Deploy

Pasos de despliegue

1. Crear carpeta llamada mobiApp.
2. Clonar repositorio del backend desde git.

```
git clone https://github.com/ormunoz/medi-app-backend.git
```

3. Clonas repositorio del frontend desde git.

```
git clone https://github.com/ormunoz/medi-app-frontend.git
```

4. Instalar docker.

💡 https://docs.docker.com/desktop/

5. Instalar docker compose.

💡 https://docs.docker.com/compose/install/

6. Ingresar a carpeta del backend.

```
cd medi-app-backend
```

7. Crear archivo .env

```
touch .env
```

8. Poblar archivo .env con lo siguiente:

```
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings

DATABASE_URL="postgresql://postgres:mediapp2023@medi-app-db.c0jnyt8s2lwd.us-east-1.rds.amazonaws.com:5432/postgres?schema=public"

SECRET_TOKEN=EstaEsMi_Clave_The_SecretaxD123

PORT=5000
```

9. Volver a la raiz he ingresar a la carpeta del frontend.

```
cd ..
cd medi-app-frontend
```

10. Crear archivo y Poblar archivo .env con lo siguiente:

```
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings

MEDI_APP_BACKEND_URL=http://localhost:5000
NODE_ENV=development
```

11. Volvemos a la carpeta raíz y crear un archivo llamado “docker-compose.yml”, en caso de tener el archivo enviado por Orlando Muñoz, pegar el archivo en la raíz y omitir el paso 11 y 12.

```
cd ..
touch docker-compose.yml
```

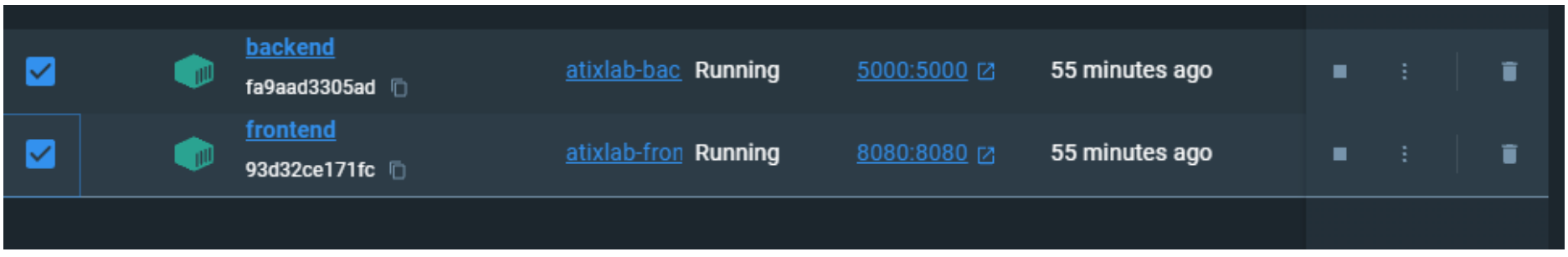
12. Copiar el siguiente código en docker-compose.yml.

```
version: '3'
networks:
  medi-net:
    driver: bridge
volumes:
  postgres:
services:
  frontend-medi-app:
    build:
      context: ./medi-app-frontend
      dockerfile: Dockerfile
    container_name: frontend
    ports:
      - "8080:8080"
    depends_on:
      - backend-medi-app
    networks:
      - medi-net
  backend-medi-app:
    build:
      context: ./medi-app-backend
      dockerfile: Dockerfile
    container_name: backend
    ports:
      - "5000:5000"
    networks:
      - medi-net
```

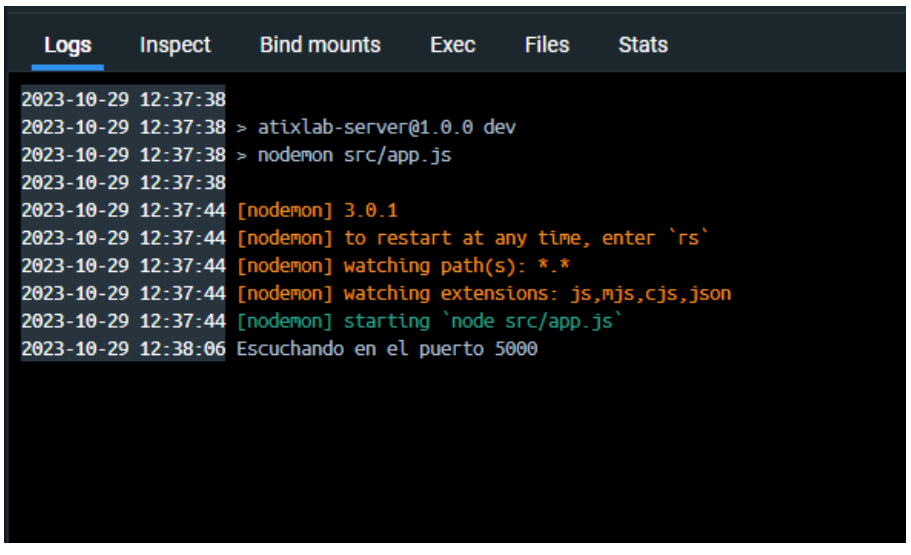
13. En la raíz ejecutamos el siguiente comando.

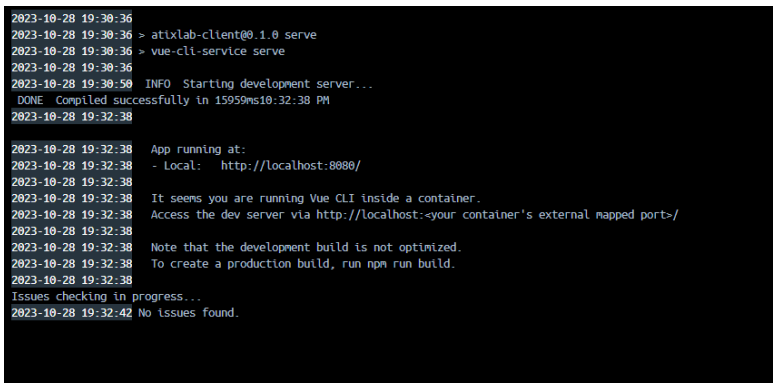
```
docker compose -f "docker-compose.yml" up -d --build
```

14. Abrimos docker, seleccionamos los container y seleccionamos start (en caso que no estén corriendo).



15. verificamos que el backend y front esten corriendo correctamente como se ve en la imagen





16. En caso de que la imagen del backend o front mande un error de dependencias, en la mismas viñetas del log ir a la pestaña Exec y ejecutar lo siguiente.
    16.1. Nota: Lo que haremos es eliminar la carpeta node_modules debido a que se instalaron mal las dependencias pero al aplicar el siguiente comando se instalaran correctamente, es un bug que tiene docker.
    16.2. Nota 2: En caso de que todo funcione como en la imagen omitir paso 16.

```
rm -rf node_modules
npm i node_modules
```

17. Abrimos el puerto que nos indica el Frontend en este caso es el: “http://localhost:8080/ ”.
18. Se puede iniciar como admin con las siguientes credenciales..

```
rut y/o UserName: admin
password: admin
```