Universidad San Francisco de Quito

Data Mining

Proyecto 01

---

1) Resumen

Construir un **pipeline de datos de backfill histórico** que extraiga información de **QuickBooks Online (QBO)** para las entidades **Invoices**, **Customers** y **Items**, y la deposite en **Postgres** dentro de un **esquema `raw`**.
La orquestación se realizará con **Mage**, el despliegue con **Docker Compose**, y todas las credenciales/tokens deberán gestionarse mediante **Mage Secrets**.

> **No incluido**: pipeline diario, transformaciones a "clean" y modelamiento dimensional.

---

2) Objetivos de aprendizaje

- Implementar una **ingesta histórica** (backfill) desde una API con **OAuth 2.0**.
- **Orquestar** y **parametrizar** un pipeline en **Mage** con **trigger de una sola ejecución** (one-time).
- Diseñar una **capa RAW** robusta con **payload completo** y **metadatos de ingesta**.
- Aplicar **idempotencia**, **paginación**, **manejo de rate limits**, **reintentos** y **buena observabilidad**.
- Asegurar la **seguridad de secretos** usando **Mage Secrets** para QBO y Postgres.

---

3) Alcance y restricciones

- **Origen:** QBO (Invoices, Customers, Items).
- **Destino:** Postgres, **esquema `raw`** (sin capa clean).
- **Orquestación: tres** pipelines en Mage: **`qb_invoices_backfill`, `qb_customers_backfill`, `qb_items_backfill`**.

- **Programación: trigger one-time** (una única ejecución) para lanzar el backfill parametrizado.
- **Despliegue:** Docker Compose con **Mage**, **Postgres y PgAdmin** en **la misma red**.
- **Zonas horarias:** filtros y marcas de tiempo en **UTC**.
- **Seguridad (obligatoria): todos** los tokens/llaves de QBO y **todas** las credenciales de Postgres deben residir en **Mage Secrets** (no en variables de entorno expuestas ni en el repositorio).

---

4) Requisitos técnicos previos

- App QBO (sandbox) con **Client ID/Secret, Realm ID, y Refresh Token**.
- **Docker** y **Docker Compose** instalados.
- Conocimientos básicos de **APIs**, **OAuth 2.0**, **SQL**, **orquestación** y **Git**.

---

5) Arquitectura esperada (alto nivel)

1. **QuickBooks API** → extracción histórica por entidad (filtros por fecha; paginación; manejo de límites).
2. **Mage** → pipelines `qb_<entidad>_backfill` con parámetros de fechas; **trigger one-time**; logs, métricas y control de errores.
3. **Postgres** → **esquema `raw`** con tablas por entidad, **payload completo** (JSON/JSONB) y **metadatos de ingesta**.
4. **Red Docker** → comunicación por **nombre de servicio** (Mage ↔ Postgres ↔ PgAdmin).
5. **Seguridad** → **Mage Secrets** centraliza todos los secretos (QBO y Postgres).

---

6) Manejo de secretos (requisito obligatorio)

- Crear en **Mage Secrets** todos los valores sensibles:
  - **QBO:** Client ID, Client Secret, Realm ID, Refresh Token y entorno (sandbox/prod).
  - **Postgres:** host (nombre del servicio), puerto, base, usuario y contraseña.
- **No permitido:** secretos en el repo, `.env` con valores reales, capturas con valores visibles.
- **Evidencias requeridas:**
  - Sección “**Gestión de secretos**” en el README con: nombres, propósito, proceso de **rotación** y responsables.

---

7) Diseño funcional del backfill (sin código)

7.1) Parámetros y segmentación

- Los pipelines `qb_<entidad>_backfill` debe aceptar al menos: **`fecha_inicio`** y **`fecha_fin`** (formato ISO, en **UTC**).
- **Segmentación (chunking):** dividir el rango en **días** (o semanas) para controlar volumen, tiempos y reintentos.
- Registrar por cada tramo: fechas procesadas, páginas leídas, filas insertadas/actualizadas, duración.

7.2) Autenticación y extracción

- **OAuth 2.0:** obtener **Access Token** desde el **Refresh Token** al inicio de cada ejecución/tramo.
- **Filtros históricos:** usar **marcas de tiempo** de última actualización o fechas de transacción en **UTC**.
- **Paginación:** leer hasta agotar resultados; limitar el tamaño de la página; **detener** cuando el último lote sea incompleto.
- **Rate limits y errores:** reintentos con backoff exponencial; política de “circuit breaker” si se exceden umbrales; logs claros.

7.3) Capa `raw` en Postgres (requisitos de diseño)

Para **cada entidad** (Invoices, Customers, Items), crear una tabla en `raw` que incluya:

- **Nombre de tabla:** qb_<entidad>, etc.
- **Clave primaria:** `id -> id de la entidad`
- **Payload completo** (JSON/JSONB) para trazabilidad total: `payload`
- **Metadatos obligatorios**:
  - `ingested_at_utc` (timestamp de carga),
  - `extract_window_start_utc` y `extract_window_end_utc`,
  - `page_number`/`page_size`
  - `request_payload`

Idempotencia: upsert por clave primaria; reejecutar un mismo tramo **no** debe duplicar filas.

7.4) Validaciones y calidad de datos

- **Integridad:** claves primarias no nulas y no duplicadas.

- **Volumetría:** conteos por entidad y por tramo; detectar regresiones fuertes (p. ej., “días vacíos” inesperados).
- **Coherencia temporal:** timestamps de ingesta y ventanas en **UTC**, consistentes con los filtros usados.
- **Reprocesos seguros:** evidencia de que reejecutar un tramo deja el mismo resultado (idempotencia).

7.5) Observabilidad y operación

- **Métricas por tramo:** registros leídos/insertados/actualizados/omitidos; duración total y por entidad; páginas leídas.
- **Logging:** eventos por fase (auth, extracción, carga, validaciones) con mensajes claros y códigos de error.
- **Runbook:** cómo **reanudar** desde el último tramo exitoso, cómo **reintentar** selectivamente y cómo **verificar** resultados.

---

8) Trigger y programación del backfill (requisito explícito)

8.1) Trigger one-time (una sola ejecución)

- Configurar un **trigger de una sola vez** en Mage para `qb_<entidad>_backfill`:
    - Fecha/hora de inicio (documentar en **UTC** y su equivalente en **America/Guayaquil**).
    - Parámetros `fecha_inicio` y `fecha_fin`.
- Al concluir exitosamente, **deshabilitar** el trigger o marcar la ejecución como **completada** para evitar relanzamientos.
- Si optan por segmentación temporal **externa** (varios eventos), planificar **múltiples one-time triggers** encadenados y documentar intervalos.

8.2) Evidencias requeridas

- Capturas de la **configuración del trigger one-time** (nombres visibles, horario, parámetros).
- Capturas de **ejecución finalizada** con métricas por tramo y totales.
- Registro de **reintentos** (si los hubo) y del **punto de reanudación**.

---

9) Entregables (en GitHub)

1. **README completo** con:
    - Descripción y **diagrama** de arquitectura.
    - Pasos para levantar contenedores y configurar el proyecto.
    - **Gestión de secretos** (nombres, propósito, rotación, responsables; sin valores).
    - **Detalle de los tres pipelines `qb_<entidad>_backfill`**: parámetros, segmentación, límites, reintentos, runbook.
    - **Trigger one-time**: fecha/hora en UTC y equivalencia a Guayaquil; política de deshabilitación post-ejecución.
    - **Esquema `raw`**: tablas por entidad, claves, metadatos obligatorios, idempotencia.
    - **Validaciones/volumetría**: cómo correrlas y cómo interpretar resultados.
    - **Troubleshooting**: auth, paginación, límites, timezones, almacenamiento y permisos.
2. **Definiciones del proyecto de Mage** y de las pipelines **`qb_<entidad>_backfill`** con su **trigger one-time**.
3. **Definiciones de base de datos** para **`raw`** (tablas y restricciones) claramente documentadas.
4. **Pruebas/validaciones** de calidad y guía para ejecutarlas.
5. **Evidencias** (carpeta):
    - Configuración de **Mage Secrets** (nombres visibles, valores ocultos).
    - Triggers one-time configurado y ejecución finalizada.
    - Tablas `raw` con registros y metadatos.
    - Reporte de volumetría por tramo y evidencia de **idempotencia**.

---

10) Rúbrica de evaluación (100 puntos)

A. Infraestructura Docker (20 pts)

- Mage y Postgres en la misma red; persistencia configurada; arranque y apagado documentados.

B. Orquestación en Mage (25 pts)

- Pipeline **`qb_<entidad>_backfill`** parametrizado; **trigger one-time** configurado; evidencia de ejecución y deshabilitación.
- Políticas de segmentación, reintento y reanudación documentadas.

C. Ingesta desde QBO (25 pts)

- OAuth 2.0 correcto (uso de Refresh Token y manejo de expiración/rotación).
- Filtros históricos en **UTC**; **paginación**; **rate limits** gestionados con backoff.
- **Idempotencia** demostrada (reprocesos sin duplicados).

D. Capa RAW en Postgres (20 pts)

- Tablas por entidad con **clave primaria**, **payload completo** y **metadatos** de ingesta.
- Restricciones mínimas y documentación del esquema.

E. Seguridad y documentación (10 pts)

- **Mage Secrets** para QBO y Postgres (evidencia sin exponer valores).
- README completo, evidencias y troubleshooting.

Bonos (+5 c/u, máx +15):

- Métricas/monitoreo visibles desde Mage (filas, duración, éxito/falla).
- Reporte automático del backfill (resumen final).
- Validaciones adicionales (detección de huecos de fechas o outliers de volumen).

---

11) Checklist de aceptación (copiar en el README y marcar)

- ☐ Mage y Postgres se comunican por **nombre de servicio**.
- ☐ **Todos** los secretos (QBO y Postgres) están en **Mage Secrets**; no hay secretos en el repo/entorno expuesto.
- ☐ Pipelines **`qb_<entidad>_backfill`** acepta `fecha_inicio` y `fecha_fin` (UTC) y segmenta el rango.
- ☐ **Trigger one-time** configurado, ejecutado y luego **deshabilitado**/marcado como completado.
- ☐ Esquema **`raw`** con tablas por entidad, **payload completo** y **metadatos** obligatorios.
- ☐ **Idempotencia** verificada: reejecución de un tramo no genera duplicados.
- ☐ **Paginación** y **rate limits** manejados y documentados.
- ☐ **Volumetría** y validaciones mínimas registradas y archivadas como evidencia.
- ☐ Runbook de reanudación y reintentos disponible y seguido.

---

12) Entrega y defensa

- **Fecha y hora límite:** 10 de septiembre (hora local **de América/Guayaquil**).
- **Entrega:** enlace al **repositorio GitHub** en la carpeta del deber.
- **Una defensa será requerida si hay sospecha de tener plagio de otro compañero o de ChatGPT**