

**UNIVERSIDAD DE GUADALAJARA**  
**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS**

**Materia:** Computación tolerante a fallas

**Clave:** I7036

**Sección:** D06



**NRC:** 179961

**Microservicios**

**Código estudiante:** 221350567

**Alumno:** Rauf Alfonso Hamden Estrada

**Carrera:** Ingeniería en Computación

**Fecha:** 12/11/2023

**Docente:** Michel Emanuel Lopez Franco

2023B

## Introducción

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, se puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno

## Desarrollo

### Cargamos la imagen

```
a16f081f8d22928e7e71702050a088c0b7384541e74227474c372e0d00b472
docker: Error response from daemon: driver failed programming external connectivity on endpoint pedantic_leakey (ecee83df336c
4be278c73d0a): Bind for 127.0.0.1:3000 failed: port is already allocated.
S C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker run -dp 127.0.0.1:3000:4000 getting-started1
5d372b293d1ec52ee3e42a0b0facf4d19d3ae594da68984db9022354afcafe6
docker: Error response from daemon: driver failed programming external connectivity on endpoint wonderful_bartik (931d33a85a7e
c22121a1bc2f5): Bind for 127.0.0.1:3000 failed: port is already allocated.
S C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker run -dp 127.0.0.1:4000:4000 getting-started1
5d2d420e95729c8539f8f6bf4b586b1829f18566e30dc514e31eaf9417a5f23
S C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker build -t getting-started2 .
[+] Building 1.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 188B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> resolve image config for docker.io/docker/dockerfile:1
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be00
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:435dcad253bb5b7f347ebc69c8cc52de7c912eb7241098b920f2fc2d7843183d
=> [internal] load build context
=> => transferring context: 5.02kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:66820f446c9aa8f7dcc78bb9eaeaf21e32823f5f9ce35a8c5c3e01906b4f1fcd7
=> => naming to docker.io/library/getting-started2

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
S C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker build -t getting-started1 .
```

### Los inicializamos

```
01306c7839a0
PS C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker run -dp 127.0.0.1:3000:3000 getting-started1
c379a378b41c61e6b94d0c8373e0a44ea500414f99b32c397e86a27f55de8701
PS C:\Users\raufa\OneDrive\Escritorio\Docker\getting-started-app> docker build -t getting-started .
[+] Building 25.2s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
```

## Imágenes creadas

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3570]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\raufa>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
55d2d420e957   getting-started1                    "docker-entrypoint.s..." 4 minutes ago  Up 4 minutes  3000/tcp, 127.0.0.1:4000->4000/tcp
32874b07a071   getting-started1                    "docker-entrypoint.s..." 12 minutes ago Up 12 minutes  127.0.0.1:3000->3000/tcp
vibrant_nash
```

## Código

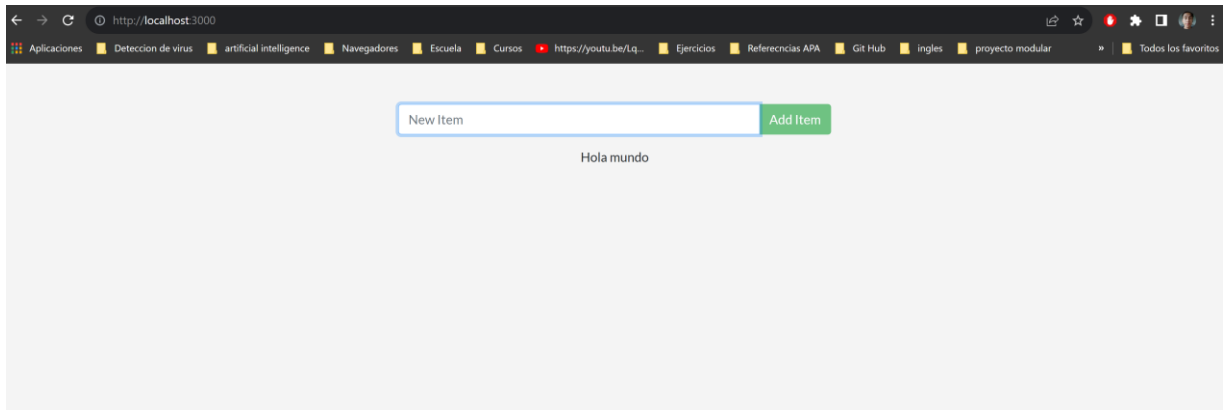
```
EXPLORER
DOCKER
  getting-started-app
    spec
    src
      persistence
      routes
      static
      css
        font-awesome
        # bootstrap.min.css
        # styles.css
      js
        app.js
        babel.min.js
        react-bootstrap.js
        react-dom.production.min.js
        react.production.min.js
        index.html
        index.js
        Dockerfile
        package.json
        README.md
        yarn.lock

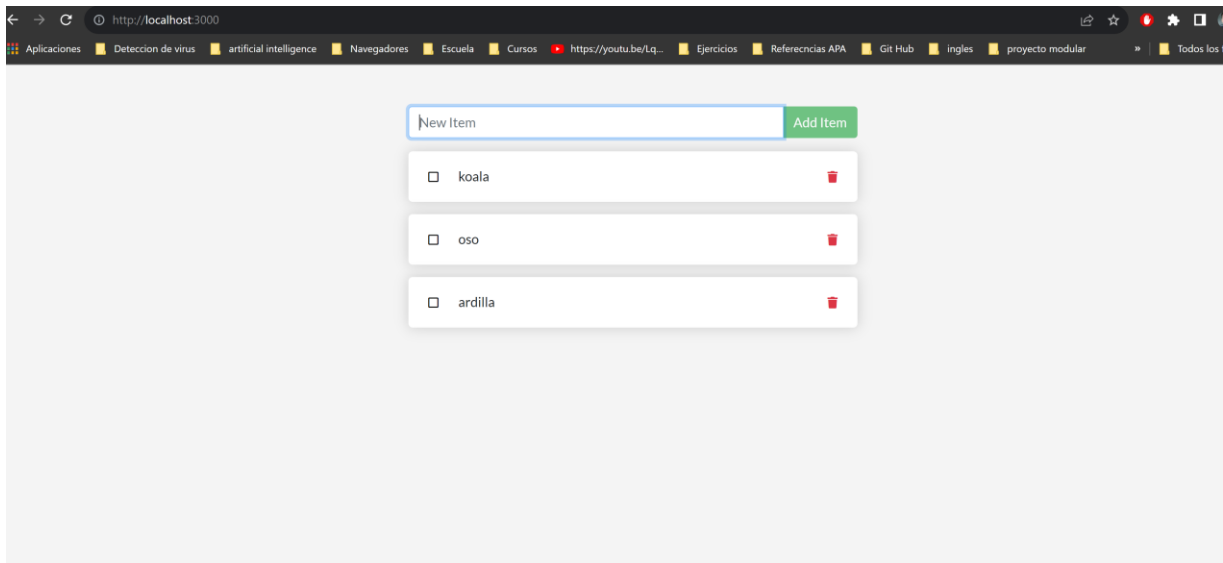
JS app.js M X Dockerfile U
getting-started-app > src > static > js > JS app.js > TodoListCard
1  function App() {
2    const { Container, Row, Col } = ReactBootstrap;
3    return (
4      <Container>
5        <Row>
6          <Col md={ { offset: 3, span: 6 } }>
7            <TodoListCard />
8          </Col>
9        </Row>
10     </Container>
11   );
12 }
13
14 function TodoListCard() {
15   const [items, setItems] = React.useState(null);
16
17   React.useEffect(() => {
18     fetch('/items')
19       .then(r => r.json())
20       .then(setItems);
21   }, []);
22
23   const onNewItem = React.useCallback(
24     newItem => {
```

## Dockfile

```
JS app.js M Dockerfile U X
getting-started-app > Dockerfile > ...
1 # syntax=docker/dockerfile:1
2
3 FROM node:18-alpine
4 WORKDIR /app
5 COPY . .
6 RUN yarn install --production
7 CMD ["node", "src/index.js"]
8 EXPOSE 3000
```

## Aplicación en el puerto 3000





## Conclusión

Docker al principio era un poco difícil de usar, sin embargo, ya he aprendido como funciona y he podido realizar la practica de micro servicios con la creación de los contenedores. Lo bueno de Docker es que con el podre correr mi código en diferentes maquinas o poder compartirlo sin la preocupación de que este en algún momento me llegue a fallas