📚 Documentación Técnica - Sistema de Recomendación a Gran Escala

Sistema de Recomendación de Películas en Gran Escala
Versión: 2.0 (Modelo Híbrido)
Última actualización: Diciembre 2025
Repositorio: Melforsy03/Recomendacion-Gran-Escala
Branch: main

📋 Tabla de Contenidos

---

1. Descripción General

1.1. Propósito

PROF

Sistema de recomendación de películas a gran escala que implementa:

- **Modelo Híbrido:** Combina ALS + Item-CF + Content-Based con estrategias configurables
- **Procesamiento Batch:** ETL, entrenamiento de modelos
- **Procesamiento Streaming:** Agregaciones en tiempo real con ventanas
- **Visualización:** Dashboard interactivo con métricas en tiempo real
- **API REST:** Acceso programático a recomendaciones y métricas

1.2. Dataset

Utiliza el dataset **MovieLens** con aproximadamente:

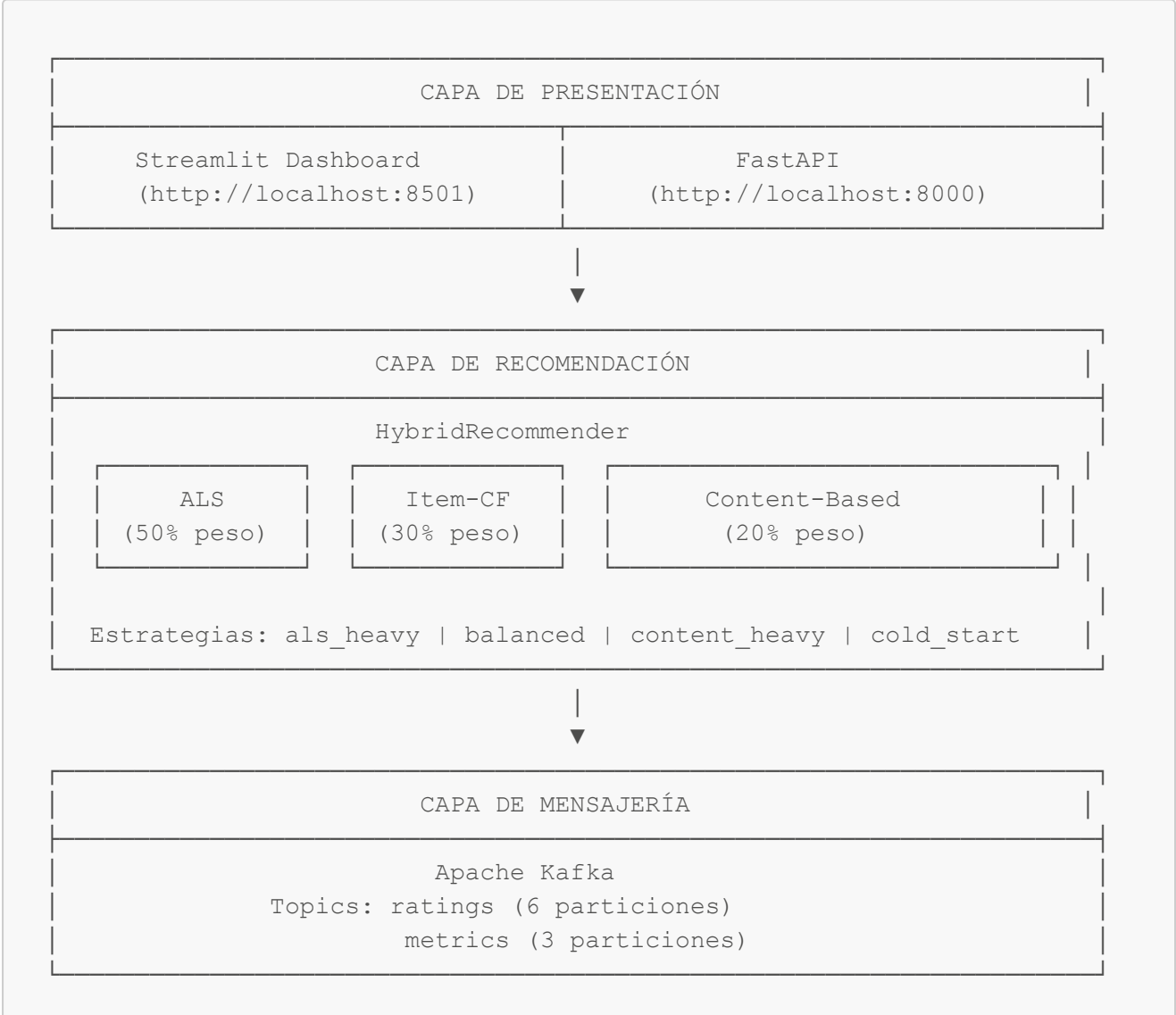
- ~20 millones de ratings
- ~27,000 películas
- 6 archivos CSV: movies, ratings, tags, genome_tags, genome_scores, links

1.3. Tecnologías

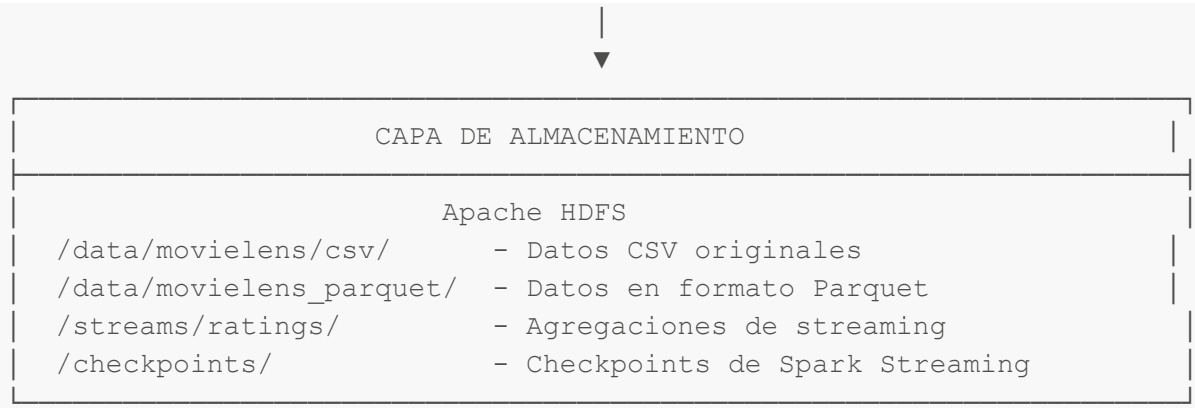
Componente	Tecnología	Versión
Almacenamiento Distribuido	Apache HDFS	3.2.1
Gestión de Recursos	Apache YARN	3.2.1
Procesamiento	Apache Spark	3.5.3
Mensajería	Apache Kafka	3.5
API	FastAPI	0.100+
Dashboard	Streamlit	1.25+
Contenedores	Docker	20.10+
Java	OpenJDK	21

2. Arquitectura del Sistema

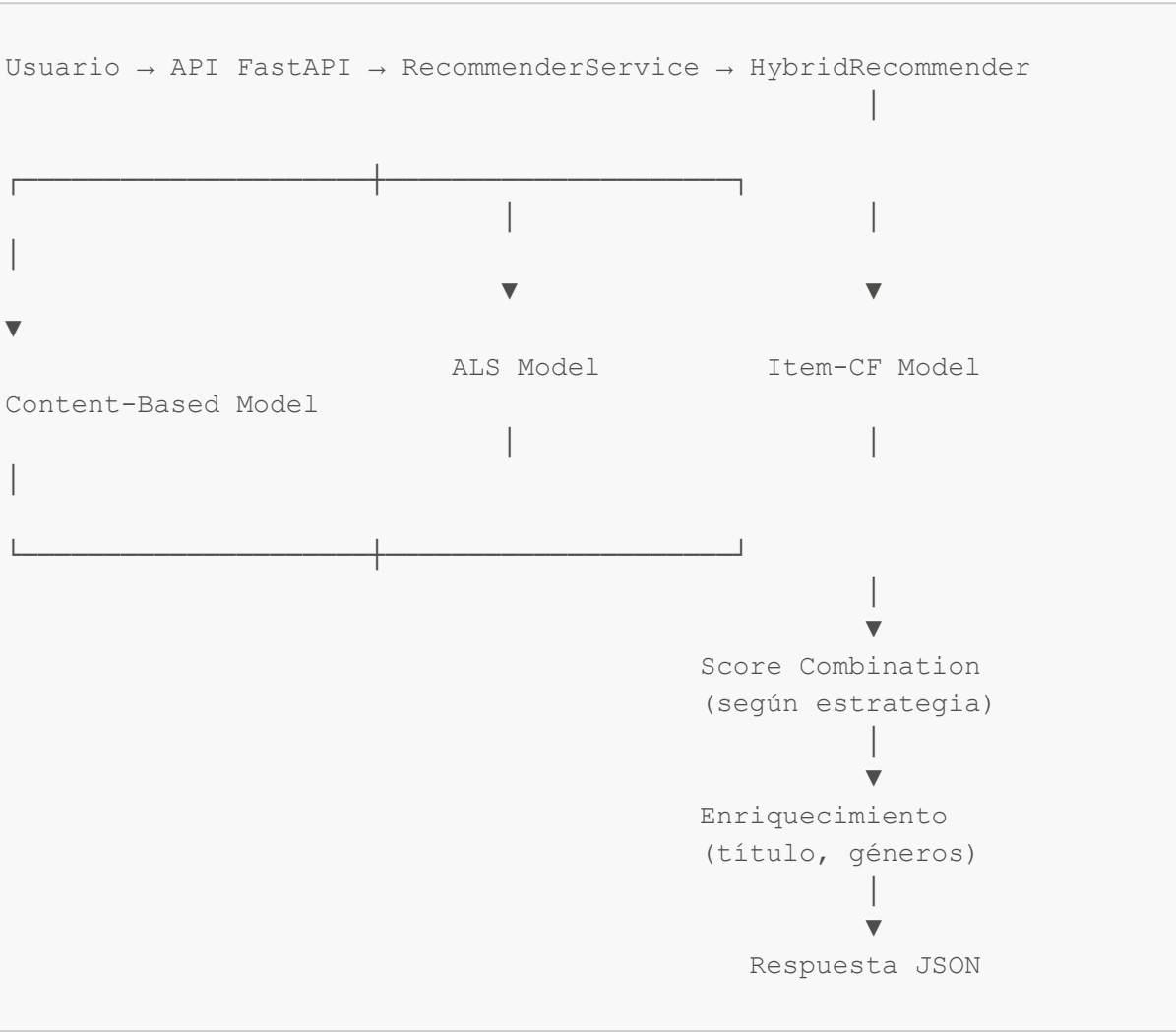
2.1. Diagrama de Arquitectura



PROF



2.2. Flujo de Recomendaciones



PROF

3. Componentes

3.1. Infraestructura Docker (10 contenedores)

Contenedor	Imagen	Puertos	Descripción

Contenedor	Imagen	Puertos	Descripción
`namenode`	bde2020/hadoop-namenode	9870, 9000	HDFS NameNode
`datanode`	bde2020/hadoop-datanode	9864	HDFS DataNode
`resourcemanager`	bde2020/hadoop-resourcemanager	8088	YARN ResourceManager
`nodemanager`	bde2020/hadoop-nodemanager	8042	YARN NodeManager
`spark-master`	bitnami/spark:3.4.1	8080, 7077	Spark Master
`spark-worker`	bitnami/spark:3.4.1	8081	Spark Worker
`zookeeper`	confluentinc/cp-zookeeper:7.5.0	2181	Zookeeper
`kafka`	confluentinc/cp-kafka:7.5.0	9092, 9093	Kafka Broker
`recs-api`	Custom (FastAPI + PySpark 3.5.3)	8000	API de Recomendaciones
`recs-dashboard`	Custom (Streamlit)	8501	Dashboard

3.2. Modelos de Machine Learning

Modelo	Algoritmo	Uso
ALS	Alternating Least Squares	Filtrado colaborativo matricial
Item-CF	Item Collaborative Filtering	Similitud entre películas
Content-Based	TF-IDF + Cosine Similarity	Features de géneros y tags
Hybrid	Combinación ponderada	Mezcla configurable de los 3

PROF

4. Sistema de Recomendación Híbrido

4.1. Estrategias Disponibles

Estrategia	ALS	Item-CF	Content	Uso Recomendado
`als_heavy`	70%	20%	10%	Usuarios con mucho historial
`balanced`	50%	30%	20%	Uso general (por defecto)
`content_heavy`	30%	20%	50%	Usuarios con poco historial
`cold_start`	0%	30%	70%	Usuarios nuevos sin historial

4.2. Estructura de Modelos

```
movies/trained_models/
├── als/
│   └── model_latest/
│       ├── spark_model/      # Modelo Spark MLlib
│       └── metadata.json     # Métricas y parámetros
├── item_cf/
│   └── model_latest/
│       └── similarity_matrix/
├── content_based/
│   └── model_latest/
│       └── movie_features/
└── hybrid/
    └── model_latest/
        └── strategies_config.json
```

4.3. Métricas del Modelo ALS

```
{
  "metrics": {
    "rmse": 0.8234,
    "mae": 0.6431,
    "mse": 0.6780,
    "r2": 0.7845
  },
  "parameters": {
    "rank": 20,
    "maxIter": 10,
    "regParam": 0.1
  }
}
```

PROF

5. API REST

5.1. Endpoints de Recomendaciones

Health Check

```
GET /recommendations/health
```

Respuesta:

```
{
  "status": "healthy",
  "model_loaded": true,
```

```
  "model_version": "hybrid_v1",
  "strategy": "balanced",
  "models": {
    "als": true,
    "item_cf": true,
    "content_based": true
  },
  "cache_stats": {
    "size": 42,
    "max_size": 1000,
    "ttl_hours": 1
  },
  "timestamp": "2025-12-10T21:00:00Z"
}
```

Obtener Recomendaciones

```
GET /recommendations/recommend/{user_id}?n=10&strategy=balanced
```

Parámetros:

- `user_id`: ID del usuario (requerido)
- `n`: Número de recomendaciones (default: 10, max: 100)
- `strategy`: Estrategia híbrida (als_heavy, balanced, content_heavy, cold_start)
- `use_cache`: Usar cache (default: true)

Respuesta:

```
{
  "user_id": 123,
  "recommendations": [
    {
      "movie_id": 126219,
      "title": "Marihuana (1936)",
      "genres": ["Documentary", "Drama"],
      "score": 3.09,
      "rank": 1
    }
  ],
  "strategy": "balanced",
  "model_version": "hybrid_v1",
  "source": "model",
  "timestamp": "2025-12-10T21:00:00Z"
}
```

PROF

5.2. Endpoints de Métricas

Endpoint	Método	Descripción
`/metrics/health`	GET	Estado del sistema de métricas
`/metrics/summary`	GET	Resumen de métricas de streaming
`/metrics/topn?limit=10`	GET	Top-N películas más vistas
`/metrics/genres`	GET	Métricas por género
`/metrics/history?limit=50`	GET	Historial de métricas

5.3. Documentación Interactiva

- **Swagger UI:** http://localhost:8000/docs
- **ReDoc:** http://localhost:8000/redoc

---

6. Configuración

6.1. Archivos de Configuración

Archivo	Ubicación	Descripción
`docker-compose.yml`	Raíz	Definición de servicios Docker
`fairscheduler.xml`	Raíz	Configuración Fair Scheduler Spark
`core-site.xml`	hadoop-conf/	Configuración core de Hadoop
`hdfs-site.xml`	hadoop-conf/	Configuración HDFS
`yarn-site.xml`	hadoop-conf/	Configuración YARN

6.2. Configuración del API (PySpark 3.5.3 + Java 21)

PROF

```
# Dockerfile del API
ENV _JAVA_OPTIONS="--add-opens=java.base/sun.nio.ch=ALL-UNNAMED \
                   --add-opens=java.base/java.nio=ALL-UNNAMED \
                   --add-opens=java.base/java.lang=ALL-UNNAMED"
```

6.3. Cache de Recomendaciones

```
class RecommenderConfig:
    CACHE_MAX_SIZE = 1000       # Máximo entradas en cache
    CACHE_TTL_HOURS = 1         # Tiempo de vida del cache
    TOP_POPULAR_N = 100         # Películas populares para fallback
    DEFAULT_STRATEGY = "balanced"
```

---

7. Scripts Disponibles

7.1. Scripts de Inicio

Script	Descripción	Uso
`start-system.sh`	Inicia toda la infraestructura	`./scripts/start-system.sh`
`run-api-kafka-producer.sh`	Productor API/Dataset → Kafka	`./scripts/run-api-kafka-producer.sh`
`run-latent-generator.sh`	Inicia generador de ratings	`./scripts/run-latent-generator.sh 100`
`run-streaming-processor.sh`	Inicia procesador streaming	`./scripts/run-streaming-processor.sh`
`run-batch-analytics.sh`	Ejecuta analytics batch	`./scripts/run-batch-analytics.sh`
`simulate-traffic.sh`	Simula tráfico HTTP	`./scripts/simulate-traffic.sh --rate 10 --duration 30`

7.2. Scripts de Entrenamiento

Script	Descripción	Uso
`train_all_models.sh`	Entrena todos los modelos	`./scripts/train_all_models.sh`
`train_all_models.sh --force`	Re-entrena todos	`./scripts/train_all_models.sh --force`

7.3. Scripts de Verificación

PROF

Script	Descripción	Uso
`check-spark-resources.sh`	Ver recursos de Spark	`./scripts/check-spark-resources.sh`
`check-status.sh`	Estado de servicios	`./scripts/check-status.sh`
`run-all-tests.sh`	Suite completa de tests	`./scripts/run-all-tests.sh`

7.4. Scripts de Mantenimiento

Script	Descripción	Uso
`stop-system.sh`	Detener todo el sistema	`./scripts/stop-system.sh`

Script	Descripción	Uso
`clean-checkpoints.sh`	Limpiar checkpoints	`./scripts/clean-checkpoints.sh all`
`spark-job-manager.sh`	Gestión de jobs Spark	`./scripts/spark-job-manager.sh list`

8. Interfaces Web

Servicio	URL	Puerto	Descripción
Dashboard Streamlit	`localhost:8501`	8501	Visualizaciones en tiempo real
API Docs (Swagger)	`localhost:8000/docs`	8000	Documentación interactiva API
API Health	`localhost:8000/recommendations/health`	8000	Estado del sistema
Spark Master UI	`localhost:8080`	8080	Jobs y recursos Spark
Spark Worker UI	`localhost:8081`	8081	Estado del worker
HDFS NameNode	`localhost:9870`	9870	Explorador de archivos
YARN ResourceManager	`localhost:8088`	8088	Gestor de recursos

9. Persistencia y Volúmenes

PROF

9.1. Volúmenes Docker

Volumen	Contenedor	Descripción
`namenode_data`	namenode	Metadatos HDFS
`datanode_data`	datanode	Datos HDFS
`spark_master_data`	spark-master	Checkpoints Spark
`kafka_data`	kafka	Datos de topics
`zookeeper_data`	zookeeper	Estado del cluster

9.2. Volúmenes del API

```
volumes:
  - ./movies/trained_models:/app/trained_models:ro
  - ./Dataset/movie.csv:/app/movies_metadata.csv:ro
  - ./movies/src:/app/movies/src:ro
  - ./movies/api/services:/app/services:ro
  - ./movies/api/routes:/app/routes:ro
```

---

10. Fair Scheduler

10.1. Configuración de Pools

```
<pool name="streaming">
  <weight>2</weight>        <!-- Prioridad ALTA -->
  <minShare>1</minShare>
</pool>

<pool name="batch">
  <weight>1</weight>        <!-- Prioridad MEDIA -->
  <minShare>1</minShare>
</pool>

<pool name="generator">
  <weight>1</weight>        <!-- Prioridad BAJA -->
  <minShare>1</minShare>
</pool>
```

10.2. Distribución de Recursos

Pool	Cores	RAM	Prioridad
streaming	2	1GB	ALTA (peso 2)
batch	2	1GB	MEDIA (peso 1)
generator	1	512MB	BAJA (peso 1)

PROF

---

11. Consumo de Recursos

11.1. Requisitos del Sistema

Recurso	Mínimo	Recomendado
RAM	8 GB	12-16 GB
CPU	4 cores	6-8 cores
Disco	20 GB	50+ GB

11.2. Distribución por Servicio

Servicio	CPU	RAM
HDFS (namenode + datanode)	0.5 cores	2GB
YARN (RM + NM)	0.5 cores	2GB
Spark Master + Worker	4-6 cores	4GB
Kafka + Zookeeper	1 core	2GB
API + Dashboard	0.5 cores	2GB
TOTAL	**~8-10 cores**	**~12GB**

---

12. Estructura del Proyecto

```
Recomendacion-Gran-Escala/
├── docker-compose.yml          # Definición de servicios
├── fairscheduler.xml           # Configuración Fair Scheduler
├── requirements.txt            # Dependencias Python
├── README.md                   # Documentación principal
│
├── Dataset/                    # Datos MovieLens
│   ├── movie.csv               # 27,278 películas
│   ├── rating.csv              # ~20M ratings
│   ├── tag.csv
│   ├── genome_tags.csv
│   ├── genome_scores.csv
│   └── link.csv
│
├── docs/                       # Documentación
│   ├── DOCUMENTACION.md        # Este archivo
│   ├── GUIA_DESPLIEGUE_INICIAL_UNICO.md
│   └── GUIA_DESPLIEGUE_REGULAR.md
│
├── movies/
│   ├── trained_models/         # Modelos entrenados
│   │   ├── als/
│   │   ├── item_cf/
│   │   ├── content_based/
│   │   └── hybrid/
│   ├── api/                    # API FastAPI
│   │   ├── Dockerfile
│   │   ├── routes/
│   │   └── services/
│   ├── dashboard/              # Dashboard Streamlit
│   └── src/
│       ├── etl/
│       ├── features/
│       ├── recommendation/     # Modelos de recomendación
```

PROF

```
│       │   └── models/
│       │       ├── als_model.py
│       │       ├── item_cf.py
│       │       ├── content_based.py
│       │       └── hybrid_recommender.py
│       └── streaming/
│
├── scripts/                    # Scripts de gestión
│   ├── start-system.sh
│   ├── stop-system.sh
│   ├── train_all_models.sh
│   └── ...
│
└── tests/                      # Scripts de prueba
```

Documentación Adicional

- **Primera Ejecución:** `docs/GUIA_DESPLIEGUE_INICIAL_UNICO.md`
- **Ejecuciones Regulares:** `docs/GUIA_DESPLIEGUE_REGULAR.md`

Mantenido por: Equipo de Desarrollo
Última actualización: Diciembre 2025