Práctica final seminario Docker

El proyecto consiste en una app que expone las siguientes urls:

- /
- /set1
- /set2

Está desarrollado con un framework llamado NextJS (desarrollado con el lenguaje Javascript) que se conecta con un servicio MongoDB.

Ejercicio 1 - Construir un Dockerfile con las siguientes características:

- La imagen base tiene que ser **`node`**
- Ejecutar el comando **`npm i -g @nestjs/cli`**
- El contenido del proyecto debe ser copiado a un folder llamado "**`myapp`**" dentro de la imágen.
- En la carpeta donde se copió el proyecto, se debe ejecutar el comando "**`npm install`**". (Ignorar los WARN cuando se ejecuta este comando)
- Se debe configurar la imágen (con una sentencia en el Dockerfile) de manera que cuando se inicialice un container, se ejecute el siguiente comando (en la carpeta donde se copió el proyecto): **`npm run start:dev`**

Validación del ejercicio

La imágen construida es una aplicación NestJS en el puerto **`3000`**. Para validar que está funcionando correctamente se debe ejecutar un contenedor (con el comando docker de línea de comando, sin utilizar de momento docker-compose) utilizando la imágen construida y realizando el mapeo de puerto correcto. Una vez ejecutada esta aplicación con un navegador acceder a **`localhost:3000`** esperar entre 10 y 20 segundos y obtener un mensaje de error similar al siguiente:

```
{"statusCode":500,"message":"Internal server error"}
```

Ejercicio 2 - Construir un docker-compose

Construir un docker compose que construya la imágen del ejercicio 1 y agregue un servicio Mongo. La imagen del servicio mongo es **`mongo`**. El servicio mongo necesita 2 variables de ambiente `MONGO_INITDB_ROOT_USERNAME` y `MONGO_INITDB_ROOT_PASSWORD`. Setear ambas en `root.` Adicionalmente requiere el mapeo de puerto `27017`. (Validar que la uri en la línea 9 del archivo app.module.ts coincida con los datos definidos para mongo en el docker-compose)

Validación del ejercicio

Para validar el ejercicio realizar los siguientes tests:

1. Abrir la url: **`localhost:3000/set1`**
2. Abrir la url: **`localhost:3000/`** y validar que muestra Valor 1.
3. Abrir la url: **`localhost:3000/set2`**.
4. Abrir la url: **`localhost:3000/`** y validar que muestra Valor 2.

Ejercicio 3 - Montar un volumen

Modificar los ejercicios anteriores para que el proyecto no sea copiado dentro de la imágen construida sino que el código fuente sea mapeado de una carpeta local.

Validación del ejercicio

1. Construir y ejecutar los containers utilizando docker compose.
2. Abrir la url: **`localhost:3000/`** y ver que el mensaje sea: "Valor: x"
3. Sin detener los contenedores, modificar el archivo `app.controllers.ts (linea 12)` y cambiar la palabra "Valor" por "Dato".
4. Recargar la url: **`localhost:3000/`** y verificar que ahora el mensaje sea "Dato: x"

Formato de entrega de los ejercicios

- Entregar los ejercicios en 3 carpetas separadas, 1 carpeta por cada ejercicio.
- La carpeta "ejercicio 1" debe contener el código del proyecto y el Dockerfile.
- La carpeta "ejercicio 2" debe contener el código del proyecto, el Dockerfile y el docker compose.
- La carpeta "ejercicio 3" debe contener los mismos archivos que el ejercicio 2, pero modificados para cumplir con el ejercicio 3.