Transferencias de autos

Despliegue de infraestructura ETL

Bases de datos

Integrantes:
Chanquía Alejo
Giusiano Santiago
Irigo, Santiago
Matias González
Martines Zacarias
Mercado Llanco

20 de mayo de 2025

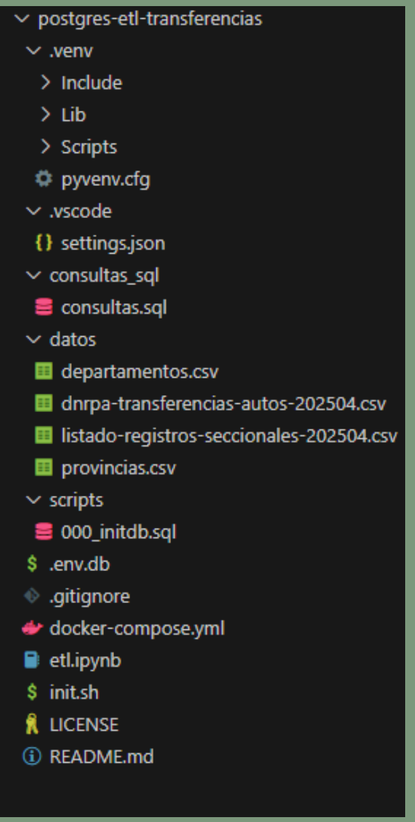
Base de datos

Introducción

Presentamos nuestro trabajo que consistió en realizar un tutorial en GitHub para guiar a un usuario a desplegar una infraestructura ETL utilizando Docker, PostgreSQL, Apache Superset y pgAdmin. Utilizamos DataSets de transferencia de vehículos, provincias, departamentos y registros, descargando los datos que obtuvimos de la página https://www.datos.gob.ar/.

Base de datos

Estructura del Proyecto



Provincias

datos > provincias.csv > data

```
1  "categoria","centroide_lat","centroide_lon","fuente","id","iso_id","iso_nombre","nombre","nombre_completo"
```

Departamentos

datos > departamentos.csv > data

```
1  "categoria","centroide_lat","centroide_lon","fuente","id","nombre","nombre_completo","provincia_id",
```

Listado de registros nacionales

datos > listado-registros-seccionales-202504.csv > data

```
1  "competencia","codigo","denominacion","encargado","encargado_cuit","domicilio","localidad","provincia_nombre",
```

Transferencias de autos

datos > dnrpa-transferencias-autos-202504.csv > data

```
1  tramite_tipo,tramite_fecha,fecha_inscripcion_inicial,registro_seccional_codigo,registro_seccional_descripcion,
```

Base de datos

ETL

```python
url = "https://datos.jus.gob.ar/dataset/f6932e82-a039-4462-968d-7dcda77d1a3e/resource/ff17485b-6711-405f-b628-676216e4d9e0/download/dnrpa-transferencias-autos-202504.csv"

departamentos_df = pd.read_csv('./datos/departamentos.csv', index_col='id')
provincia_df = pd.read_csv('./datos/provincias.csv', index_col='id')
registro_df = pd.read_csv('./datos/listado-registros-seccionales-202504.csv', index_col='codigo')

# Descargo el csv desde la url donde lo actualizan a diario.
transferencias_df = pd.read_csv(url, encoding='utf-8-sig', sep=',', low_memory=False)
```

Python

```python
#Muestra las primeras 5 filas del DataFrame transferencias_df.
transferencias_df.head()
```

Python

	tramite_tipo	tramite_fecha	fecha_inscripcion_inicial	registro_seccional_codigo	registro_seccional_descripcion	registro_seccional_provincia	automotor_origen	automotor_anio_modelo	automotor_tipo_codigo
0	TRANSFERENCIA NACIONAL	2025-04-01	1984-07-03	1146	MERLO Nº 2	Buenos Aires	Nacional	1984.0	12
1	TRANSFERENCIA NACIONAL	2025-04-01	2007-11-26	1146	MERLO Nº 2	Buenos Aires	Nacional	2007.0	17
2	TRANSFERENCIA NACIONAL C/PEDIDO	2025-04-01	2019-01-25	1146	MERLO Nº 2	Buenos Aires	Nacional	2019.0	22
3	TRANSFERENCIA NACIONAL	2025-04-01	2001-01-22	1146	MERLO Nº 2	Buenos Aires	Nacional	2001.0	13
4	TRANSFERENCIA IMPORTADO C/PEDIDO	2025-04-01	2011-04-19	1146	MERLO Nº 2	Buenos Aires	Protocolo 21	2011.0	05

5 rows × 25 columns

```python
columnas_deseadas = ['registro_seccional_descripcion','tramite_fecha','automotor_origen','automotor_anio_modelo','automotor_tipo_codigo','automotor_tipo_descripcion','automotor_modelo_descrip
transferencias_transformado_df = transferencias_df[columnas_deseadas]

transferencias_transformado_df.head()
```

Pythor

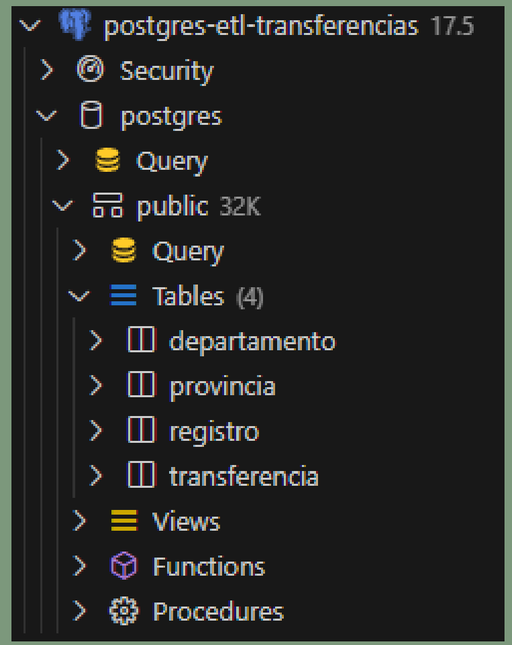
	registro_seccional_descripcion	tramite_fecha	automotor_origen	automotor_anio_modelo	automotor_tipo_codigo	automotor_tipo_descripcion	automotor_modelo_descripcion
0	MERLO Nº 2	2025-04-01	Nacional	1984.0	12	PICK-UP	F-100 / 1984
1	MERLO Nº 2	2025-04-01	Nacional	2007.0	17	SEDAN 5 PTAS	206 X-LINE 1.4 5P
2	MERLO Nº 2	2025-04-01	Nacional	2019.0	22	FURGON	SPRINTER 411 CDI/F 3250 STREET V1
3	MERLO Nº 2	2025-04-01	Nacional	2001.0	13	RURAL 3 PUERTAS	PARPADDAA
4	MERLO Nº 2	2025-04-01	Protocolo 21	2011.0	05	SEDAN 5 PTAS	FIT LX

Base de datos

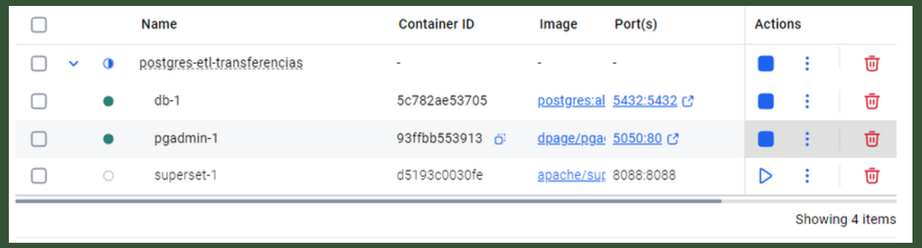
Docker Compose

```yaml
networks:
  net:
    external: false

volumes:
  postgres-db:
    external: false

services:
  db:
    image: postgres:alpine
    env_file:
      - .env.db
    restart: unless-stopped
    environment:
      - POSTGRES_INITDB_ARGS=--auth-host=md5 --auth-local=trust
    healthcheck:
      # Prueba de salud para el contenedor
      test: [ "CMD-SHELL", "pg_isready" ]
      interval: 10s
      timeout: 2s
      retries: 5
    ports:
      - 5432:5432
    volumes:
      - postgres-db:/var/lib/postgresql/data
      - ./scripts:/docker-entrypoint-initdb.d
      - ./datos:/datos
    networks:
      - net
```

```yaml
superset:
  image: apache/superset:4.0.0
  restart: unless-stopped
  env_file:
    - .env.db
  ports:
    - 8088:8088
  depends_on:
    db:
      condition: service_healthy
  networks:
    - net

pgadmin:
  image: dpage/pgadmin4
  restart: unless-stopped
  env_file:
    - .env.db
  ports:
    - 5050:80
  depends_on:
    db:
      condition: service_healthy
  networks:
    - net
```



Contenedores




```
PS C:\Users\zacam\Desktop\postgres-etl-transferencias> docker compose up -d
[+] Running 42/42
 ✓ pgadmin Pulled
 ✓ db Pulled
 ✓ superset Pulled
[+] Running 5/5
 ✓ Network postgres-etl-transferencias_net                Created
 ✓ Volume "postgres-etl-transferencias_postgres-db"       Created
 ✓ Container postgres-etl-transferencias-db-1             Healthy
 ✓ Container postgres-etl-transferencias-superset-1       Created
 ✓ Container postgres-etl-transferencias-pgadmin-1        Started
dependency failed to start: container postgres-etl-transferencias-db-1 is unhealthy
```

Base de datos

Init.sh

```
#/bin/bash
echo "Inicializamos el usuario de superset"
docker compose exec -it superset superset fab create-admin \
                --username admin \
                --firstname Superset \
                --lastname Admin \
                --email admin@superset.com \
                --password admin
echo "Migramos la base de datos"
docker compose exec -it superset superset db upgrade
echo "Seteamos los Roles"
docker compose exec -it superset superset init
```

Base de datos

Tablas definitivas y temporales

```sql
/*
Se crean las tablas temporales para cargar los datos
*/

CREATE TEMPORARY TABLE temp_provincia (
    categoria VARCHAR,
    centroide_lat FLOAT,
    centroide_lon FLOAT,
    fuente VARCHAR,
    id VARCHAR,
    iso_id VARCHAR,
    iso_nombre VARCHAR,
    nombre VARCHAR,
    nombre_completo VARCHAR
)

CREATE TEMPORARY TABLE temp_departamento (
    categoria VARCHAR,
    centroide_lat FLOAT,
    centroide_lon FLOAT,
    fuente VARCHAR,
    id VARCHAR,
    nombre VARCHAR,
    nombre_completo VARCHAR,
    provincia_id VARCHAR,
    provincia_interseccion FLOAT,
    provincia_nombre VARCHAR
)

CREATE TEMPORARY TABLE temp_registro (
    competencia VARCHAR,
    codigo VARCHAR,
    denominacion VARCHAR,
    encargado VARCHAR,
    encargado_cuit VARCHAR,
    domicilio VARCHAR,
    localidad VARCHAR,
    provincia_nombre VARCHAR,
    provincia_letra VARCHAR,
    codigo_postal VARCHAR,
    telefono VARCHAR,
    horario_atencion VARCHAR,
    provincia_id VARCHAR,
    nombre_completo VARCHAR
)
```

```sql
CREATE TABLE public.provincia (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    nombre_completo VARCHAR(50)
);

CREATE TABLE public.departamento (
    id SERIAL PRIMARY KEY,
    nombe VARCHAR(50) NOT NULL,
    nombre_completo VARCHAR(50),
    provincia_id INT,
    CONSTRAINT fk_provincia FOREIGN KEY (provincia_id) REFERENCES public.provincia (id)
);

CREATE TABLE public.registro (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    nombre_completo VARCHAR(50),
    codigo_postal VARCHAR(10),
    denominacion VARCHAR(50),
    provincia_id INT,
    CONSTRAINT fk_provincia FOREIGN KEY (provincia_id) REFERENCES public.provincia (id),
);

CREATE TABLE public.transferencia (
    id SERIAL PRIMARY KEY,
    registro_seccional_descripcion VARCHAR(50) NOT NULL,
    tramite_fecha DATE NOT NULL,
    automotor_origen VARCHAR(50),
    automotor_anio_modelo INT,
    automotor_tipo_codigo VARCHAR(50),
    automotor_tipo_descripcion VARCHAR(50),
    automotor_modelo_descripcion VARCHAR(50),
    departamento_id INT,
    provincia_id INT,
    registro_id INT,
    CONSTRAINT fk_departamento FOREIGN KEY (departamento_id) REFERENCES public.departamento (id),
    CONSTRAINT fk_provincia FOREIGN KEY (provincia_id) REFERENCES public.provincia (id),
    CONSTRAINT fk_registro FOREIGN KEY (registro_id) REFERENCES public.registro (id)
);
```

```sql
-- SQLBook: Code
/*
Verificamos borrando las tablas si existen
*/
DROP TABLE IF EXISTS public.departamento;

DROP TABLE IF EXISTS public.provincia;

DROP TABLE IF EXISTS public.registro;

DROP TABLE IF EXISTS public.transferencia;
```

```sql
CREATE TEMPORARY TABLE temp_transferencia (
    tramite_tipo VARCHAR,
    tramite_fecha DATE,
    fecha_inscripcion_inicial DATE,
    registro_seccional_codigo VARCHAR,
    registro_seccional_descripcion VARCHAR,
    registro_seccional_provincia VARCHAR,
    automotor_origen VARCHAR,
    automotor_anio_modelo INT,
    automotor_tipo_codigo VARCHAR,
    automotor_tipo_descripcion VARCHAR,
    automotor_marca_codigo VARCHAR,
    automotor_marca_descripcion VARCHAR,
    automotor_modelo_codigo VARCHAR,
    automotor_modelo_descripcion VARCHAR,
    automotor_uso_codigo VARCHAR,
    automotor_uso_descripcion VARCHAR,
    titular_tipo_persona VARCHAR,
    titular_domicilio_localidad VARCHAR,
    titular_domicilio_provincia VARCHAR,
    titular_genero VARCHAR,
    titular_anio_nacimiento INT,
    titular_pais_nacimiento VARCHAR,
    titular_porcentaje_titularidad DECIMAL,
    titular_domicilio_provincia_id VARCHAR,
    titular_pais_nacimiento_id VARCHAR
)
```

Base de datos

Carga de datos en las tablas temporales

```
/*
Cargo los datos en las tablas temporales
*/

/
▷Run
COPY temp_provincia
FROM '/datos/provincias.csv' DELIMITER ',' CSV HEADER;

▷Run | Select | Ask AI
INSERT INTO
    public.provincia (
        id,
        nombre,
        nombre_completo,
        centroide_lat,
        centroide_lon,
        categoria
    )
SELECT
    id::INTEGER,
    nombre,
    nombre_completo,
    centroide_lat,
    centroide_lon,
    categoria
FROM provincias_temp;

▷Run
COPY temp_departamento
FROM '/datos/departamentos.csv' DELIMITER ',' CSV HEADER;
```

```
INSERT INTO
    public.departamento (
        id,
        nombre,
        nombre_completo,
        centroide_lat,
        centroide_lon,
        categoria,
        provincia_id
    )
SELECT
    id::INTEGER,
    nombre,
    nombre_completo,
    centroide_lat,
    centroide_lon,
    categoria,
    provincia_id::INTEGER
FROM temp_departamentos;

▷Run
COPY temp_registro
FROM '/datos/listado-registros-seccionales-202504.csv' DELIMITER ',' CSV HEADER;

▷Run | Select | Ask AI
INSERT INTO
    public.registro (
        id,
        nombre,
        nombre_completo,
        provincia_id
    )
SELECT
    id::INTEGER,
    nombre,
    nombre_completo,
    provincia_id::INTEGER
FROM temp_registros;
```

Base de datos

Carga de datos en las tablas definitivas

```sql
INSERT INTO
    public.provincia (id, nombre)
SELECT DISTINCT
    titulaer_domicilio_provincia_id,
    titular_domicilio_provincia
FROM temp_transferencia
WHERE
    titular_domicilio_provincia_id NOT IN (
        SELECT id
        FROM public.provincia
    );

INSERT INTO
    public.departamento (id, nombre)
SELECT DISTINCT
    titular_domicilio_departamento_id,
    titular_domicilio_departamento
FROM temp_transferencia
WHERE
    titular_domicilio_departamento_id NOT IN (
        SELECT id
        FROM public.departamento
    );

INSERT INTO
    public.registro (id, nombre,codigo_postal,denominacion)
SELECT DISTINCT
    id_registro_seccional,
    registro_seccional_descripcion,
    codigo_postal,
    denominacion
FROM temp_transferencia
WHERE
    id_registro_seccional NOT IN (
        SELECT id
        FROM public.registro
    );
INSERT INTO
    public.transferencia (id,descripcion,fecha,automotor_origen,automotor_anio_modelo,automotor_tipo_codigo,automotor_tipo_descripcion,automotor_modelo_descripcion)
SELECT DISTINCT
    id,
    descripcion,
    fecha,
    automotor_origen,
    automotor_anio_modelo,
    automotor_tipo_codigo,
    automotor_tipo_descripcion,
    automotor_modelo_descripcion
FROM temp_transferencia
WHERE
    id NOT IN (
        SELECT id
        FROM public.transferencia
    );
```

Base de datos

Consultas SQL

```sql
SELECT
    registro_seccional.descripcion AS registro_seccional_descripcion,
    automotor.automotor_anio_modelo,
    COUNT(*) AS cantidad
FROM
    transferencia
    INNER JOIN registro_seccional ON automotor.registro_seccional_id = registro_seccional.id
GROUP BY
    registro_seccional.descripcion,
    automotor.automotor_anio_modelo;

SELECT
    tipo_automotor.codigo AS automotor_tipo_codigo,
    tipo_automotor.descripcion AS automotor_tipo_descripcion,
    automotor.modelo_descripcion AS automotor_modelo_descripcion,
    COUNT(*) AS cantidad
FROM
    transferencia
    INNER JOIN tipo_automotor ON automotor.tipo_automotor_id = tipo_automotor.id
GROUP BY
    tipo_automotor.codigo,
    tipo_automotor.descripcion,
    automotor.modelo_descripcion;

SELECT automotor.tramite_fecha, automotor.automotor_origen, COUNT(*) AS cantidad
FROM transferencia
GROUP BY
    automotor.tramite_fecha,
    automotor.automotor_origen
ORDER BY automotor.tramite fecha DESC;
```

Muchas Gracias