Documento Técnico de Arquitectura General

IMDb Scraper

Versión [1.1]

Elaborado por:	Andrés Ruiz
Fecha de Elaboración:	2025-08-04

CONTROL DE DOCUMENTOS.

FECHA	ELABORADOR	VERSIÓN	REVISADO POR	DESCRIPCIÓN DE LA PUBLICACIÓN
2025-08-04	Andrés Ruiz	1.0		PRIMERA LIBERACIÓN
2025-08-04	Andrés Ruiz	1.1		CONTENEDOR DE DEPENDENCIAS

CONTENIDO.

1. Introducción

Este documento presenta la estructura técnica del proyecto IMDb Scraper, desarrollado bajo los principios de Clean Architecture y Domain-Driven Design (DDD). El sistema está diseñado para ser modular, desacoplado y fácilmente escalable. Incluye persistencia híbrida (CSV y PostgreSQL), proxies anónimos vía TOR y VPN, y despliegue con Docker.

El objetivo del proyecto es extraer y estructurar los datos del Top 250 de IMDb para su posterior análisis de tendencias cinematográficas y la posible alimentación de un data warehouse interno.

2. Arquitectura del Sistema

El sistema está diseñado bajo los principios de Clean Architecture y DDD. Se implementaron 4 capas:

- **Presentation:** contiene el archivo run_scraper.py, que inicia el proceso de scraping.
- **Application:** orquesta los casos de uso como SaveMovieWithActorsCompositeUseCase, que permite guardar los resultados tanto en CSV como en PostgreSQL.
- **Domain:** define las entidades principales (Movie, Actor, MovieActor) y las interfaces (MovieRepository, etc.) que desacoplan la lógica de persistencia.
- **Infrastructure:** contiene las implementaciones reales de los repositorios para CSV y PostgreSQL, utilitarios de red (proxies, TOR), y un Contenedor de Dependencias centralizado que se encarga de construir y conectar todos los componentes del sistema.

Todo el flujo de dependencias respeta la regla de que las dependencias apuntan hacia el dominio, garantizando desacoplamiento y testabilidad.

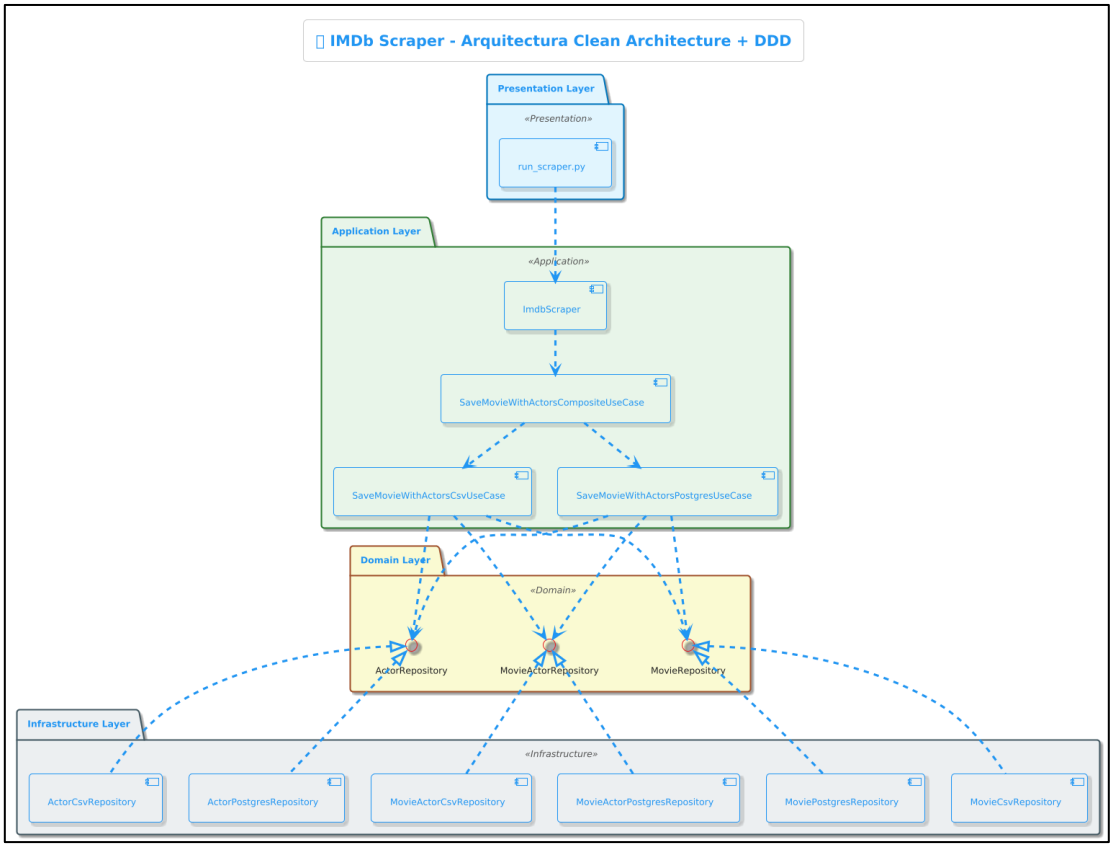

Figura 1. Arquitectura

3. Modelo de Datos

El modelo de datos refleja una relación de muchos a muchos entre películas y actores. Se implementó la entidad intermedia MovieActor para modelar esta relación tanto en base de datos como en CSV.

- Cada Movie contiene una lista de Actor directamente desde el scraping, facilitando el flujo de datos.
- Luego, en la persistencia, se normaliza esa relación para mantener integridad y consultas eficientes.
- Se diseñó el modelo para soportar expansión futura (por ejemplo: directores, géneros, etc.).

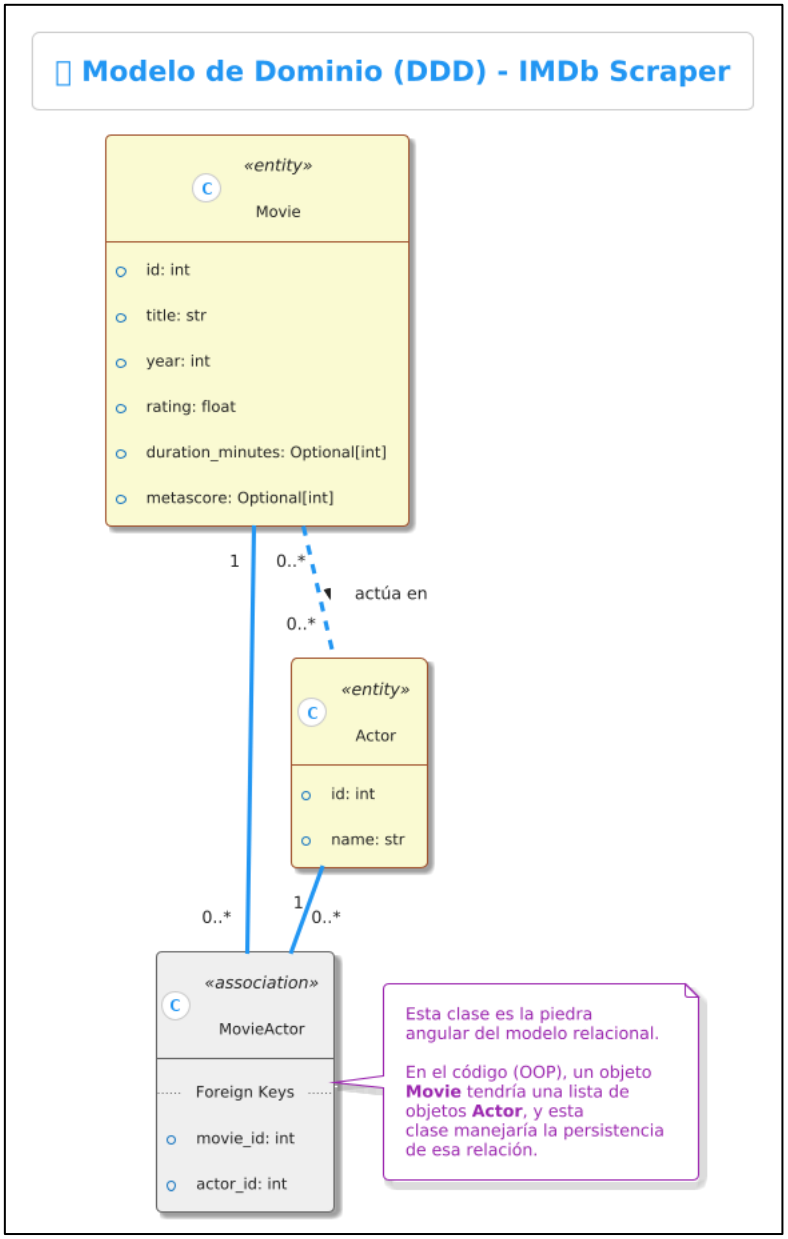

Figura 2. Modelo del dominio

4. Flujo del Scraper

El proceso comienza desde run_scraper.py, el cual instancia ImdbScraper y ejecuta el método scrape(). Este scraper accede a IMDb usando una combinación de:

- HTML scraping para obtener los 25 primeros registros y sus cookies.
- GraphQL (endpoint oficial de IMDb) para obtener los siguientes 100 registros del Top 250.
- Lógica concurrente (ThreadPoolExecutor) para extraer detalles individuales de cada película.

Cada película es enviada a un caso de uso compuesto, que luego la guarda tanto en CSV como en PostgreSQL usando interfaces desacopladas.

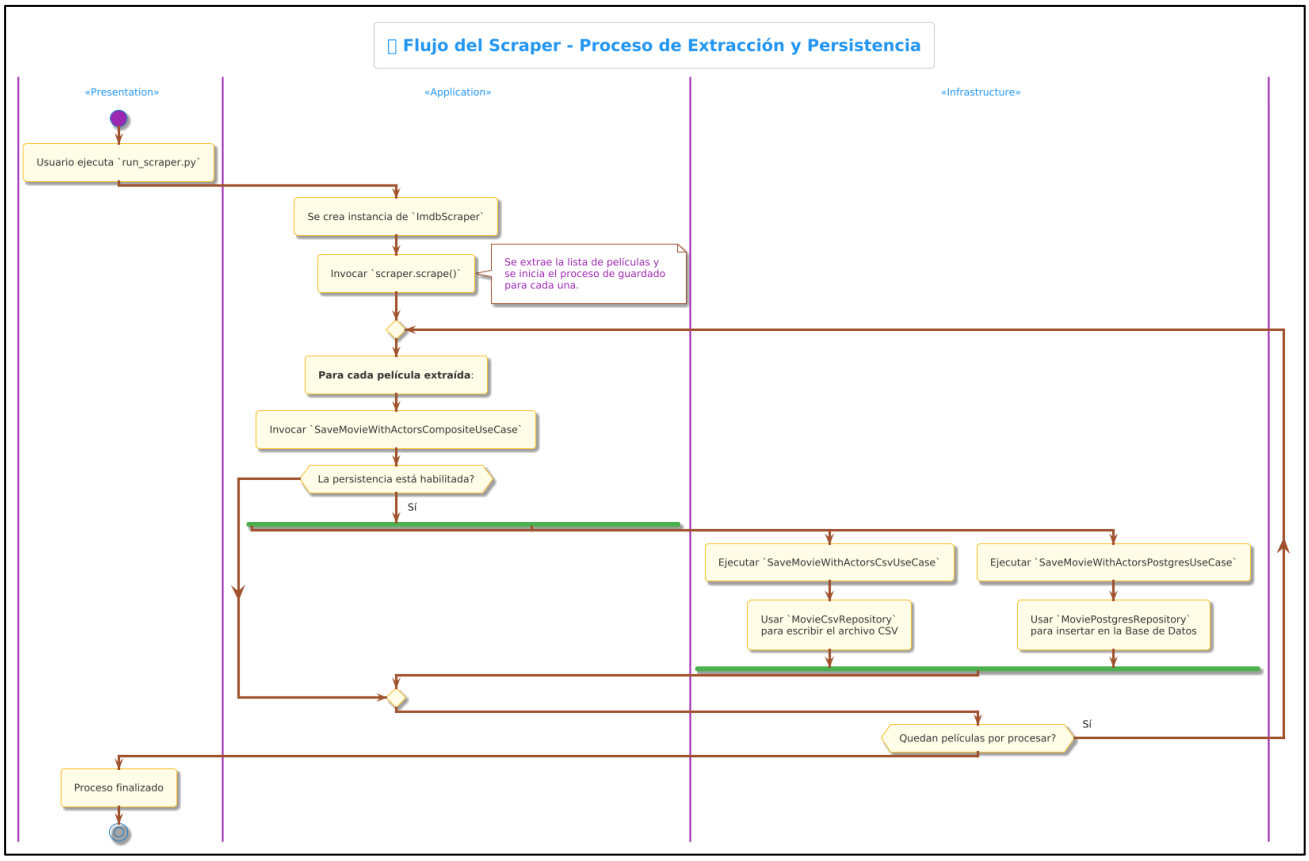

Figura 3. Flujo Scraper

5. Componentes de Red (Docker)

En el entorno de ejecución se usó Docker para montar todos los componentes:

- Scraper: contiene el código de scraping. Se conecta a TOR, a proxies premium y está enrulado a través de una VPN.
- PostgreSQL: base de datos donde se guarda la información estructurada.
- TOR: se utiliza como fallback en caso de que los proxies premium fallen. La rotación se controla vía script.
- VPN (ProtonVPN): encapsula todo el tráfico de red del contenedor scraper, brindando anonimato geográfico (por ejemplo: Argentina).

Se validó que la red interna (vpn_net) conecta correctamente todos los servicios necesarios para evitar bloqueos por parte de IMDb.

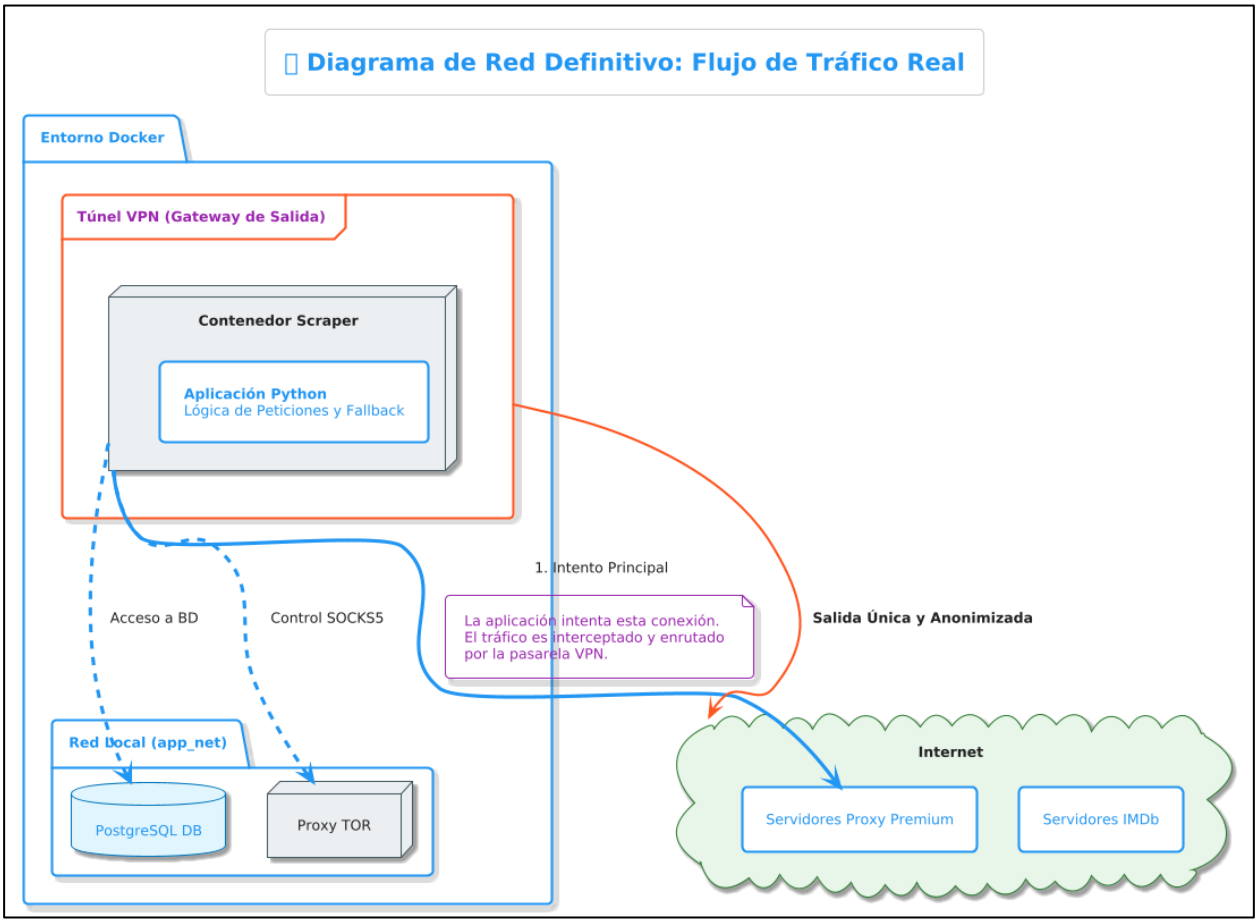

Figura 4. Diagrama de Red