MANUAL TECNICO

1) Back End

Para el Back End se utilizó el puerto 3000 de localhost, se realizó un servidor en lenguaje de programación Go, con conexión a base de datos mongo.

Contiene un método Post con la extensión “ObtenerOperacion”, que funciona para almacenar una operación enviada desde el Front End, calcularla, obtener su fecha y hora de realización y almacenarla en la base de datos de mongo llamada SO1_Practica1 en la colección Operaciones.

Contiene un método Get con la extensión “RegresarOperaciones” que realiza una consulta en la base de datos de mongo llamada SO1_Practica1 en la colección Operaciones, para obtener todos los registros de operaciones almacenadas.

2) Front End

Cuenta con una interfaz sencilla y amigable donde se utilizó el puerto 4200 de localhost, realizando petición a la aplicación Back End de Go para mandar operaciones, o regresar registros de operaciones realizadas en formato json.

3) Mongo DB

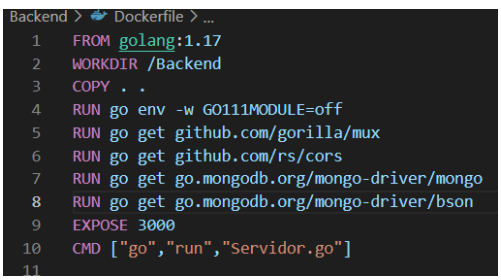
Se realizo una instalación de base de datos sin cliente o contraseña en el puerto 27017 ya sea local, o en contenedor de Docker.

4) Docker

- Imágenes

  Se utilizo la imagen oficial de Mongo DB.

  Se creo una imagen para la aplicación de servidor en Go por medio de un archivo Docker file con los siguientes parámetros:



Se creo una imagen a partir de una aplicación estática de React JS generada con la ayuda de nginx, por medio de un archivo Docker file con los siguientes parámetros:

frontend > Dockerfile > ...

```
FROM nginx:1.21.6-alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY ./build/ /usr/share/nginx/html

```

Y Se utilizo una configuración de nginx en el siguiente archivo nginx.conf:

frontend > nginx.conf

```
events{}
http{
    include /etc/nginx/mime.types;
    server{
        listen 4200;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html;
        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

- Docker-Compose.yaml

El archivo Docker-Compose utiliza la versión 3.9 creando una Network llamada practica1_201908335 con los siguientes servicios.

Cuenta con un servicio de bases de datos de mongo a partir de la imagen oficial de Mongo DB mapeada en el puerto 27017, creando volúmenes en la ruta especificada dentro del almacenamiento del equipo, cuando se desea ejecutar este archivo, es importante cambiar esta ruta, a una ruta valida dentro de nuestro equipo, teniendo el siguiente formato: -@Ruta:/data/db

```
services:
  db:
    image: mongo
    container_name: dbmongo
    ports:
      - "27017:27017"
    restart: always
    volumes:
      - C:\Users\cocac\Documents\GitHub\SO1_Practica1_201908335\DBVolumenes:/data/db
    networks:
      - practica1_201908335
```

También cuenta con un servicio de Back End que se comunica con el contenedor de base de datos mongo, donde obtiene la imagen a utilizar desde Docker Hub: cocacore7/backend_p1_201908335. Este servicio depende de la base de datos y se mapea en el puerto 3000.

```yaml
backend:
  image: cocacore7/backend_p1_201908335
  container_name: backend_p1_201908335
  environment:
    MONGO_HOST: db
  ports:
  - "3000:3000"
  restart: always
  depends_on:
  - db
  networks:
  - practica1_201908335
```

Por ultimo cuenta con un servicio de Front End que se comunica con el contenedor de Back End, donde obtiene la imagen desde Docker Hub: cocacore7/frontend_p1_201908335. Este servicio depende del servicio de Back End y se mapea en el puerto 4200.

```yaml
frontend:
  image: cocacore7/frontend_p1_201908335
  container_name: frontend_p1_201908335
  ports:
  - "4200:4200"
  restart: always
  depends_on:
  - backend
  networks:
  - practica1_201908335
```

Se mapea el volumen de la base de datos Mongo y se establece la Network como tipo bridge.

```yaml
volumes:
  mongodata:

networks:
  practica1_201908335:
    name: "practica1_201908335"
    driver: bridge
```