

# dotnet-stack

## EJERCICIO 1



### 0.Red lemoncode-challenge:

Para crear la red bastará con usar el siguiente comando por consola:

```
docker network create lemoncode-challenge
```

### 1.DB some-mongo

Creamos el directorio data:

 data	09/12/2024 20:27	Carpeta de archivos
 dotnet-stack	09/12/2024 20:26	Carpeta de archivos

2. Dentro de data creamos db y creamos el fichero para generar la estructura de la base de datos:

 db	09/12/2024 20:38
 init-mongo.js	19/11/2024 20:20

El archivo tendría este contenido, es importante conservar la estructura ya que serán los datos que el backend y el frontend necesitarán:

```
db = db.getSiblingDB("TopicstoreDb");

db.createCollection("Topics");

db.Topics.insertOne({
  "_id": ObjectId("5fa2ca6abe7a379ec4234883"),
  "topicName": "Contenedores"
});

// Añadir 4 registros más
db.Topics.insertMany([
  {
    "_id": ObjectId("64ab9cdbe6a9172efc12d124"),
    "topicName": "Virtualización"
```

```

    },
    {
      "_id": ObjectId("78b12dc0f5c9a4e3f8112fa6"),
      "topicName": "Kubernetes"
    },
    {
      "_id": ObjectId("8a3c1a2b75c4a0e5d631b12d"),
      "topicName": "Microservicios"
    },
    {
      "_id": ObjectId("93e1bfb9a3d7c5f890b01c4d"),
      "topicName": "DevOps"
    }
  ]
});

```

#### 4. Levantar docker some-mongo:

```

docker run --name some-mongo -d --network lemoncode-challenge -v
C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge\data\init-
mongo.js:/docker-entrypoint-initdb.d/init-mongo.js -v
C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge\data\db:/data/db -p
27017:27017 mongo:4.4

```

## 2.Backend topics-api

Nos situamos en el directorio backend:

Generamos el siguiente dockerfile:

```

# Etapa 1: Construcción de la aplicación

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build

WORKDIR /app

```

```
# Copiar los archivos del proyecto al contenedor
```

```
COPY . .
```

```
# Restaurar las dependencias y publicar la aplicación
```

```
RUN dotnet restore
```

```
RUN dotnet publish -c Release -o out
```

```
# Etapa 2: Imagen de runtime
```

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1
```

```
WORKDIR /app
```

```
# Copiar los archivos publicados desde la etapa de build

COPY --from=build /app/out .

# Exponer el puerto 5000

EXPOSE 5000

# Definir el punto de entrada

ENTRYPOINT ["dotnet", "backend.dll"]
```

Antes de generar la imagen hay que cambiar el connectionString y pasarlo de localhost al nombre del contenedor de mongo ficheros:

El fichero a modificar appsettings.json:

```
{
  "TopicstoreDatabaseSettings": {
    "ConnectionString": "mongodb://some-mongo:27017",
    "TopicsCollectionName": "Topics",
    "DatabaseName": "TopicstoreDb"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

```
    },  
    "AllowedHosts": "*" }  
}
```

En el fichero Topics.cs se debe colocar el elemento con el mismo nombre con el que aparece en la base de datos de mongo db en este caso "topicName":

```
using MongoDB.Bson.Serialization.Attributes;  
  
using MongoDB.Bson;  
  
namespace backend.Models  
{  
  
    public class Topic  
    {  
  
        [BsonId]  
  
        [BsonRepresentation(BsonType.ObjectId)]  
  
        public string Id { get; set; }  
  
  
        [BsonElement("topicName")]  
  
        public string TopicName { get; set; }  
  
    }  
  
}
```

Ahora quedaría crear la imagen

```
docker build -t topics-api ./backend/ --no-cache
```

Levantamos el backend importante usar `-e ASPNETCORE_URLS` ya que esta es necesaria para indicar el puerto de escucha para Kestrel :

```
docker run -d -p 5000:5000 --name topics-api --network lemoncode-challenge -e ASPNETCORE_URLS=http://+:5000 topics-api
```

```
C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge\dotnet-stack>docker logs f4c24e996516
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://[::]:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
```

### Frontend front-api:

Nos situamos en el directorio del frontend y creamos el Dockerfile:

```
# Usa una imagen base de Node.js
```

```
FROM node:18
```

```
# Establece el directorio de trabajo dentro del contenedor
```

```
WORKDIR /usr/src/app
```

```
# Copia los archivos necesarios para instalar dependencias
```

```
COPY package*.json ./
```

```
# Instala las dependencias
```

```
RUN npm install
```

```
# Copia todos los archivos del frontend al contenedor
```

```
COPY . .
```

```
# Expone el puerto en el contenedor
```

```
EXPOSE 8080
```

```
# Comando para iniciar la aplicación
```

```
CMD ["node", "server.js"]
```

Ahora hay que verificar que esté este index.ejs con el nombre del campo que coincida con el backend y mongo db, en este caso topicsName:

```
<!doctype html>

<html lang="en">

<head>

  <!-- Required meta tags -->

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">


  <!-- Bootstrap CSS -->

  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css"
rel="stylesheet"

    integrity="sha384-
KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZ0J3BCsw2P0p/We"
crossorigin="anonymous">


  <title>Lemoncode Challenge</title>

</head>
```



```
<body>

  <div class="container">

    <div class="row">

      <h1>Topics</h1>

    </div>

    <div class="row">

      <ul class="list-group">

        <% topics.forEach(function(topic){ %>

          <li class="list-group-item">

            <%- topic.topicName %>

          </li>

          <%});%>

        </ul>

      </div>

    </div>

    <!-- Option 1: Bootstrap Bundle with Popper -->

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min
.js"

    integrity="sha384-
```

```
U1DAWAznBHeqEILVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFihvj"

    crossorigin="anonymous"></script>

</body>

</html>
```

Seguidamente hacemos el build:

```
docker build -t front-api ./frontend/ --no-cache
```

Levantamos el docker usando la variable de entorno API\_URI la cual venia preconfigurada en el código(server.js) para conectarnos con backend:

```
docker run -d -p 8080:3000 --name front-app --network lemoncode-challenge -e  
API_URI=http://topics-api:5000/api/topics front-api
```

### Comandos básicos:

para listar contenedores :

```
docker ps
```

Parar contenedores:

```
docker stop id_delcontenedor
```

Para eliminar todo una vez parado los contenedores se usa:

```
docker ps -a
```




Con el comando anterior mostrará las ID de los contenedores parados por lo que una vez identificados podemos proceder a borrarlos:

```
docker rm id_del_contenedor1 id_del_contenedor2 id_del_contenedor3
```

## EJERCICIO 2

### Docker-compose

Nos situamos en la carpeta del proyecto y creamos el docker compose:

nombre	fecha de modificacion	tipo	tamaño
 data	09/12/2024 21:51	Carpeta de archivos	
 dotnet-stack	09/12/2024 20:26	Carpeta de archivos	
 docker-compose.yml	12/12/2024 19:12	Archivo de origen ...	2 KB

```
services:
  some-mongo:
    image: mongo:4.4
    container_name: some-mongo
    networks:
      - lemoncode-challenge
    volumes:
      - C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge\data\init-mongo.js:/docker-entrypoint-initdb.d/init-mongo.js
      - C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge\data\db:/data/db
    ports:
      - "27017:27017"

  topics-api:
    build:
      context: ./dotnet-stack/backend # Ruta al directorio donde se encuentra el código del backend
      dockerfile: Dockerfile # Opcional si se llama Dockerfile por defecto
    image: topics-api:1.0
    container_name: topics-api
    networks:
      - lemoncode-challenge
    environment:
      - ASPNETCORE_URLS=http://+:5000
    ports:
      - "5000:5000"

  frontend-app:
    build:
```

```
context: ../dotnet-stack/frontend # Ruta al directorio donde se
encuentra el código del frontend
dockerfile: Dockerfile # Opcional si se llama Dockerfile por defecto
image: front-api:1.0
container_name: front-api
networks:
  - lemoncode-challenge
environment:
  - API_URI=http://topics-api:5000/api/topics
ports:
  - "8080:3000"

networks:
  lemoncode-challenge:
    driver: bridge
```

Para levantar los contenedores sería:

```
docker compose up -d
```

Para pararlos sería:

```
docker compose down
```

Para eliminar todo una vez parado los contenedores se usa:

```
docker ps -a
```

Con el comando anterior mostrará las ID de los contenedores parados por lo que una vez identificados podemos proceder a borrarlos:

```
docker rm id_del_contenedor1 id_del_contenedor2 id_del_contenedor3
```

Para listar las imágenes se usa:

```
docker images
```

Para borrar las imágenes:

```
docker rmi id_del_imagen1 id_del_imagen2 id_del_imagen3
```

```
C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
front-api	1.0	7b8b8eb6f6d3	2 days ago	1.59GB
topics-api	1.0	f4513c0554e1	2 days ago	311MB
mongo	4.4	52c42cbab240	9 months ago	616MB

Para listar los volúmenes se usa:

```
docker volume ls
```

```
C:\Users\hugin\Desktop\entregar_docker\lemoncode-challenge>docker volume ls
```

DRIVER	VOLUME NAME
local	1005c13f5ff318d355befe110a82be79ad1a70482caf316e4a3cc3c35fcdd5ba

Para borrar los volúmenes se usa:

```
docker volume rm id_del_volumen1
```