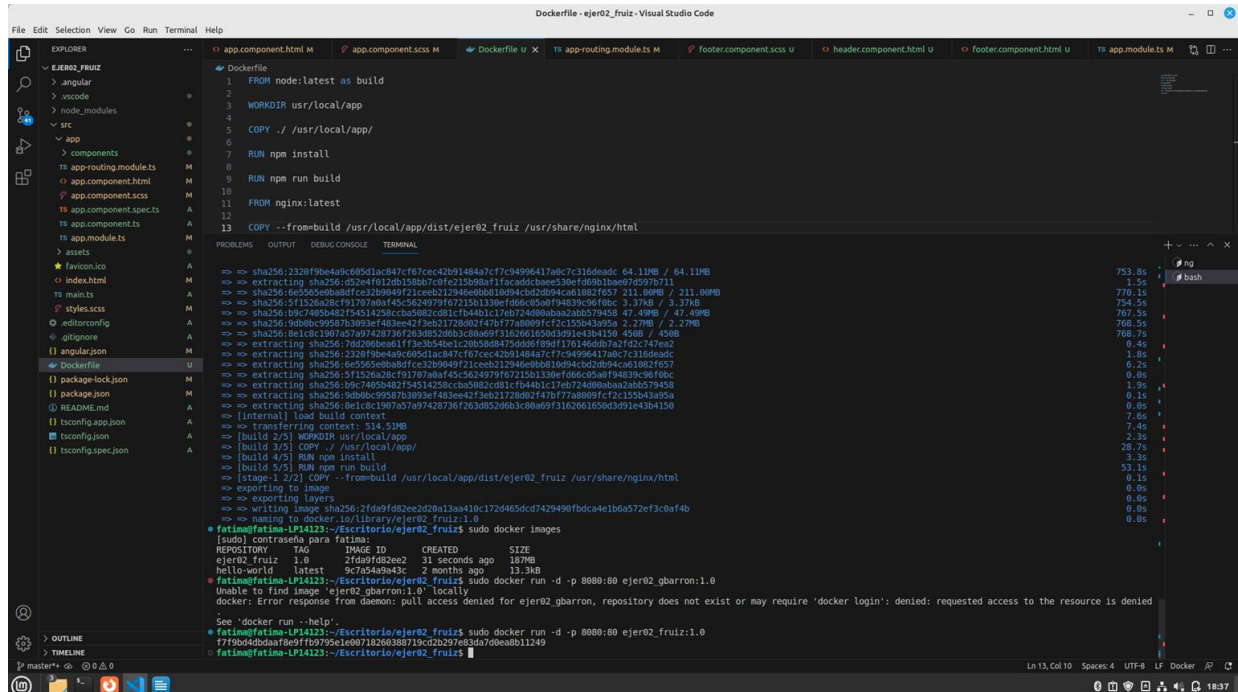
The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor. The Dockerfile contains the following instructions:

```
1 FROM node:latest as build
2
3 WORKDIR /usr/local/app
4
5 COPY ./ /usr/local/app/
6
7 RUN npm install
8
9 RUN npm run build
10
11 FROM nginx:latest
12
13 COPY --from=build /usr/local/app/dist/ejer02_fruiuz /usr/share/nginx/html
```

The output window shows the build process for the Docker image. It starts with pulling the base images (node:latest and nginx:latest) and then proceeds to build the application. The build process includes copying the application code to the container, installing dependencies, and running the build command. The final output shows the Docker image being pushed to the registry.

Ejecutar la imagen y exponerla en el puerto 8080

`sudo docker run -d -p 8080:80 ejer02_gbarron:1.0`



The screenshot shows the Visual Studio Code interface with the same Dockerfile open. The output window shows the result of running the Docker image. The output indicates that the image was successfully built and is now available in the registry. The command `docker run -d -p 8080:80 ejer02_gbarron:1.0` was executed, and the output shows that the container is running and exposing port 8080.

Mostrar un listado de los contenedores en ejecución

sudo docker ps

The screenshot shows the Visual Studio Code interface with a Dockerfile open in the editor and its execution output in the terminal. The Dockerfile contains the following instructions:

```

FROM node:latest as build
WORKDIR /usr/local/app
COPY ./ /usr/local/app/
RUN npm install
RUN npm run build
FROM nginx:latest
COPY --from=build /usr/local/app/dist/ejer02_fruiz /usr/share/nginx/html

```

The terminal output shows the execution of the Dockerfile, including the extraction of layers and the final image creation. The output also shows the execution of the command `sudo docker images`, which lists the images in the local repository:

```

[fa]@fatima-LP14123:~/Escritorio/ejer02_fruiz$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ejer02_fruiz        1.0                2fd99f82ee2d20a13aa410c172d465dc7429490fbdca4e1b6a572ef3c0af4b
hello-world         latest             9cf549a9a943c      2 months ago       13.3kB

```

The output also shows the execution of the command `sudo docker run -d -p 8080:80 ejer02_gbarron:1.0`, which failed due to a pull access denied error. The error message is:

```

Unable to find image 'ejer02_gbarron:1.0' locally
docker: Error response from daemon: pull access denied for ejer02_gbarron, repository does not exist or may require 'docker login': denied: requested access to the resource is denied

```