PROYECTO FINAL

FASES DEL CICLO DE VIDA DEL DATO

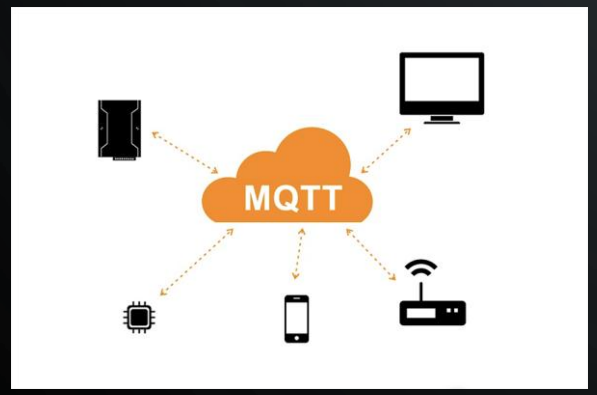
RUBÉN, ENEKO Y MANEL

HERRAMIENTAS UTILIZADAS

PostgreSQL

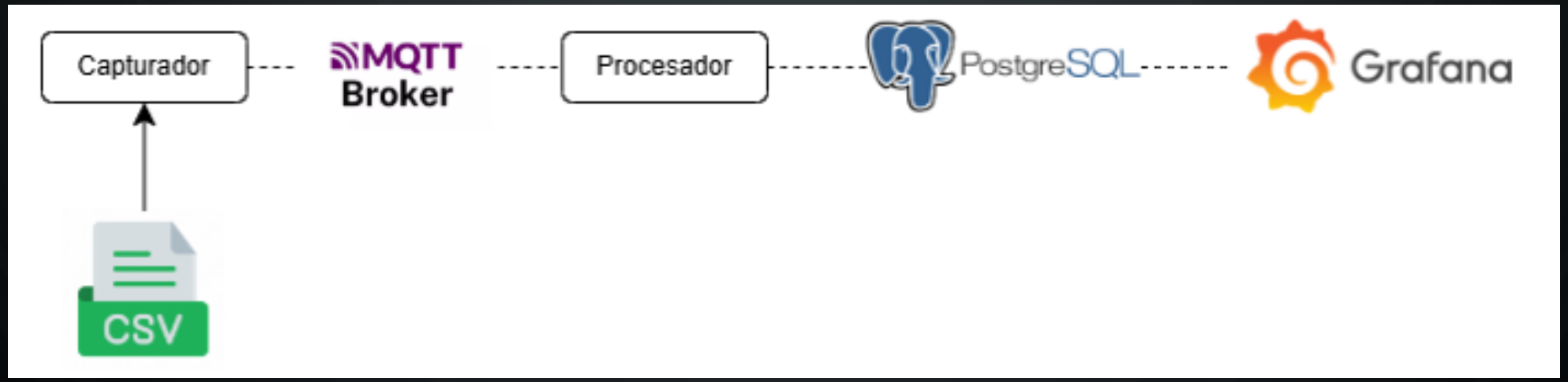
Grafana

python™



docker

ARQUITECTURA DEL PROYECTO



DOCKER-COMPOSE.YML

```yaml
broker:
  image: eclipse-mosquitto:latest
  container_name: Broker
  ports:
    - "1883:1883"
  volumes:
    - ./mosquitto/config:/mosquitto/config
  networks:
    - mqtt-net
```

```yaml
capturador:
  build: ./capturador
  container_name: Capturador
  environment:
    - MQTT_BROKER=broker
  depends_on:
    - broker
  networks:
    - mqtt-net
  ports:
    - "8000:8000"
  volumes:
    - ./capturador:/capturador
```

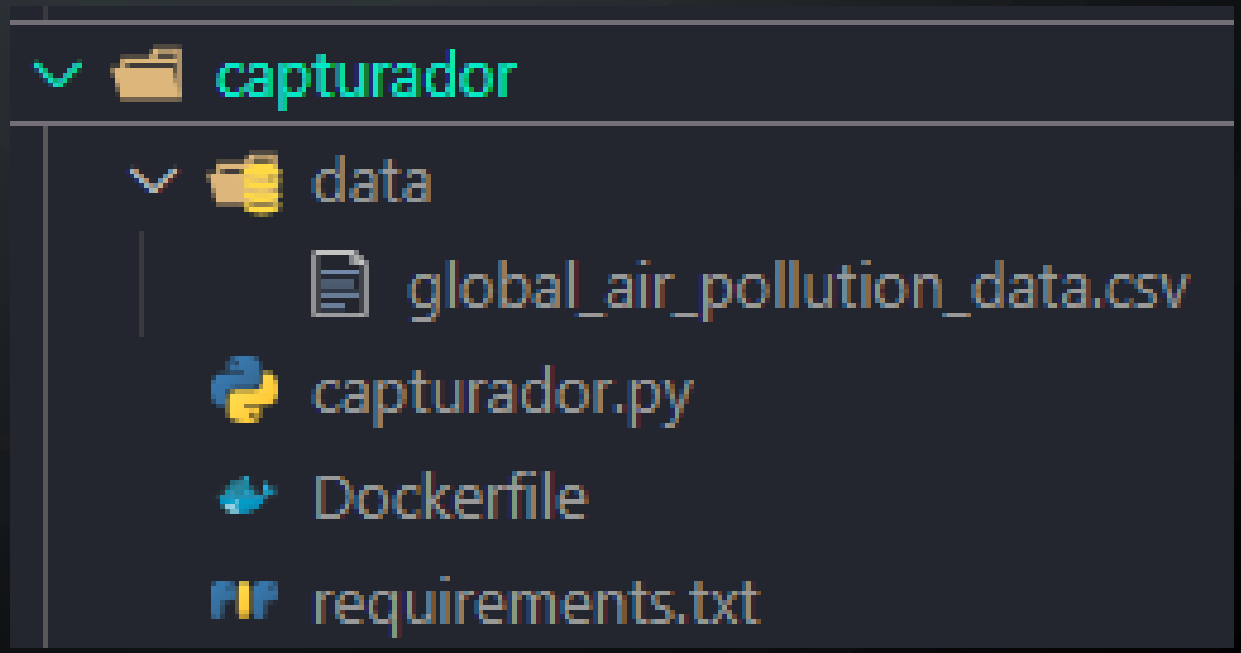
```yaml
procesador:
  build: ./procesador
  container_name: Procesador
  environment:
    - MQTT_BROKER=broker
    - POSTGRES_HOST=db
    - POSTGRES_PORT=5432
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=postgres
    - POSTGRES_DB=calidad_aire
  depends_on:
    - broker
    - db
  networks:
    - mqtt-net
  ports:
    - "8001:8000"
  volumes:
    - ./procesador:/procesador
```

```yaml
db:
  image: postgres:15
  container_name: Postgres
  restart: always
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: postgres
    POSTGRES_DB: calidad_aire
  ports:
    - "5432:5432"
  volumes:
    - pgdata:/var/lib/postgresql/data
    - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql
  networks:
    - mqtt-net
```

```yaml
grafana:
  image: grafana/grafana:11.6.0
  container_name: Grafana
  restart: always
  environment:
    - GF_SECURITY_ADMIN_USER=admin
    - GF_SECURITY_ADMIN_PASSWORD=admin
  ports:
    - "3000:3000"
  volumes:
    - grafana_data:/var/lib/grafana
    - ./grafana/dashboards:/var/lib/grafana/dashboards
    - ./grafana/provisioning:/etc/grafana/provisioning
  depends_on:
    - db
  networks:
    - mqtt-net
```

APP “CAPTURADOR”

- **Fase de Captura**
- **Fase de Envío (1)**



CAPTURADOR

capturador > Dockerfile > ...

```
1  FROM python:3.12.3
2  WORKDIR /capturador
3  COPY requirements.txt .
4  RUN pip install --no-cache-dir -r requirements.txt
5  COPY . .
6  CMD ["python", "capturador.py"]
7
```

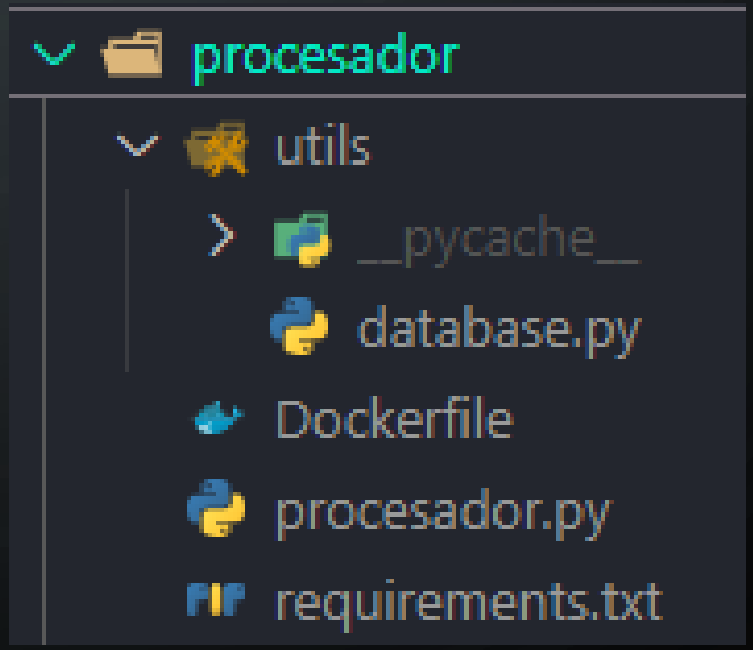
capturador > requirements.txt

```
1  paho-mqtt==2.1.0
2  pandas==2.2.0
```

- Captura de datos históricos de la calidad del aire desde un archivo CSV.
- Convertir cada fila del CSV en un objeto JSON y **publicarlo** mediante MQTT.

APP “PROCESADOR”

- **Fase de Envío (2)**
- **Fase de Procesamiento**
- **Fase de Persistencia**



PROCESADOR

procesador > Dockerfile > ...

```
1  FROM python:3.12.3
2  WORKDIR /procesador
3  COPY requirements.txt .
4  RUN pip install --no-cache-dir -r requirements.txt
5  COPY . .
6  CMD ["python", "procesador.py"]
```

procesador > requirements.txt

```
1  paho-mqtt==2.1.0
2  pydantic==2.6.3
3  sqlalchemy==2.0.29
4  psycopg2-binary==2.9.9
```

- Se **suscribe** a los datos enviados por el Capturador.
- Se validan los tipos de datos, campos requeridos y formatos de los datos entrantes con Pydantic.
- Se insertan en la base de datos de Postgres.

PERSISTENCIA

db > init.sql

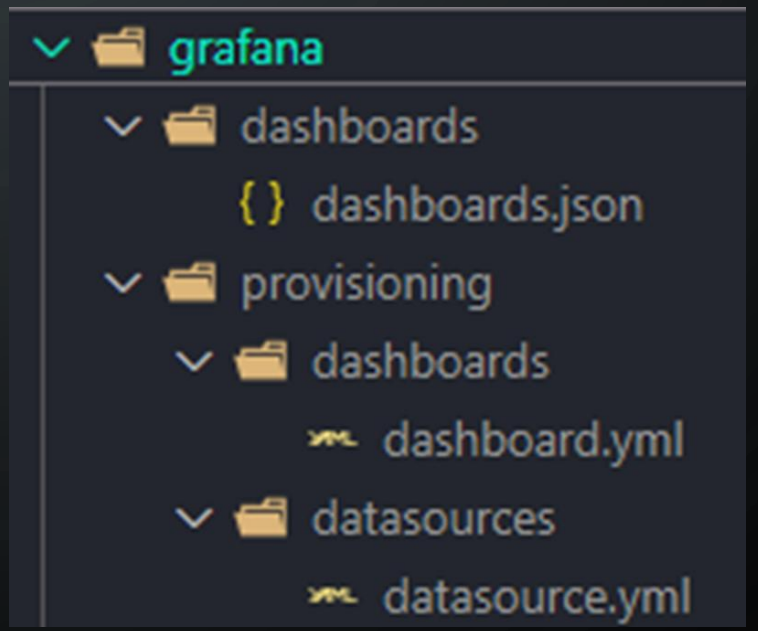
```sql
DROP TABLE IF EXISTS calidad_aire;

CREATE TABLE IF NOT EXISTS calidad_aire (
    id SERIAL PRIMARY KEY,
    pais VARCHAR,
    ciudad VARCHAR,
    valor_calidad_aire INTEGER,
    categoria_calidad_aire VARCHAR,
    categoria_monoxido_carbono VARCHAR,
    valor_ozono INTEGER,
    categoria_ozono VARCHAR,
    valor_dioxido_nitrogeno INTEGER,
    categoria_dioxido_nitrogeno VARCHAR,
    valor_particulas_finas INTEGER,
    categoria_particulas_finas VARCHAR
);
```

procesador > utils > database.py > ...

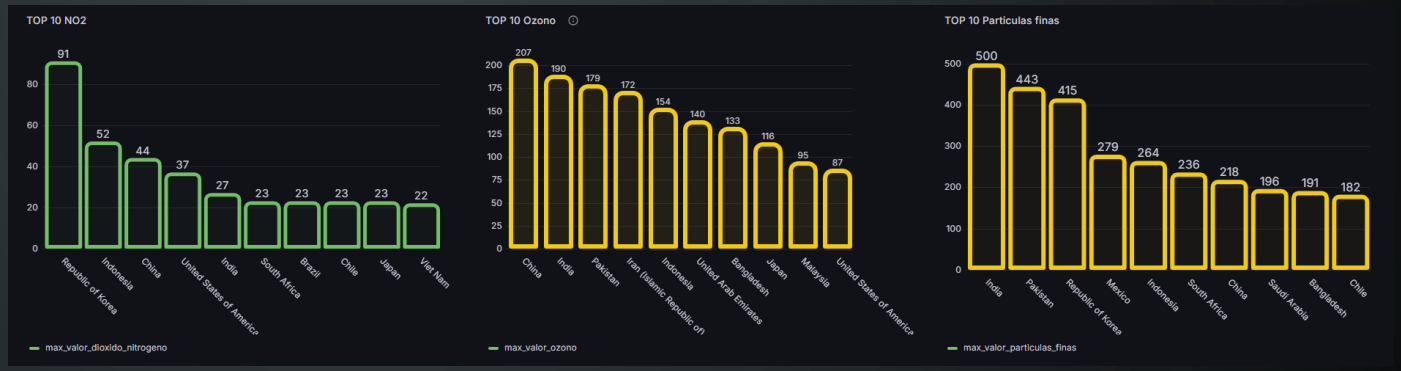
```python
import os
import logging
from sqlalchemy import create_engine, Column, Integer, String, MetaData, Table
from sqlalchemy.orm import sessionmaker

logger = logging.getLogger('database')
PG_HOST = os.environ.get("POSTGRES_HOST", "db")
PG_PORT = os.environ.get("POSTGRES_PORT", "5432")
PG_USER = os.environ.get("POSTGRES_USER", "postgres")
PG_PASSWORD = os.environ.get("POSTGRES_PASSWORD", "postgres")
PG_DB = os.environ.get("POSTGRES_DB", "postgres")
DATABASE_URL = f"postgresql://{PG_USER}:{PG_PASSWORD}@{PG_HOST}:{PG_PORT}/{PG_DB}"
engine = create_engine(DATABASE_URL)
metadata = MetaData()

calidad_aire_table = Table(
    "calidad_aire",
    metadata,
    Column("id", Integer, primary_key=True),
    Column("pais", String),
    Column("ciudad", String),
    Column("valor_calidad_aire", Integer),
    Column("categoria_calidad_aire", String),
    Column("categoria_monoxido_carbono", String),
    Column("valor_ozono", Integer),
    Column("categoria_ozono", String),
    Column("valor_dioxido_nitrogeno", Integer),
    Column("categoria_dioxido_nitrogeno", String),
    Column("valor_particulas_finas", Integer),
    Column("categoria_particulas_finas", String),
)

Session = sessionmaker(bind=engine)

def guardar_datos_calidad_aire(datos):
    try:
        with Session() as session:
            datos_dict = datos.dict() if hasattr(datos, 'dict') else datos
            ins = calidad_aire_table.insert().values(**datos_dict)
            session.execute(ins)
            session.commit()
            logger.info(f"Datos guardados correctamente: {datos_dict['ciudad']}, {datos_dict['pais']}")
            return True
    except Exception as e:
        logger.error(f"Error al guardar datos en la base de datos: {e}")
        return False
```

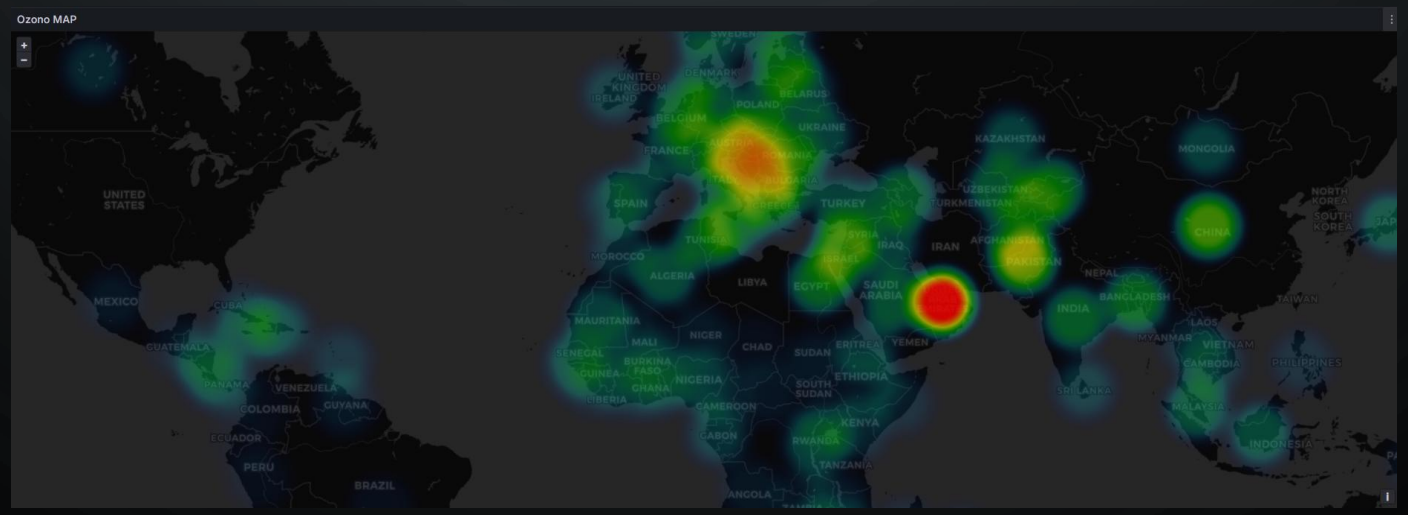
GRAFANA

- **Fase de Visualización**



VISUALIZACIÓN





- Fase de Análisis

analizador > analizador.ipynb > columnas_object = df_datos.select_dtypes(include='object')

Generate | Code | Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline | ...

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df_datos = pd.read_csv('../capturador/data/global_air_pollution_data.csv')

df_datos.head(5)
```

[2] ✓ 1.8s

	country_name	city_name	aqi_value	aqi_category	co_aqi_value\t	co_aqi_category	ozone_aqi_value	ozone_aqi_category	no2_aqi_value	no2_aqi_category	pm2.5_aqi_value	pm2.5_aqi_category
0	Russian Federation	Praskoveya	51	Moderate	1	Good	36	Good	0	Good	51	Moderate
1	Brazil	Presidente Dutra	41	Good	1	Good	5	Good	1	Good	41	Good
2	Italy	Priolo Gargallo	66	Moderate	1	Good	39	Good	2	Good	66	Moderate
3	Poland	Przasnysz	34	Good	1	Good	34	Good	0	Good	20	Good
4	France	Punaauia	22	Good	0	Good	22	Good	0	Good	6	Good

```
df_datos.info()
```

[3] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   country_name        2052 non-null   object
 1   city_name           2075 non-null   object
 2   aqi_value           2075 non-null   int64
 3   aqi_category        2075 non-null   object
 4   co_aqi_value              2075 non-null   int64
 5   co_aqi_category     2075 non-null   object
 6   ozone_aqi_value     2075 non-null   int64
 7   ozone_aqi_category  2075 non-null   object
 8   no2_aqi_value       2075 non-null   int64
 9   no2_aqi_category    2075 non-null   object
 10  pm2.5_aqi_value     2075 non-null   int64
 11  pm2.5_aqi_category  2075 non-null   object
dtypes: int64(5), object(7)
memory usage: 194.7+ KB
```

CONCUSIONES

- La Republica de Corea casi duplica al segundo en niveles máximos de NO2.
- China supera a India en el apartado de Ozono.
- El mapa resalta los niveles altos en el apartado de Ozono medio de Europa y en Arabia Saudí.
- Grafana no ha identificado a Estados Unidos en el mapa por el nombre (United States of America en vez de United States). *Podría ser una mejora a llevar a cabo la modificación del nombre de ese país.*

DEMO