Prueba técnica – Refactor + Validación + DAG en Airflow

Se entregará al/la candidato/a:

- El **código de la Lambda** (adjunto).
- El **`docker-compose` de Airflow** (adjunto).

Objetivo

Tomar la Lambda provista y **refactorizarla** para ejecutar el proceso con la secuencia: **Extracción → Validación → Escritura**, orquestado en **Airflow**.

Alcance

1. **Modularización**
   - Separar el código en **dos módulos**:
     - `extracción (scraping)`
     - `escritura (persistencia)`
   - Mantener intacta la **lógica de scraping** ya existente (no cambiar decisiones de omitir filas, reglas específicas ni criterios actuales).
2. **Validación**
   - Intercalar una **etapa de validación** entre extracción y escritura.
   - Verificar **tipo de dato** y **regex** por **campo**.
   - Reglas:
     - Si un **campo** no cumple, ese **campo** no se escribe (queda vacío/`NULL`).
     - Si **campos obligatorios** no cumplen, **descartar la fila completa**.
   - Las **reglas** (tipos, regex, obligatoriedad) deben estar **configurables** (archivo simple, por ejemplo YAML/JSON).
3. **Airflow (con `docker-compose`)**
   - Crear un **DAG** con tres tareas: **Extracción → Validación → Escritura**.
   - Usar la **misma base de datos Postgres** que levanta el `docker-compose` de Airflow como destino.
   - **Modificar** el acceso a base de datos en el código (ya **no** usar Secrets Manager).
4. **Idempotencia básica**
   - Evitar inserciones duplicadas de acuerdo con los criterios que ya incorpora el código base (no cambiar la lógica, solo reutilizarla).

Entregables

- **Repositorio** con:
  - Código refactorizado (módulos de extracción, validación y escritura).
  - **DAG de Airflow** funcional con la secuencia indicada.

    - **Archivo de reglas de validación** (tipos/regex/obligatoriedad).
    - **Esquema/DDL** de las tablas destino (para crear si no existen).
    - **README** breve con cómo levantar el entorno y ejecutar el DAG (alto nivel).
    - **Logs** claros que muestren: totales extraídos, descartes por validación y filas insertadas.

Criterios de evaluación

- **Correctitud**: se mantiene la lógica de scraping; la validación respeta reglas; el DAG corre end-to-end y escribe en la DB de Airflow.
- **Diseño**: separación clara por etapas; configuración de reglas sin tocar código.
- **Operabilidad**: repositorio entendible; variables por entorno; README suficiente (sin tutoriales extensos).
- **Calidad**: manejo sencillo de errores y logs útiles.
- **Idempotencia**: no duplica registros.

Entorno

- Se usará el **`docker-compose`** provisto para Airflow.
- La **base Postgres** del mismo `docker-compose` es el **destino** de escritura.
- Credenciales por **variables de entorno**.