Descripción Proceso ETL de carga y procesamiento de información de variables de indicadores

Proyecto: Contrato de prestación de servicios n° 202500136 de 2025 celebrado entre la corporación Ruta N - Medellín y Guane Enterprises S.A.S
Fecha de elaboración: 2026-04-17

Tabla de contenido

Resumen ejecutivo 2
Introducción 2
Zonas del datalake 2
1. Insumos del proceso de ETL 3
1.1. El instrumento de reporte de variables (marco lógico de datos) 3
1.2. Archivo de configuración config.json 4
1.3. Diccionario geográfico (estándar FO-GINF-041) 5
2. Procesamiento de variables tabulares 6
2.1. Fase 1 — Bronce: configuración e ingesta 6
2.2. Fase 2 — Plata: transformación y validación 6
Campos representativos de logs.json (variables) 7
2.3. Fase 3 — Oro: publicación en TBL_VALORVARIABLE 8
2.4. Diagramas de flujo (variables) 9
Transformación del instrumento 9
Validación con catálogo y periodicidad 10
Macro Bronce → Plata → Oro (variables) 11
3. Procesamiento de capas geográficas 11
3.1. Propósito y modelo de capas 12
3.2. Fase 1 — Bronce: extracción SQL 12
3.3. Fase 2 — Plata: validación FO-GINF-041 13
3.4. Fase 3 — Plata: procesamiento SCD2 y geoproceso 13
3.5. Fase 4 — Oro: CRE y VVA geo 14
3.6. Diagrama de flujo (capas) 15
3.7. Logs de capas 16
4. Artefactos y logs 16
5. Pruebas de calidad 17
5.1. Casos de prueba ETL Variables 17
5.2. Casos de prueba ETL Capas 17
6. Anexos 18

Anexo A — Ejemplo de config.json 18
Anexo B — Layout landing zone y datalake 19

Resumen ejecutivo

El presente documento describe los flujos técnicos de ingesta, validación y publicación de datos del Plan de Ordenamiento Territorial (POT) en el datalake del DAP. Operan dos pipelines paralelos orquestados en Azure Data Factory (ADF): **ETL-CARGA-VARIABLES** (instrumento Excel tabular) y **ETL-CARGA-CAPAS** (extracción SQL de capas geográficas). Ambos recorren la misma cadena **Bronce → Plata → Oro**: copia fiel o extracción en Bronce, reglas de negocio y procesamiento en Plata, persistencia en PostgreSQL Oro (vars.*). **Estos procesos no calculan indicadores**: publican variables y capas resultado para que un **proceso externo** evalúe fórmulas y calcule indicadores. La trazabilidad se garantiza mediante artefactos logs.json y lock.json por periodo; la calidad mínima se aplica antes de publicar en Oro.

Introducción

Los actores principales son las **dependencias proveedoras del dato** y el **equipo de datos del DAP**. El objetivo común es garantizar trazabilidad (qué insumo se reportó, en qué periodo y con qué contenido), calidad estructural y de dominio antes de publicar variables en Oro, y recuperación operativa cuando un archivo o capa debe corregirse sin perder el registro original en Bronce.

Variables y capas pueden ejecutarse en el **mismo periodo** (YYYY_MM) en orden libre: no hay dependencia estricta entre ambos pipelines, aunque comparten config.json, catálogo POT en PostgreSQL y convenciones de directorio en el datalake. El detalle operativo día a día permanece en el manual de operación; este informe es la vista técnica de proceso.

Zonas del datalake

Zona	Rol	Formato típico	Ambiente
Bronce	Copia fiel de insumos (inmutable)	.xlsx (variables), .parquet (capas)	bronce-ENV
Plata	Validado y procesado	Parquet columnar, GeoJSON maestra/periodo	plata-ENV
Oro	Publicación consultable	PostgreSQL vars.* (VVA, CRE, HIN)	BD indicadores-ENV

En producción los contenedores Bronce/Plata usan los nombres bronce y plata (sin sufijo); la base Oro es indicadores.

---

1. Insumos del proceso de ETL

Para controlar la recolección e ingesta se acordó un protocolo basado en el **instrumento de reporte de variables** (Excel) y un archivo **config.json** en la landing zone del servidor de origen del DAP. Las capas geográficas por el momento **no** se depositan manualmente en ese servidor: su extracción se define en el mismo config.json y la ejecuta ADF contra PostgreSQL origen.

1.1. El instrumento de reporte de variables (marco lógico de datos)

Se dispone de un libro en formato Excel que concentra, por diseño, tanto las **reglas por columna** como los **hechos reportados**:

- Las **primeras filas** (hasta la fila de encabezado de atributos) constituyen la **grilla de metadatos**: clasificador del campo, observaciones de negocio, si el campo admite nulos, descripción semántica, longitud máxima esperada, tipo de dato lógico y el **nombre del atributo** tal como se usará en el resto del pipeline. Esta grilla está alineada al diseño definido en código (instrument_layout.py).
- A partir de la fila de datos inicial, cada fila representa un **reporte de variable** con valores alineados a esas cabeceras.

Hojas operativas:

Hoja	Contenido
Reporte_de_Variables	Hechos reportados (filas de variables por indicador)
Dominios	Reglas de diseño y dominios por columna

Errores de estructura del diseño (filas de metadatos incompletas, reglas de nulabilidad ausentes) se registran en el stage sheet_layout del artefacto logs.json (véase §4).

Los atributos del instrumento vinculan el dato operativo con su contexto de política pública: identificador y nombre del indicador, identificador y nombre de la variable, fechas de reporte y de validez del periodo, valor y unidad, responsable y entidad que reportan, y fuente documental que respalda el dato. Esta estructura permite auditar no sólo el valor numérico o textual, sino el vínculo indicador–variable y la procedencia del reporte.

Entre las reglas de negocio explícitas en la grilla figuran, por ejemplo: la definición del valor que se asocia al reporte a partir de una fecha dada; y el criterio de que, si una variable alimenta varios indicadores con periodicidades distintas, el reporte debe asegurarse por cada indicador con su correspondiente periodicidad. Estas reglas son criterios de calidad del reporte que el instrumento deja documentados para el usuario reportante y que el ETL usa como referencia de consistencia en las etapas de validación.

1.2. Archivo de configuración config.json

La ingesta hacia Bronce no recorre el árbol de carpetas de forma arbitraria: se gobierna con **config.json**, ubicado en la raíz del share de origen del SHIR en el servidor del DAP
(junto a la jerarquía dependencias/{DEP}/{YYYY_MM}/).

Ese archivo es **estático** respecto de cada corrida: no incorpora la fecha de ejecución ni el periodo concreto, sino las **reglas generales** que el ETL aplicará en todas las cargas.

Clave	**Rol**
configuracion.patron_periodo	Convención de etiqueta de carpeta de periodo (YYYY_MM)
archivos_variables[]	{ dependencia, archivos_esperados } → manifiesto Excel vía data flow VariablesDataFlow
capas_estandarizadas[]	{ dependencia, capa, query } → manifiesto geo vía data flow CapasDataflow

Reglas operativas de las queries de capas (INFO BASE del manual):

- El campo capa debe alinearse al diccionario POT (DIC_NOMBRE).
- Toda exportación debe incluir columna **objectid** (p. ej. id AS objectid cuando el origen usa otro nombre).
- Alias de columnas en el SELECT ≤ 10 caracteres.
- Geometría en CRS **EPSG:9377** (MAGNA-SIRGAS / Origen Nacional IGAC).

La carpeta {DEP}/{PER}/variables/ debe existir en landing aunque esté vacía si la dependencia figura en archivos_variables[]. Cualquier cambio operativo —nueva dependencia, nuevo patrón de archivo o capa adicional— implica actualizar config.json en el servidor de origen antes de la corrida ADF.

Para un ejemplo recortado del archivo operativo, véase Anexo A.

1.3. Diccionario geográfico (estándar FO-GINF-041)

Las capas geográficas se validan contra el diccionario **DiccionarioDatosGeograficos.xlsx** (estándar FO-GINF-041), hoja DiccionarioDatos. El ETL lo materializa como especificación por feature class (GeoCapaSpec) y contrasta cada Parquet Bronce con ese contrato.

Columnas Excel relevantes:

Columna Excel	**Uso en validación**
Nombre del feature class	Clave de grupo de la capa
Alias FC	Nombre humano; base del slug canónico
Geometría / Tipo Dato	Tipo geométrico esperado (Point, Line, Polygon…)
Sistema de coordenadas	CRS declarado → resuelto a EPSG
Nombre del campo	Nombre técnico de columna **obligatoria** en Parquet
Alias Campo	Alias alternativo para match con exportaciones origen
Tipo de dato / Longitud / Acepta nulos	Contrato de atributos

El nombre del archivo Parquet se empareja con feature class o Alias FC (incluye variantes singular/plural y tipos POT conocidos). Criterios duros en validación: columna objectId (case-insensitive), columnas del diccionario presentes, CRS **EPSG:9377**, geometría no nula y topológicamente válida. Columnas extra generan **warning** no bloqueante (stage geo_validation en logs).

---

2. Procesamiento de variables tabulares

Pipeline maestro ADF: **ETL-CARGA-VARIABLES**. Sub-pipelines: 01-CARGA-ARCHIVOS-BRONCE → 02-VALIDAR-ARCHIVOS-PLATA → 03-CARGA-VARIABLES-ORO.

2.1. Fase 1 — Bronce: configuración e ingesta

Propósito: materializar en el datalake una **copia de respaldo e histórico** de los libros Excel entregados en landing, **antes** de aplicar validaciones de negocio.

Actividades del sub-pipeline 01-CARGA-ARCHIVOS-BRONCE:

1. CopiarConfiguracionArchivosServidor — Copy SHIR config.json → {bronce}/config/
2. ProcesarConfiguracion — Data flow VariablesDataFlow → CSV manifiesto en Bronce
3. ListarArchivos — Lookup del manifiesto
4. CopiarArchivosServidorBronce — ForEach + Copy de .xlsx → servidor/pot/dependencias/{DEP}/{PER}/variables/

Cálculo de periodo: si la ejecución no recibe un periodo explícito (p_periodo), el sistema determina el **mes anterior** respecto de la fecha de ejecución. No se permite procesar un periodo posterior al mes calendario actual.

Logs de copia: {bronce}/servidor/pot/logs-carga/{PER}/*.txt (actividad Copy ADF). Un FilesWritten=0 inesperado debe investigarse antes de continuar a Plata.

Política Bronce: **ingesta fiel** — ningún archivo se rechaza por calidad de datos; los rechazos por reglas de negocio ocurren en Plata, dejando siempre el raw disponible.

2.2. Fase 2 — Plata: transformación y validación

Propósito: interpretar el instrumento, transformarlo a representación operativa estable, validar contra catálogo POT y periodicidad de indicadores, y publicar en Plata solo el conjunto coherente.

Cuaderno Synapse: **ETLValidarArchivosVariables** (sub-pipeline 02-VALIDAR-ARCHIVOS-PLATA).

Bloques lógicos:

1. **Bootstrap:** periodo (p_periodo), ambiente (p_env), usuario pipeline (p_correo_usuario — debe existir en pot.TBL_USUARIO), conexión PostgreSQL (p_postgres_connection_string).
2. **Catálogo indicadores:** periodicidades y variables aplicables al periodo desde PostgreSQL (pot.*).
3. **Resolución incremental (default):** reprocesa archivos en estado errors del último logs.json más archivos **nuevos** respecto a esa corrida. Con p_procesamiento_completo=true se fuerza escaneo total del periodo en Bronce.
4. **Validación por workbook/hoja:**
   - Layout y transformación (process_variables_transform, sheet_layout)

- Validación catálogo y periodicidad por variable
- **Completitud de variables por indicador** (indicator_completeness): verifica que estén presentes las variables requeridas en el periodo; **no evalúa fórmulas ni calcula indicadores**

5. **Salidas Plata:** Parquet bajo variables_procesadas/{PER}/; TSV intermedio opcional; **logs.json** en pot/logs/archivos/{PER}/.

Lock Plata: pot/logs/archivos/{PER}/lock.json — indica que el periodo ya fue validado; confirmar con soporte antes de reprocesar.

Campos representativos de logs.json (variables)

Campo	Significado
execution_id	Identificador único de la corrida
errors[]	Incidencias con stage, dependencia, file, sheet
errors[].payload.errores[]	Detalle por fila: fila, atributo, tipo_error, valor_incorrecto
rows_validated_output	Filas que pasaron validación
indicators_due_in_period / indicators_incomplete	Cobertura de variables requeridas por indicador (sin cálculo de indicadores)
timings_ms	Tiempos por etapa (process_files_ms, completeness_ms, …)

Stages frecuentes: process_variables_transform, sheet_layout, indicator_completeness.

2.3. Fase 3 — Oro: publicación en TBL_VALORVARIABLE

Propósito: persistir valores validados en la base de publicación.

Sub-pipeline **03-CARGA-VARIABLES-ORO**, cuaderno **ETLCargaVariablesOro**.

Pasos lógicos:

1. Lee Parquet Plata del periodo en plata/pot/variables_procesadas/{PER}/ — requiere validación Plata OK.
2. **Lock Oro** — impide doble carga; respeta flag p_can_upload_variables.

3. INSERT en vars.TBL_VALORVARIABLE con ON CONFLICT DO NOTHING sobre (VVA_ID_VARIABLE_FK, VVA_PERIODO).
4. Auditoría en TBL_LOGAUDITORIA; JSON de salida con inserted_rows y execution_id.

2.4. Diagramas de flujo (variables)

Transformación del instrumento

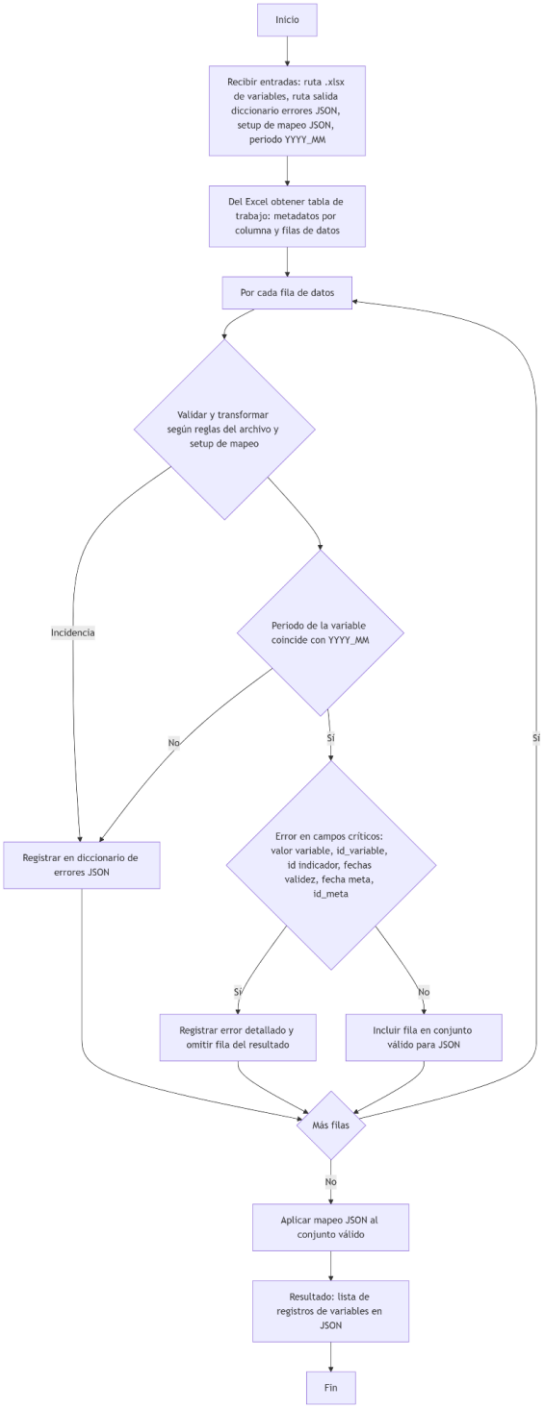


Figura 1: Proceso de transformación del instrumento de avances

Validación con catálogo y periodicidad

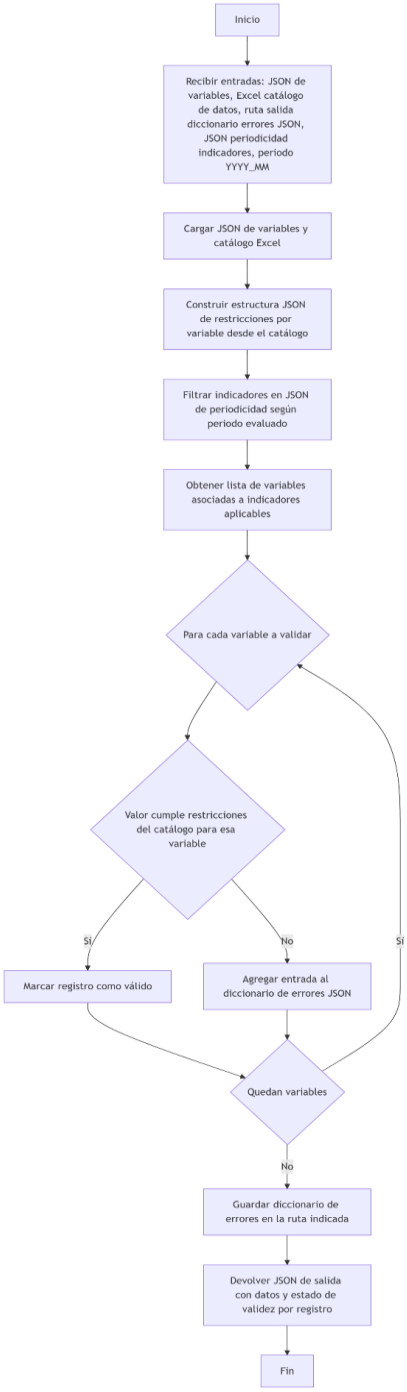


Figura 2: Validaciones variables

Macro Bronce → Plata → Oro (variables)

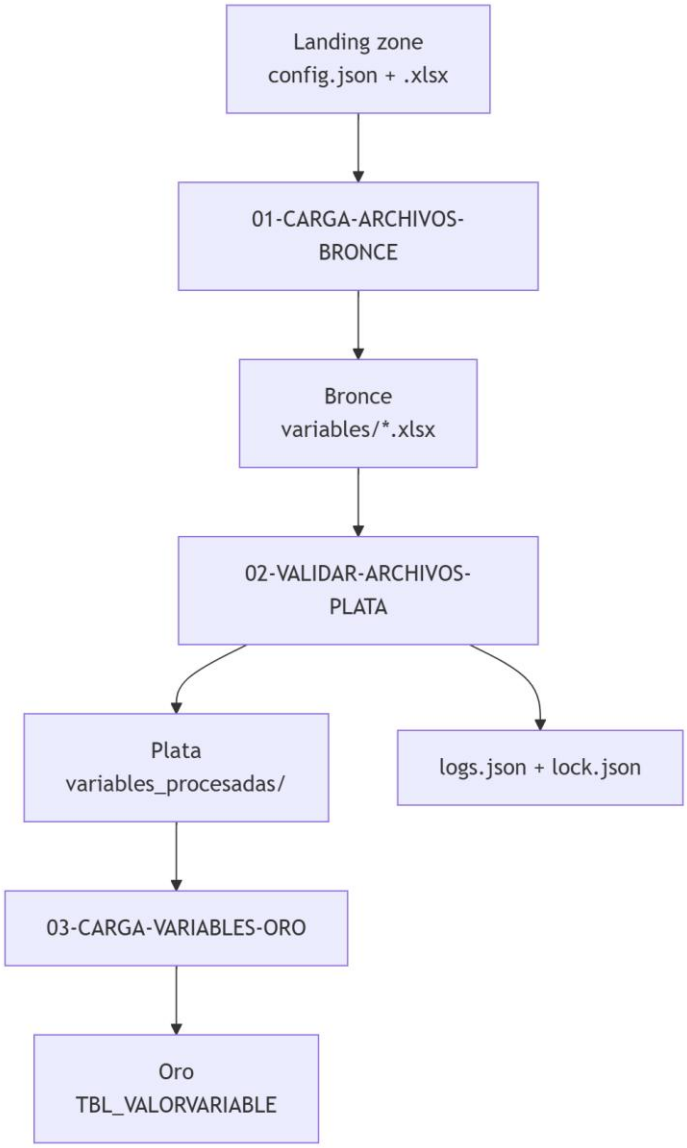


Figura 3: Flujo datalake

3. Procesamiento de capas geográficas

Pipeline maestro ADF: **ETL-CARGA-CAPAS**. Sub-pipelines: 01-CARGA-CAPAS-BRONCE → 02-VALIDAR-CAPAS-PLATA → 03-CARGA-CAPAS-ORO.

Flujo macro: **validate + process** (ETLValidarCapasGeograficas) → **promote_geo_to_oro** → **upload_geo_oro_data** (Oro fuera del cuaderno de procesamiento Plata).

3.1. Propósito y modelo de capas

Término	Definición	Dónde vive
Capa base (Bronce)	Feature class del diccionario FO-GINF-041	Parquet en dependencias/{DEP}/{PER}/capas/
Capa maestra (Plata SCD2)	Historial versionado de entidades con vigencias	capas_procesadas/tablas_maestras/{slug}.geojson
Capa resultado (periodo)	Subconjunto vectorial del periodo listo para indicador	capas_procesadas/periodos/{slug}/{PER}/capa_resultado.geojson
Capa intermedia (runtime)	Salida temporal de un paso de geoproceso (Cap_*, Res_*, V_*)	Solo en memoria durante geoproceso
Producto escalar	Valor numérico agregado asociado a un var_id	valor_escalar.json → luego TBL_VALORVARIABLE

Las capas **Cap_*/Res_*/V_*** son productos de geoproceso en runtime; **no** se depositan en landing ni entran por Bronce como insumo manual.

3.2. Fase 1 — Bronce: extracción SQL

Propósito: materializar en Bronce las capas definidas en config.json mediante consulta a PostgreSQL origen.

Sub-pipeline **01-CARGA-CAPAS-BRONCE:**

1. CopiarConfiguracionCapasServidor — Copy config.json SHIR → Bronce
2. ProcesarConfiguracion — Data flow CapasDataflow → CSV manifiesto capas/queries
3. ListarCapas — Lookup del manifiesto
4. ForEach1 / CopyCapaPostgreSQL — Ejecuta query → servidor/pot/dependencias/{DEP}/{PER}/capas/{capa}.parquet

Las capas **no** se depositan manualmente en la landing zone del DAP; la extracción es exclusivamente vía ADF.

3.3. Fase 2 — Plata: validación FO-GINF-041

Cuaderno **ETLValidarCapasGeograficas** (sub-pipeline 02-VALIDAR-CAPAS-PLATA), bloque de validación:

- **Resolución de fuentes:** incremental (`E ∪ N` — capas en error del último log más capas nuevas) vs p_procesamiento_completo=true (escaneo total Bronce).
- **Criterios duros:** objectId obligatorio; columnas del diccionario; CRS **EPSG:9377**; geometría válida.
- **Salida:** listas capas_aprobadas / capas_rechazadas; logs en pot/logs/capas/{PER}/{execution_id}/logs.json.

Warnings (columnas opcionales ausentes, unidad inferida) no bloquean — stages geo_validation, geo_scalar_unidad.

3.4. Fase 3 — Plata: procesamiento SCD2 y geoproceso

SCD2 paralelo por capa aprobada → actualiza tablas_maestras/{slug}.geojson con columnas de control:

Columna	Significado
sk_row	Surrogate key de fila
id_entidad	Identificador de negocio (desde objectId)
fecha_inicio_vigencia / fecha_fin_vigencia	Ventana temporal SCD2
hash_atributos	Hash SHA-256 de atributos para detectar cambios
tipo_operacion	INSERT / UPDATE / DELETE

Geoprocesos secuenciales desde catálogo POT (TBL_PASOGEOPROCESO): regla de **un solo alias Cap_*** en planes multi-paso; operaciones vectoriales (FILTRO, INTERSECCION, UNION, …) y escalares (AREA, CONTEO, SUMA, …). Productos por periodo: capa_resultado.geojson + valor_escalar.json.

Completitud geo: stage geo_coverage e indicator_completeness en logs — contrasta variables geo esperadas vs escalares producidos; validación de insumos, no cálculo de indicadores.

Lock geo Plata: pot/logs/capas/{PER}/lock.json.

3.5. Fase 4 — Oro: CRE y VVA geo

Sub-pipeline **03-CARGA-CAPAS-ORO**, cuaderno **ETLCargarGeodatosOro:**

1. Auto-descubre último run Plata con scalars / rutas_disposicion.
2. Requiere p_approval_status=approved (aprobación QA).
3. INSERT en vars.TBL_CAPARESULTADO (capas resultado) y escalares en vars.TBL_VALORVARIABLE.
4. **Lock Oro geo** tras carga exitosa — re-ejecución bloqueada sin intervención de soporte.

3.6. Diagrama de flujo (capas)

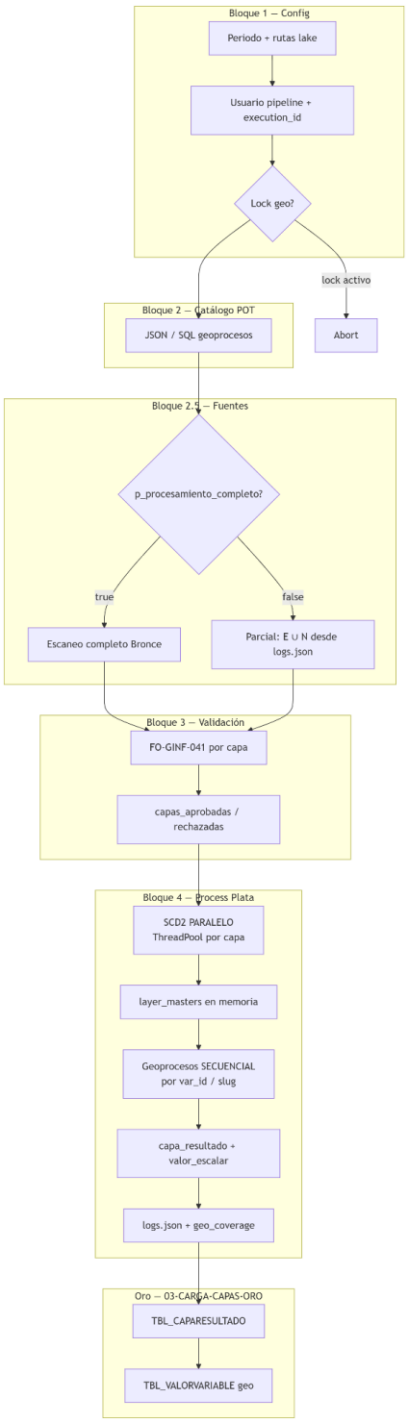


Figura 4: Flujo de capas

3.7. Logs de capas

Basado en sección de **capas** de los ejemplos de logs:

Campo / stage	**Significado**
capas_aprobadas[]	Slugs validados con path Parquet y alias FC
capas_rechazadas[]	Slugs descartados en validación
scalars[]	var_id, scalar_value, valor_escalar_path
geo_coverage	Métricas de cobertura indicadores/capas/geoprocesos
geo_validation	Errores/warnings de contrato FO-GINF-041
geo_scalar_unidad	Warnings de unidad inferida desde diccionario
indicator_completeness	Indicadores con variables geo faltantes

---

4. Artefactos y logs

Rol	**Variables**	**Capas**
Config maestro	config.json en share origen	mismo archivo
Bronce	.xlsx en dependencias/{DEP}/{PER}/variables/	.parquet en dependencias/{DEP}/{PER}/capas/
Plata	Parquet variables_procesadas/{PER}/	GeoJSON maestra + periodo + valor_escalar.json
Oro	vars.TBL_VALORVARIABLE	vars.TBL_CAPARESULTADO + VVA geo
Trazabilidad	logs/archivos/{PER}/logs.json + lock.json	logs/capas/{PER}/…/logs.json + lock.json
Copia ADF	logs-carga/{PER}/*.txt	logs-carga/{PER}/*.txt

Referencia cruzada: el detalle operativo (códigos ERR-CV / ERR-CC, parámetros ADF, re-ejecución por sub-pipeline) permanece en ManualOperacionETLs Este informe documenta la vista técnica de proceso; el manual es la guía del operador.

---

5. Pruebas de calidad

La siguiente información consolida los casos de prueba ejecutados para los procesos de carga de **variables tabulares** y **capas geográficas** del POT Medellín. Los ETL cubren recepción, validación, transformación y publicación en las zonas Bronce, Plata y Oro.

5.1. Casos de prueba ETL Variables

- **CP-001:** Conexión y lectura del catálogo GDB del DAP.
- **CP-002:** Ingesta cruda de capas a Zona Bronce sin alterar origen.
- **CP-003:** Validación del sistema de referencia CTM12.
- **CP-004:** Consistencia de atributos contra el Diccionario de datos.
- **CP-005:** Presencia y validez del campo fecha_actualizacion.
- **CP-006:** Unicidad e inmutabilidad del objectid.
- **CP-007:** Detección de capa sin cambios y extensión de vigencia.
- **CP-008:** Detección de cambios y ejecución de geoprocesos.
- **CP-009:** Gestión de versiones en la Tabla Maestra (Plata).
- **CP-010:** Consolidación y promoción de variables a Zona Oro.
- **CP-011:** Bitácora consolidada de trazabilidad.
- **CP-012:** Reconstrucción multitemporal del estado oficial.
- **CP-013:** Ejecución total, parcial y bajo demanda.
- **CP-014:** Indisponibilidad del servidor del DAP.
- **CP-015:** Acceso de solo lectura a la fuente.
- **CP-016:** Aprobación o rechazo de la carga (UAT).

5.2. Casos de prueba ETL Capas

- **CP-001:** Conexión y lectura mediante SHIR.
- **CP-002:** Manejo de fallo del SHIR.
- **CP-003:** Carga de archivos a Zona Bronce.
- **CP-004:** Calidad de datos en validación Plata.
- **CP-005:** Transformaciones de archivos Excel.
- **CP-006:** Manejo de errores y bloqueo de carga.
- **CP-007:** Trazabilidad y control de lock.
- **CP-008:** Disponibilidad de datos en Oro.

- **CP-009:** Carga parcial por periodo.
- **CP-010:** Control de reprocesamiento con lock.
- **CP-011:** Errores en carga parcial.
- **CP-012:** Trazabilidad con metadatos.
- **CP-013:** Logs de carga parcial.
- **CP-014:** Registro de usuario ETL.
- **CP-015:** Auditoría de carga en tbl_logauditoria.
- **CP-016:** Parametrización de entornos ETL_ENV.
- **CP-017:** Integridad de creado_por_fk.
- **CP-018:** Diagnósticos de rechazo de variables.

---

6. Anexos

Anexo A — Ejemplo de config.json

Define dependencias de variables y capas con queries de extracción. Archivo operativo en share folder del SHIR en servidor de origen del DAP.


```
{
 "configuracion": {
  "patron_periodo": "YYYY_MM"
 },
 "archivos_variables": [
  {
   "dependencia": "Secretaria_Salud",
   "archivos_esperados": ["*.xlsx"]
  },
  {
   "dependencia": "Servicios_Publicos",
   "archivos_esperados": ["*.xlsx"]
  },
  {
   "dependencia": "Gestion_Control_Territorial",
   "archivos_esperados": ["*.xlsx"]
  },
  {
   "dependencia": "Secretaria_Movilidad",
   "archivos_esperados": ["*.xlsx"]
  },
  {
   "dependencia": "Varias_Secretarias",
   "archivos_esperados": ["*.xlsx"]
  }
 ],
```

```
 "capas_estandarizadas": [
   {
    "dependencia": "Secretaria_Movilidad",
    "capa": "ciclorrutas",
    "query": "SELECT id, ST_AsGeoJSON(geom)::text AS geom, objectid, nombre, label, estado, fecha_adop, fecha_actu, shape_leng FROM movilidad.ciclorrutas;"
   },
   {
    "dependencia": "Dpto_Administrativo_Planeacion",
    "capa": "clasificacion_suelo",
    "query": "SELECT id, ST_AsGeoJSON(geom)::text AS geom, objectid, codigo_tra, nombre, clase_suel, categoria_, catdllores, fecha_adop, fecha_actu, shape_leng, shape_area FROM variables.clasificacion_suelo;"
   },
   {
    "dependencia": "Dpto_Administrativo_Gestion_Riesgos_Desastres",
    "capa": "estaciones_siata",
    "query": "SELECT id AS objectid, ST_AsGeoJSON(geom)::text AS geom, descrip, \"long\" AS long, lat, subcuenca, vereda, corregimie, comuna, barrio, direccion, fecha_inst, estado, tipo, red, nombre, codigo FROM variables.estaciones_siata;"
   }
 ]
}
```


(Queries adicionales de capas en el archivo operativo completo para algunas capas: equipamientos, eventos_desastres, muertes_eventos, patrimonio_bic, patrimonio_licbic_a, vias_urbanorural.)

Anexo B — Layout landing zone y datalake

Landing zone (servidor de origen del DAP):

```
{share}/config.json
{share}/dependencias/{DEP}/{YYYY_MM}/variables/*.xlsx
# Las capas NO se depositan aquí — extracción vía ADF desde PostgreSQL origen
```

Datalake ADLS (servidor/pot/):

```
servidor/pot/
├── config/
├── dependencias/{DEP}/{PER}/
│   ├── variables/*.xlsx
│   └── capas/*.parquet
├── logs/
│   ├── archivos/{PER}/logs.json, lock.json
│   └── capas/{PER}/…/logs.json, lock.json
└── logs-carga/{PER}/

Plata (contenedor plata-*):
├── variables_procesadas/{PER}/
```

```
└── capas_procesadas/
    ├── tablas_maestras/{slug}.geojson
    └── periodos/{slug}/{PER}/capa_resultado.geojson, valor_escalar.json
```

Oro: PostgreSQL vars.* (TBL_VALORVARIABLE, TBL_CAPARESULTADO, TBL_HISTORICOINDICADOR)