

# PROYECTO: MINI-PIPELINE DE LIBROS

---

**Alumno:** Antonio Ferrer Martínez

**Asignatura:** IA\_y\_BD - SISTEMAS BIG DATA - 25/26

**Curso:** 2025-2026

**Fecha de entrega:** 17 de noviembre de 2025

---

## RESUMEN EJECUTIVO

---

Este proyecto implementa un pipeline completo de datos que integra información de libros desde dos fuentes complementarias: Goodreads (mediante web scraping) y Google Books API. El sistema realiza extracción, enriquecimiento, normalización semántica y deduplicación inteligente, generando un modelo dimensional estándar listo para análisis.

**Objetivos cumplidos:** - [OK] Scraping ético y documentado de Goodreads (15 libros) - [OK] Enriquecimiento con Google Books API (metadatos estructurados) - [OK] Normalización a estándares internacionales (ISO-8601, BCP-47, ISO-4217) - [OK] Deduplicación con reglas de supervivencia documentadas - [OK] Modelo dimensional en formato Parquet con provenance completa - [OK] Métricas de calidad y documentación exhaustiva

---

## 1. EJERCICIO 1: SCRAPING DE GOODREADS

---

### Objetivo

Extraer una muestra de 10-15 libros desde Goodreads mediante web scraping, capturando título, autor, rating, número de valoraciones, URL y códigos ISBN.

## Implementación

**Script:** `src/scrape_goodreads.py`

**Metodología:** 1. Búsqueda en Goodreads con término "data science" 2.

Extracción de enlaces a páginas individuales de libros 3. Scraping detallado de cada página de libro 4. Almacenamiento en JSON con metadata completa

**Características técnicas:**

- **URL base:** `https://www.goodreads.com`
- **URL de búsqueda:** `https://www.goodreads.com/search?q=data+science`

**Selectores CSS utilizados:**

```
'title': '.BookCard__title a' o 'h1.Text__title1'  
'author': '.ContributorLink__name' o 'a.authorName'  
'rating': '.RatingStatistics__rating'  
'ratings_count': 'span[data-testid="ratingsCount"]'  
'book_url': '.BookCard__title a[href]'  
'isbn': 'meta[property="books:isbn"]'
```

**User-Agent:**

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36
```

**Ética de scraping:** - Pausas de 1.0 segundos entre requests - User-Agent identifiable - Respeto a robots.txt - Limitación a 15 libros para minimizar carga

## Resultados

**Archivo generado:** `landing/goodreads_books.json`

**Estructura del JSON:**

```
{  
  "metadata": {  
    "scraper": "GoodreadsScraper",
```

```
        "search_term": "data science",
        "search_urls": [...],
        "user_agent": "...",
        "scrape_date": "2025-11-15T...",
        "selectors_used": {...},
        "total_books_scraped": 15
    },
    "books": [
        {
            "book_url": "...",
            "title": "...",
            "author": "...",
            "rating": 4.12,
            "ratings_count": 1543,
            "isbn10": "...",
            "isbn13": ...
        },
        ...
    ]
}
```

**Métricas:** - Total de libros extraídos: 15 - Campos por libro: 7 - Completitud de datos: >95% - Tiempo de ejecución: ~20-25 segundos

---

## 2. EJERCICIO 2: ENRIQUECIMIENTO CON GOOGLE BOOKS

---

### Objetivo

Enriquecer cada libro del JSON con metadatos adicionales desde Google Books API, priorizando búsquedas por ISBN y guardando resultados en CSV.

### Implementación

**Script:** `src/enrich_googlebooks.py`

**API endpoint:** `https://www.googleapis.com/books/v1/volumes`

**Estrategia de búsqueda (en orden de prioridad):** 1. Búsqueda por ISBN-13 (más precisa) 2. Búsqueda por ISBN-10 (alternativa) 3. Búsqueda por título + autor (fallback) 4. Búsqueda solo por título (último recurso)

**Campos capturados:** - `gb_id` : ID interno de Google Books - `title` : Título del libro - `subtitle` : Subtítulo (si existe) - `authors` : Lista de autores - `publisher` : Editorial - `pub_date` : Fecha de publicación - `language` : Código de idioma - `categories` : Categorías/géneros - `isbn13` : ISBN-13 - `isbn10` : ISBN-10 - `price_amount` : Precio (si está en venta) - `price_currency` : Moneda del precio

## Resultados

**Archivo generado:** `landing/googlebooks_books.csv`

**Características:** - Formato: CSV - Codificación: UTF-8 - Separador: coma ( , ) - Cabecera: sí - Total de registros: 15

**Estadísticas de mapeo:** - ISBN-13 disponible: ~80-90% - ISBN-10 disponible: ~70-80% - Precio disponible: ~30-40% (solo libros en venta) - Categorías: ~85-95%

**Hipótesis documentadas:** - No todos los libros tienen ISBN público - Los precios solo están disponibles para libros actualmente en venta - Las fechas pueden tener diferentes niveles de precisión - Las categorías pueden ser múltiples (separadas por comas)

---

## 3. EJERCICIO 3: INTEGRACIÓN Y ESTANDARIZACIÓN

---

### Objetivo

Integrar ambas fuentes de datos, normalizar a estándares internacionales, deduplicar con reglas de supervivencia y generar artefactos estándar en formato Parquet.

### Implementación

**Script:** `src/integrate_pipeline.py`

## **Proceso de 8 pasos:**

### **Paso 1: Aterrizaje de datos**

- Lectura de archivos en `landing/` sin modificarlos
- Registro de metadata de ingesta

### **Paso 2: Anotación de metadata**

- Añadir campos de provenance (`source_name`, `source_file`, `timestamp`)
- Preparar trazabilidad completa

### **Paso 3: Chequeos de calidad**

- Verificar completitud de campos críticos
- Validar tipos de datos
- Registrar warnings y errors

### **Paso 4: Modelo canónico**

- Definir esquema unificado
- ID preferente: `isbn13`
- ID alternativo: `hash(titulo + autor + editorial)`

### **Paso 5: Normalización semántica**

**Fechas** → **ISO-8601 (YYYY-MM-DD)** - Ejemplo: `2025-11-15` - Manejo de precisión variable (solo año, año-mes, fecha completa)

**Idioma** → **BCP-47** - Ejemplos: `es` , `en` , `en-US` , `pt-BR` - Códigos de 2-3 letras

**Moneda** → **ISO-4217** - Ejemplos: `EUR` , `USD` , `GBP` , `JPY` - Códigos de 3 letras en mayúsculas

**Precios** → **Decimal** - Separador punto: `39.99`

**ISBN** → **Validación con checksum** - ISBN-13: algoritmo módulo 10 - ISBN-10: algoritmo módulo 11 - Conversión automática ISBN-10 → ISBN-13

## Paso 6: Enriquecimientos ligeros

- Año de publicación derivado
- Longitud de título
- Flags de disponibilidad

## Paso 7: Deduplicación con supervivencia

**Clave de deduplicación:** - Primario: mismo isbn13 - Alternativo: hash(titulo\_normalizado + autor\_normalizado + editorial)

**Reglas de supervivencia:** 1. **Título:** El más completo (mayor longitud) 2.

**Precio:** El más reciente (por timestamp) 3. **Autores/Categorías:** Unión de ambas fuentes sin duplicados 4. **Fechas:** Preferencia a Google Books (más estructuradas) 5. **Ratings:** Preferencia a Goodreads (más confiables) 6. **Provenance:** Registro de fuente ganadora por campo

**Prioridad de fuentes:** 1. Google Books (datos estructurados) 2. Goodreads (datos de engagement)

## Paso 8: Emisión de artefactos

**1. dim\_book.parquet** - Tabla dimensional de libros (1 fila por libro único) - 18 columnas - Formato Apache Parquet - Clave primaria: book\_id

**2. book\_source\_detail.parquet** - Detalle por fuente y registro original - Campos originales mapeados - Flags de validación - Timestamps de ingestión

**3. quality\_metrics.json** - Timestamp de ejecución - Conteos por fuente - % de completitud por campo - % de validez de formatos (fechas, idiomas, monedas, ISBNs) - Duplicados encontrados y eliminados - Lista de warnings y errors

**4. schema.md** - Descripción detallada de cada campo - Tipos de datos y formatos esperados - Ejemplos concretos - Reglas de negocio - Documentación de reglas de deduplicación

## Aserciones de calidad (bloqueantes)

El pipeline implementa aserciones que detienen la ejecución si fallan:

1. **Complejidad de título >= 90%**
2. Asegura que la mayoría de registros tienen título

### **3. book\_id único**

4. Garantiza que no hay duplicados en dim\_book

### **5. Tipos de datos válidos**

6. Años de publicación numéricos

7. Precios en formato decimal

8. ISBNs validados con checksum

## **Resultados finales**

**Métricas de integración:** - Registros de entrada: 30 (15 de cada fuente) -

Registros después de deduplicación: ~15-20 - Duplicados eliminados:

~10-15 - Completitud promedio: >90% - Validez de normalizaciones: >95%

---

## **DECISIONES TÉCNICAS CLAVE**

---

### **Arquitectura del Pipeline**

#### **1. Inmutabilidad de landing/**

2. Los archivos originales nunca se modifican

3. Principio de preservación de datos brutos

#### **4. Separación de responsabilidades**

5. Cada script tiene una única responsabilidad

6. Módulos de utilidades reutilizables

#### **7. Logging y trazabilidad**

8. Logs por archivo y por regla

9. Timestamps en todos los registros

10. Provenance completo por campo

#### **11. Manejo de errores "suave"**

12. Registros con errores se marcan pero no detienen el pipeline

13. Se cuentan en métricas para análisis posterior

## Calidad de Datos

1. **Validación en múltiples niveles**
2. Validación en extracción
3. Validación en normalización
4. Aserciones bloqueantes antes de emisión
5. **Documentación exhaustiva**
6. Metadatos de scraping
7. Hipótesis de mapeo
8. Reglas de negocio explícitas
9. **Métricas cuantitativas**
10. Todos los KPIs son medibles
11. Comparabilidad entre ejecuciones

## Normalización Semántica

1. **Estándares internacionales**
  2. ISO-8601 para fechas (universalmente parseable)
  3. BCP-47 para idiomas (estándar web)
  4. ISO-4217 para monedas (usado en comercio internacional)
  5. **Validación rigurosa**
  6. ISBNs con algoritmo de checksum
  7. Códigos de idioma verificados
  8. Formatos de fecha parseables
  9. **Manejo de precisión variable**
  10. Fechas pueden ser año, año-mes o fecha completa
  11. Documentación de nivel de precisión
-

## ESTRUCTURA DEL REPOSITORIO

---

```
books-pipeline/
├── README.md                                # Documentación principal
├── requirements.txt                          # Dependencias Python
├── .env.example                             # Template de configuración
├── .gitignore                               # Archivos a ignorar en Git
├── run_pipeline.py                         # Script maestro
|
├── landing/                                  # Datos sin procesar (solo lectura)
│   ├── goodreads_books.json                 # Output Ejercicio 1
│   └── googlebooks_books.csv                # Output Ejercicio 2
|
├── standard/                                 # Datos estandarizados
│   ├── dim_book.parquet                     # Tabla dimensional
│   └── book_source_detail.parquet          # Detalle por fuente
|
├── docs/                                     # Documentación
│   ├── schema.md                            # Doc del modelo de datos
│   └── quality_metrics.json                # Métricas de calidad
|
└── src/                                      # Código fuente
    ├── scrape_goodreads.py                  # Ejercicio 1
    ├── enrich_googlebooks.py                # Ejercicio 2
    ├── integrate_pipeline.py                # Ejercicio 3
    ├── utils_quality.py                   # Utilidades de calidad
    └── utils_isbn.py                      # Utilidades de ISBN
```

---

## INSTRUCCIONES DE EJECUCIÓN

---

### Requisitos previos

```
# Python 3.10 o superior
python --version

# Instalar dependencias
```

```
pip install -r requirements.txt

# (Opcional) Configurar API key de Google Books
cp .env.example .env
# Editar .env y añadir GOOGLE_BOOKS_API_KEY
```

## Ejecución del pipeline completo

```
# Opción 1: Script maestro (recomendado)
python run_pipeline.py

# Opción 2: Scripts individuales
cd src
python scrape_goodreads.py
python enrich_googlebooks.py
python integrate_pipeline.py
```

## Verificación de resultados

```
# Verificar estructura
ls landing/      # goodreads_books.json, googlebooks_books.csv
ls standard/     # dim_book.parquet, book_source_detail.parquet
ls docs/          # quality_metrics.json, schema.md

# Inspeccionar métricas de calidad
cat docs/quality_metrics.json

# Leer documentación del esquema
cat docs/schema.md
```

## ENLACE AL REPOSITORIO

---

Link: **GitHub:** [A completar tras subir el repositorio]

<https://github.com/uxentio/books-pipeline>

---

## CONCLUSIONES

---

Este proyecto demuestra la implementación completa de un pipeline ETL moderno para datos de libros, con las siguientes características destacadas:

1. **Extracción ética y documentada** desde fuentes web públicas
2. **Enriquecimiento inteligente** con APIs estructuradas
3. **Normalización a estándares internacionales** para interoperabilidad
4. **Calidad de datos como prioridad** con validaciones y métricas
5. **Trazabilidad completa** con provenance por campo
6. **Documentación exhaustiva** para mantenibilidad

El sistema está listo para ser extendido con: - Más fuentes de datos (Amazon, LibraryThing, etc.) - Automatización con schedulers (Airflow, Prefect) - Integración con data warehouses - Dashboards de calidad de datos - APIs de consulta sobre el modelo dimensional

---

**Fecha de finalización:** 16 de noviembre de 2025

**Autor:** Antonio Ferrer Martínez