NiFi Weather Data Flow

Descripción

Este flujo de **Apache NiFi** obtiene datos meteorológicos desde la API pública de **Open-Meteo**, los transforma y los almacena en una base de datos **PostgreSQL** alojada en **Supabase**.
Además, se genera un archivo `.json` en el sistema local para validación y respaldo.

El flujo está diseñado como **pipeline ETL ligero** (Extract, Transform, Load).

Procesadores utilizados y su función

1. **InvokeHTTP**
    - **Qué hace:** Llama periódicamente a la API `https://api.open-meteo.com/v1/forecast?...` y obtiene datos en formato JSON.
    - **Configuración clave:**
        - `HTTP Method = GET`
        - `Run Schedule = 300 sec (5 min)` → evita sobrecarga en la API y la base de datos.
    - **Problema encontrado:** Inicialmente lo ejecutábamos cada 5s → generaba cientos de registros en minutos.
    - **Solución:** Ajustamos el `Run Schedule`.
2. **EvaluateJsonPath**
    - **Qué hace:** Extrae los campos relevantes del JSON (latitud, longitud, temperatura, etc.) y los convierte en atributos de FlowFile.
    - **Uso:** Facilita el mapeo hacia las tablas de base de datos.

3. **AttributesToJSON**
    - **Qué hace:** Convierte los atributos extraídos en un objeto JSON.
    - **Uso:** Estándar intermedio para que otros procesadores trabajen con datos uniformes.
4. **ConvertRecord**
    - **Qué hace:** Convierte el JSON a un formato de registro estructurado usando `JsonTreeReader` y `JsonRecordSetWriter`.
    - **Uso:** Necesario para alimentar al procesador `PutDatabaseRecord`.
5. **PutDatabaseRecord**
    - **Qué hace:** Inserta los datos en la tabla `weather_data` de Supabase.
    - **Configuración clave:**
        - `Database Type = Generic`
        - `Statement Type = INSERT`
        - `Database Connection Pooling Service = DBCPConnectionPool`
        - `Table Name = weather_data`
6. **UpdateAttribute**
    - **Qué hace:** Renombra el archivo antes de guardarlo localmente.
    - **Configuración:**
        - `filename = weather.json`
        - Esto hace que siempre se sobreescriba el mismo archivo → evita miles de ficheros.
7. **PutFile**
    - **Qué hace:** Guarda una copia del JSON en el directorio local (`C:/Users/Nhoeli/Desktop/nifi-output/weather.json`).

- **Uso:** Validación y respaldo de los datos descargados.

Base de Datos (Supabase)

- **Tabla usada:** `weather_data`.
- **Problema principal:** Conexión desde NiFi fallaba porque Supabase usa **IPv6 por defecto**.
- **Solución:**
  - Usamos el **Session Pooler** con puerto `5432` (IPv4 compatible).

Cadena de conexión usada en NiFi:

- `jdbc:postgresql://aws-1-eu-west-3.pooler.supabase.com:5432/postgres`
- Usuario: `postgres`
- Password: configurado en `Database Settings` de Supabase.

Nota: En el plan gratuito de Supabase **no hay IPv4 dedicado**, solo el pooler compartido.

Dificultades encontradas

1. **Driver PostgreSQL**
   - Tuvimos que descargar manualmente el `.jar` de PostgreSQL y añadirlo en `lib/` de NiFi para que `DBCPConnectionPool` reconociera el driver.
2. **IPv4 vs IPv6**
   - NiFi no lograba conectar al host `xxxxx.supabase.co` (IPv6).
   - Se solucionó usando el **pooler IPv4** (`aws-1-eu-west-3.pooler.supabase.com`).

3. **Frecuencia de llamadas**
    - Ejecutar cada 5s saturaba la tabla.
    - Se ajustó a 3600s (60 min).