DOCUMENTACION SCRIPT GENERADOR DE TICKETS SINTÉTICOS – PEAJE

GRUPO 1

Contenido

Introducción 3

Entradas, salidas y estructura de carpetas. 3

Archivos de entrada 3

Archivos de salida 3

Dependencias y utilidades 3

Normalización, de provincias y detección de tramo 4

Carga de tickets base 4

Generacion de tickets de peaje 5

Exportacion y aplanado 5

Flujo de generacion 6

Diferencias clave: Gasolina vs Eléctrico 6

Introducción

Estos notebook generan tickets sintéticos de PEAJE a partir de:

- Tickets base de gasolina (`tickets_sinteticos.json`) o eléctricos (`tickets_ev_sinteticos.json`).
- Tabla de peajes de vehículos ligeros (`peajes_ligeros.csv`) con autopistas, tramos y precios.

Produce documentos de pejae cohetentes con: concesionaria, autopista, tramo (entrada-salida), importe y hora de paso ajustada de forma determinista para variabilidad más realista posible

Flujos:

- Peaje Gasolina → Categoría de vehículo "B"
- Peaje Eléctrico → Categoría de vehículo "A"

Entradas, salidas y estructura de carpetas.

Archivos de entrada

- `tickets_sinteticos.json` (GASOLINA) o `tickets_ev_sinteticos.json` (ELÉCTRICO).
- peajes_ligeros.csv (tarifas de peajes para ligeros, con autopista y tramo).
- En la versión eléctrica, se incluye `find_file()` para localizar ficheros por el árbol de carpetas.

Archivos de salida

- GASOLINA → `tickets_peaje.json` y `tickets_peaje.csv`
- ELÉCTRICO → `tickets peaje electrico.json` y `tickets peaje electrico.csv`

Dependencias y utilidades

- Python estándar: `json, uuid, hashlib, unicodedata, re, pathlib, datetime, random`.
- `pandas` para manipulación tabular.
- Funciones clave
  - `strip_accents() y norm_prov()` → normalización robusta de provincias (quita acentos, unifica sinónimos).
  - `split_tramo()` → divide el campo Tramo en origen y destino.
  - `eur_str()` → formatea precios como string con coma y símbolo: `5,90 €`.
  - `gen_ref()` → referencia única tipo `PASO-XXXXXXXX`.
  - `hora_desplazada()` → ajusta la hora ±180 min de forma determinista (según clave estable), limitada al mismo día.

Normalización, de provincias y detección de tramo

`norm_prov()`

- Elimina acentos y normaliza variantes:
  - GASOLINA: `VIZCAYA→BIZKAIA, GUIPUZCOA→GIPUZKOA, LA CORUÑA→A CORUÑA, GERONA→GIRONA, LÉRIDA→LLEIDA, ALAVA/ARABA→ALAVA`, etc.
  - ELÉCTRICO (amplía): además unifica `MADRID, COMUNIDAD DE→MADRID, MURCIA, REGION DE→MURCIA, ARAGON→ZARAGOZA, PAIS VASCO→BIZKAIA, PALMAS LAS→LAS PALMAS`, etc.

`load_peajes()`

- Estándar de columnas esperadas (renombra si procede):
  - `{concesionaria, autopista, tramo, precio_ligeros}` y opcional `fecha_tarifa`.
- Procesamiento
  - Se separa `tramo` en `origen` y `destino`.
  - `precio_ligeros` a `float` (soporta coma decimal).
  - `autopista_norm`: mayúsculas sin espacios, p. ej. `AP-68`.
  - `provincias_norm`: lista de provincias para la autopista (a partir de `AUTOPISTA_PROVINCIAS`).
  - Si faltan provincias, intenta deducir con palabras clave del tramo (`PROV_KEYWORDS`).
  - Se filtra a filas con `provincias_norm` no vacía.

`build_index_por_prov()`

- Construye un índice: `{provincia_norm → [ {concesionaria, autopista, origen, destino, precio_ligeros}, … ] }` que acelera la selección de tramos por provincia.

Carga de tickets base

`load_base_tickets()` proyecta los tickets (gasolina o eléctricos) a las columnas mínimas:

```
{
  'idTicket', 'idEmpresa', 'empresaNombre', 'idUsuario',
  'provincia', 'provincia_norm', 'fecha', 'hora', 'metodoPago'
}
```

- GASOLINA: `PATH_TICKETS = "tickets_sinteticos.json"`
- ELÉCTRICO: `PATH_TICKETS = find_file("tickets_ev_sinteticos.json")`

Generacion de tickets de peaje

`generar_ticket_peaje(row, idx_peajes)`

- Toma la `provincia_norm` del ticket base y busca candidatos en `idx_peajes`.
- Selecciona pseudoaleatoriamente un tramo (con `Random(42))`.
- Calcula `hora_alt` con `hora_desplazada()` a partir de una clave estable: `idTicket`, o `idUsuario`, o bien `provincia+fecha+hora`.
- Ensambla el diccionario del ticket de peaje:
  - GASOLINA: `"categoriaVehiculo": "B"`
  - ELÉCTRICO: `"categoriaVehiculo": "A"`
  - Importe como string `("5,90 €")`, referencia `PASO-XXXX`, copia de empresa/usuario y provincia original.


```
{
  "tipoDocumento": "Peaje",
  "concesionaria": "Estatal",
  "autopista": "AP-61",
  "localizacion": {
    "tramo": "Conexión AP-6 (El Espinar) ┼ Ortigosa",
    "entrada": "Conexión AP-6 (El Espinar)",
    "salida": "Ortigosa"
  },
  "fechaHora": "2025-04-13 01:32:15",
  "categoriaVehiculo": "A",
  "importe": "1,40 €",
  "ivaIncluido": "IVA incluido",
  "formaPago": "Tarjeta crédito",
  "referencia": "PASO-B279B985",
  "idEmpresa": "EMP001",
  "empresaNombre": "Transporte_01 S.L.",
  "idUsuario": "EMP001-U28",
  "provincia": "Madrid"
```


Exportacion y aplanado

`generar_y_exportar_JSON(base_df, idx_peajes, out_json, out_csv=None)`

- Recorre los tickets base → genera peajes → devuelve lista de tickets.
- Escribe JSON con indentación.
- Si se pasa `out_csv`, exporta un CSV aplanado.

`flatten_for_csv(tk)` → Convierte localización anidada en columnas planas, la cabecera típica es la siguiente

`tramo,entrada,salida,tipoDocumento,concesionaria,autopista,fechaHora,categoriaVehiculo,importe,ivaIncluido,formaPago,referencia,idEmpresa,empresaNombre,idUsuario,provincia`

Flujo de generacion

1. Carga/normalización de tickets base (gasolina o eléctrico) y normalización de provincias
2. Carga de peajes, normalizar y enriquecer provincias (`provincias_norm`)
3. Construir idnice por provincia → Tramos candidatos
4. Generar tickest de peaje por cada ticket base
5. Exportar a JSON

Diferencias clave: Gasolina vs Eléctrico

- Rutas de entrada/salida
  - GASOLINA → `tickets_sinteticos.json` → `tickets_peaje.json` y `tickets_peaje.csv`.
  - ELÉCTRICO → `tickets_ev_sinteticos.json` (vía `find_file`) → `tickets peaje electrico.json` y `tickets peaje electrico.csv`.
- Normalización de provincia: la versión eléctrica amplía sinónimos y variantes administrativas.
- Categoría de vehículo: “B” (gasolina) vs “A” (eléctrico).
- Mensajes de log: gasolina imprime resumen y rutas para control de calidad; eléctrico retorna la lista