DOCUMENTACION SCRIPT GENERADOR DE TICKETS SINTÉTICOS

GRUPO 1

Contenido

Introducción

Este notebook genera tickets de repostaje sintéticos (facturas simplificadas) para varias empresas de transporte y sus usuarios, usando:

- Un catálogo de estaciones (con provincia, coordenadas, grupo/marca)
- Un mapa de NIF/CIF por compañía/grupo/marca.
- Una tabla de precios por provincia, producto y mes (o por fecha diaria).

El resultado se guarda en:

- `data/tickets/tickets_sinteticos.jsonl` (un ticket por línea).
- `data/tickets/tickets_sinteticos.json` (lista JSON completa).

Entradas, salidas y estructura de carpetas.

Archivos de entrada

Ubicados en /data

- EstacionesDeServicio.csv
  - Columnas flexibles (se autodetectan): id de estación, nombre/rotulo/grupo, provincia, municipio, dirección, latitud/longitud
- nif_empresas_gasolineras_es.csv
  - Columnas flexibles: NIF/CIF y compañía/grupo/razón social y/o marca/rotulo. Se acepta múltiples alias por celda con diferentes separadores
- PreciosProvincia.csv
  - Columnas flexibles: provincia, producto, mes o fecha, y precio. El año se filtra a YEAR=2025

El lector es tolerante a diferentes encoding.

Archivos de salida

Se crean automáticamente

- Data/
  - EstacionesDeServicio.csv
  - Nif_empresas_gasolineras_es.csv
  - PreciosProvincia.csv
  - Tickets/
    - Tickets_sinteticos.jsonl
    - Tickets_sinteticos.json

Parámetros globales y aleatoriedad

- Replicabilidad: Se añade una semilla para asegurar que los tickets obtenidos son coherentes con los empleados
- Rango temporal:
  - `YEAR = 2025`
  - `FECHA_INI = 2025-01-01`, `FECHA_FIN = 2025-07-31`
- Población sintética:
  - `N_EMPRESAS_TRANSP = 3`
  - `USUARIOS_POR_EMPRESA = 9` *(se sobreescribe a 9 más adelante)*
  - `TICKETS_POR_USUARIO = 50`
- Producto/combustible por usuario: 1 producto asignado al azar entre:
  - `"Gasolina 95 E5"`, `"Gasolina 98 E5"`, `"Gasóleo A"`, `"Gasóleo Premium"`
- Métodos de pago:
  - `PESO_TARJETA = 0.85` → 85% "Tarjeta crédito", 15% "Efectivo".
- Cantidad de litros:
  - Distribución triangular: `min=10`, `moda=35`, `max=80` (redondeo a 2 decimales).
- Precio por estación y día:
  - Base mensual por (provincia normalizada, producto canonizado, mes)
  - Desplazamiento de estación determinista por `id_estacion`: `[-0.02, 0.02] €/L`.
  - Ruido diario: `[-0.01, 0.01] €/L`.
- IVA:
  - `IVA_TIPO = 0.21` → cálculo de base imponible/IVA/total coherente por redondeos.

Normalización, limpieza y detección de columnas

Utilidades clave

- `norm_txt(x)`
  Quita tildes, pasa a minúsculas y recorta espacios.
- `_simplify(s)`
  Deja solo `[a-z0-9]` para comparar nombres de columnas robustamente.
- `to_float_locale(series)`
  Convierte números con coma decimal a `float`.
- `prov_key(x)`
  Normaliza nombres de provincia (corrige sinónimos: *vizcaya→bizkaia, guipuzcoa→gipuzkoa, la coruña→a coruña, orense→ourense, etc.*).
- `canon_producto(x)`
  Mapea alias de combustible a los 4 valores canónicos.

Detección flexible de columnas

Para Estaciones:

- Id estación: busca entre `["ideess","id","idestacion","id_estacion","codigo","id_ees s"]` (si no, genera `_idgen`).
- Nombre/rotulo: busca entre `["rotulo","estacion","nombre","razon_social","razonsocial" ,"marca","rótulo"]`.
- Grupo/compañía: busca entre `["grupo","empresa","rotulo","marca","compania","compañia", "rótulo"]`.
- Provincia: obligatoria, varios alias (`"provincia"`, `"desc_provincia"`, etc.) → si no existe, lanza `ValueError`.
- Municipio/dirección: opcional.
- Lat/Lon: detecta por sufijos y sinónimos (lat/lon/long/lng); invierte si parece que están cruzadas; descarta filas sin coordenadas si la cobertura >80%.

Para NIFs:

- Detecta una columna de NIF/CIF y otra de `compañía/grupo` y/o `marca/rotulo`. Si falta, `ValueError`.
- Construye `map_grupo_to_nif` con alias múltiples por marca.

Para Precios:

- Admite esquema por fecha (con `fecha_precio`) o **por mes** (con `mes`); filtra al `YEAR`.
- Convierte precio a float tolerando coma decimal.
- Resultado: diccionario `precios_idx[(prov_norm, prod_canon, mes)] = precio`.
  Si hay filas "Nacional/Total", crea `precios_nac_idx[(prod_canon, mes)]`.

Lógica de precios

1. `precio_base_mes(prov_norm, prod, mes)`
   - Devuelve el precio medio mensual si existe para esa provincia-producto-mes.
   - Si falta, usa media nacional del mes para ese producto.
   - Si tampoco hay nacional, usa la media del mes para ese producto entre todas las provincias.
   - Último recurso: 1.50 €/L.
2. `station_offset(id_estacion)`
   - Desplazamiento determinista uniforme en `[-0.02, 0.02]` por id de estación.
3. `precio_diario(prov_norm, prod, fecha, id_estacion)`
   - `precio = base_mes + offset_estación + ruido_diario`
   - Redondeo a 3 decimales, mínimo 0.5 €/L.

Geolocalización y validación de coordenadas

- `_coords_from_row(row)`:
  - Intenta `lat, lon`; si no parecen válidas en el rango España `(lat 27–44.5, lon −20–5.5)`, prueba invertidas `lon, lat`.
  - Si siguen inválidas, devuelve `NaN, NaN`.
- `generar_ticket`() reintenta hasta 5 veces muestreando estaciones aleatorias hasta conseguir coordenadas válidas; si no, puede dejar `lat/lon = null`.

Identificadores y fiscalidad

- NIF de la gasolinera:
  - `nif_de_grupo(grupo_texto)` busca en `map_grupo_to_nif`.
  - Si no hay, genera un CIF plausible con `cif_generate()` (control de dígito/letra basado en reglas españolas simplificadas).
- Cálculo de importes (`calcular_importes`):
  - `total = litros * precio_unit` (2 decimales).
  - `base = total / (1 + IVA), iva = total - base` (redondeados).
  - Ajusta centésimas para que `base + iva == total`.

Modelo de datos de salida (Ticket JSON)

- Cada **ticket** (`dict`) contiene:


```
{
  "idTicket": "T-86837D3A0D41",
  "idEmpresa": "EMP001",
  "empresaNombre": "Transporte_01 S.L.",
  "idUsuario": "EMP001-U1",
  "fechaEmision": "2025-04-13",
  "horaEmision": "04:23:15",
  "metodoPago": "Tarjeta crédito",
  "estacion": {
    "id": "1492",
    "nombre": "COOP. LA SIBERIA EXTREMEÑA",
    "provincia": "BADAJOZ",
    "municipio": "TALARRUBIAS",
    "direccion": "AVENIDA EXTREMADURA, 118",
    "lat": 39.042917,
    "lon": -5.234583,
    "grupo": "COOP. LA SIBERIA EXTREMEÑA",
    "nifEmpresa": "U40781619"
  },
  "lineas": [
    {
      "producto": "Gasóleo A",
      "litros": 17.44,
      "precioUnitario": 1.46,
      "importe": 25.46
    }
  ],
  "baseImponible": 21.04,
  "iva": 4.42,
  "total": 25.46,
  "moneda": "EUR",
  "tipoDocumento": "Factura simplificada"
},
```

Flujo de generación

1. Carga/normalización de estaciones (`est_std`), NIFs (`map_grupo_to_nif`) y precios (`precios_idx` / `precios_nac_idx`).
2. Empresas y usuarios:
   - `empresas`: `EMP001...EMP003`
   - `usuarios`: por empresa, USUARIOS_POR_EMPRESA=9, con 1 producto asignado por usuario.
3. Pool de estaciones: `EST_POOL` con `lat/lon` numéricos; asegura que hay > 0 estaciones.
4. Generación:
   - `generar_todos()` itera empresas→usuarios→`TICKETS_POR_USUARIO` y llama a `generar_ticket()`.
5. Escritura a disco:
   - JSONL y JSON en `data/tickets/`.

Funciones principales (índice breve)

- **Lectura/Limpieza**
  - `load_csv_guess(path)` – lector robusto CSV.
  - `norm_txt`, `_simplify`, `to_float_locale`, `prov_key`, `canon_producto`.
- **Precios**
  - `precio_base_mes`, `station_offset`, `precio_diario`.
- **Identificación fiscal**
  - `nif_de_grupo`, `cif_generate`.
- **Aleatoriedad y formato**
  - `random_fecha`, `random_hora`, `random_litros`, `elegir_metodo_pago`,
  - `str_fecha`, `str_hora`, `round2`, `round3`.
- **Cálculo económico**
  - `calcular_importes`.
- **Coordenadas**
  - `_to_float_locale`, `_coords_from_row`.
- **Modelo y generación**
  - `Empresa`, `Usuario` (dataclasses),
  - `generar_ticket(empresa, usuario)`,
  - `generar_todos()`.

Validaciones y asunciones

- Se asume que **meses disponibles** en Precios cubren **enero–julio** (1–7).
  El código filtra: `parsed = parsed.loc[parsed["_mes"].isin(range(1,8))]`.
- Si falta precio provincial, se usa nacional; si no hay, media del mes; si no, 1.50 €/L.
- Lat/Lon:
  - Intento de autocorrección cuando están invertidas.
  - Si no son plausibles, null.

- IVA fijo al 21%.
- Productos: 4 categorías canónicas.
- Usuarios por empresa**:** el valor inicial `USUARIOS_POR_EMPRESA=3` se sobrescribe a `9` (ojo a esto si reusas la celda anterior).