

Laboratorio N3.

Curso: ST0263 – TET.

Título: Aplicación Web N-Tier Dockerizada

Objetivo: Desplegar una aplicación web dockerizada en entorno de desarrollo que conste de un front end, back end, que acceda recursos en un servicio de persistencia de datos gestionado.

Duración: 45 mins

Tabla de Contenido

1	Introducción.	2
2	Comandos y herramientas para utilizar:	3
3	Recursos.....	3
4	Web Application: Book Store.....	3
4.1	Arquitectura de la Aplicación.....	4
4.2	Vistas de la Aplicación.....	4
4.3	Front End.....	5
4.4	Back End.....	6
4.5	Persistencia de Datos.....	6
4.6	Código del Proyecto.....	6
4.6.1	Descripción del Código del Back End	6
4.6.2	Descripción del Código del Front End.....	7
5	Clonar el Repositorio del Proyecto.....	8
6	Workflow de la Aplicación	8
7	Creando la capa de Persistencia de Bases de Datos: Dynamo DB.....	9
7.1	Creando una Tabla en Dynamo DB	9
7.2	Creando Ítems en la Tabla	9
8	Ejecutando BookStore- WebApp Localmente	11
8.1	Contenerizar las aplicaciones	11
8.2	Instalando Docker Desktop.....	12
8.3	Docker	12
8.4	Docker Compose	14
8.4.1	Creando los Archivos de Variables de Entorno	16
8.4.2	Ejecutando la Aplicación – Modo Desarrollo.....	18
8.4.3	Verificando Funcionalidad.....	19

1 **Introducción.**

Para el caso de este laboratorio, se ha decidido por la opción de desarrollar una aplicación web empleando un estilo arquitectónico por división en capas, específicamente se van a considerar, tres capas: presentación, lógica de negocio y persistencia de datos.

Al respecto, la capa de presentación se desarrolla empleando React.js, para la capa de back end se decidió optar por Node.js y para la capa de persistencia de datos, se empleó, Dynamo DB, el cual es el motor de bases de datos NoSQL de Amazon Web Services (AWS).

Es así como para efectos de su desarrollo, pruebas y despliegue, se va a dockerizar la aplicación y probarla en un entorno de ejecución local empleando Docker y Docker Compose.

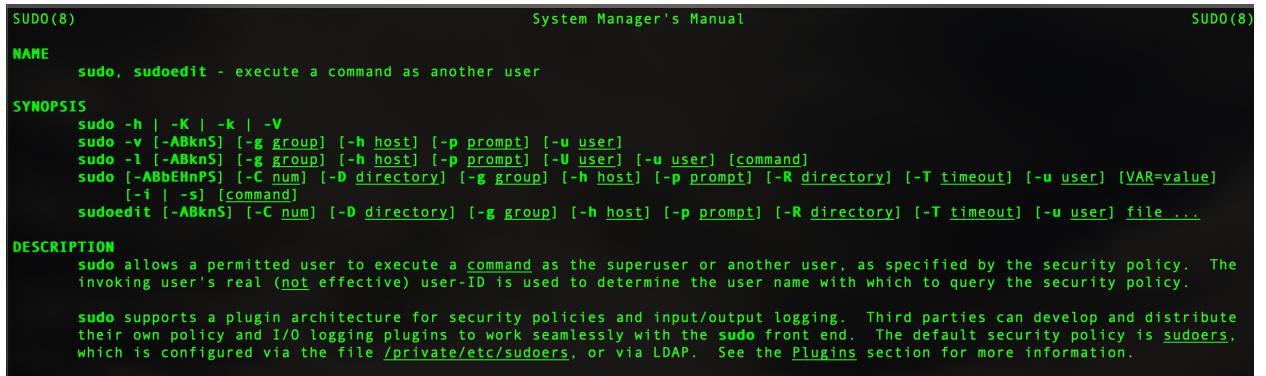
2 Comandos y herramientas para utilizar:

- **Git**: sistema de control de versiones distribuido.

En términos generales puede utilizar las siguientes ayudas para comandos en máquinas Linux:

- **man**: es un comando que permite obtener una descripción de los diferentes comandos. Por ejemplo, si queremos información sobre el comando sudo, digitamos lo siguiente:

```
$ man sudo
```



The screenshot shows a terminal window with the title 'SUDO(8)'. The content is the System Manager's Manual for the sudo command. It includes sections for NAME, SYNOPSIS, and DESCRIPTION, detailing the command's usage and security features.

```
SUDO(8)                               System Manager's Manual                               SUDO(8)

NAME      sudo, sudoedit - execute a command as another user

SYNOPSIS
        sudo -h | -K | -k | -V
        sudo -v [-ABknS] [-g group] [-h host] [-p prompt] [-u user]
        sudo -l [-ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
        sudo [-ABBEHnPS] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-T timeout] [-u user] [VAR=value]
              [-i | -s] [command]
        sudoedit [-ABknS] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-T timeout] [-u user] file ...

DESCRIPTION
        sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy. The invoking user's real (not effective) user-ID is used to determine the user name with which to query the security policy.

        sudo supports a plugin architecture for security policies and input/output logging. Third parties can develop and distribute their own policy and I/O logging plugins to work seamlessly with the sudo front end. The default security policy is sudoers, which is configured via the file /private/etc/sudoers, or via LDAP. See the Plugins section for more information.
```

- En la siguiente url: <https://linux.die.net/man/> puede encontrar una referencia web para los comandos en Linux.

3 Recursos

Para el desarrollo del siguiente laboratorio se dispondrán de los siguientes recursos:

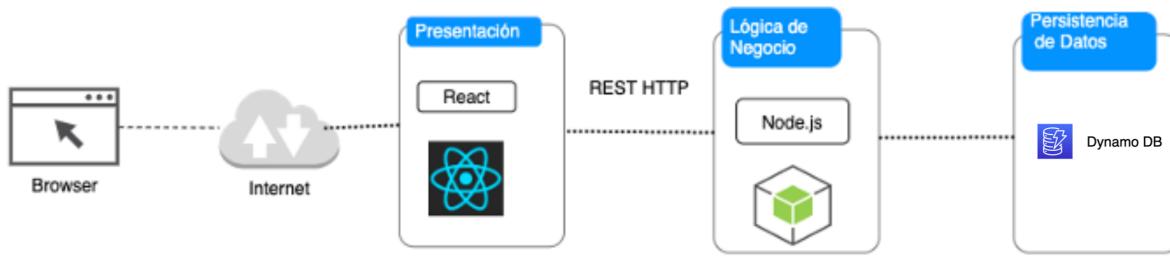
- Docker
- Docker Compose
- Dynamo DB

4 Web Application: Book Store

4.1 Arquitectura de la Aplicación

La aplicación web Book Store implementa un estilo arquitectónico por división en capas (layers). Es así como se tienen las siguientes capas:

- Capa de presentación
- Capa de lógica de negocio
- Capa de persistencia de datos.



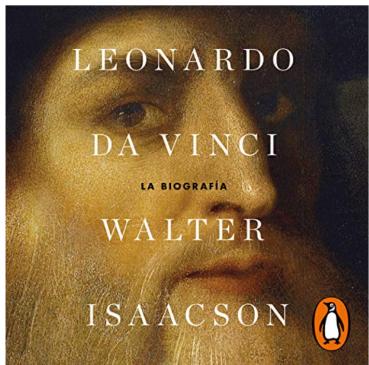
Es importante mencionar que tanto la aplicación del front end (react) y la del back end, cada uno emplea un estilo arquitectónico monolítico y por componentes.

4.2 Vistas de la Aplicación

La aplicación permite visualizar una colección de recursos, para efectos de este caso, libros. Igualmente, cuando el usuario selecciona alguno de los recursos, se ofrece una vista con información detallada sobre el recurso seleccionado. La información de los recursos (libros) se encuentra almacenada en base de datos. La aplicación tiene tres (vistas): raíz (“/”, home), descripción detallada de los libros y acerca de.

La captura de pantalla muestra la interfaz de usuario de la aplicación Book Store. En la parte superior, hay un menú con los ítems "MI TIENDA DE LIBROS" (en un cuadro oscuro), "INICIO" y "ACERCA DE". Abajo de esto, se titula "CATÁLOGO DE LIBROS". Se muestran cuatro tarjetas de libro:

- Inteligencia Genial** (Leonardo da Vinci)
\$30.000.00
- Leonardo Davinci: La Biografía** (WALTER ISAACSON)
\$50.000.00
- Las Meditaciones de Marco Aurelio**
\$50.000.00
- Henry Kissinger LIDERAZGO**
\$99.000.00

[REGRESAR AL CATÁLOGO](#)

LEONARDO DAVINCI: LA BIOGRAFIA

Autor: Walter Isaacson

Descripción: Basándose en las miles de páginas de los cuadernos manuscritos de Leonardo y nuevos descubrimientos sobre su vida y su obra, Walter Isaacson teje una narración que conecta el arte de Da Vinci con sus investigaciones científicas, y nos muestra cómo el genio del hombre más visionario de la historia nació de habilidades que todos poseemos y podemos estimular, tales como la curiosidad incansable, la observación cuidadosa y la imaginación juguetona. Su creatividad, como la de todo gran innovador, resultó de la intersección entre la tecnología y las humanidades. Despelajeó y estudió el rostro de numerosos cadáveres, dibujó los músculos que configuran el movimiento de los labios y pintó la sonrisa más enigmática de la historia, la de la Mona Lisa. Exploró las leyes de la óptica, demostró como la luz incida en la córnea y logró producir esa ilusión de profundidad en la Última cena.

Estado: Disponible (2) uds

Precio: \$50.000.oo

Todos los derechos reservados © MyCompanyStore

Este sitio web está concebido como una tienda virtual de obras literarias, la cual permite obtener el listado de obras disponibles para reservar y/o comprar. El sitio fue desarrollado para que los usuarios pueda obtener información útil acerca de...

4.3 Front End.

Esta capa se encarga de todos los aspectos relacionados con la interfaz de usuario. A esta capa se accede a través del browser. Igualmente, se encarga de la comunicación con el back end, en este caso, utilizando una API REST (empleando el paquete axios).

Particularmente en este laboratorio, la capa de presentación o front end se encuentra desarrollado utilizando React. Ésta se define como una librería de java script para desarrollar la capa de presentación. En este sentido, la principal función de react es renderizar código html en el browser. Para efectos del manejo de estilos (CSS), se empleó react-bootstrap

4.4 Back End.

Esta es la capa de la mitad en la cual se ejecuta y lleva a cabo la lógica de negocio. El back end se encuentra desarrollado empleando tecnología de scripting del lado del servidor, para este caso node.js. Igualmente utilizamos un framework de desarrollo de aplicaciones web denominado express.

4.5 Persistencia de Datos

En esta capa es donde la información es gestionada y almacenada. En esta aplicación, los datos se almacenan en un motor de bases de datos NoSQL, específicamente Dynamo DB.

4.6 Código del Proyecto.

En esta sección se describe la estructura del proyecto.

```
└── backend  
└── frontend  
└── README.md
```

Como se puede observar el proyecto se compone de dos grandes carpetas: backend y frontend. La carpeta del back es un proyecto desarrollado en node.js mientras que el front es desarrollado en React.js.

Finalmente, el archivo de README.md que es un archivo con la información del proyecto.

4.6.1 Descripción del Código del Back End

```
└── config  
└── controller  
└── package-lock.json  
└── package.json  
└── routes  
└── server.js
```

Como se puede apreciar en la estructura del directorio **backend**, se observan los siguientes directorios y archivos:

- Directorios (en negrita):
 - **config**: en esta carpeta se encuentra el archivo para realizar la conexión con la base de datos.
 - **controller**: en esta carpeta se encuentra el archivo para gestionar la conexión y consultas a la base de datos.
 - **routes**: en esta carpeta se encuentra el archivo encargado de gestionar la ruta adecuada acorde al path solicitado.
- Archivos:
 - Server.js: archivo principal de la aplicación de back end.

- package.json - package-lock.json: archivos de dependencias del proyecto node.

4.6.2 Descripción del Código del Front End

```

├── README.md
├── nginx.conf
├── package-lock.json
├── package.json
└── public
    └── src

```

Como se puede apreciar en la estructura del directorio **frontend**, se observan los siguientes directorios y archivos:

- Directorios:
 - **public**: en este directorio esta creada la carpeta imágenes, la cual contiene las imágenes alusivas a cada modelo.
 - **src**: en este directorio se encuentran todos los archivos fuentes creados para el proyecto de React. Es importante mencionar que esta librería para el desarrollo de sus proyectos se basa en el concepto de componentes reutilizables, por esto se observa esta carpeta (**components**). De igual forma, se puede visualizar el subdirectorio de **screens**, donde se encuentran las diferentes vistas. En el subdirectorio **public**, ubicamos las imágenes que tenemos para cada modelo.

```

├── App.js
├── bootstrap.min.css
└── components
    ├── Footer.js
    ├── Header.js
    └── Model.js
    └── index.css
    └── index.js
└── screens
    ├── AboutScreen.js
    ├── HomeScreen.js
    └── ModelScreen.js

```

- Archivos:
 - nginx.conf: En este archivo se encuentra la estructura del archivo de configuración para el servidor web nginx. Este archivo es pasado a la imagen de Docker de nginx al momento de creación de la imagen.

```

server {
    listen 80;

    location / {
        root /usr/share/nginx/html/;
        include /etc/nginx/mime.types;
        try_files $uri $uri/ /index.html;
    }

    location /api/books {
        proxy_pass http://models_backend:5001/api/books;
    }
}

```

- package-lock.json
- package.json
- README.md

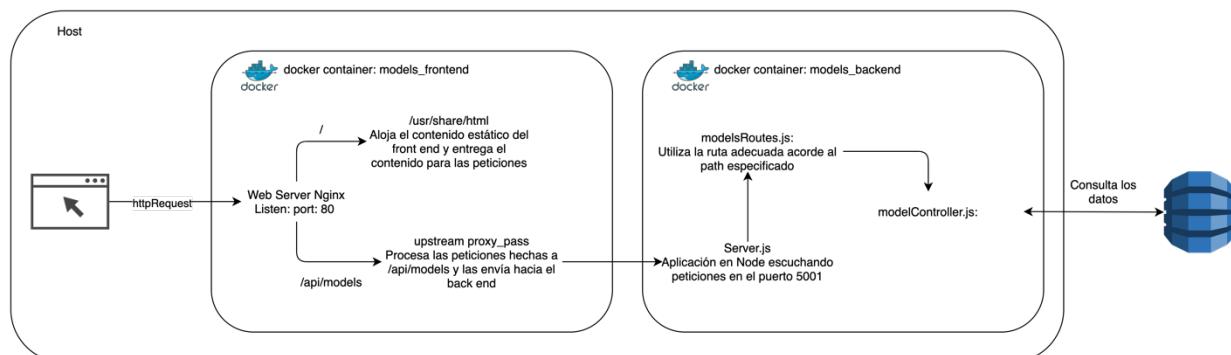
5 Clonar el Repositorio del Proyecto.

En este punto se procede a clonar el repositorio del proyecto a su estación de trabajo local. Para esto emplee y ejecute el siguiente comando. Recuerde que debe tener instalado Git para poder realizar esta operación.

```
$ git clone https://github.com/Course-ST0911/bookstore.git
```

6 Workflow de la Aplicación

En el contexto de este laboratorio, el workflow de la aplicación funcionará acorde a lo que se observa en la figura. Toda la aplicación se ejecutará de manera local a través del uso de contenedores. Para esto, emplearemos, Docker y Docker compose.



Desde el browser se realiza una petición, ésta es atendida por front end, específicamente, por el web server (nginx) que se ejecuta en éste y el cual aloja el contenido estático de la aplicación. En este punto es importante mencionar que nginx funcionará tanto como web server así como proxy inverso. Ahora, cuando se realiza una petición a la API que se ejecuta en el back end, la petición es atendida, también por

el web server. Aquí el web server (nginx) opera como proxy inverso y a través de la directiva proxy_pass, se envía la petición al back end, el cual está escuchando las peticiones en el puerto 5001. La petición se procesa y se realiza la consulta a la base de datos. Se obtiene los datos, y se envían devuelta al cliente.

7 Creando la capa de Persistencia de Bases de Datos: Dynamo DB

7.1 Creando una Tabla en Dynamo DB

Para efectos de este laboratorio, vamos a crear una tabla en Dynamo DB que va a ser la encargada de mantener la información de los modelos que se visualizan en el sitio web.

- En la consola de AWS, por favor busque el servicio de Dynamo DB.
- Click en Create table.
- **Table Details:**
 - Table Name: **tb_books**
 - Partition key: **id**
 - Seleccione que el tipo de dato para id es: **String**
 - Click en **Create table**

Al final debe observar que se creó la tabla “tb_books” de manera correcta:

Tables (1) <small>Info</small>												
	Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode	Total size	Table class	Actions	Delete
<input type="checkbox"/>	tb_books	Active	id (\$)	-	0	Off	Provisioned (5)	Provisioned (5)	0 bytes	Standard	Create table	

7.2 Creando ítems en la Tabla

En este punto se agregan los ítems a la tabla creada anteriormente. Para esto ejecute los siguientes pasos:

- Click en la tabla “tb_books”
- Click en “Explore table ítems”
- Click en Create item.

Agregue los atributos, así como la información de cada uno tal como se ilustra en la figura:

DynamoDB > Explore items: tb_books > Create item

Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attributes		Type	Add new attribute ▾
Attribute name	Value	Type	
id - Partition key	1	String	
name	Leonardo Davinci: La Biografia	String	Remove
image	/images/img-ld-labiografia.jpeg	String	Remove
author	Walter Issacson	String	Remove
description	Basándose en las miles de páginas de los cuadernos manuscritos de Leonardo y nuevos descubrimientos sobre su vida y su obra, Walter Isaacson teje una narración que conecta el arte de Da Vinci con sus investigaciones científicas, y nos muestra cómo el genio del hombre más visionario de la historia nació	String	Remove
countInStock	2	String	Remove
price	\$50.000.00	String	Remove

[Cancel](#) [Create item](#)

- Click en Create Item
- Al final debe observar lo siguiente:

DynamoDB > Explore items > tb_books

tb_books

Scan or query items

Completed. Read capacity units consumed: 0.5

Items returned (1)

	id (String)	author	countInStock	description	image	name	price
1	1	Walter Issac...	2	Basándose en...	/images/im...	Leonardo D...	\$50.000.00

```
{
  name: 'Leonardo Davinci: La Biografia',
  image: '/images/img-ld-labiografia.jpeg',
  author: 'Walter Issacson',
  description:'Basándose en las miles de páginas de los cuadernos manuscritos de Leonardo y nuevos descubrimientos sobre su vida y su obra, Walter Isaacson teje una narración que conecta el arte de Da Vinci con sus investigaciones científicas, y nos muestra cómo el genio del hombre más visionario de la historia nació de habilidades que todos poseemos y podemos estimular, tales como la curiosidad incansable, la observación cuidadosa y la imaginación juguetona. Su creatividad, como la de todo gran innovador, resultó de la intersección entre la tecnología y las humanidades. Despellejó y estudió el rostro de numerosos cadáveres, dibujó los músculos que configuran el movimiento de los labios y pintó la sonrisa'
```

más enigmática de la historia, la de la Mona Lisa. Exploró las leyes de la óptica, demostró como la luz incidía en la córnea y logró producir esa ilusión de profundidad en la Última cena.',
 countInStock: '2',
 price: '\$50.000.oo',
},
{
 name: 'Inteligencia Genial',
 image: '/images/img-ld-inteligenciagenial.jpeg',
 author: 'Michael Gelb',
 description: 'El que, para muchos, ha sido el mayor genio de todos los tiempos, Leonardo da Vinci, puede servir de inspiración a todo aquel que quiera desarrollar al máximo su potencial intelectual y su creatividad. Paso a paso, mediante ejercicios, técnicas y lecciones, este libro muestra el camino para ampliar los horizontes de la mente',
 countInStock: '3',
 price: '\$30.000.oo',
},
{
 name: 'Liderazgo: Seis estudios sobre estrategia mundial',
 image: '/images/img-hk-liderazgo.jpeg',
 author: 'Henry Kissinger',
 description: 'Henry Kissinger, uno de los principales estrategas políticos del siglo xx, analiza en este nuevo libro los perfiles de seis de los líderes mundiales más fascinantes e influyentes del pasado reciente: Konrad Adenauer, Charles de Gaulle, Richard Nixon, Anwar Sadat, Lee Kuan Yew y Margaret Thatcher. Todos ellos se formaron en un periodo en el que las instituciones establecidas se derrumbaban en Europa, las estructuras coloniales daban paso a estados independientes en Asia y África y hubo que crear un nuevo orden internacional a partir de los vestigios del anterior.',
 countInStock: '1',
 price: '\$99.000.oo',
},

Nota:

- Al final puede observar que la tabla contiene dos ítems cuya primary key es "id" que es de tipo string.
- Puede agregar muchos más ítems a su consideración. Por favor agregue la imagen alusiva a cada libro en la carpeta ./public/images/ ubicada en el proyecto en el directorio frontend

8 Ejecutando BookStore- WebApp Localmente

8.1 Contenerizar las aplicaciones

Dentro de los beneficios de usar contenedores para el desarrollo de aplicaciones es que a través de esta tecnología se permite el uso de un entorno de desarrollo consistente para todos los miembros del equipo. Es así como todos los desarrolladores emplearan el mismo OS, las mismas librerías, etc, independiente del sistema operativo que tenga la estación del desarrollador. Al respecto, al contenerizar las aplicaciones para desarrollo, evita al desarrollador el tener que instalar de manera nativa múltiples lenguajes de programación, así como versiones de estos.

Por otro lado, otra de las ventajas de los contenedores en ambientes de desarrollo, es que nos permite tener toda la arquitectura de la aplicación ejecutándose de manera local en un equipo de escritorio/portátil. De esta forma, vamos a poder probar de manera local nuestra aplicación, es decir, ejecutar el front y el back en nuestra máquina local.

8.2 Instalando Docker Desktop

Es así como para ver esta utilidad, vamos a emplear como en primera instancia, se procede con los siguientes pasos:

- Descargar e instalar Docker desktop para su sistema operativo: [Docker Desktop: The #1 Containerization Tool for Developers | Docker](#)
- Ejecute la aplicación docker desktop.

8.3 Docker

Docker es una plataforma que permite el crear, implementar y desplegar contenedores. Recuerde que un contenedor se puede entender como un componente estandarizado que combina el código fuente de la aplicación con las diferentes librerías y dependencias que éste requiere para su ejecución.

Un Docker se crea a partir de un archivo Dockerfile. En el presente laboratorio se tienen un dockerfile tanto para el frontend como para el backend. De esta forma se automatiza la creación de la imagen del contenedor. Este archivo consta de un conjunto de instrucciones/comandos que el “engine” de Docker procesará para la creación de la imagen.

Nota: Se debe crear un archivo con el nombre **Dockerfile** en la carpeta frontend.

Dockerfile (Frontend):

```
FROM node:16-alpine as builder

# Create the app directory
WORKDIR /app

# Copy app files
COPY . .

# Install the required dependencies
RUN npm ci

# Build the front end app
RUN npm run build

# Bundle static assets with nginx
FROM nginx:1.21.0-alpine as development
```

```

# Copy built assets from `builder` image
COPY --from=builder /app/build /usr/share/nginx/html

# Add your nginx.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Expose port
EXPOSE 80

# Start and run the nginx
CMD ["nginx", "-g", "daemon off;"]

```

- From: proporciona una imagen base para la imagen del contenedor.
- Workdir: indica el directorio en el Docker sobre el cual se van a aplicar los comandos que siguen. En este caso, se estable que el directorio de la aplicación es /app.
- Copy: permite realizar la copia de archivos desde nuestra máquina local a la imagen del contenedor. En este caso, copia los archivos fuentes del proyecto.
- Run: ejecuta el comando que se indica. En este caso, se crea el código a distribuir para la aplicación front end.
- Expose: le indica al contenedor en que puerto va a escuchar peticiones. En este caso, el puerto 80.
- CMD: se emplea para ejecutar un comando específico en el contenedor. En este caso la ejecución de nginx.

Nota: Se debe crear un archivo con el nombre **Dockerfile** en la carpeta backend.

Dockerfile (Backend)

```

FROM node:lts-buster

# Create app directory
WORKDIR /usr/src/app

# Copy dependency definitions
COPY package.json ./package.json
COPY package-lock.json ./package-lock.json

# Install the required dependencies
RUN npm ci

# Get all the code needed to run the app
COPY . .

# Expose the port the app runs in
EXPOSE 5001

```

```
# Serve the app
CMD ["npm", "start"]
```

- From: proporciona una imagen base para la imagen del contenedor.
- Workdir: indica el directorio en el Docker sobre el cual se van a aplicar los comandos que siguen. En este caso, se estable que el directorio de la aplicación es /usr/src/app.
- Copy: permite realizar la copia de archivos desde nuestra máquina local a la imagen del contenedor. En este caso, copia los archivos de dependencias del proyecto node (package.json y package-lock.json)
- Run: ejecuta el comando que se indica para la instalación de las dependencias.
- Expose: le indica al contenedor en que puerto va a escuchar peticiones. En este caso, el puerto 5001.
- CMD: se emplea para ejecutar un comando específico en el contenedor. En este caso la ejecución de node.js.

8.4 Docker Compose

Docker compose es una herramienta para la gestión de aplicaciones distribuidas que se utiliza para definir, así como ejecutar aplicaciones contenerizadas (dockerizadas), las cuales se componen de varios contenedores (docker). De esta forma, con el uso de esta herramienta, se simplifica el uso de Docker.

Antes de ejecutar el Docker compose, vamos a analizar la estructura e información de este archivo. En términos generales para todo el proyecto contamos con un archivo de Docker compose para el front, uno para el back, y otro que nos permite ejecutar tanto el front como el back conjuntamente. Para efectos de esta sección vamos a trabajar solamente con el archivo denominado: [docker-compose.yml](#).

Nota: Se debe crear un archivo con el nombre [docker-compose.yml](#) en el directorio raíz del proyecto.

A continuación, se presenta el archivo Docker compose para el entorno de desarrollo.

```
services:
  frontend:
    build: ./frontend
    container_name: models_frontend
    ports:
      - 80:80
    restart: always
    env_file:
      - "./frontend/.env"
    networks:
      - front-tier
      - back-tier
  backend:
    build: ./backend
    container_name: models_backend
```

```

ports:
  - 5001:5001
restart: always
env_file:
  - "./backend/.env"
networks:
  - front-tier
  - back-tier
networks:
  front-tier:
  back-tier:

```

- **services:** En esta sección se definen las imágenes de contenedores que se van a crear. Como se puede observar se definen dos servicios: frontend y backend.
 - **Frontend:** este servicio es una aplicación en react.
 - *build*: Se toman los archivos del directorio frontend para realizar la construcción de la imagen Docker.
 - *container_name*: La imagen para el contenedor se denomina models_frontend.
 - *ports*: Indica los puertos y el mapeo de estos en los cuales se va a ejecutar la aplicación en el host y el redireccionamiento que se hace al puerto del contenedor. Para este caso, puerto 80 y se redirige al puerto 80.
 - *restart*: este parámetro configurado a “always” permite que el contenedor siempre se reinicie en caso tal se presente una falla.
 - *env_file*: esto nos permite especificar el archivo del cual se van a leer las variables de entornos. Para este caso, el archivo se denomina “.env”.
 - *networks*: el contenedor se va a conectar a dos redes: frontier y backtier.
 - **Backend:**
 - *build*: Se toman los archivos del directorio backend para realizar la construcción de la imagen Docker.
 - *container_name*: La imagen para el contenedor se denomina models_backend.
 - *ports*: Indica los puertos y el mapeo de estos en los cuales se va a ejecutar la aplicación en el host y el redireccionamiento que se hace al puerto del contenedor. Para este caso, puerto 5001 y se redirige al puerto 5001.
 - *restart*: este parámetro configurado a “always” permite que el contenedor siempre se reinicie en caso tal se presente una falla.
 - *env_file*: esto nos permite especificar el archivo del cual se van a leer las variables de entornos. Para este caso, el archivo se denomina “.env”
 - *networks*: el contenedor se va a conectar a dos redes: frontier y backtier.
- **networks:** permite definir el componente de networking para que las aplicaciones se comuniquen entre sí. Como se evidencia se crean dos redes: front-tier y back-tier.

8.4.1 Creando los Archivos de Variables de Entorno

Como se observó, el archivo de Docker compose toma como parámetros dos archivos de variables de entornos que se denominan y encuentran en: ./frontend/.env (para la aplicación front end al interior de este directorio de trabajo) y ./backend/.env (para la aplicación back end al interior de este directorio de trabajo), uno para cada servicio definido.

- **Nota:** Los archivos *.env realmente no se colocan en los repositorios. Estos archivos, normalmente, tienen información que no debe ser distribuida, como son las variables de entorno.

Al respecto, los archivos .env definidos anteriormente, van a contener información de variables de entorno que se requieren para la aplicación de back como del front. Estos archivos son pasados como parámetros desde el archivo `docker-compose.yml`. Este archivo es el archivo de la herramienta Docker Compose el cual nos permitirá ejecutar los dos contenedores (front y back) en una sola máquina local.

Nota: Se debe crear un archivo con el nombre `.env` en el directorio `/frontend` del proyecto.

Creando .env en el directorio /frontend

Para el caso particular de este laboratorio, para la ejecución de la aplicación correctamente, vamos a requerir algunas variables. Específicamente para el caso del front end, se requiere un parámetro para el servidor web NGINX que se denomina PROXY_PASS así como el puerto en el cual se va a ejecutar. A continuación, la definición de las variables de entorno para el front end:

```
PROXY_PASS=http://models_backend:5001/api/models  
NGINX_PORT=80
```

Recuerde que el NGINX entrega la aplicación del front a los clientes (browsers). En el momento en que los clientes realizan una petición, se la pasan al contenedor que ejecuta el backend (`models_backend`) el cual está escuchando peticiones en el puerto 5001.

Nota: Se debe crear un archivo con el nombre `.env` en el subdirectorio frontend del proyecto.

Creando .env en el directorio backend

Para el caso particular de este laboratorio, para la ejecución de la aplicación correctamente, vamos a requerir algunas variables como son si estamos en ambiente de “development” o “production”. De igual manera, vamos a requerir el puerto en que se va a ejecutar la aplicación.

Por otro lado, para poder acceder a los recursos de AWS desde la aplicación backend, vamos a requerir que ésta se autentique. Por tal razón se hace necesario facilitar la región donde se encuentran los servicios que quiero acceder, así como las credenciales de acceso. Al respecto, estas credenciales se van a facilitar como variables de entorno.

A continuación, se encuentran las variables de entorno definidas

```
NODE_ENV=  
PORT=
```

```
AWS_REGION=
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_SESSION_TOKEN=
```

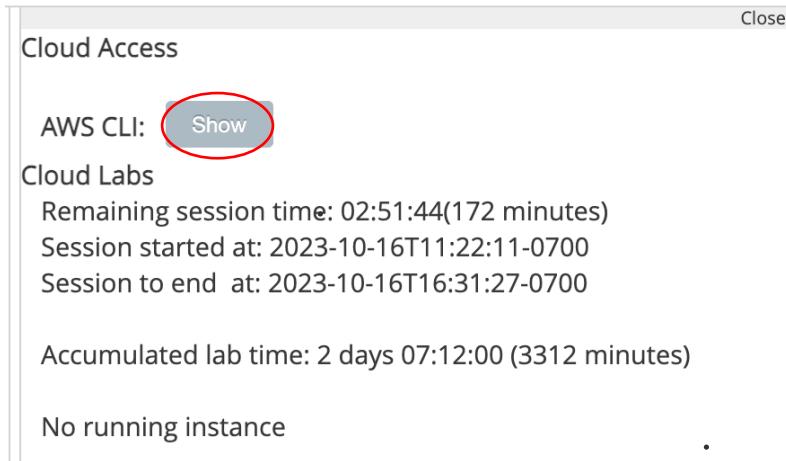
Para efectos de este laboratorio, coloque el siguiente valor:

```
NODE_ENV=development
PORT=5001
AWS_REGION=us-east-1
```

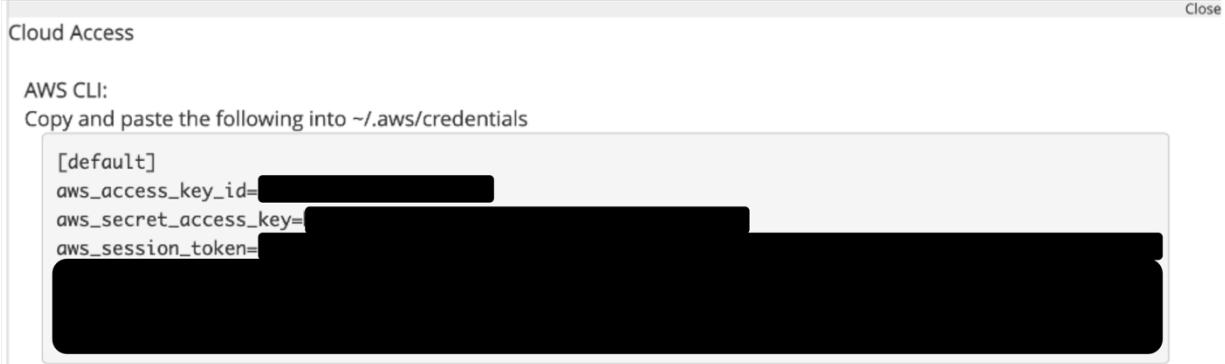
El valor de las credenciales lo obtiene de la siguiente forma. En la consola de administración/acceso de su curso de AWS academy, de click en AWS Details.



Le va a aparecer la siguiente pantalla y va a dar click en Show.



Una vez de click, va a poder observar el valor de estas variables. Por favor, copiar y pegar el valor en su archivo de backend.env.



Nota: Recuerde que esta información es sensible y no se debe compartir. Igualmente, cada vez que expira la sesión de AWS Academy (4:00 horas), se actualizan estos valores. Por tal razón, los debe cambiar, en caso tal sea necesario.

Nota: Se debe crear un archivo con el nombre **.env** en el subdirectorio backend del proyecto.

Al final, la estructura de archivos de su proyecto debe observarse de la siguiente forma:

```
.
├── README.md
├── backend
│   ├── Dockerfile
│   ├── config
│   ├── controller
│   ├── package-lock.json
│   ├── package.json
│   ├── routes
│   └── server.js
└── docker-compose.yml
└── frontend
    ├── Dockerfile
    ├── README.md
    ├── nginx.conf
    ├── package-lock.json
    ├── package.json
    ├── public
    └── src
```

8.4.2 Ejecutando la Aplicación – Modo Desarrollo

Para ejecutar la aplicación, vamos a emplear la utilidad Docker compose como se mencionó anteriormente. Ahora se procede a ejecutar el Docker compose. Para ejecutar este comando, debe estar ubicado en el directorio que contiene el archivo docker-compose.yml.

Ahora, ejecute el siguiente comando:

```
$ docker compose docker-compose.yml up -d
```

Tenga presente que al momento de ejecutar el docker compose, este busca los archivos Dockerfile en cada carpeta (tanto para el front como el back) y las imágenes se construyen a partir de los comandos especificados en cada Dockerfile.

8.4.3 Verificando Funcionalidad

Se puede verificar la funcionalidad observando el estado de las imágenes, así como de los contenedores. Igualmente verificando que la aplicación funcione correctamente.

Para verificar las imágenes creadas:

```
$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
bookstore2024-backend	latest	fdb9686eae84	6 minutes ago	1.11GB
bookstore2024-frontend	latest	2aa3b331448f	31 minutes ago	24.1MB

Como se puede apreciar, se crearon las dos imágenes tanto para el frontend y de backend. Igual lo puede observar en la herramienta Docker desktop en el menú de images:

The screenshot shows the Docker Desktop interface under the 'Images' tab. It displays two images: 'bookstore2024-backend' (latest tag, fdb9686eae84, created 8 minutes ago, 1.1 GB) and 'bookstore2024-frontend' (latest tag, 2aa3b331448f, created 33 minutes ago, 24.12 MB). The interface includes a search bar, filter buttons for Local, Hub, and Artifactory, and an 'EARLY ACCESS' button. The status bar at the bottom indicates 'Last refresh: 9 hours ago'.

Si se desea observar los contenedores que se están ejecutando, digite el comando:

```
$ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7b909b4be5cb	bookstore2024-frontend	"/docker-entrypoint..."	7 minutes ago	Up 7 minutes	0.0.0.0:80->80/tcp	models_frontend
aae43b632335	bookstore2024-backend	"docker-entrypoint.s..."	7 minutes ago	Up 7 minutes	0.0.0.0:5001->5001/tcp	models_backend

Igual lo puede observar en la herramienta Docker desktop en el menú de containers:

Containers		Give feedback			
Container CPU usage		Container memory usage		Show charts	
0.00%	/ 1000% (10 cores allocated)	55.74MB	/ 7.59GB	Show charts	▼
<input type="text"/> Search		<input checked="" type="radio"/>	Only show running containers	<input type="button" value="Delete"/>	<input type="button" value=""/>
Name	Image	Status	CPU (%)	Port(s)	Last started
bookstore2024		Running (2/2)	0%	6 minutes ago	<input type="button" value=""/>
models_frontend	bookstore2024-frontend 7599b4be5cb	Running	0%	80:80	6 minutes ago
models_backend	bookstore2024-backend aae43b632335	Running	0%	5001:5001	6 minutes ago

Para verificar el funcionamiento de la aplicación en el entorno local, por favor abra un browser y digite lo siguiente: <http://localhost/>

The screenshot shows a web browser window with the URL "localhost" in the address bar. The page title is "MI TIENDA DE LIBROS". Below the title, there is a section titled "CATÁLOGO DE LIBROS" featuring four book cards:

- Inteligencia Genial** by Leonardo da Vinci (Author), Walter Isaacson (Editor). Price: \$30.000.00.
- Leonardo Da Vinci: La Biografía** by Walter Isaacson. Price: \$50.000.00.
- Las Meditaciones de Marco Aurelio** by Marco Aurelio. Price: \$50.000.00.
- Liderazgo: Seis estudios sobre estrategia mundial** by Henry Kissinger. Price: \$99.000.00.

De click en el cualquier libro:

localhost/book/2

MI TIENDA DE LIBROS

INICIO ACERCA DE

REGRESAR AL CATÁLOGO

INTELIGENCIA GENIAL

Estado: Disponible (3) uds

Precio: \$30.000,00

Autor: Michael Gelb

Descripción: El que, para muchos, ha sido el mayor genio de todos los tiempos, Leonardo da Vinci, puede servir de inspiración a todo aquel que quiera desarrollar al máximo su potencial intelectual y su creatividad. Paso a paso, mediante ejercicios, técnicas y lecciones, este libro muestra el camino para ampliar los horizontes de la mente.

En este momento la aplicación está ejecutando el contenedor del front, back y conectándose a Dynamo de manera local.

Igualmente, puedes verificar el funcionamiento de la aplicación backend dockerizada a través de la siguiente URL <http://localhost:5001/api/books/>.

```
localhost:5001/api/books/
1 |   {
2 |     "image": "/images/img-ld-inteligenciagenial.jpeg",
3 |     "countInStock": "3",
4 |     "price": "$30.000,00",
5 |     "description": "El que, para muchos, ha sido el mayor genio de todos los tiempos, Leonardo da Vinci, puede servir de inspiración a todo aquel que quiera desarrollar al máximo su potencial intelectual y su creatividad. Paso a paso, mediante ejercicios, técnicas y lecciones, este libro muestra el camino para ampliar los horizontes de la mente.",
6 |     "id": "2",
7 |     "name": "Inteligencia Genial",
8 |     "author": "Michael Gelb"
9 |   },
10 |   {
11 |     "image": "/images/img-ld-labiografia.jpeg",
12 |     "countInStock": "2",
13 |     "price": "$50.000,00",
14 |     "description": "Basándose en las miles de páginas de los cuadernos manuscritos de Leonardo y nuevos descubrimientos sobre su vida y su obra, Walter Isaacson teje una narración que conecta el arte de Da Vinci con sus inventos, su científico y matemático genio y su increíble habilidad para observar y describir el mundo que lo rodeaba. Su creatividad, como la de todo gran innovador, resultó de la intersección entre la tecnología y las humanidades. Describió y estudió el rostro de numerosos cadáveres, dibujó los músculos que configuran el movimiento de los labios y pintó la sonrisa más enigmática de la historia, la de La Mona Lisa. Exploró las leyes de la óptica, demostró como la luz incidía en la cínea y logró producir esa ilusión de profundidad en la Última cena.",
15 |     "author": "Walter Isaacson"
16 |   },
17 |   {
18 |     "image": "/images/img-ma-meditaciones.jpeg",
19 |     "countInStock": "1",
20 |     "price": "$30.000,00",
21 |     "description": "Las Meditaciones del gran emperador-filósofo romano Marco Aurelio son sencillas aunque profundas obras de filosofía estoica que, a día de hoy, continúan ofreciendo a muchos orientación y consuelo con su elocuencia, sabiduría y bondad. A lo largo de la historia, algunos libros han cambiado el mundo. Han transformado la manera en que nos vemos a nosotros mismos y a los demás. Han inspirado el debate, la guerra, la revolución, la humanidad, indignado, provocado y consolado. Han enriquecido vidas, y también las han destruido. Taurus publica las obras de los grandes pensadores, pioneros, radicales y visionarios cuyas ideas sacudieron la civilización y nos impulsaron a ser quienes somos.",
22 |     "id": "3",
23 |     "name": "Las Meditaciones de Marco Aurelio",
24 |     "author": "Marco Aurelio"
25 |   },
26 |   {
27 |     "image": "/images/img-hk-liderazgo.jpeg",
28 |     "countInStock": "1",
29 |     "price": "$30.000,00",
30 |     "description": "Henry Kissinger, uno de los principales estrategas políticos del siglo xx, analiza en este nuevo libro los perfiles de seis de los líderes mundiales más fascinantes e influyentes del pasado reciente: Konrad Adenauer, Charles de Gaulle, Richard Nixon, Anwar Sadat, Lee Kuan Yew y Margaret Thatcher. Todos ellos se formaron en un período en el que las instituciones establecidas se derrumbaban en las estremecidas culturas europeas y americanas, estando permanentemente en conflicto entre la tradición y la modernidad. De Gaulle partió contra la Francia postmaterial, la rehabilitación llevada a cabo por Adenauer de una Alemania devastada por la guerra y el éxito del experimento de la pequeña ciudad Estado de Lee Kuan Yew en Singapur. El análisis de estos procesos sirve para mostrar las estrategias de gobierno de unos líderes que, impulsados por un alto sentido de Estado, se propusieron posicionar a sus respectivos países en el centro del mundo. Una perspectiva que no tiene parangón: es la de un historiador de primer orden que conoció y estuvo implicado en los acontecimientos que se relatan. La experiencia como alto representante público, el conocimiento personal de los protagonistas y la carrera política de Kissinger enriquecen un libro que atestigua como la combinación del carácter de los personajes y las circunstancias de cada situación es lo que acaba dando forma a la historia."
31 |     "id": "4",
32 |     "name": "Liderazgo: Seis estudios sobre estrategia mundial",
33 |     "author": "Henry Kissinger"
34 |   }
35 | }
```