Etapas 1 & 2: Abastecimiento y Conexión a la Base de Datos

Configuración de Infraestructura con PostgreSQL y Docker

1. Abastecimiento de Infraestructura con Docker

Para la gestión de la base de datos, se seleccionó **PostgreSQL 16**, un motor de base de datos de código abierto, robusto y ampliamente utilizado para aplicaciones analíticas. Esta elección se basó en la familiaridad con la tecnología y su facilidad de implementación.

La instalación se realizó utilizando **Docker** para fomentar las buenas prácticas de contenerización, aislando el entorno de la base de datos del sistema operativo base.

Pasos de Instalación

1. **Creación del archivo `docker-compose.yml`**: Se definió un servicio de base de datos con la imagen oficial de `postgres:16`. El archivo de configuración es el siguiente:

```
version: '3.8'

services:
  db:
    image: postgres:16
    container_name: postgres_transporte
    restart: always
    environment:
      POSTGRES_DB: transporte_medellin
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: admin
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data

volumes:
  pgdata:
```

2. **Levantamiento del Contenedor**: Se ejecutó el siguiente comando en la terminal, en la raíz del proyecto, para iniciar el contenedor en modo "detached" (en segundo plano):

```
docker-compose up -d
```

Configuración del Contenedor

Imagen	postgres:16
Nombre del Contenedor	postgres_transporte
Volumen Persistente	Se mapeó un volumen nombrado `pgdata` al directorio de datos de PostgreSQL dentro del contenedor para asegurar que los datos no se pierdan al detener o reiniciar el contenedor
Puerto	Se expuso el puerto `5432` del contenedor al puerto `5432` de la máquina local

2. Configuración de Conexión Remota desde un IDE

Se estableció una conexión a la base de datos desde un IDE (se utilizó **DBeaver** por su facilidad de instalación y uso, además de haber sido presentado en clase) para facilitar la ejecución de scripts y la administración.

Parámetros de Conexión

Parámetro	Valor
Motor de Base de Datos	PostgreSQL
Host / Servidor	`localhost`
Puerto (TCP)	`5432`
Base de Datos	`transporte_medellin` (definida en `POSTGRES_DB`)
Usuario	`postgres` (definido en `POSTGRES_USER`)
Contraseña	`admin` (definida en `POSTGRES_PASSWORD`)

Habilitación de Puertos

La conexión es posible gracias a la directiva `ports: - "5432:5432"` en el archivo `docker-compose.yml`. Esta línea mapea el puerto **5432/TCP** del contenedor de PostgreSQL al puerto **5432** de la máquina anfitriona (`localhost`), permitiendo que las aplicaciones locales se conecten al servicio de base de datos que se ejecuta dentro del contenedor.

3. Gestión de Usuarios y Privilegios

El requisito del examen especifica la creación de un usuario y la asignación de privilegios mínimos. En esta implementación con Docker, este proceso se maneja de forma automática y eficiente:

Configuración Automática

- **Creación de Usuario:** La imagen oficial de PostgreSQL utiliza las variables de entorno del archivo `docker-compose.yml` para la configuración inicial. El usuario especificado en `POSTGRES_USER` (en nuestro caso, `postgres`) es creado automáticamente cuando el contenedor se inicia por primera vez.
- **Asignación de Privilegios:** Por defecto, el usuario creado se establece como **propietario** de la base de datos definida en `POSTGRES_DB` (`transporte_medellin`). Como propietario, el usuario `postgres` tiene **todos los privilegios** sobre esta base de datos (crear tablas, insertar datos, ejecutar funciones, etc.), cumpliendo así con el requisito de tener los permisos necesarios para operar. No se requieren sentencias `GRANT` adicionales para el funcionamiento del proyecto.

NOTA: (sé que no es lo ideal si hubiera otros usuarios, para la próxima hare usuarios con privilegios limitados, así como le gusta profe :D)