Facultad de Ingeniería y Ciencias

Tarea 2 de Algoritmo de ruteo y redes resilientes

Integrantes	:	*David Salas*
Sección	:	*1*
Semestre	:	*2023-2*
Profesor	:	*Nicolas Boettcher*
Fecha de A	:	*04/12/2023*

Todos los archivos utilizados se pueden ver en este github:

Tarea2_Github

Descargar infraestructura

Para esta parte se descarga la infraestructura a utilizar, que en este caso es la de fibra óptica, en donde se descarga directamente del canvas en modulos, el archivo "PortalTransparencia_Yafun.zip", luego descomprimimos lo que tiene al interior y obtenemos cinco archivos que son:

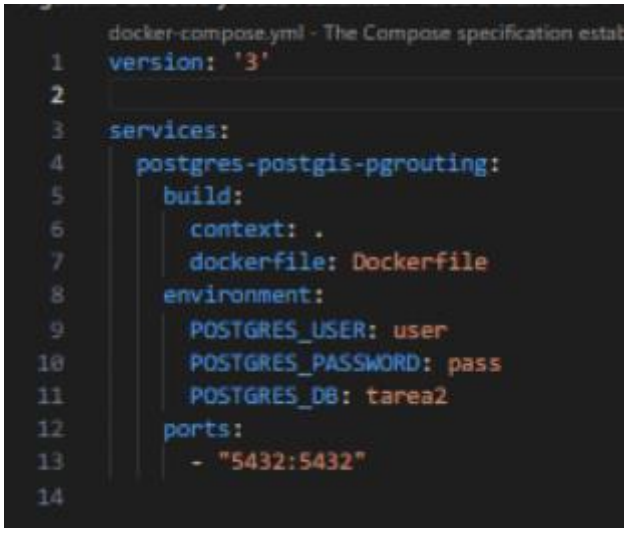
- Fibra_optica_detectada.dbf
- Fibra_optica_detectada.prj
- Fibra_optica_detectada.qpj
- Fibra_optica_detectada.shp
- Fibra_optica_detectada.shx

Creación de la tabla con la información

Luego, creamos un archivo **Dockerfile** que contiene Postgres:16 como imagen y las instalaciones de todo lo necesario, tanto de los documentos mencionado anteriormente, como la instalación de postgres16, postgist16 y pgrouting.

```
# Utiliza la imagen oficial de PostgreSQL con PostGIS
FROM postgres:16

# Variables de entorno para la configuración de la base de datos
ENV POSTGRES_USER user
ENV POSTGRES_PASSWORD pass
ENV POSTGRES_DB tarea2

# Copia el script SQL para la inicialización de la base de datos
COPY init.sql /docker-entrypoint-initdb.d/
COPY fibra_optica_detectada.* /docker-entrypoint-initdb.d/

# Descarga y compila pgrouting
RUN apt-get update
RUN apt-get install -y --no-install-recommends postgresql-16-postgis-3 postgresql-16-pgrouting gdal-bin
RUN apt-get clean && rm -rf /var/lib/apt/lists/*

# Expone el puerto de PostgreSQL
EXPOSE 5432
```

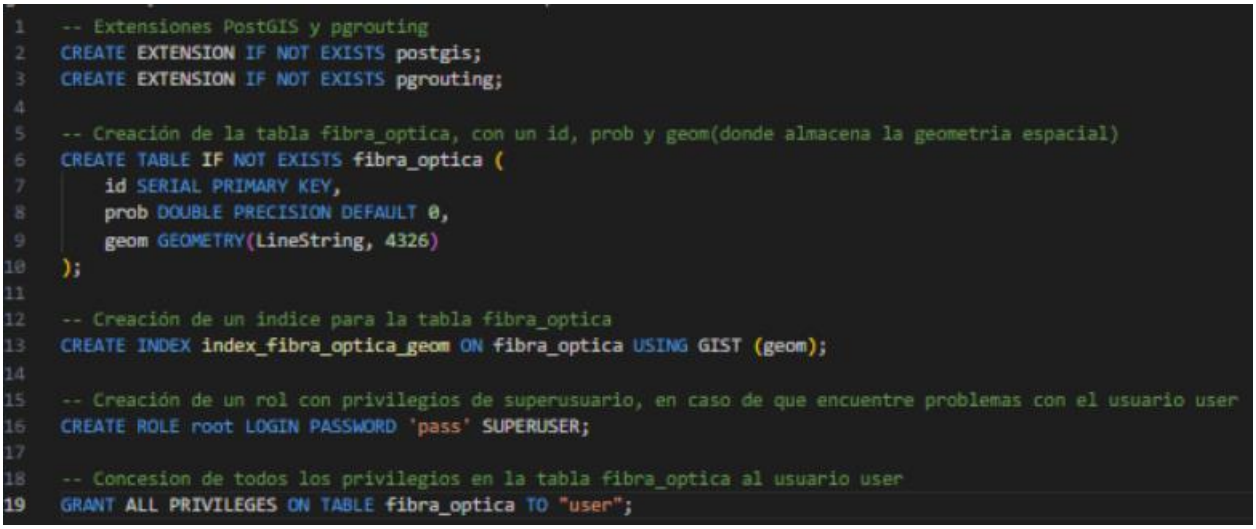
Figura 1: Archivo "Dockerfile"



```yaml
version: '3'

services:
  postgres-postgis-pgrouting:
    build:
      context: .
      dockerfile: Dockerfile
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: pass
      POSTGRES_DB: tarea2
    ports:
      - "5432:5432"
```

Figura 2: Archivo “docker-compose.yml”

Información del archivo init.sql, para que cree la tabla requerida para este trabajo



```sql
-- Extensiones PostGIS y pgrouting
CREATE EXTENSION IF NOT EXISTS postgis;
CREATE EXTENSION IF NOT EXISTS pgrouting;

-- Creación de la tabla fibra_optica, con un id, prob y geom(donde almacena la geometria espacial)
CREATE TABLE IF NOT EXISTS fibra_optica (
    id SERIAL PRIMARY KEY,
    prob DOUBLE PRECISION DEFAULT 0,
    geom GEOMETRY(LineString, 4326)
);

-- Creación de un indice para la tabla fibra_optica
CREATE INDEX index_fibra_optica_geom ON fibra_optica USING GIST (geom);

-- Creación de un rol con privilegios de superusuario, en caso de que encuentre problemas con el usuario user
CREATE ROLE root LOGIN PASSWORD 'pass' SUPERUSER;

-- Concesion de todos los privilegios en la tabla fibra_optica al usuario user
GRANT ALL PRIVILEGES ON TABLE fibra_optica TO "user";
```

Figura 3: Archivo “init.sql”

Donde tenemos las extensiones de postgis y pgrouting, la creación de la tabla a utilizar, un index para el id de la tabla “fibra_optica”, y por ultimo un rol de super usuario y conceder todos los privilegios al usuario de “user”.

Luego de haber ejecutado el dockerfile, con el siguiente comando:

```
Docker compose build
Docker compose up
```

Tenemos lo siguiente:

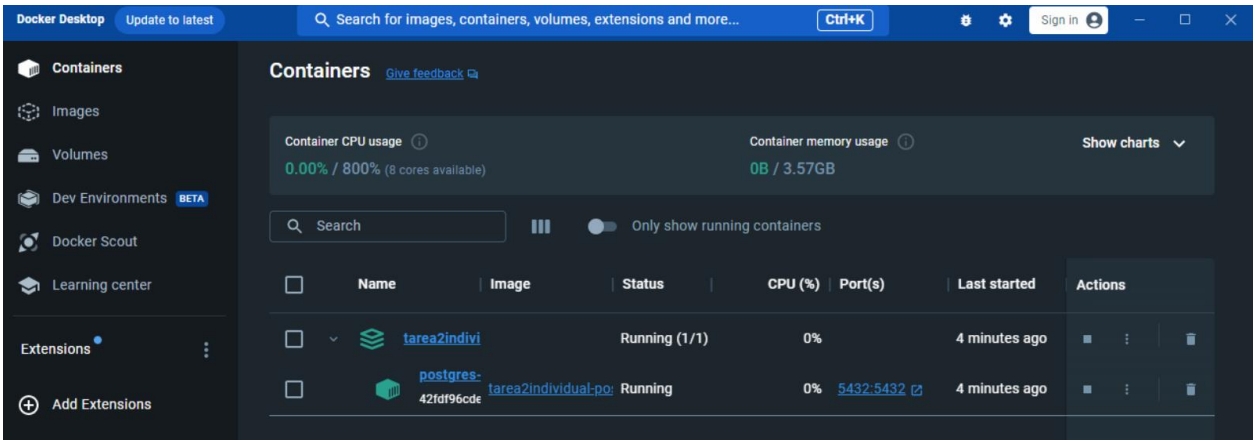


Figura 4: El contenedor levantado

En donde podemos visualizar que los contenedores creados por el dockerfile están levantados, pero aún no tenemos la información de la fibra_optica_detectada.
Para esto utilizaremos una herramienta llamada **“ogr2ogr”**, que nos sirva para transformar archivo de tipo SHP, en archivo que nosotros queramos, en este caso, sería de transformarlo a un formato de la base de datos SQL para adquirir sus datos a la tabla creada “fibra_optica”. Todo esto dentro de la bash del contenedor, con el siguiente comando podemos entrar a la bash de este:

```
docker exec -it 42fdf96cdec3 bash
```

En donde, el identificador “**42fdf96cdec3**”, sería el id del contenedor usado.

Con la siguiente línea de comando podemos hacer lo mencionado anteriormente:

```
ogr2ogr -f "PostgreSQL" PG:'dbname=tarea2 user=user password=pass' -nln fibra_optica -lco GEOMETRY_NAME=geom -append fibra_optica_detectada.shp
```

Donde el resultado entregado es el siguiente:

Figura 5: Comprobación comando ogr2ogr

Cabe destacar que, para que ejecute como debe el comando, este debe encontrarse con el archivo “fibra_optica_detectada.shp” en el mismo destino, por se ingresa al archivo “docker-entrypoint-initdb.d”.

Una vez realizado esto, procedemos a verificar si existe la tabla y sus datos correspondiente de lo realizado anteriormente, con el siguiente comando en la bash del contenedor, entramos a PostgreSQL:

```
psql -U user -d tarea2
```

Luego, con el comando dentro de psql, "***\dt",*** podemos visualizar las tablas que hay dentro.

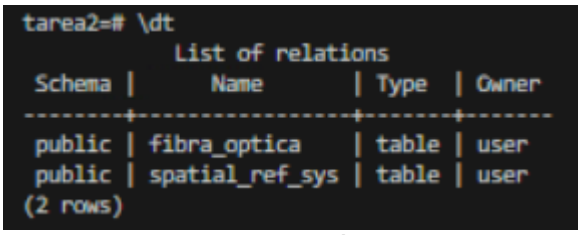


Figura 6: Verificación de tablas

Y con este comando sabemos si posee los datos que les añadimos con el comando ogr2ogr:

```
SELECT COUNT(*) FROM fibra_optica;
```

Con el siguiente resultado:

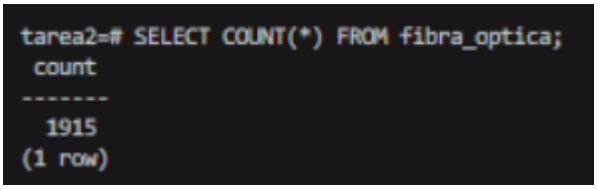


Figura 7: Comprobación de datos

Conversión a red

Una vez, ya realizado todo lo mencionado anteriormente, se procede a la conversión a red, para hacer uso de pgrouting.

En donde, se hace unos cambios a la tabla "fibra_optica", para poder convertirla en una topología red, que ente caso creamos dos columnas, que serian el inicio y final, que se representaran como nodos de inicio y final, valga la redundancia, para cada segmento de la red

```
ALTER TABLE fibra_optica ADD COLUMN inicio integer;
ALTER TABLE fibra_optica ADD COLUMN final integer;
```

Una vez agregadas las tablas, se realiza el siguiente comando de SQL, para agregarle valores correctos a los nodos mencionado anteriormente.

```
UPDATE fibra_optica SET
  inicio = CAST(ST_X(nodes.inicio) AS integer),
  final = CAST(ST_X(nodes.final) AS integer)
FROM (
  SELECT
    edge.id AS edge_id,
    edge.inicio,
    edge.final
  FROM (
    SELECT
      id,
      ST_StartPoint(geom)::geometry(Point, 4326) AS inicio,
      ST_EndPoint(geom)::geometry(Point, 4326) AS final
```

```
    FROM fibra_optica
  ) AS edge
  JOIN (
    SELECT
      id,
      ST_StartPoint(geom)::geometry(Point, 4326) AS inicio,
      ST_EndPoint(geom)::geometry(Point, 4326) AS final
    FROM fibra_optica
  ) AS nodes
  ON edge.id = nodes.id
) AS nodes
WHERE fibra_optica.id = nodes.edge_id;
```

Donde el resultado se puede ver de la siguiente Figura:

```
tarea2=# ALTER TABLE fibra_optica ADD COLUMN inicio integer;
ALTER TABLE
tarea2=# ALTER TABLE fibra_optica ADD COLUMN final integer;
ALTER TABLE
tarea2=# UPDATE fibra_optica SET
tarea2-#   inicio = CAST(ST_X(nodes.inicio) AS integer),
tarea2-#   final = CAST(ST_X(nodes.final) AS integer)
tarea2-# FROM (
tarea2(#   SELECT
tarea2(#     edge.id AS edge_id,
tarea2(#     edge.inicio,
tarea2(#     edge.final
tarea2(#   FROM (
tarea2(#     SELECT
tarea2(#       id,
tarea2(#       ST_StartPoint(geom)::geometry(Point, 4326) AS inicio,
tarea2(#       ST_EndPoint(geom)::geometry(Point, 4326) AS final
tarea2(#     FROM fibra_optica
tarea2(#   ) AS edge
tarea2(#   JOIN (
tarea2(#     SELECT
tarea2(#       id,
tarea2(#       ST_StartPoint(geom)::geometry(Point, 4326) AS inicio,
tarea2(#       ST_EndPoint(geom)::geometry(Point, 4326) AS final
tarea2(#     FROM fibra_optica
tarea2(#   ) AS nodes
tarea2(#   ON edge.id = nodes.id
tarea2(# ) AS nodes
tarea2-# WHERE fibra_optica.id = nodes.edge_id;
UPDATE 1915
```

Figura 8: Comprobación de los datos agregados

Ya agregado estos nodos y configurados con los valores correspondiente, se procede a crear la topología con el siguiente comando:

```
SELECT pgr_createTopology('fibra_optica', 0.0001, 'geom', 'id', 'inicio',
'final');
```

En donde el resultado de crear esta topología se ve en la siguiente figura:

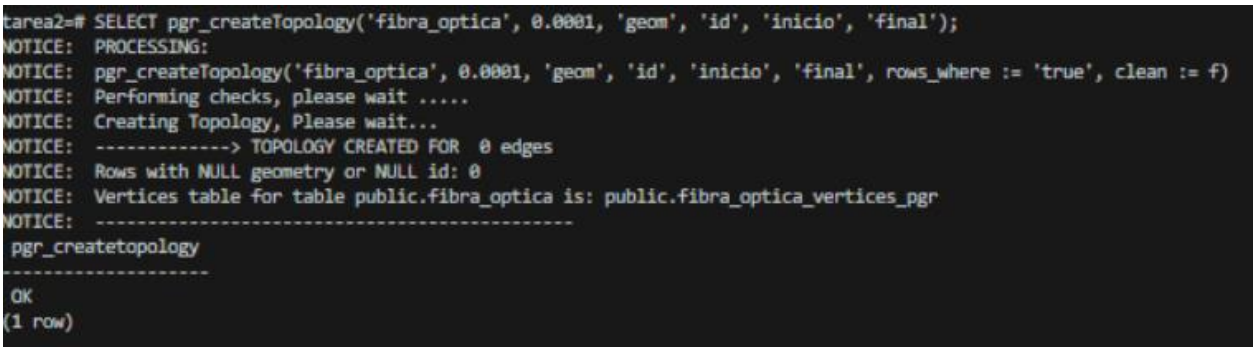


Figura 9: Creación de topología red con “pgr_createTopology”

Debido a problemas de visualización para ver los datos, no puedo mostrar una figura con los datos ya realizado:

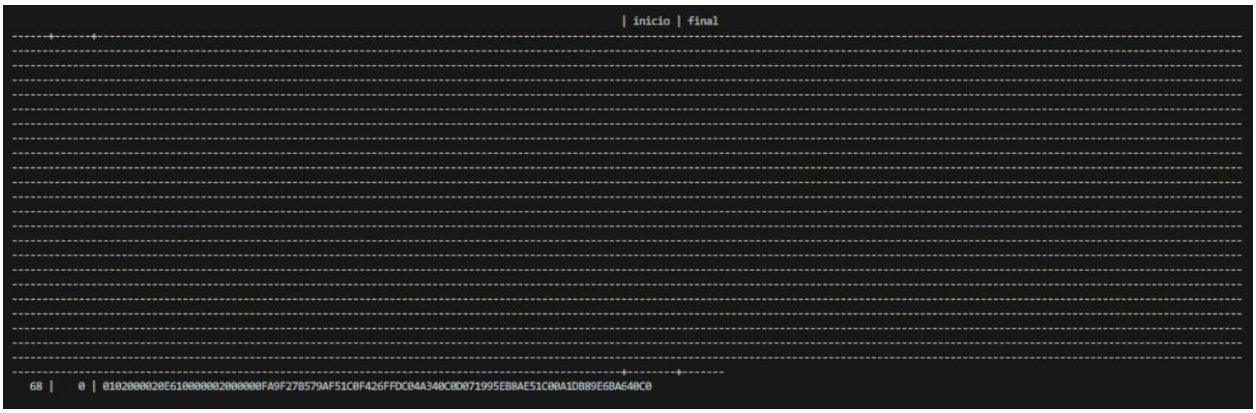


Figura 10: Mala visualización de los datos

Script ATTR

Para este caso el script de ATTR que utilizaremos es el siguiente:

Algoritmo de ruteo y redes resilentes > Tarea 2 individual > $ ATTR.sh

```
#!/bin/bash

# Variables con la información de la base de datos
DB_NAME="tarea2"
DB_USER="user"
DB_PASSWORD="pass"

# Comando SQL para calcular el ATTR
SQL_COMMAND="SELECT 1 - AVG(start.prob * final.prob) AS a2tr
FROM fibra_optica start
JOIN fibra_optica final ON start.final = final.inicio
WHERE start.id != final.id;"

# Ejecutar la consulta SQL y almacenar el resultado
RESULTADO=$(docker exec -i tarea2individual-postgres-postgis-pgrouting-1 psql -U $DB_USER -d $DB_NAME -c "$SQL_COMMAND")

# Verificar el resultado de la ejecución de la consulta SQL
if [ $? -ne 0 ]; then
  echo "Error al ejecutar la consulta."
  exit 1
fi

# Guardar el resultado en un archivo de texto (bloc de notas)
echo "El ATTR promedio obtenido fue: $RESULTADO" > resultado.txt

echo "Resultado guardado en resultado.txt"

# Pausar la ejecución del script para que no se cierre inmediatamente
read -p "Presiona Enter para salir
```

Figura 11: Script promedio ATTR

En donde, calculamos el ATTR promedio entre los nodos de inicio y final, luego el resultado obtenido de esto se guarda en un archivo llamado “resultado.txt”, donde nos dio el siguiente resultado:

Algoritmo de ruteo y redes resilentes > Tarea 2 individual >

```
El ATTR promedio obtenido fue:  a2tr
------
     1
(1 row)
```

Figura 12: Resultado de aplicar ATTR promedio

Que, en este caso, podemos decir que lo obtenido fue un 100%, ya que este cálculo se realiza antes de la falla, donde tenemos que todas las probabilidades asignadas son de 0, dada la creación de la tabla sin valores.