

Scripting GS1

Indice

1. Estructura del Scripting GS1 Helper
2. Extractor de XLSX
3. Motor Loader Balanced / Procesar lotes masivos GTINs y hacer mas balanceado y carga efectiva en menor tiempo, con respuesta feedback al final. Reporte CSV
4. FAQ

1. Estructura del Scripting

El script actual esta conformado de un [gtin_extractor.py](#), para poder leer los XLSx y generar los gtins.

El [gtin_engineLoader_balanced.py](#), nos va servir para procesar el [gtin_extracted.txt](#) generado por el [gtin_extractor.py](#). Puede cargar por lotes de 100, 500 a 1000,

o igual podemos asignar GTINs de forma manual, sin leer el xlsx. La ventaja que nos proporciona el [engineLoader.py](#), es que podra cargar en forma concurrente, y balanceado, y cargar

en forma de lotes, en forma optima y nos va devolver un reporte en CSV, sobre el status generar de la carga. exitosas, fallidas, y reintentos y graficas en. [batch_comparison_results/](#).

Mas adelante les explicamos como usarlo.

Automático ▾



```
gtin_processor_project/
├── requirements.txt
├── gtin_extractor.py
├── gtin_engineLoader_balanced.py
├── Productos_a_cargar.xlsx
├── gtins_extracted.txt (se genera)
├── venv/
├── batch_comparison_results/ (alli se obtiene reporte, de como fue la
carga y observar estadisticas, exitos y pendientes)
└── logs/
```

2. Extractor de XLSx

Applicativo para extraer los gtins en los xlsx que tenga el cliente

Estructura de archivos:

Automático (Lua) ▾



```
tu_carpeta/  
├── gtin_extractor.py      (motor para extraer GTINs)  
├── Productos_a_cargar.xlsx (tu archivo Excel)  
└── gtins_extracted.txt   (output: se generará automáticamente)
```

Características principales:

1. Extrae GTINs de la primera columna de tu archivo Excel de 24MB
2. Valida los GTINs (solo números de 8-14 dígitos)
3. Divide en lotes de 1000 automáticamente
4. Genera código Python en el formato que solicitas

Uso del script: **Importante!**

1. Instala las dependencias:

Automático ▾



```
pip install pandas openpyxl
```

2. Coloca el archivo **Productos_a_cargar.xlsx** en la misma carpeta
3. Ejecuta el script:

Automático ▾



```
python gtin_extractor.py
```



Salida esperada:

El script generará un archivo `gtins_extracted.py` con este formato:

Automático (Bash) ▾



```
# Lista de GTINs dividida en lotes de 1000
# Total de GTINs: 90,000

# Lote 1 - 1000 GTINs
GTINS_LOTE_1 = [
    "8808563464114", "7506022300843", "887961858785", "606068888606",
    "8002820001955", "27045773751", "785639100474", "7730720009009",
    # ... más GTINs
]

# Lote 2 - 1000 GTINs
GTINS_LOTE_2 = [
    "7501967779983", "7503005496128", "7501088919381", "94922315717",
    # ... más GTINs
]

# Lista consolidada de todos los lotes
ALL_GTINS = [
    *GTINS_LOTE_1,
    *GTINS_LOTE_2,
    *GTINS_LOTE_3,
    # ... todos los lotes
]
```



Optimizaciones incluidas:

- **Memoria eficiente:** Solo lee la primera columna

- **Validación de GTINs:** Filtra valores inválidos
- **Preserva ceros iniciales:** Mantiene GTINs como strings
- **Manejo de errores:** Logs detallados del proceso
- **Estadísticas:** Muestra información del procesamiento

3. Motor Loader Balanced / Script para cargar Gtins

gtin_engineLoader_balanced.py

Applicativo para tomar los GTINs y subirlos al GS1, y genere las descripciones

1. Lee automáticamente del archivo **gtins_extracted.txt**
2. Respeta el límite de **TOTAL_GTINS_TO_PROCESS = 1000**
3. Mantiene toda la lógica de chunks, workers y reintentos
4. Añade flexibilidad para elegir lotes específicos

Modos de uso:

Modo 1: Usar primeros 1000 GTINs del Excel

Automático (Python) ▾



```
USE_EXTRACTED_FILE = True
MANUAL_BATCHES = []           # Vacío = usar todos, pero limitado a 1000
```

Modo 2: Usar primeros 1000 GTINs del Excel

python

Automático (Python) ▾



```
USE_EXTRACTED_FILE = True
MANUAL_BATCHES = ["GTINS_LOTE_1"] # Solo lote 1
```

Modo 3: Usar múltiples lotes específicos

Automático (Python) ▾



```
USE_EXTRACTED_FILE = True
MANUAL_BATCHES = [
    "GTINS_LOTE_1",
    "GTINS_LOTE_2" # 2000 GTINS, pero limitado a 1000
]
```

Modo 4: Usar carga manual, y poner lotes gtins propios

Automático (Python) ▾



```
USE_EXTRACTED_FILE = False
### --- codigo ---##
MANUAL_BATCHES = True
BACKUP_GTINS = [
    "07502209290686", "07501943474307", "07506052540714", "00613008738884"
    "07501010789211", "07502214983573", "07502214983726", "07502214983726"
]
```

Configuración su acceso de API y Auth0: **Importante!**

(batch_processor_enhanced.py)

Automático (PowerShell) ▾



```
API_URL = "http://127.0.0.1:8000/api/v1/product/description/generate"
AUTH0_URL = "https://dev1-gs1mx.us.auth0.com/oauth/token"
AUTH0_PAYLOAD = {
    "grant_type": "password",
    "username": "jflores@gs1mexico.org",
    "password": "test123$%",
```

```
"audience": "https://api.gs1.neurotry.services",
"scope": "openid profile email",
"client_id": "BKrpZDHjtvbzcVjXQN4RioAJY8Jk4m7b",
"client_secret":
"toK2LxRXg5nKa0sN6mXW4KnbiqShTbPWrpJpuddEJCLn0GJfiMK69iBx0K0km9HrV"
}
```

Technical pasos, para instalar preparar ambiente:

Automático (Bash) ▾



1. Crear entorno virtual

```
python3 -m venv venv
```

2. Activar entorno

```
source venv/bin/activate          # Linux/Mac
```

```
venv\Scripts\activate            # Windows
```

3. Instalar dependencias

```
pip install -r requirements.txt
```

Listo a correr, para cargar

#1 ero parametrizar como queremos correr - **Modos de Uso 1,2 o 4**, una vez ajustado.

#Nota: recomiendo probar primero con lote menor de 30. y luego irlo ampliando de 50, a 100, y a 1000.

Automático ▾



```
python3
```

Resultados

Allí podremos ver en CSV el resultado, y ver cuales quedaron exitos, y pendientes

📁 batch_comparison_results/

Estructura del proyecto actual

Automático ▾



```
gtin_processor_project/
├── requirements.txt
├── gtin_extractor.py
├── gtin_engineLoader_balanced.py
├── Productos_a_cargar.xlsx
├── gtins_extracted.txt (se genera)
├── venv/
├── batch_comparison_results/ (allí se obtiene reporte, de como fue la
carga y observar estadísticas, exitos y pendientes)
└── logs/
```

FAQ

===== SOLUCIÓN DE PROBLEMAS

=====

🩺 SOLUCIÓN DE PROBLEMAS COMUNES:

✗ Error: 'python3: command not found'

Solución: Instalar Python 3.8+ desde python.org

✗ Error: 'No module named pandas'

Solución: Activar el venv y reinstalar: pip install pandas

✗ Error: 'Permission denied'

Solución: No usar sudo, usar entorno virtual

✗ Error: 'QueuePool limit reached'

Solución: Reducir NUM_WORKERS en el script

✗ Error: Archivo Excel muy grande

Solución: Usar solo lotes específicos en MANUAL_BATCHES