MANUAL TECNICO

Redes 2 -Practica 2

GRUPO #17

201246079 AXEL SMAYLIE LOPEZ XUM
201503910 ARIEL ALEJANDRO BAUTISTA MÉNDEZ
201709164 JHONNATAN ENMANUEL ORANTES GARCIA
201504284 WENDY ARACELY CHAMALÉ BOCH
201503470 YIMMI DANIEL RUANO PERNILLO

SERVIDOR

Se crearon tres servidores utilizando node.js los cuales pertenecen a un grupo de red separado estos servidores solo tienen conexión con la base de datos y no es posible acceder a ello a no ser que sea por el puerto que queda libre para el front

Son tres servidores que se crearon como replica para poder ser manejadas con un load balancer estos servidores contiene tres endpoint los cuales son :

```
router.route('/reporte/insertar').post(reporte.insertar)
router.route('/reporte/obtener/:carnet').get(reporte.obtener)
router.route('/reporte/obtenertodos').get(reporte.obtenerTodos)
```

endpoints

Reporte/Insertar

Se obtienen los datos en un formato json y se ingresan en input que es una variable que mas adelante se usa para manejar los datos.

Se comprueba que los datos estén completos para poder seguir de lo contrario se regresa un res.status(422) indicando que los datos no están completos

se inicia un .query que es la sentencia de mysql que se ejecutara para mandar los datos a la base de datos si los datos no están en orden se regresa un res.status(500) indicando que la consulta no se realizo por la carencia de los datos solicitados, y si todo esta en orden se creara la consulta y se retorna un estatus 200

```
function insertar(req, res) {
    let input = req.body;
    console.log(req.body);
    if (input.carnet != "") {
        try {
            conn.query(
                "INSERT INTO reporte (carnet, nombre, curso, descripcion) VALUES (?,?,?,?)"
,
                [input.carnet, input.nombre, input.curso, input.descripcion],
                (error, results) => {
                    if (error) {
                        res.status(500).json({
                            Mensaje: "Error en la consulta, verifique los campos de entrada
",
                            Error: error
```

```
                    });
                } else {
                    res.status(200).json({
                        Mensaje: "Reporte Ingresado Correctamente",
                        Servidor: firma.servidor.dos
                    });
                }
            }
        );
    } catch (error) {
        res.status(500).json({
            Mensaje: "Error catch",
        });
    }
} else {
    res.status(422).json({
        Mensaje: "Faltan campos obligatorios en el JSON"
    });
}
}
```

Reporte/obtener

Se debe ingresar el carnet de no tener el carnet se regresará un error indicando que los datos no son los correctos

Se crea un Query que toma todos los valores de la tabla reporte que coincida con el carnet que se manda

```
function obtener(req, res) {
    let input = req.params.carnet

    try {
        conn.query(
            "SELECT * FROM reporte WHERE carnet = ?",
            [input],
            (error, results) => {
                if (error) {
                    console.log(error);
                    res.status(500).json({
                        Mensaje: "Error en la consulta, verifique los campos de entrada",
                    });
                } else {
                    console.log(results);
                    res.status(200).json({
                        data: results,
```

```
                        Servidor: firma.servidor.dos
                    });
                }
            }
        );
    } catch (error) {
        res.status(500).json({
            Mensaje: "Error catch",
        });
    }
}
```

Reporte/obtenerTodos

Un get que obtiene todos los datos en la tabla de reportes

```
function obtenerTodos(req, res) {

    try {
        conn.query(
            "SELECT * FROM reporte",
            (error, results) => {
                if (error) {
                    res.status(500).json({
                        Mensaje: "Error en la consulta",
                    });
                } else {
                    res.status(200).json({
                        data: results,
                        Servidor: firma.servidor.dos
                    });
                }
            }
        );
    } catch (error) {
        res.status(500).json({
            Mensaje: "Error catch",
        });
    }
}
```

DOCKER

Cada uno de los servidores se ingresó en un contenedor esto para tener un manejo aislado de la red en la que estos trabajan

Docker File

En el Docker file se especifica el tipo de lenguaje al cual pertenece la imagen, el directorio en el que se trabajara dentro del conteiner, los archivos que se deben copiar del la carpeta en la que se encuentre el Dockerfile en este caso se copiara todo .

Siguiendo con un run npm instal que instala todas las dependencias que se encuentre en nuestro package.json para luego correr el index.js donde se levanta nuestro servidor

```
FROM node:alpine
WORKDIR /src
COPY . .
RUN npm install
EXPOSE 3002
ENTRYPOINT ["node", "index.js"]
```

Este archive es útil para crear el Docker compose ya que con esto se creara la imagen

```
  dockerfile: Dockerfile
      context: ./Servidor1
```

Dockercompose

Este es un archivo que se utilixa para levantartodo los servidores si como la base de datos

En services se puede ver que se crea mysql que es donde se le colocan losdatos para crear la base de datos des mysql dejando expuesto el puerto 3306

```
version: '3.1'

services:
  mysql:
    image: mysql
    expose:
        - "3306"
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    volumes:
     - ./DataBase/mysql_data:/var/lib/mysql
    environment:
     MYSQL_ROOT_PASSWORD: grupo17
```

```
    ports:
      - "3036:3306"
```

Se crear el server1 con su nombre de imagen seguido de build que creara la imagen a partir del Docker file que esta en carpeta que se le indica este estar corriendo en el puerto 3001

```
  server1:
    image: server1
    build:
      dockerfile: Dockerfile
      context: ./Servidor1
    ports:
      - "3001:3001"
```

Load blancer

El load balancer esta creado en nginx este se encarga de balacera las peticiones para que todas puedan ser atendidas lo mas pronto posible por los servidores que este libres, balanceara las peticiones que le mandemos a los 3 servidores que estará en nuestro Docker compose

Este siempre manda mensajes para verificar si los servidores esta activos de no estar activos los saca de las rutas posibles y no mandan data al port donde están escuchando

El balanceador se crea en el puerto 4000 con el host siendo el mencionado en la parte location

```
events {}
http {
    upstream servidores {
        server server2:5000 fail_timeout=10s max_fails=5;
        server server1:5001 fail_timeout=10s max_fails=5;
        server server3:5002 fail_timeout=10s max_fails=5;

    }

    server {
        listen 4000;

        location / {
            proxy_pass http://servers;
        }
    }
}
```

Desde Docker compose la forma de implementar el balanceador es la siguiente:

Se le manda el archivo. conf que es el anterior visto y se crea una imagen a partir de una imagen de nginx que se obtiene desde dockehub

Se le especifica sobre que servidores actuar en este caso el server 1,2 y 3 y se especifica la red a la que pertenece

Por ultimo se cran las redes que estar siendo usadas dentro del Docker dando un driver bridge

```
  nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - server1
      - server2
      - server3
    ports:
      - "4000:4000"
    networks:
      - red-servidores
            Ipv4_address:172.35.7173


  frontend:
    #...
    networks:
      -red-servidores
      -red-basededatos
networks:

  red-servidores:
    driver: bridge
```

Frontend

El front esta implementado en angular 12, este cuenta con las páginas:

Enviar-reportes
Lista de reportes
Ver reportes

Dentro del fronte se hacen las llamas a las apis que están corriendo en los servidores

```
 getReporte()
  {
    return this.http.get<any>(`${this.api1}/reporte/obtenerTodos`);
  }

  Buscar(carnet:any){
    return this.http.post<any>(`${this.api1}/reporte/obtener`,carnet);
  }
}
```