Mini-pipeline de libros

Proyecto_UT1_RA1_SBA

Rafael García López

Noviembre 2025

Índice

Books Pipeline (Goodreads → Google Books → Parquet)

(README.md)

Ejecución del pipeline completo

BLOQUE 1 - Scraping (Goodreads → JSON)

Archivo: `src/scrape_goodreads.py`

Este script extrae libros desde la búsqueda pública de Goodreads sin autenticación y genera un fichero JSON con los resultados.

Opcionalmente, descarga portadas locales y extrae ISBN/ASIN accediendo a la ficha de cada libro.

1.1 URL utilizada

```
https://www.goodreads.com/search?q={query}&page={page}
```

donde `{query}` es la cadena de búsqueda con espacios reemplazados por `+`.

1.2 Selectores CSS (Cascading Style Sheets) empleados

Campo	Selector CSS
Filas	`table.tableList tr`
Título	`a.bookTitle`
Autor	`a.authorName`
Rating / votos	`span.minirating`
Portada (URL img)	`img.bookCover`

1.3 Campos extraídos

Cada libro incluye:

- `title` — Título.
- `author` — Autor/a.
- `rating` — Valoración media (`float`).
- `ratings_count` — Número de valoraciones (`int`).
- `book_url` — URL absoluta a la ficha del libro.
- `cover_url` — URL de la imagen de portada (si existe).
- `cover_local_path` — Ruta relativa al fichero de portada descargado (ej. `covers/<titulo>.jpg`).
- `isbn10`, `isbn13` y `asin` — Extraídos opcionalmente desde la ficha del libro.

  Por defecto, `isbn10`, `isbn13` y `asin` se dejan en `null`.
  La extracción de ISBN/ASIN se activa con el parámetro de entorno `GOODREADS_FETCH_ISBN = true`

1.4 Salida generada

- Archivo: `landing/goodreads_books.json`
- Codificación: UTF-8
- Contenido: Lista JSON con hasta el número solicitado de libros (`GOODREADS_MAX_BOOKS`, por defecto 15).

La ruta base del proyecto se calcula a partir de la ubicación del script y se asegura la existencia del directorio `landing/` antes de escribir el fichero.

1.5 Descarga de portadas

Cuando el libro tiene portada (`img.bookCover`):

1. Se obtiene la URL (`cover_url`).
2. Se genera un nombre de fichero a partir del título (`<titulo>.jpg`), saneando caracteres no válidos.
3. Se descarga la imagen y se guarda en el directorio `covers/<nombre_saneado>.jpg`
4. La ruta relativa se guarda en el campo `cover_local_path` del JSON.
5. Si la descarga falla, se registra y se muestra un mensaje de error y `cover_local_path` queda en `null`.

1.6 Pausas y buenas prácticas de scraping

Para evitar cargas excesivas sobre Goodreads:

- Se incluye una pausa corta (0.5s) entre peticiones a páginas y fichas.
- Se utiliza un User-Agent identificable y configurable por `.env`.

1.7 Configuración y backend

Los parámetros del scraping se leen de variables de entorno (con valores por defecto si no existen):

Variable	Descripción	Valor por defecto
`GOODREADS_SEARCH_QUERY`	Término de búsqueda	"Big Data"
`GOODREADS_MAX_BOOKS`	Nº máximo de libros a extraer	15
`GOODREADS_USER_AGENT`	User-Agent HTTP	"Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
`GOODREADS_BACKEND`	Backend scraping `requests/playwright`	"requests"
`GOODREADS_FETCH_ISBN`	Activar extracción ISBN/ASIN desde ficha	"true"

Ejemplo de flujo de ejecución (`main()`):

1. Lee configuración desde `.env`
2. Determina el backend (`requests` o `playwright`)
3. Ejecuta `scrape_goodreads_search()`
4. Recorre páginas hasta obtener el límite solicitado
5. Entra en cada ficha y extrae ISBN/ASIN (Opcional)
6. Descarga portadas
7. Escribe el JSON resultante en `landing/goodreads_books.json`.

1.8 Herramienta de debugging opcional (`debug_goodreads.py`)

Archivo: `src/debug_goodreads.py`

Este script no forma parte del pipeline principal, pero resulta muy útil para inspeccionar cómo Goodreads estructura los datos internos de una ficha de libro, especialmente cuando cambian los HTML o cuando hay problemas al extraer `isbn10`, `isbn13` o `asin`.

Funcionalidades:

- **Descarga la página HTML completa** de un libro y la guarda como `debug_goodreads.html` para inspección directa en el navegador.
- **Busca todas las apariciones de `isbn` e `isbn13` en el HTML crudo**, incluyendo valores embebidos en JSON o scripts internos.
- **Analiza el bloque `#bookDataBox`** con BeautifulSoup y muestra todas las filas (`label → valor`) que presente Goodreads.
- **Revela cambios, estructuras ocultas o variaciones** que afecten la extracción de identificadores.

Uso rápido:

```
from debug_goodreads import debug_goodreads
debug_goodreads("https://www.goodreads.com/book/show/<ID>")
```

Permite localizar manualmente dónde aparece realmente cada identificador para ajustar el scraper cuando Goodreads modifica su HTML.

BLOQUE 2 — Enriquecimiento (Google Books → CSV)

Archivo: **src/enrich_googlebooks.py**

Este script consulta la API pública de Google Books para enriquecer la información procedente de Goodreads (`landing/goodreads_books.json`).

Genera un CSV con metadatos adicionales como autores, editorial, categorías, idiomas, identificadores y precios.

2.1 Endpoint utilizado

`https://www.googleapis.com/books/v1/volumes`

Parámetros enviados:

Parámetro	Descripción
`q`	Query de búsqueda (ISBN o título+autor).
`maxResults`	Siempre `1` → se toma el primer resultado devuelto.
`key`	`GOOGLE_BOOKS_API_KEY` (opcional).

2.2 Prioridad de búsqueda

Para cada libro del JSON de Goodreads se construye una query aplicando la siguiente prioridad:

1. `isbn13`
2. `isbn10`
3. `asin` (se usa también como `isbn:<asin>`)
4. Título + autor (usando campos entrecomillados):

   `intitle:"<título>" inauthor:"<autor>"`

Esto garantiza que se priorice la búsqueda por identificadores únicos y solo se use `title+author` como fallback

2.3 Campos extraídos de Google Books

De la respuesta de Google Books se extrae:

- Identificación y trazabilidad
  - `gb_id` — ID interno de Google Books.
  - `original_title` — Título original del JSON de Goodreads.
  - `original_author` — Autor original del JSON de Goodreads.
- Campos de Volumen:
  - `title` — Título normalizado por Google Books.
  - `subtitle` — Subtítulo (si existe).
  - `authors` — Lista de autores serializada con `|`
  - `publisher` — Editorial.
  - `pub_date` — Fecha de publicación (`publishedDate`).
  - `language` — Idioma (código ISO-639).
  - `categories` — Categorías serializadas con `|`

- Identificadores:
  - isbn10 / isbn13 — Extraídos de `industryIdentifiers`.
  - asin — Si aparece como identificador con tipo `ASIN`.
- Precio (`saleInfo`)
  - `price_amount` — Importe del precio (si existe `listPrice` o `retailPrice`).
  - `price_currency` — Moneda del precio (código ISO-4217).

Nota: Los campos ISBN/ASIN pueden sobreescribir o completar la información original de Goodreads, ya que se toman de la respuesta de Google Books.

2.4 Formato para campos multi-valor

Los campos con múltiples valores (varios autores o categorías) se serializan como cadenas separadas por `|`:

```
authors = "Autor1|Autor2"
categories = "Cat1|Cat2"
```

Si el campo no contiene valores, se deja vacío `""` en el CSV, lo que se interpreta como `null` al leer desde pandas.

2.5 Hipótesis de mapeo

- Se selecciona únicamente el primer elemento de items devuelto por Google Books.
- El primer ISBN disponible en `industryIdentifiers` con tipo `ISBN_13` o `ISBN_10` se usa como valor final.
- `ASIN` se usa solo si aparece explícitamente en la respuesta.
- `listPrice` se considera la fuente de precio preferente; `retailPrice` es fallback.
- Autores y categorías se serializan en formato de lista separada por `|`.
- Si Google Books no devuelve resultados para un libro, no se genera fila para él en el CSV.

2.6 Manejo de errores

El acceso a Google Books se realiza mediante la función `call_google_books_api`, que aplica:

- Reintentos automáticos cuando Google Books devuelve errores temporales:
  - Códigos 503 (Service Unavailable) o 429 (Too Many Requests).
  - Se realiza hasta `max_retries` intentos (por defecto 3), con backoff incremental (espera que aumenta en cada intento).
- Manejo explícito de fallos de red:
  - Cualquier `RequestException` se captura y se informa con un mensaje `[ERROR RED]`.
- Otros errores HTTP:
  - Para códigos distintos de 200/503/429, se muestra `[ERROR HTTP]` con el código y un fragmento de la respuesta.
- Control de excepciones generales por libro:
  - En el bucle principal se envuelve la llamada en un `try/except` de tipo "catch-all" para que, si ocurre algo inesperado con un libro concreto, el script continúe con el resto.
- Logging por libro. Para cada libro se imprime:
  - El título y autor originales.
  - La query enviada a Google Books.
  - Mensajes de enriquecido correcto, sin resultados o errores.

Se introduce una pequeña pausa entre llamadas (`time.sleep(0.3)`) para no saturar la API de Google Books.

Si `call_google_books_api` no devuelve ningún resultado (`item is None`), no se genera fila en el CSV de salida.

Si no se encuentra la variable de entorno `GOOGLE_BOOKS_API_KEY`, se muestra un aviso, pero el script intenta llamar igualmente a la API sin clave.

Si no existe el fichero de entrada `landing/goodreads_books.json`, se lanza un `FileNotFoundError` y el script termina.

2.7 Configuración

Variable de entorno:

`GOOGLE_BOOKS_API_KEY` — Clave de API para Google Books (opcional pero recomendada).

Ruta de entrada:

`landing/goodreads_books.json` — JSON con los libros procedentes de Goodreads.

Ruta de salida:

`landing/googlebooks_books.csv` — CSV enriquecido con datos de Google Books.

Las rutas se calculan a partir de la raíz del proyecto, utilizando `BASE_DIR` y `LANDING_DIR`.

2.8 Salida

Archivo generado: `landing/googlebooks_books.csv`

- Separador: `;`
- Codificación: `utf-8`
- Cabecera incluida
- Columnas (en este orden): `gb_id`, `original_title`, `original_author`, `title`, `subtitle`, `authors`, `publisher`, `pub_date`, `language`, `categories`, `isbn13`, `isbn10`, `asin`, `price_amount`, `price_currency`

BLOQUE 3 - Integración y normalización (JSON + CSV → Parquet)

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

Este módulo integra los datos procedentes de Goodreads (`landing/goodreads_books.json`) y Google Books (`landing/googlebooks_books.csv`) sin modificar los ficheros originales de la carpeta `landing/`.

La integración se basa en una tabla de `staging/` (intermedia), donde se normalizan campos clave, se aplican controles de calidad, se generan identificadores canónicos y se deduplican los registros para producir las tablas finales en `standard/`, junto con métricas y documentación.

3.1 Integración de fuentes y construcción de la tabla staging

La función `load_sources()` carga ambas fuentes en memoria en formato pandas.

Después, `build_staging()` realiza:

- Alineación de columnas entre Goodreads y Google Books:
  - Goodreads aporta: `title`, `author`, `rating`, `ratings_count`, `book_url`, `isbn10`, `isbn13`, `asin`.
  - Google Books aporta: `title`, `authors`, `publisher`, `pub_date`, `language`, `categories`, `isbn10`, `isbn13`, `asin`., `price_amount`, `price_currency`.
- Se renombran columnas para unificarlas (`titulo`, `autor_principal`, `editorial`, `precio`, `moneda`, etc.)
- Se añaden metadatos de trazabilidad:
  - `source_name` (`goodreads` / `googlebooks`)
  - `source_file` (`goodreads_books.json` / `googlebooks_books.csv`)
  - `row_number` (fila original dentro de cada fuente)
- Se escriben los registros combinados en la tabla staging/books_staging.parquet (fuera de landing/, cumpliendo el requisito de no modificar los ficheros originales).

Esta tabla staging es la base para normalización, calidad y deduplicación.

3.2 Normalización semántica

A partir de los datos combinados en **staging** se aplican las siguientes normalizaciones y derivaciones:

Fechas (ISO-8601)

Función: `normalize_date`:

- Soporta formatos típicos `YYYY`, `YYYY-MM`, `YYYY-MM-DD`.
- Si no se puede parsear, la fecha normalizada queda en `null`.
- Se mantiene también el campo original como `fecha_publicacion_raw`
- La fecha normalizada se convierte a formato `YYYY-MM-DD`, rellenando componentes faltantes con `01`

```
2018     → 2018-01-01
2017-05  → 2017-05-01
```

Idioma (BCP-47 simplificado)

- Se transforma a minúsculas.
- Se eliminan espacios.
- Si viene vacío o `NaN`, devuelve `null`.
- Se valida posteriormente con una aproximación a BCP-47 con `idioma_valido`

```
"EN" → "en"
"pt-BR" → "pt-br"
```

Moneda (ISO-4217)

Función `normalize_currency`:

- Convierte a mayúsculas.
- Elimina espacios.
- Si viene vacío o `NaN`, devuelve `null`.
- Se valida posteriormente con `moneda_valida` (3 letras mayúsculas).

```
"usd" → "USD"
" eur " → "EUR"
```

Autores / Categorías

- Campo original en texto: `autores`, `categorias` .
- Cadenas tipo `"Autor1|Autor2"` se convierten en listas `["Autor1", "Autor2"]`.
- Se eliminan vacíos y espacios.
- Se guardan como listas `autores_list` y `categorias_list`.

Otros derivados

Además se generan:

- `titulo_normalizado` — título en minúsculas.
- `autor_principal` — si falta, se toma el primer autor de `autores_list`.
- `anio_publicacion` — año extraído de fecha_publicacion.
- `longitud_titulo` — longitud del título (usada como criterio de desempate).

3.3 Identificador canónico (`book_id`)

Función: `generate_book_id_from_row`

1. Si existe `isbn13`, se utiliza directamente como `book_id`.
2. Si no, se genera un hash (SHA-1) estable a partir de:
   - `titulo_normalizado | autor_normalizado | editorial_normalizada | anio_publicacion`

Los campos normalizados `autor_normalizado` y `editorial_normalizada` se generan en `deduplicate()` a partir de `autor_principal` / `autor` / `author` y `editorial` / `publisher` respectivamente.

3.4 Deduplicación

La deduplicación se realiza en dos fases:

1. Anotación de calidad
2. Elección de fila ganadora.

La función `deduplicate(staging)` aplica las reglas sobre la tabla de staging:

1. Flags y prioridad:
   - `has_isbn13` — `True` si `isbn13` no es nulo.
   - `has_precio` — `True` si `precio` no es nulo.
   - `prioridad_fuente` — `googlebooks = 3`, `goodreads = 2`, otras = 1.
2. Validación y errores (soft fail)
   - Se anotan errores con `annotate_errors`
   - Se añaden `error_codes` (lista) y `has_error` (booleano) según reglas:
     - `R1_MISSING_KEY_TITULO_AUTOR`
     - `R2_INVALID_DATE`
     - `R3_INVALID_LANGUAGE`
     - `R4_INVALID_CURRENCY`
     - `R5_INVALID_RATING`
   - Los registros con errores (`has_error == True`) no participan en la deduplicación final de `dim_book` (pero sí aparecen en `book_source_detail`).
3. Selección del ganador por `book_id`
   - Para cada conjunto con el mismo `book_id`, se elige la primera fila ganadora ordenando por:
     - `has_isbn13` (primero los que tienen `isbn13`)
     - `has_precio` (después los que tienen precio)
     - `prioridad_fuente` (`googlebooks > goodreads > otras`)
     - `longitud_titulo` (título más completo)
4. Unión de listas
   - Autores y categorías se unen sin duplicados sobre las filas válidas del mismo `book_id`:
     - `autores_unificados` (unicón sin duplicados)
     - `categorias_unificadas` (unicón sin duplicados)
5. Construcción de `dim_book`.
   - La tabla resultante contiene **una fila única por libro**, con:
     - información ganadora,
     - autores y categorías unificados,
     - normalizaciones semánticas,
     - trazabilidad (`fuente_ganadora`, `ts_ultima_act`).

3.5 Salidas

Además de un Parquet de staging (`staging/books_staging.parquet`), el módulo genera las siguientes salidas:

`standard/dim_book.parquet`

Tabla canónica con una fila por `book_id` (solo registros válidos). Incluye:

Campo	Tipo	Nulo	Descripción
book_id	string	no	Identificador canónico del libro (isbn13 o hash estable).
titulo	string	sí	Título del libro tal y como aparece en la fuente ganadora.
titulo_normalizado	string	sí	Título en minúsculas, sin espacios extra (para comparación/deduplicación).
autor_principal	string	sí	Autor principal del libro.
autores	array	sí	Lista de autores (fusionados de todas las fuentes).
editorial	string	sí	Editorial del libro.
anio_publicacion	string	sí	Año de publicación (`YYYY`) derivado de la fecha de publicación.

Campo	Tipo	Nulo	Descripción
fecha_publicacion	date	sí	Fecha de publicación en formato ISO-8601 (`YYYY-MM-DD`).
idioma	string	sí	Idioma en BCP-47 simplificado (ej. `en`, `es`, `pt-br`).
isbn10	string	sí	Identificador ISBN-10 (si existe).
isbn13	string	sí	Identificador ISBN-13 (si existe).
asin	string	sí	Identificador ASIN (si existe).
paginas	int	sí	Número de páginas (no disponible en este pipeline).
formato	string	sí	Formato físico/digital (no disponible en este pipeline).
categoria	array	sí	Lista de categorías temáticas.
precio	float	sí	Precio del libro (si está disponible en Google Books).
moneda	string	sí	Moneda en ISO-4217 (ej. `EUR`, `USD`).
fuente_ganadora	string	no	Fuente del registro ganador (`googlebooks` o `goodreads`).
ts_ultima_act	timestamp	no	Marca temporal de generación del registro canónico (ISO-8601).

`standard/book_source_detail.parquet`

Detalle completo de todas las filas de `staging` (válidas y con error), por fuente y fila original, ya ordenadas por prioridad, incluyendo campos normalizados, flags de calidad y trazabilidad. Incluye:

- Campos originales unificados de ambas fuentes (título, autor, rating, etc.).
- Campos normalizados (titulo_normalizado, idioma, fecha_publicacion, etc.).
- Listas derivadas:
  - `autores_list` — Lista de autores ya tokenizados.
  - `categorias_list` — Lista de categorías normalizadas.
- Flags y prioridad:
  - `has_isbn13` — Flag: indica si la fila tiene `isbn13` no nulo.
  - `has_precio` — Flag: indica si la fila tiene `precio` no nulo.
  - `prioridad_fuente` — Prioridad de la fuente en la regla de supervivencia (`googlebooks` > `goodreads`).
  - `has_error` — Indica si esa fila tuvo algún problema durante el procesamiento (`true`/`false`)
  - `error_codes` — Lista o cadena con los códigos de error que afectaron a la fila.
- Trazabilidad y claves:
  - `source_name` — Nombre de la fuente (`goodreads`, `googlebooks`).
  - `source_file` — Nombre del fichero de origen en `landing/` (`goodreads_books.json` / `googlebooks_books.csv`)
  - `source_id` — ID secuencial en book_source_detail
  - `row_number` — Número de fila original dentro de cada fuente.
  - `book_id` — Identificador canónico asignado a la fila.
  - `book_id_candidato` — Usada para trazabilidad; coincide con `book_id`
  - `ts_ingesta` — Timestamp de carga en `staging`

La trazabilidad de procedencia se mantiene mediante `source_name`, `source_file` y `book_id_candidato`, que permiten identificar el origen de cada registro.

En `dim_book` el campo `fuente_ganadora` indica el origen del valor final seleccionado tras la deduplicación.

Comparativa: `dim_book` vs `book_source_detail`

Aspecto	`dim_book`	`book_source_detail`
Propósito	Modelo canónico final, listo para consumo	Detalle por fuente y registro original
Nivel	Capa estándar / Gold	Capa de detalle / Silver-raw
Cardinalidad	1 fila por `book_id`	Varias filas por `book_id` (una por fuente y fila original)
Estado del dato	Normalizado, deduplicado y sin errores	Incluye filas válidas y con error

Aspecto	dim_book	book_source_detail
Origen de los valores	Fila ganadora según reglas de deduplicación	Datos tal y como llegan de cada fuente + normalizaciones básicas
Uso principal	Analítica, reporting, consumo downstream	Auditoría, trazabilidad, debugging del pipeline
Trazabilidad	book_id, fuente_ganadora, timestamps	source_name, source_file, row_number, book_id_candidato

docs/quality_metrics.json Fichero JSON con métricas de calidad y trazabilidad. Incluye:

- Sobre dim_book:
  - Número de filas y columnas.
  - % de nulos por campo.
- Sobre book_source_detail:
  - Número de filas y columnas.
  - % de nulos por campo.
  - Conteo de duplicados por isbn13, (titulo_normalizado, autor_principal, editorial)
  - Filas por fuente (goodreads, googlebooks).
- Validaciones globales:
  - porcentaje_idiomas_validos
  - porcentaje_monedas_validas
  - porcentaje_fechas_validas
  - porcentaje_clave_titulo_autor_presente
  - porcentaje_filas_validas
  - porcentaje_ratings_validos (si existe rating)
  - % de nulos en campos clave: titulo, isbn13, precio
  - porcentaje_registros_invalidos (filas con has_error = True)
- Logs de reglas de calidad:
  - logs.por_archivo — número de errores por source_file y código de regla.
  - logs.por_regla — número total de errores por cada código de regla.
- Metadatos de entrada (metrics["entradas"]):
  - Para goodreads y googlebooks: ruta, n_filas, n_columnas y tamano_bytes

docs/schema.md

Se genera un fichero de documentación con el esquema de las tablas para dim_book y book_source_detail:

- Nombre de columna
- Tipo
- Nullability (incluyendo % de nulos)
- Un valor de ejemplo

CONCLUSIÓN

El proyecto implementa un pipeline ETL completo:

- Extrae libros desde Goodreads y los enriquece con Google Books.
- No modifica landing/ (solo lectura).
- Normaliza semánticamente fechas, idiomas y monedas.
- Integra ambas fuentes en una tabla de staging con normalización y reglas de calidad.
- Deduplica con un identificador canónico (book_id) y criterios de supervivencia claros.
- Produce Parquets limpios (dim_book, book_source_detail).
- Genera métricas de calidad (quality_metrics.json)
- Documenta completamente el esquema (schema.md).

Resumen Books Pipeline (Goodreads → Google Books → Parquet)

(README_resumen.md)

---

1. Objetivo

Este proyecto implementa un pipeline completo de **Extracción → Enriquecimiento → Integración** para consolidar datos de libros a partir de:

- **Scraping de Goodreads**
- **Google Books API**
- **Integración con normalización semántica y deduplicación**

El resultado final produce:

- Un **modelo canónico limpio en Parquet**
- Un archivo de **detalle por fuente**
- Un conjunto de **métricas de calidad**.

---

2. Tecnologías utilizadas

- Python
- Requests
- BeautifulSoup4
- LXML
- python-dotenv
- Playwright (opcional)
- pandas

---

3. Estructura del repositorio

```
books-pipeline/                          Proyecto completo del pipeline de libros

    landing/                            Datos "crudos" recién extraídos de las fuentes
      goodreads_books.json              Datos sin procesar obtenidos de Goodreads
      googlebooks_books.csv             Datos sin procesar obtenidos de Google Books

    staging/                            Datos transformados parcialmente (limpios pero no finales)
      book_staging.parquet              Dataset unificado y normalizado previo al estándar

    standard/                           Datos finales listos para análisis
      dim_book.parquet                  Dimensión de libros estandarizada
      book_source_detail.parquet        Detalles de origen por libro (trazabilidad)

    covers/                             Carpeta para almacenar portadas descargadas

    docs/                               Documentación del pipeline
      schema.md                         Descripción de los esquemas de datos
      quality_metrics.json              Métricas de calidad del pipeline/dataset

    src/                                Código fuente del pipeline
       debug_goodreads.py               Herramienta para depurar fichas de Goodreads y localizar ISBN/ASI
```

```
    scrape_goodreads.py          Script para extraer datos de Goodreads
    enrich_googlebooks.py        Script para enriquecer datos usando Google Books
    integrate_pipeline.py        Script principal que orquesta todo el pipeline

README.md                        Descripción general del proyecto, cómo usarlo
requirements.txt                 Lista de dependencias de Python necesarias
.env.example                     Fichero de ejemplo de variables de entorno (API keys, ...)
.env                             Variables de entorno (API keys, ...)
```

---

4. Flujo del pipeline

1. **Scraping (Goodreads → JSON)**
   Script: `src/scrape_goodreads.py`
   - Lanza una búsqueda en Goodreads (`GOODREADS_SEARCH_QUERY`).
   - Uso de User-Agent personalizado y pausas opcionales para un scraping responsable.
   - Extrae: título, autor, rating, número de valoraciones y URL del libro.
   - Genera: `landing/goodreads_books.json`.
2. **Enriquecimiento (Google Books → CSV)**
   Script: `src/enrich_googlebooks.py`
   - Para cada libro de Goodreads, llama a Google Books API.
   - Prioriza `isbn13` > `isbn10` > `ASIN` > combinación `título + autor`.
   - Extrae: título, autores, editorial, fecha de publicación, idioma, categorías, ISBNs, precio/moneda.
   - Genera: `landing/googlebooks_books.csv` (`;` y UTF-8).
   - Extrae título, autores, editorial, fecha, idioma, categorías, precios, ISBNs.
3. **Integración y estandarización (JSON + CSV → Parquet)**
   Script: `src/integrate_pipeline.py`
   - Lee los ficheros de `landing/`.
   - *Los ficheros de* ***`landing/`*** *no se modifican;* ***`staging/`*** *y* ***`standard/`*** *contienen las salidas del pipeline.*
   - Normaliza fechas (ISO-8601), idioma (BCP-47) y moneda (ISO-4217).
   - Genera un `book_id` canónico (ISBN13 o hash).
   - Deduplica registros aplicando reglas de supervivencia (ISBN13 > precio > fuente > longitud).
   - Salidas:
     - `standard/dim_book.parquet` → tabla canónica de libros.
     - `standard/book_source_detail.parquet` → detalle por fuente y registro.
     - `docs/quality_metrics.json` → métricas de calidad (nulos, duplicados, formatos, etc.).
     - `docs/schema.md` → descripción de los esquemas de datos

---

5. Instalación y configuración

5.1. Instalar dependencias

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

5.2. Crear archivo `.env`

Puedes copiarlo desde `.env.ejemplo`. Ejemplo:

```
GOOGLE_BOOKS_API_KEY=TU_API_KEY_AQUI
GOODREADS_BACKEND=requests/playwright
GOODREADS_FETCH_ISBN=true/false
GOODREADS_SEARCH_QUERY=big data
GOODREADS_MAX_BOOKS=25
```

```
GOODREADS_USER_AGENT=Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

5.3. Ejecución

1. Scraping Goodreads → JSON

   ```
   python src/scrape_goodreads.py
   ```

   Salidas generadas:

   - `landing/goodreads_books.json`

2. Enriquecimiento Google Books API → CSV

   ```
   python src/enrich_googlebooks.py
   ```

   Salidas generadas:

   - `landing/googlebooks_books.csv`

3. Integración y estandarización → Parquet

   ```
   python src/integrate_pipeline.py
   ```

   Salidas generadas:

   - `standard/dim_book.parquet`
   - `standard/book_source_detail.parquet`
   - `staging/books_staging.parquet`
   - `docs/quality_metrics.json`
   - `docs/schema.md`

---

6. Publicación web (GitHub)

- https://github.com/IA-CSG/books-pipeline