Universidad Europea

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

TRABAJO FIN DE MÁSTER

Gobernanza del dato en sistemas analíticos y de IA en AWS

Jonathan Palan Murillo

Dirigido por

Joaquín García Onrubia

CURSO 2024 - 2025

TÍTULO: Gobernanza del dato en sistemas analíticos y de IA en AWS

AUTOR: Jonathan Palan Murillo

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

DIRECTOR DEL PROYECTO: Joaquín García Onrubia

FECHA: Julio de 2025

RESUMEN

El creciente volumen y complejidad de los datos en sistemas analíticos y de inteligencia artificial plantea la necesidad de contar con marcos de gobernanza robustos que garanticen la calidad, seguridad y cumplimiento normativo de los datos. Este proyecto aborda dicho reto mediante el diseño e implementación de una arquitectura de gobernanza del dato en Amazon Web Services (AWS), aplicada a un caso práctico: una aplicación web de recomendaciones de libros basada en los datasets de Goodreads Book Graph.

La solución propuesta integra diversos servicios de AWS para cubrir el ciclo de vida completo de los datos. Amazon S3 se emplea como data lake para datos en bruto y transformados, mientras que AWS Glue y su Data Catalog permiten la transformación, clasificación y catalogación de los datos. AWS Lake Formation y IAM aseguran un control de acceso granular y permisos basados en roles, garantizando la seguridad y el acceso gobernado. El monitoreo y la auditoría continua se logran mediante AWS CloudTrail y CloudWatch. Sobre este pipeline gobernado se desarrolló un modelo de recomendación, integrado en una aplicación web y complementado con dashboards de analítica de negocio en Amazon QuickSight.

Los resultados obtenidos muestran la viabilidad de aplicar un marco de gobernanza alineado con el modelo DAMA dentro de AWS, manteniendo la escalabilidad y eficiencia en costes. La implementación no solo garantizó el cumplimiento y la transparencia en el uso de los datos, sino que también mejoró la fiabilidad de los resultados de aprendizaje automático al asegurar datasets consistentes y de calidad.

En conclusión, el proyecto valida a AWS como una plataforma eficaz para combinar gobernanza del dato, analítica e inteligencia artificial, y resalta su potencial para futuras extensiones tanto en el ámbito técnico como empresarial.

Palabras clave: AWS, Aprendizaje automático, Big Data, Gobernanza del dato.

ABSTRACT

The increasing volume and complexity of data in analytical and artificial intelligence systems raises the need for robust data governance frameworks to ensure data quality, security, and compliance. This project addresses such challenge by designing and implementing a data governance architecture in Amazon Web Services (AWS), applied to a practical use case: a book recommendation web application based on datasets from Goodreads Book Graph.

The proposed solution integrates multiple AWS services to cover the full data lifecycle. Amazon S3 serves as the data lake for raw and transformed data, while AWS Glue and its Data Catalog enable data transformation, classification, and cataloging. AWS Lake Formation and IAM enforce fine-grained access control and role-based permissions, ensuring secure and governed data access. Continuous monitoring and auditing are achieved through AWS CloudTrail and CloudWatch. On top of this governed pipeline, a recommendation model was developed, integrated into a web application and complemented with business intelligence dashboards in Amazon QuickSight.

The results demonstrate the feasibility of applying a governance framework aligned with the DAMA model within AWS, while maintaining scalability and cost efficiency. The implementation not only ensured compliance and transparency in data usage, but also improved the reliability of machine learning outcomes by guaranteeing consistent and high-quality datasets.

In conclusion, the project validates AWS as an effective platform for combining data governance, analytics, and artificial intelligence, highlighting its potential for future extensions in both technical and business domains.

Keywords: AWS, Machine Learning, Big Data, Data governance

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1. INTRODUCCIÓN

Este capítulo enmarca el resumen del trabajo realizado. Los puntos detallados pueden ser consultados en sus capítulos correspondientes.

1.1 Contexto y justificación

Con los avances tecnológicos y la actual era digital, los datos se han convertido en uno de los activos más importantes y valiosos para las organizaciones. Sin embargo, esto ha demandado un crecimiento en volumen y complejidad, lo cual plantea distintos retos para mantener la calidad, seguridad y el uso ético. La gobernanza del dato surge como un planteamiento esencial para establecer procesos, políticas y seguridad que aseguren un manejo eficaz y responsable de la información a lo largo de todo su ciclo de vida. Lo que permite no solo cumplir normas regulatorias, sino que impulsa la toma de decisiones basadas en datos, mejorando el rendimiento y los objetivos empresariales.

Por su parte, la computación en la nube ha cambiado la manera en la que las empresas procesan, almacenan y analizan datos. Las actuales plataformas que marcan tendencia en el mercado como Azure, AWS y Google Cloud ofrecen servicios flexibles y escalables que facilitan el acceso a herramientas avanzadas para el control de recursos. A pesar de ello, este entorno plantea nuevos desafíos a personas, empresas y sectores relacionados con la gobernanza del dato, como el control de acceso a la información, el tratamiento de datos sensibles, la protección contra amenazas informáticas y el cumplimiento de normas.

Bajo ese contexto, este trabajo se centra en la intersección que existe entre la gobernanza del dato y el uso de computación en la nube en la gestión de los datos, explorando los servicios disponibles que pueden integrarse en sistemas de desarrollo analítico. Se busca responder algunas preguntas como: ¿Qué servicios específicos de AWS facilitan la implementación de estrategias para la gobernanza del dato?¿Cuál es la influencia de un gobierno del dato correctamente implementado dentro de proyectos de analítica e IA desplegados en entornos cloud?

1.2 Planteamiento del problema

El auge de herramientas para analítica avanzada e inteligencia artificial basadas en la nube plantea la implementación de estrategias de gobernanza del dato para garantizar que los datos almacenados sean accesibles y seguros, implicando prácticas como catalogación de datos, control de metadatos, roles y accesos, y un monitoreo continuo que asegura una gestión adecuada del valor del dato a la vez que minimiza el riesgo operativo.

Para ello, se propone una metodología combinada entre el análisis teórico de los distintos servicios disponibles en AWS para la gobernanza de datos y el estudio de caso práctico que permita demostrar y aprovechar las capacidades analíticas y de inteligencia artificial ofrecidas en entornos cloud, proponiendo soluciones asociadas a riesgos de mal uso o gestión inadecuada de los datos.

1.3 Objetivos del proyecto

El objetivo general de este trabajo es analizar los servicios disponibles en AWS que facilitan la gobernanza de datos en sistemas analíticos e inteligencia artificial en la nube. Se pretende comprender cómo estas herramientas permiten gestionar de forma segura y eficiente los datos, garantizando su calidad, accesibilidad y cumplimiento normativo en entornos cloud.

Para ello, se plantean objetivos específicos que incluyen el estudio de las funcionalidades clave de AWS para la gobernanza del dato, el diseño y aplicación de un caso práctico que demuestre la integración segura de datos en proyectos de IA, la definición de políticas y roles para un control adecuado de accesos, la implementación de mecanismos automatizados de monitoreo y auditoría, y el establecimiento de procesos que aseguren la preparación y disponibilidad de datos para el entrenamiento y despliegue de modelos de machine learning.

1.4 Resultados obtenidos

El proyecto logró implementar un sistema de análisis de datos orientado a la gobernanza del dato en entornos de analítica y machine learning sobre AWS, cumpliendo todos los objetivos planteados. Se desplegaron servicios como AWS Lake Formation para la administración granular de permisos sobre datos en S3, Glue Data Catalog para centralizar la catalogación y clasificación de metadatos, e IAM para definir roles diferenciados y restringir accesos según las necesidades de administradores, analistas y científicos de datos. El ciclo de vida de los datos se estructuró en capas (raw, transformed, join) y se aplicaron ETL con AWS Glue, garantizando la calidad e integridad de los datasets y facilitando su utilidad en dashboards y modelos de recomendación, todo bajo un enfoque de mínimo privilegio y trazabilidad con CloudTrail y CloudWatch.

Asimismo, se diseñó y aplicó un caso práctico integrando servicios analíticos y de inteligencia artificial: un sistema de recomendación de libros basado en filtrado en contenido y colaborativo, desplegado sobre Lambda y accesible vía API Gateway. El monitoreo y auditoría automatizados con CloudTrail y CloudWatch permitieron obtener registros precisos de accesos y métricas de rendimiento, asegurando la protección, optimización de recursos y cumplimiento normativo. En conjunto, estas acciones consolidaron un ecosistema robusto y seguro para la explotación de datos en AWS, estableciendo buenas prácticas de gobernanza, observabilidad y gestión de identidades en ambientes analíticos y de inteligencia artificial.

1.5 Estructura de la memoria

Capítulo 1: Introducción Este capítulo es introductorio donde se explicarán los objetivos y motivaciones que han dado lugar al desarrollo de este Trabajo de Fin de Máster.

Capítulo 2: Antecedentes y Estado del Arte Este capítulo detallará los conceptos necesarios que se aplicarán en el desarrollo del trabajo, sirve como fundamentos en basados en bases teóricas.

Capítulo 3: Objetivos El capítulo describirá los objetivos a alcanazar con el desarrollo del proyecto, general como específicos, se basa en el proceso que se realizará para cumplir el planteamiento del problema, de la misma manera describirá los beneficios que se esperan del desarrollo del mismo.

Capítulo 4: Desarrollo del proyecto Este capítulo mostrará el proceso realizado en el desarrollo del cumplimiento del tema y los objetivos planteados, mediante actividades y metodologías, diseños, herramientas, algoritmos, entre otros.

Capítulo 5: Discusión Este capítulo enmarca el análisis crítico que enmarca la conexión de la teoría con la práctica. Se muestra la efectividad del uso de distintos servicios dentro del marco de gobernaza propuesto.

Capítulo 6: Conclusión Capítulo que presentará la descripcion de los resultados obtenidos en relación con el objetivo general y resultados personales en experiencia con la realización del proyecto, importancia y aprendizaje.

Capítulo 7: Futuras lineas de trabajo El capítulo presentará las propuestas que podrían aplicarse para continuar el trabajo, aspectos que se pueden tomar en cuenta y que pueden desarrollarse para complementar un aspecto en específico.

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE

En este capítulo se plantea los antecedentes y estado del arte, el contexto, la justificación y el planteamiento del problema a responder con este trabajo. El capítulo desarrollará los antecedentes en la industria actual, posibles brechas y cómo poder resolverlas.

2.1 Estado del arte

Para poder entender la situación actual que abordan los sectores tecnológicos y de inteligencia artificial, es necesario comprender conceptos clave acerca de la gobernanza del dato.

2.1.1. Gobernanza del dato

La gobernanza de datos se refiere al conjunto de políticas, procesos y estándares que una organización establece para gestionar sus activos de datos de manera efectiva [1]. De esta manera una gobernanza del dato maneja el uso de los datos como propiedad basandose en la administración y seguridad esperando que estos puedan ser accesibles, confiables y fiables.

La gobernanza del dato se basa en algunos componentes básicos:

Calidad de datos (Data Quality)

Este concepto se basa en el estado en el que se encuentran los datos en cuanto a su capacidad para cumplir con los requisitos de usos específicos.

Seguridad de datos (Data Security)

Se basa en un conjunto de prácticas que tratan de protejer los datos contra accesos no permitidos, evitar pérdidas y robos cuando estos están dentro de la organización o en tráfico.

Metadatos y catálogo (Metadata Management)

Los metadatos son un concepto que se basa en describir otros datos, proporcionando información adicional estructurada sobre la información principal, tales como su origen, formato, fecha o modificaciones.

Mientras que la catalogización de datos es el proceso de recopilar, organizar y gestionar datos a través de un sistema o herramienta, de esta manera se puede usar como inventario centralizado, donde se pueden explorar y comprender la existencia de los datos [2].

Lineage y trazabilidad (Data Lineage)

Este concepto se define como el registro detallado del flujo de los datos: desde su origen, pasando por todas las transformaciones, movimientos y sistemas por los que transitan, hasta su destino final o consumo. Permite conocer cómo, cuándo y por quién han sido modificados los datos, así como las reglas o procesos aplicados en cada etapa [3].

Privacidad y cumplimiento (Data Privacy & Compliance)

La privacidad se basa en la protección de la información personal y confidencial de los individuos frente a accesos no autorizados, usos indebidos y divulgaciones inadecuadas [4].

Dentro del cumplimiento se encuentran las organizaciones que aseguran que el tratamiento de los datos se realice conforme a las leyes, regulaciones y estándares aplicables como la GDPR en europa.

2.1.2. Marcos y estándares

Los marcos y estándares proporcionan las directrices, mejores prácticas y estructuras necesarias para que las organizaciones gestionen sus activos de datos de manera eficiente, segura y alineada con los objetivos estratégicos y regulatorios.

DAMA

Proporciona un compendio de mejores prácticas, procesos, roles y áreas de conocimiento para la gestión integral de los datos, incluyendo calidad, seguridad, metadatos, arquitectura y gobierno. Es una referencia central para la definición de políticas, estándares y responsabilidades en la organización The Data Management Body of Knowledge [5].

COBIT

COBIT incluye principios y prácticas aplicables a la gobernanza de datos, especialmente en lo relacionado con la seguridad, calidad y cumplimiento normativo [6].

ISO 8000

Es el estándar internacional más relevante para la calidad de datos, proporcionando principios y directrices para la gestión y aseguramiento de la calidad en los activos de datos.

Normativa europea y marcos regulatorios

El Reglamento (UE) 2022/868 (Reglamento de Gobernanza de Datos) y el RGPD (GDPR) son referencias obligadas en papers y estudios sobre gobernanza, ya que establecen requisitos legales y técnicos para la gestión, intercambio y protección de datos en la Unión Europea [7].

La gobernanza del dato es una pieza central en la construcción de sistemas analíticos e inteligentes fiables. Su implementación efectiva requiere tanto de marcos normativos sólidos como de una arquitectura tecnológica que facilite su despliegue.

Figura 2.1. Representación del gobierno del dato según DAMA [8]

2.1.3. Arquitecturas de Sistemas Analíticos

Sistema tradicional vs Moderno

Las arquitecturas de sistemas analíticos han evolucionado significativamente en la última década. Tradicionalmente, las organizaciones utilizaban Data Warehouses (almacenes de datos) como único repositorio centralizado donde se cargaban datos estructurados desde fuentes transaccionales, utilizando procesos ETL (Extract, Transform, Load) altamente controlados.

Los almacenes de datos tradicionales procesan información típicamente en ventanas de procesamiento por lotes que promedian 24 horas, con el 85 % de los datos críticos empresariales experimentando este retraso antes de estar disponibles para análisis [9]. Esta latencia inherente representa una limitación fundamental para la toma de decisiones en tiempo real.

Los estudios empíricos revelan que las arquitecturas monolíticas tradicionales experimentan una degradación del rendimiento de aproximadamente 28 % cuando los volúmenes de datos exceden el 70 % de su capacidad diseñada [9].

Sin embargo, con el auge del Big Data, surgió la necesidad de procesar grandes volúmenes de datos no estructurados y semiestructurados provenientes de múltiples fuentes (logs, sensores, redes sociales, etc.). Esto impulsó el desarrollo de Data Lakes, repositorios en bruto que permiten almacenar cualquier tipo de dato en su formato original y escalar de forma eficiente.

Más recientemente, ha emergido el enfoque híbrido conocido como Data Lakehouse, que combina las ventajas del almacenamiento flexible de los Data Lakes con las garantías transaccionales y de gobernanza propias de los Data Warehouses.

Componentes de un sistema analítico moderno

Un sistema analítico moderno suele estar compuesto por los siguientes bloques funcionales:

- **Ingesta de datos:** Se refiere a la captura de datos desde múltiples fuentes, en tiempo real o por lotes.
- **Almacenamiento:** Entornos de almacenamiento de datos, estos pueden ser locales, cloud o híbridos.
- **Procesamiento:** Incluye el tratamiento y transformación de los datos mediante pipelines ETL/ELT.
- **Visualización:** Permite a los usuarios de negocio acceder a dashboards e informes interactivos.

2.1.4. Gobernanza del Dato en Sistemas de Inteligencia Artificial

El desarrollo de sistemas de inteligencia artificial (IA), especialmente los basados en aprendizaje automático (machine learning), depende directamente de la calidad, disponibilidad y representatividad de los datos utilizados en el entrenamiento. Si los datos son incompletos, erróneos o reflejan sesgos estructurales, el sistema puede producir predicciones injustas o incluso discriminatorias. Además, la reutilización de datos personales en contextos distintos a los previstos inicialmente puede violar normativas de privacidad o principios éticos [10].

Principios éticos aplicados a la IA

Diversos organismos y marcos normativos han propuesto principios éticos para el uso responsable de datos en inteligencia artificial. Entre los más aceptados se encuentran:

- **Transparencia:** Los sistemas deben explicar, en la medida de lo posible, cómo toman decisiones, especialmente en contextos sensibles como salud, crédito o justicia.
- **Equidad:** Evitar la reproducción de sesgos o desigualdades sociales presentes en los datos.
- **Responsabilidad:** Existencia de mecanismos de rendición de cuentas ante errores o impactos negativos.
- **Privacidad:**Protección de los datos personales y consentimiento informado.

- **Seguridad:**Resiliencia ante ataques adversariales o filtraciones de información

La Unión Europea, mediante su propuesta de AI, y organismos como la OECD o la Unesco han reforzado estos principios mediante guías y recomendaciones específicas para sistemas algorítmicos [10].

2.1.5. Computación en la Nube y AWS

Beneficios de la nube para la gobernanza del dato

La computación en la nube ha transformado profundamente la forma en que las organizaciones almacenan, procesan y gestionan sus datos. Frente a las arquitecturas tradicionales on-premise, los entornos cloud ofrecen escalabilidad dinámica, alta disponibilidad, y una seguridad granular que permite aplicar políticas de gobernanza más efectivas y automatizadas [11].

Entre los beneficios clave para la gobernanza del dato destacan:

- **Escalabilidad horizontal y vertical:** Permite adaptar rápidamente los recursos a la cantidad de datos, usuarios o cargas de trabajo, sin comprometer la integridad del sistema.
- **Seguridad y control de acceso detallado:** A través de políticas como IAM (Identity and Access Management), se puede restringir el acceso a los datos según roles, proyectos o niveles de confidencialidad.
- **Auditoría y trazabilidad:**Servicios como CloudTrail permiten registrar de manera automatizada cada operación sobre los datos, facilitando la trazabilidad completa.
- **Cumplimiento normativo:** Los principales servicios en la nube cumplen con estándares internacionales como ISO/IEC 27001, HIPAA, SOC 2 o GDPR, lo cual permite a las organizaciones alinearse más fácilmente con marcos regulatorios.

AWS establece un modelo de responsabilidad compartida. Este modelo define con claridad qué aspectos de la seguridad y gobernanza son gestionados por el proveedor cloud (AWS) y cuáles recaen en el cliente:

- **Responsabilidad de AWS:** Seguridad de la infraestructura física, redes, hardware, virtualización, centros de datos y servicios nativos.
- **Responsabilidad del cliente:** Seguridad y gobernanza de los datos, usuarios, configuraciones, cifrado, auditoría, y cumplimiento normativo.

Servicios de AWS relevantes para la gobernanza del dato

AWS permite implementar en práctica los requerimientos del modelo de gobierno de datos según el marco regulatorio de DAMA. Definiendo a la arquitectura de datos como la estructura general y los recursos relacionados con los datos como una parte integral de la arquitectura empresarial [12].

Los servicios que cumplen las recomendaciones para cada dominio son:

- **Almacenamiento y control de acceso**

Amazon S3: Servicio de almacenamiento de objetos con control de versiones, cifrado en reposo y en tránsito, y gestión de políticas de acceso granular [13].

AWS Lake Formation: Permite construir data lakes gobernados, estableciendo permisos a nivel de tabla, columna o fila sobre datos almacenados en S3 [14].

- **Catalogación y transformación de datos**

AWS Glue y Glue Data Catalog: Herramientas serverless para catalogar, perfilar, limpiar y transformar datos mediante ETL o ELT. Glue Crawler [15] detecta esquemas automáticamente y los registra para su consulta con Athena o Redshift [16].

AWS DataBrew: Interfaz visual para limpieza y estandarización de datos por parte de perfiles no técnicos [17].

- **Consultas y análisis**

Amazon Athena: Motor serverless para consultar directamente datos en S3 mediante SQL, ideal para arquitecturas orientadas a datos abiertos (data lake) [18].

Amazon Redshift: Almacén de datos en la nube optimizado para análisis a gran escala con funciones avanzadas de seguridad y auditoría [19].

- **Gobernanza en IA y MLOps**

Amazon SageMaker Clarify y Model Monitor: Herramientas para detectar sesgos, explicar modelos y monitorear cambios en los datos que afectan el rendimiento [20].

SageMaker Pipelines: Orquestación del ciclo de vida de modelos de ML con trazabilidad y validación en cada paso [20].

- **Auditoría y cumplimiento**

AWS CloudTrail: Registro de actividad y cambios en recursos de AWS, útil para auditorías [21].

AWS Config: Supervisa configuraciones de recursos y verifica el cumplimiento de políticas [22].

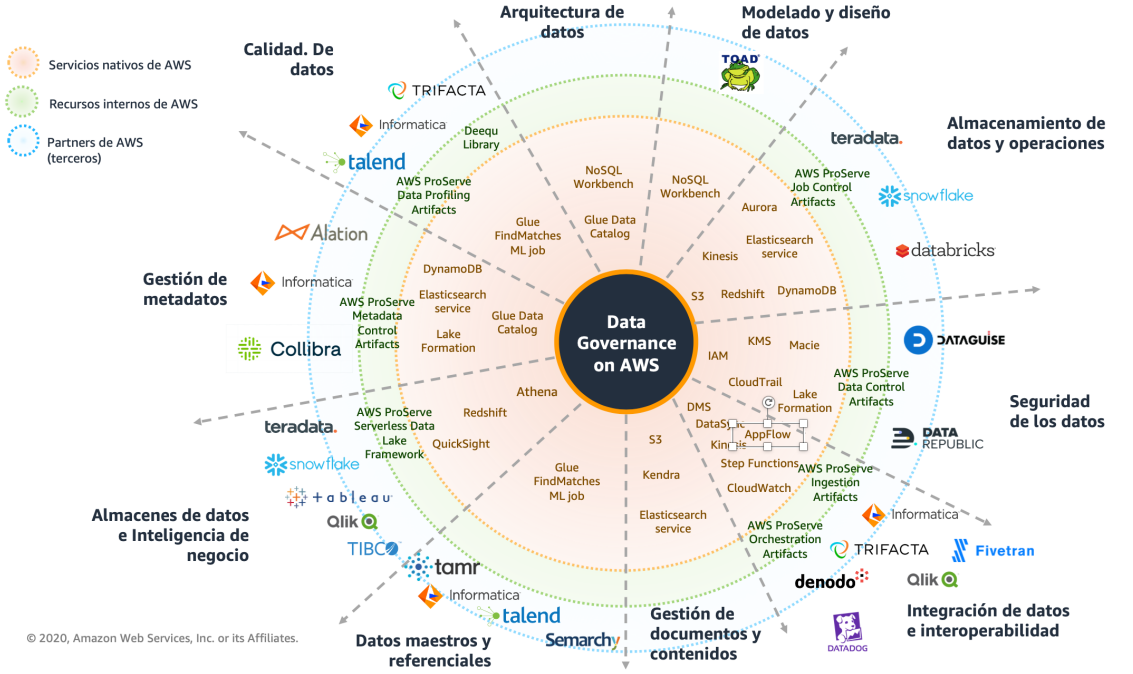


Figura 2.2. Mapeo de soluciones y relación con DAMA [12]

La gobernanza del dato actúa como eje vertebrador que garantiza la calidad, seguridad, trazabilidad y cumplimiento normativo a lo largo de todo el ciclo de vida de la información. Por su parte, los sistemas analíticos y los modelos de inteligencia artificial dependen intrínsecamente de datos confiables, éticos y bien gobernados para ofrecer valor real a las organizaciones. Los servicios cloud, especialmente AWS, habilitan esta integración mediante servicios escalables, auditables y automatizables que permiten implementar prácticas de gobernanza sólidas desde la ingesta hasta el consumo analítico o predictivo. Comprender cómo se articulan estos elementos resulta fundamental para diseñar soluciones que no solo sean técnicamente eficientes, sino también responsables, confiables y alineadas con principios éticos y regulatorios.

2.2 Contexto y justificación

Teniendo en cuenta los cambios tecnológicos y avances en el consumo, almacenamiento y procesamiento de datos, es necesario comprender el impacto que puede tener la información; los datos se han consolidado como el activo más valioso de las organizaciones modernas, impulsando la innovación y la diferenciación competitiva. El volumen de datos generados a nivel global ha experimentado un crecimiento exponencial, pasando de 4.4 zettabytes en 2013 a una proyección de 44 zettabytes para 2020 [23].

La Necesidad Imperante de Gobernanza en AWS para Analítica e IA

La rápida adopción de sistemas analíticos y de IA basados en la nube, particularmente en plataformas como AWS, introduce complejidades únicas en la gobernanza. Aunque AWS

proporciona una infraestructura robusta y segura , la responsabilidad de la seguridad y el control de los datos.

Componentes Esenciales de un Marco de Gobernanza del Dato

Un marco integral de gobernanza del dato generalmente comprende los siguientes componentes:

- **Estrategia y Objetivos:** Define la misión, los objetivos corporativos y alinea la gobernanza del dato con los objetivos generales del negocio.
- **Roles y Responsabilidades:** Establece roles claros, como el Chief Data Officer (CDO), propietarios de datos, administradores de datos y custodios de datos, y sus respectivas responsabilidades.
- **Políticas y Estándares:** Documenta las reglas para la clasificación de datos, retención, cifrado, acceso y uso.
- **Procesos y Procedimientos:** Define los flujos de trabajo para las solicitudes de acceso a datos, la remediación de la calidad y la gestión del ciclo de vida
- **Tecnología y Herramientas:** Implica la implementación de catálogos de datos, herramientas de linaje, herramientas de garantía de calidad y mecanismos de control de acceso.
- **Formación y Culturización:** Educación a los empleados sobre el uso ético de los datos, la protección y el cumplimiento.

Impacto de la gobernanza de datos y las normativas en el mundo empresarial

La gobernanza de datos se ha consolidado como un factor esencial para la competitividad y sostenibilidad de las empresas, especialmente en sectores regulados como finanzas, salud y retail. La correcta gestión de datos bajo marcos normativos como el RGPD (Reglamento General de Protección de Datos) y el nuevo Reglamento Europeo de Gobernanza de Datos 2022/868 garantiza la protección, trazabilidad y calidad de la información, al tiempo que responde a demandas crecientes de transparencia y responsabilidad en la toma de decisiones [24].

La práctica de modelos de gobernanza y cumplimiento con el GDPR implica ventajas importantes, como el aumento de la confianza del cliente, la disminución de riesgos legales y el impulso a la innovación analítica, según distintas investigaciones. La adaptación de la empresa al RGPD también conlleva la oportunidad de eludir castigos altos, asegurar la protección de los clientes actuales y futuros, y fortalecer la imagen corporativa en mercados difíciles. En realidad, en Europa y en el mundo entero, cumplir con las normas y poner en práctica herramientas tecnológicas que sean seguras y auditables son requisitos cada vez más necesarios para poder conseguir contratos, acuerdos estratégicos y licitaciones.

Ejemplos como Grupo Danone (Global) que lanzó el programa “One Source of Truth” para centralizar y gobernar sus datos de productos, proveedores y logística en más de 60 países, obteniendo una reducción del “time-to-market” de productos en casi 20 %, Banco Ga-

licia(Argentina) que implementó un programa de gobierno de datos a nivel corporativo para mejorar la calidad de la información financiera, reduciendo el 30 % en errores de reconciliación de cuentas y mayor eficiencia en la generación de informes regulatorios, y Mercado Libre(Latinoamérica) que a medida que escalaron en volumen de usuarios y operaciones, implementaron gobernanza automatizada de datos sobre su data lake basado en AWS. Establecieron reglas de acceso granulares, automatizaron la anonimización de datos personales y desplegaron paneles de monitoreo de calidad, demuestran que adoptar prácticas avanzadas de gobierno del dato y automatización cloud logra mejoras significativas en la calidad de los informes, cumplimiento normativo y agilidad de negocio [25].

2.3 Planteamiento del problema

Las organizaciones actualmente gestionan volúmenes masivos de datos procedentes de múltiples fuentes: IoT, transaccional, logs, redes sociales entre otros. Este escenario ha impulsado el despliegue de sistemas analíticos e IA, pero también ha revelado fallos críticos relacionados con la gobernanza del dato, esenciales para sustentar decisiones confiables y conformes a regulaciones.

Sin embargo, este crecimiento acelerado ha puesto en evidencia una problemática crítica: la falta de gobernanza efectiva de los datos. Muchas empresas adoptan tecnologías analíticas e IA sin contar con una estrategia clara que garantice la calidad, seguridad, trazabilidad y uso ético de los datos que alimentan estos sistemas. Esto genera riesgos significativos, como:

- Sesgos y errores en los modelos de IA debido a datos incompletos, mal etiquetados o no representativos.
- Fugas o accesos indebidos a información sensible, comprometiendo la privacidad y el cumplimiento de regulaciones como el GDPR.
- Falta de trazabilidad (lineage) que impide auditar el origen y transformaciones de los datos usados en decisiones críticas.
- Duplicidad, inconsistencia y silos de datos, que dificultan la analítica confiable y el gobierno corporativo.
- Desconexión entre equipos técnicos y de negocio, generando una brecha entre los datos disponibles y su uso estratégico.

Una gobernanza débil se traduce en datos inconsistentes, no confiables, duplicados o inaccesibles, lo cual afecta directamente la calidad de los modelos analíticos e inteligencia artificial. De hecho, se estima que el 85 % de los proyectos de IA fallan o no alcanzan los resultados esperados debido a problemas relacionados con la calidad y trazabilidad de los datos.

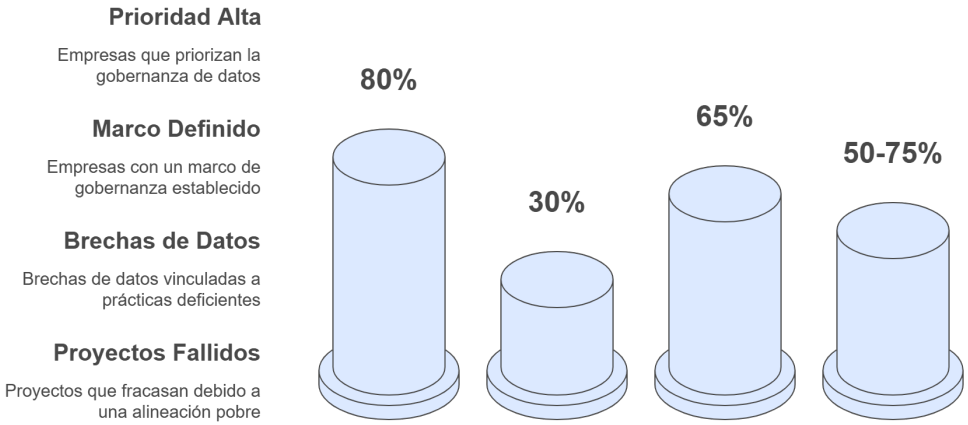


Figura 2.3. Priorización de un gobierno de datos

Sin un gobierno claro sobre el linaje de los datos, sus transformaciones y uso, los modelos operan sin rumbo u objetivo, generando decisiones automatizadas sin una base transparente o ética.

Capítulo 3. OBJETIVOS

El objetivo general de este trabajo es analizar los servicios disponibles capaces de integrarse en sistemas analíticos y de inteligencia artificial para aplicar la gobernanza de datos en la plataforma de computación en la nube AWS.

3.1 Objetivos específicos

Los objetivos específicos del proyecto son:

- Describir las funciones y servicios de AWS para la gobernanza de datos, como AWS Lake Formation, AWS Glue Data Catalog, AWS CloudTrail, AWS IAM, enfocados en la catalogación, control de metadatos, gestión de roles y accesos, y monitoreo continuo de los datos.
- Establecer procesos para extracción, transformación y carga, catalogación y clasificación de datos que faciliten la calidad, integridad y disponibilidad de los datos para el entrenamiento de modelos de inteligencia artificial.
- Definir políticas, roles y responsabilidades necesarias para implementar una gobernanza de datos, incluyendo el control de acceso y la gestión de identidades mediante servicios como AWS Lake Formation Y AWS IAM.
- Diseñar y aplicar un caso práctico que utilice servicios analíticos e inteligencia artificial en AWS para demostrar la integración efectiva de servicios de gobernanza del dato, garantizando el acceso controlado a datos y su posterior uso y despliegue en modelos de inteligencia artificial.
- Implementar mecanismos de monitoreo, auditoría y cumplimiento normativo automatizados utilizando herramientas de AWS como AWS CloudTrail y AWS CloudWatch para asegurar la protección.

3.2 Beneficios del proyecto

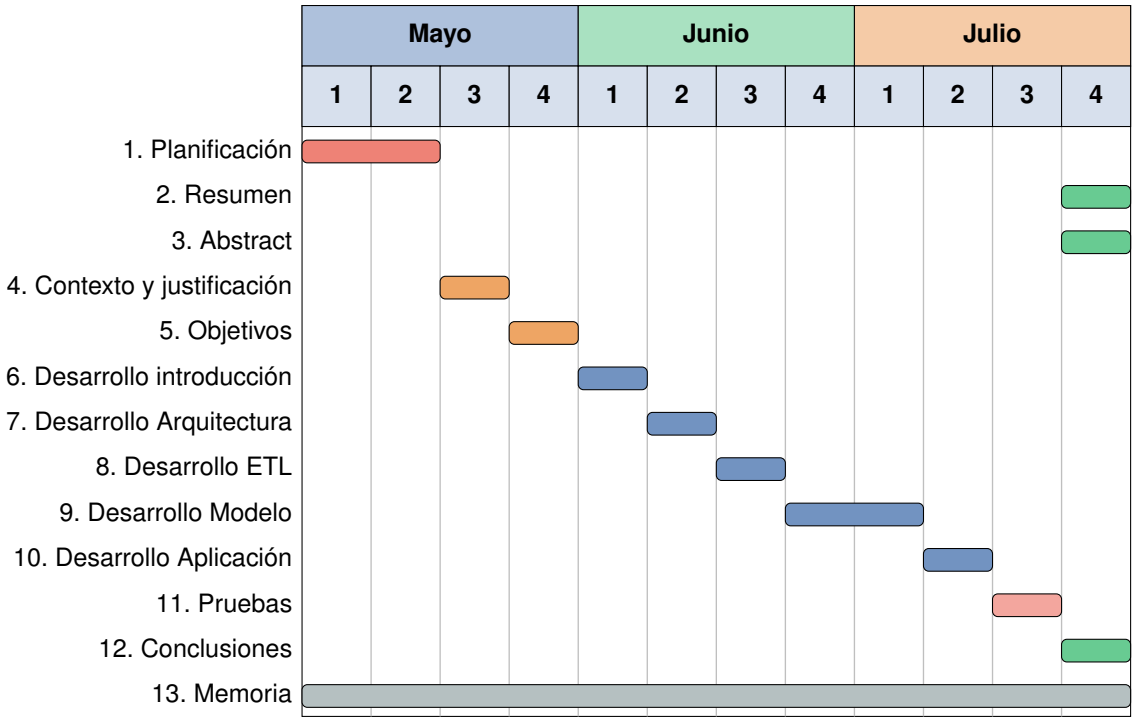
Los beneficios esperados de este proyecto son comprender y mejorar significativamente la calidad, seguridad y uso de datos para la toma de decisiones basadas en información y que se encuentren accesibles, precisos y confiables para el uso de modelos de inteligencia artificial sin riesgos operativos y cumpliendo normativas mediante monitoreo continuo. Esto a través de servicios de AWS, automatizando procesos de gobernanza del dato, con el fin de reducir costos, escalar la gestión de los datos y mantener la trazabilidad de la información.

Capítulo 4. DESARROLLO DEL PROYECTO

Este capítulo describe las actividades desarrolladas en el proyecto. Se menciona la planificación, los recursos necesarios para el desarrollo, presupuesto y viabilidad. El proyecto presenta el desarrollo e implementación de un sistema de recomendaciones basado en un modelo de aprendizaje automático, todo esto soportado sobre infraestructura en la nube. Se muestran todos los servicios usados en AWS y el desarrollo completo desde los procesos ETL sobre los datos, la creación del modelo y el uso de datos para la aplicación final.

4.1 Planificación del proyecto

Tabla 4.1. Diagrama de Gantt del proyecto



4.2 Implementación y desarrollo

Se plantea el siguiente caso de uso, para un negocio dedicado a la venta de libros, es necesario desarrollar una aplicación que facilite la compra de libros a los usuarios basados en recomendaciones de otros libros que hayan usado. Toda la infraestructura se encontrará sobre la plataforma en la nube de AWS y los datos que serán utilizados corresponden a datos de reseñas de libros del público, datos de libros y datos de clientes.

Sistema de recomendaciones empleando herramientas para gobernanza de datos sobre infraestructura en la nube

Para el despliegue de este sistema y con el objetivo de que sea escalable y replicable, se usa el servicio de CloudFormation de AWS; este servicio nos ayuda a modelar y configurar recursos en AWS, reduciendo los tiempos de despliegue [26]. Esta selección se prefiere sobre

otras herramientas como AWS CDK o Terraform por su despliegue automático de recursos en AWS.

Para el desarrollo del proyecto se ha definido una serie de etapas, permitiendo un entorno organizado. A continuación, se indican las etapas:

1. **Arquitectura:** Planteamiento y creación de la arquitectura del sistema completo, los servicios que son usados y los procesos y etapas del proyecto.
2. **Obtención de los datos:** Recopilación de los datos necesarios para el sistema y el modelo de machine learning.
3. **Preparación de los datos - Proceso ETL:** Implica los procesos de extracción, transformación y carga, sobre los cuales se aplica servicios de AWS para la gobernaza de datos.
4. **Desarrollo del modelo de recomendaciones:** Selección y creación del algoritmo adecuado para el sistema de recomendaciones. Las métricas obtendidas pueden tener un respectivo seguimiento a través de servicios de AWS.
5. **Desarrollo de la aplicación:** Uso de los datos sobre el modelo de machine learning el cual funciona sobre una aplicación web despegada en la nube.

4.2.1. Arquitectura

La arquitectura propuesta está diseñada para implementar una solución de gobernanza del dato en sistemas analíticos e inteligencia artificial utilizando servicios de AWS.

La base del sistema parte de una fuente de datos semi-estructurados en formato JSON (JavaScript Object Notation), el cual es un formato de datos que sigue la estructura de objetos en JavaScript [27] y culmina con una aplicación de recomendaciones de libros basadas en machine learning, complementada con capacidades de análisis de negocio.

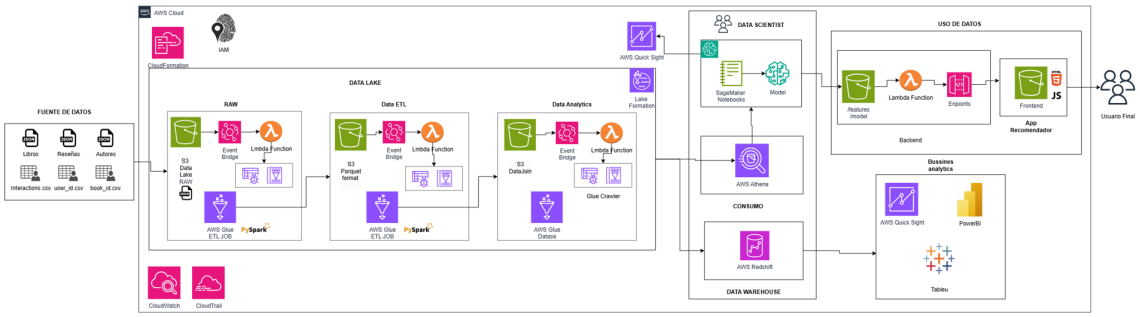


Figura 4.1. Arquitectura general del sistema

Como se muestra en la figura 4.1 el sistema comienza con la ingesta de datasets que contienen información sobre:

- Libros

- Reseñas
- Autores

Los archivos en formato JSON, representan datos semiestructurados y heterogéneos que son la base de los procesos analíticos y de recomendación.

También se tienen los siguientes archivos:

- interactions
- user_id
- book_id

Estos corresponden a información complementaria y cumplen un papel fundamental en la preparación de los datos para el modelo de machine learning.

A continuación, los archivos JSON y CSV son almacenados inicialmente en un Data Lake construido sobre buckets Amazon S3 [13]. El bucket se estructura de la siguiente manera:

- books/
- authors/
- reviews/
- books_id
- reads_int
- user_id

Mediante un proceso de transformación usando Spark [28] sobre ETL Jobs [29], una herramienta disponible de AWS Glue, se pasa de formato JSON a Parquet [30] un formato columnar que mejora la comprensión de los datos sin perder información, mejorando el rendimiento de consultas analíticas. Estos archivos son guardados en otro bucket S3.

Para garantizar la gobernanza del dato, se utilizan AWS Glue Crawlers para catalogar automáticamente los datos transformados en el AWS Glue Data Catalog. Esto permite su consulta posterior mediante Amazon Athena, lo que facilita la exploración de los datos sin necesidad de moverlos.

A continuación, se implementa un proceso ETL (Extract, Transform, Load) con AWS Glue Jobs, donde se aplican transformaciones adicionales necesarias: limpieza, normalización y enriquecimiento de los datasets.

Los datos transformados se integran en un único dataset de valor analítico a través de procesos de join entre libros, reseñas e interacciones, y se almacenan en un nuevo bucket S3 DataJoin.

Los datos del bucket DataJoin son nuevamente catalogados mediante Glue y puestos a disposición para análisis con Athena o ingestados directamente en Amazon Redshift, un almacén de datos columnar optimizado para grandes volúmenes de consultas analíticas complejas.

Redshift centraliza los datos necesarios para:

- Entrenar modelos de machine learning.
- Generar reportes analíticos con herramientas de Business Intelligence (BI).

Desde los datos almacenados en Redshift o S3, se habilita un entorno de ciencia de datos donde se desarrollan notebooks y scripts de entrenamiento de modelos. Este entorno puede implementarse sobre instancias EC2 o servicios como Amazon SageMaker, según los requerimientos.

El modelo entrenado es un sistema de recomendación de libros basado en técnicas como filtrado colaborativo, el cual se entrena con las interacciones usuario-libro. Posteriormente, el modelo se despliega como un servicio accesible desde la capa de aplicación.

La recomendación de libros se materializa en una aplicación web compuesta por:

- Un frontend que permite al usuario visualizar libros y calificar/reseñar.
- Un backend que expone un API para acceder a recomendaciones y persistir información de usuarios.

Ambos componentes están almacenados y desplegados en buckets de Amazon S3, pudiendo además integrarse con AWS API Gateway y AWS Lambda o ECS con Fargate para mayor escalabilidad y control.

El sistema también contempla una capa de análisis de negocio que permite evaluar el comportamiento de los usuarios, los libros más recomendados y el rendimiento del modelo. Para ello, se utilizan herramientas como:

- Amazon QuickSight

O puede ser usado para otras herramientas externas como:

- Power BI
- Tableau

Estos conectores pueden acceder tanto a los datos de Redshift como a Athena, lo que permite obtener dashboards e informes interactivos.

Toda la infraestructura es monitoreada y auditada a través de servicios como:

- AWS CloudWatch para logs y métricas.
- AWS CloudTrail para auditoría de eventos.
- IAM (Identity and Access Management) para controlar los permisos y asegurar el acceso a los recursos según el principio de menor privilegio.

Además, la infraestructura puede ser provisionada y mantenida de forma declarativa mediante AWS CloudFormation, lo cual mejora la reproducibilidad y el control de versiones.

4.2.2. Obtención de los datos

Para el sistema en general, el modelo de recomendaciones se basará en los datos obtenidos del departamento de ciencia de la computación e ingeniería de la universidad UC San Diego Jacobs School of engineering, exactamente publicado por el profesor Julian McAuley [31].

El conjunto de datos consta de distintos datasets recolectados mediante webscraping de la plataforma de libros y reseñas online GoodReads. Los cuales fueron obtenidos solo para uso

académico. Además, se garantiza la seguridad de los datos personales dado que los ID de usuarios e ID de revisión están anonimizados.

Se recopila tres conjuntos de datos:

- Metadatos de los libros
- Interacciones entre libros y usarios
- Reseñas detalladas de los libros de los usuarios

Estos conjuntos de datos se pueden unir a través de los identificadores libro/usuario/reseña [31].

Estadísticas básicas de los datos:

- 2.360.655 libros (1.521.962 obras, 400.390 series de libros, 829.529 autores)
- 876.145 usuarios
- 228.648.342 interacciones de libros de usuario en las estanterías de los usuarios (incluye 112.131.203 lecturas y 104.551.549 valoraciones)[31]

En total, los datasets son de alrededor de 15 GB a 20 GB, dependiendo de si se decide usar el conjunto de datos de interacciones detalladas de usuario-libro.

Inspección de los datos iniciales

El objetivo es conocer la estructura de los datos en cada conjunto, analizar las columnas y verificar la calidad de los datos en bruto.

Para el primer dataset Books se tiene la siguiente estructura:


```
{

        "isbn": "0312853122",
        "text_reviews_count": "1",
        "series": [],
        "country_code": "US",
        "language_code": "",
        "popular_shelves": [
            {
                "count": "3",
                "name": "to-read"
            },
            {
                "count": "1",
                "name": "p"
            },
            {
                "count": "1",
                "name": "collection"
            },
            {
                "count": "1",
```

```
                "name": "w-c-fields"
            },
            {
                "count": "1",
                "name": "biography"
            }
        ],
        "asin": "",
        "is_ebook": "false",
        "average_rating": "4.00",
        "kindle_asin": "",
        "similar_books": [],
        "description": "",
        "format": "Paperback",
        "link": "https://www.goodreads.com/book/show/5333265-w-c-
           fields",
        "authors": [
            {
                "author_id": "604031",
                "role": ""
            }
        ],
        "publisher": "St. Martin's Press",
        "num_pages": "256",
        "publication_day": "1",
        "isbn13": "9780312853129",
        "publication_month": "9",
        "edition_information": "",
        "publication_year": "1984",
        "url": "https://www.goodreads.com/book/show/5333265-w-c-
           fields",
        "image_url": "https://images.gr-assets.com/books/1310220028
           m/5333265.jpg",
        "book_id": "5333265",
        "ratings_count": "3",
        "work_id": "5400751",
        "title": "W.C. Fields: A Life on Film",
        "title_without_series": "W.C. Fields: A Life on Film"
}
```


La tabla 4.2 muestra el tipo y la descripción de los campos.

Tabla 4.2. Campos del JSON del libro *W.C. Fields: A Life on Film*

Campo	Tipo	Descripción
isbn	string	ISBN de 10 dígitos del libro.
text_reviews_count	string (numérico)	Cantidad de reseñas escritas por usuarios.
series	array	Lista de series a las que pertenece el libro.
country_code	string	Código de país asociado al libro (ISO 3166).
language_code	string	Código del idioma. Vacío si no está especificado.
popular_shelves	array de objetos	Estanterías populares donde los usuarios han guardado el libro.
asin	string	Código ASIN de Amazon. Vacío si no aplica.
is_ebook	string (booleano)	Indica si el libro es un eBook ("true/"false").
average_rating	string (numérico)	Valoración promedio del libro (1.0 a 5.0).
kindle_asin	string	Código ASIN para Kindle. Vacío si no aplica.
similar_books	array	Lista de libros similares.
description	string	Descripción del contenido del libro.
format	string	Formato del libro (ej. Paperback, Hardcover).
link	string (URL)	Enlace a la página del libro en Goodreads.
authors	array de objetos	Lista de autores con ID y rol.
publisher	string	Nombre de la editorial.
num_pages	string (numérico)	Número de páginas del libro.
publication_day	string (numérico)	Día de publicación.
isbn13	string	ISBN de 13 dígitos.
publication_month	string (numérico)	Mes de publicación.
edition_information	string	Información adicional de la edición.
publication_year	string (numérico)	Año de publicación.
url	string (URL)	URL del libro (alternativa a ‘link‘).
image_url	string (URL)	Enlace a la imagen de la portada.
book_id	string (numérico)	ID único del libro en Goodreads.
ratings_count	string (numérico)	Total de valoraciones recibidas.
work_id	string (numérico)	ID del trabajo literario en Goodreads.
title	string	Título completo del libro.
title_without_series	string	Título sin el nombre de la serie.

Esta entidad presenta cada título único en la base de datos, con su información bibliográfica, metadatos, atributos de contenido y popularidad, lo que ayudará a la recomendación a los usuarios.

Esta entidad se convierte en un punto central que ayuda a conectar con varias relaciones.

Libros - Autores: para recomendar libros del mismo autor.

Libros - Géneros/Etiquetas: para entender preferencias temáticas.

Libros - Otros libros: por similitud (formato, género, tema, popularidad, etc.)

También puede aportar características necesarias para modelos basados en contenido.

- Género o etiquetas (popular_shelves)
- Número de páginas (num_pages)

- Autor(es)
- Valoración media (average_rating)
- Formato (Paperback, eBook, etc.)
- Descripción (description)
- Idioma (language_code)

Estas variables ayudan a construir modelos que recomienden libros similares a los que le han gustado a un usuario.

El segundo dataset Reviews tiene la siguiente estructura:


```
{
    "user_id": "8842281e1d1347389f2ab93d60773d4d",
    "book_id": "24375664",
    "review_id": "5cd416f3efc3f944fce4ce2db2290d5e",
    "rating": 5,
    "review_text": "Mind blowingly cool. Best science fiction I
       've read in some time. I just loved all the descriptions
        of the society of the future - how they lived in trees,
        the notion of owning property or even getting married
       was gone. How every surface was a screen. \n The
       undulations of how society responds to the Trisolaran
       threat seem surprising to me. Maybe its more the Chinese
        perspective, but I wouldn't have thought the ETO would
       exist in book 1, and I wouldn't have thought people
       would get so over-confident in our primitive fleet's
       chances given you have to think that with superior
       science they would have weapons - and defenses - that
       would just be as rifles to arrows once were. \n But the
       moment when Luo Ji won as a wallfacer was just too cool.
        I may have actually done a fist pump. Though by the way
       , if the Dark Forest theory is right - and I see no
       reason why it wouldn't be - we as a society should
       probably stop broadcasting so much signal out into the
       universe.",
    "date_added": "Fri Aug 25 13:55:02 -0700 2017",
    "date_updated": "Mon Oct 09 08:55:59 -0700 2017",
    "read_at": "Sat Oct 07 00:00:00 -0700 2017",
    "started_at": "Sat Aug 26 00:00:00 -0700 2017",
    "n_votes": 16,
    "n_comments": 0
}
```


El tipo de variable y su descripción se muestra en la tabla 4.3

Tabla 4.3. Descripción de los campos de la entidad *Review*

Campo	Tipo	Descripción
user_id	string (hash/UUID)	Identificador único del usuario que escribió la reseña. Se usa para relacionarlo con la entidad Usuario.
book_id	string (numérico)	ID del libro reseñado. Relación directa con la entidad Libro.
review_id	string (hash/UUID)	Identificador único de la reseña. Sirve como clave primaria.
rating	integer	Calificación numérica otorgada al libro (por lo general de 1 a 5).
review_text	string (texto)	Opinión escrita por el usuario. Puede incluir análisis, impresiones y spoilers.
date_added	string (timestamp)	Fecha y hora en la que la reseña fue registrada por primera vez.
date_updated	string (timestamp)	Fecha de la última edición de la reseña, si fue modificada.
read_at	string (timestamp)	Fecha en que el usuario terminó de leer el libro. Útil para análisis de comportamiento lector.
started_at	string (timestamp)	Fecha en la que el usuario comenzó a leer el libro. Permite calcular el tiempo de lectura.
n_votes	integer	Número de votos recibidos por la reseña (por ejemplo, votos de utilidad).
n_comments	integer	Número de comentarios que otros usuarios hicieron sobre esta reseña.

Esta entidad representa una relación directa y explícita entre libro y usuario, donde el usuario no solo califica al libro a través del campo rating, sino que además es posible encontrar una opinión escrita, indicando aspectos y experiencias de la lectura.

Para cada reseña se puede tener:

- Una calificación numérica
- Texto libre
- Tiempos de lectura
- Información sobre utilidad y comentarios de otros usuarios

Si el objetivo de los datos es usarlos para un sistema de recomendaciones, esta entidad es importante dado que es posible personalizar las recomendaciones de forma más precisa.

Esta entidad se relaciona directamente con ***Book***, lo que permitiría entrenar modelos de Collaborative Filtering [32] utilizando la lógica usuario-libro-rating e identificar patrones de comportamiento lector(frecuencia de lectura, gustos).

Otros campos como review_text pueden usarse para análisis de sentimientos, extracción de temas, filtrado de recomendaciones según preferencias o contenido sensible(lenguaje explícito, detección de spoilers).

Esta entidad representa un feedback explícito de los usuarios, aportando contenido textual

de los libros, permitiendo entrenar modelos más precisos, analizando el comportamiento del lector.

El siguiente dataset llamado Authors, contiene la siguiente estructura.

```
{
"average_rating": "3.98",
"author_id": "604031",
"text_reviews_count": "7",
"name": "Ronald J. Fields",
"ratings_count": "49"
}
```

El tipo de variable y su descripción se muestra en la tabla:

Tabla 4.4. Descripción de los campos de la entidad *Author*

Campo	Tipo	Descripción
author_id	string (numérico o UUID)	Identificador único del autor. Se utiliza para relacionarlo con los libros en los que ha participado.
name	string	Nombre del autor. Puede incluir nombres reales o seudónimos.
average_rating	float	Promedio de calificaciones recibidas por todos los libros del autor. Útil para evaluar su recepción general.
ratings_count	integer	Número total de puntuaciones recibidas por los libros del autor. Puede indicar popularidad o alcance.
text_reviews_count	integer	Número total de reseñas escritas por los lectores sobre los libros del autor. Puede reflejar nivel de engagement.

Esta entidad muestra una descripción detallada de los distintos autores.

Para los archivos complementarios.

La tabla Interactions como se muestra en la tabla 4.5, muestra las interacciones entre usuarios y libros.

Tabla 4.5. Descripción de los campos de la entidad *interactions*

Campo	Tipo	Descripción
user_id	integer	ID interno (indexado) del usuario. Necesita mapearse a user_id_map.csv para obtener el ID real.
name	string	Nombre del autor. Puede incluir nombres reales o seudónimos.
book_id	integer	ID interno del libro. Se mapea a book_id_map.csv.
is_read	boolean	Indica si el usuario ha leído el libro. Variable binaria..
rating	integer	Calificación otorgada por el usuario (por ejemplo, de 1 a 5).
is_reviewed	boolean	indica si el usuario escribió una reseña sobre ese libro.

user_id_map ayuda a mapear el identificador interno con el identificador real del usuario. Esto ayuda a tener una mejor eficiencia computacional.

Su estructura se muestra en la tabla 4.6.

Tabla 4.6. Descripción de los campos de la entidad *user_id_map*

Campo	Tipo	Descripción
user_id_csv	integer	ID numérico interno del usuario usado en interactions.csv
user_id	string	Hash o UUID real del usuario.

book_id, esta entidad permite la traducción entre el ID interno del libro y su ID real. Su estructura se muestra en la siguiente tabla 4.7

Tabla 4.7. Descripción de los campos de la entidad *book_id_map*

Campo	Tipo	Descripción
book_id_csv	integer	ID numérico interno del libro usado en interactions.csv
book_id	string	ID real del libro (como el que aparece en Books)

El conjunto de datos en bruto se encuentra distribuido en múltiples formatos, principalmente JSON y CSV, lo que dificulta su integración y análisis. Para garantizar un procesamiento uniforme y optimizar su consumo por parte de la aplicación y los procesos analíticos, es necesario unificar el formato y almacenarlos en una plataforma confiable, escalable y orientada a análisis.

Para ello, se implementa un flujo de preprocesamiento y transformación utilizando servicios gestionados de AWS. En particular, se emplean AWS Glue ETL Jobs, que permiten ejecutar transformaciones mediante scripts basados en PySpark, facilitando tareas como:

- Conversión de formatos (de JSON/CSV a Parquet, formato columnar optimizado para análisis de grandes volúmenes de datos).
- Limpieza y estandarización de datos (corrección de valores atípicos, imputación de valores nulos, normalización de textos)
- Validación y ajuste de tipos de variables para asegurar la consistencia semántica.
- Optimización de compresión y particionamiento para mejorar el rendimiento en consultas posteriores.

Una vez procesados, los archivos en formato Parquet se almacenan en un Data Lake en Amazon S3, lo que permite un acceso escalable, seguro y de bajo coste. Estos datos son catalogados mediante AWS Glue Data Catalog, habilitando su consulta directa a través de Amazon Athena para análisis exploratorio y extracción de información.

Finalmente, los datos transformados se integran en Amazon Redshift, que actúa como Data Warehouse para soportar la aplicación de recomendaciones, la generación del modelo de Machine Learning y la visualización de métricas en Amazon QuickSight.

4.2.3. Preparación de los datos - Procesos ETL

Como se vio en el apartado anterior, los datos necesitan pasar por una capa de preprocesado antes de ser utilizados en sistemas analíticos.

En este proyecto, los datos en bruto provienen del conjunto Goodreads Book Graph Datasets, distribuidos en formatos heterogéneos como JSON y CSV. Dichos formatos, si bien son adecuados para el almacenamiento inicial, no resultan óptimos para el consumo analítico ni para el entrenamiento de modelos de machine learning debido a su falta de estandarización, volumen y redundancia.

Para resolver estos inconvenientes, el preprocesamiento se centra en tres objetivos principales:

- La unificación de todos los archivos en un único formato columnar (Parquet) que permite mayor eficiencia en consultas y compresión de datos.
- La validación de tipos de variables, eliminación de inconsistencias y normalización de campos clave.
- La preparación para el consumo analítico y de IA, almacenamiento de los datos procesados en un Data Lake confiable y escalable (Amazon S3), con catalogación en AWS Glue para su explotación posterior en servicios como Amazon Athena, Amazon Redshift y Amazon SageMaker.

De esta manera, el preprocesamiento asegura que la información fluya desde su estado crudo hasta convertirse en un activo gobernado y listo para aplicaciones analíticas y predictivas dentro de la arquitectura en AWS.

Cabe mencionar que se aplica infraestructura como código para la creación de la arquitectura en AWS. Para esto, se usa AWS CloudFormation el servicio nativo de Infrastructure as Code (IaC) de AWS.

Algunas características de que comparten con otras plataformas se pueden ver en la tabla 4.8.

Tabla 4.8. Comparación entre AWS CloudFormation, Terraform y AWS CDK

Característica	**AWS CloudFormation**	**Terraform**	**AWS CDK**
Proveedor	AWS nativo	HashiCorp (multicloud)	AWS nativo
Lenguaje	Declarativo (YAML / JSON)	Declarativo (HCL)	Imperativo (Python, TypeScript, Java, etc.)
Integración con AWS	Total, soporta todos los servicios en cuanto se lanzan	Buena, pero puede tardar en soportar nuevos servicios	Total, transpila a CloudFormation
Multicloud / Híbrido	No, solo AWS	Sí, soporta AWS, Azure, GCP, Kubernetes, etc.	No, solo AWS
Curva de aprendizaje	Baja (plantillas YAML/JSON legibles)	Media (HCL específico de Terraform)	Media/Alta (requiere conocimientos de programación)
Ecosistema y comunidad	Limitado a AWS	Muy grande y activa	Creciente, enfocada en AWS
Manejo de estados	Implícito, gestionado por AWS	Explícito, requiere backend	Gestionado por CloudFormation
Gobernanza y seguridad	Alta, integración con IAM y AWS Organizations	Media, requiere configuración adicional	Alta, igual que CloudFormation

El uso de AWS CloudFormation frente a alternativas como Terraform o AWS CDK se justifica principalmente porque se trata de un servicio nativo de AWS, totalmente integrado con su ecosistema, lo que garantiza soporte inmediato para todos los servicios en el momento de su lanzamiento y una gestión simplificada del ciclo de vida de la infraestructura.

Al estar completamente gestionado por AWS, CloudFormation elimina la necesidad de mantener manualmente estados de despliegue, ofreciendo mayor seguridad y gobernanza al integrarse de forma natural con IAM. Esto lo convierte en la opción más adecuada en un proyecto centrado en gobernanza del dato dentro de AWS, ya que prioriza la trazabilidad, la consistencia y la conformidad normativa, aspectos críticos para entornos analíticos y de inteligencia artificial en la nube.

Lanzamiento de infraestructura en CloudFormation

Para ello en la consola de AWS en el buscador seleccionamos el servicio de AWS CloudFormation.

Se abrirá la ventana de inicio, creamos un nuevo Stack(Pila), y seleccionamos nuestra plantilla en formato yaml, como se muestra en la figura 4.2, el cual es un lenguaje de serialización de datos, fácil de comprender y mayormente utilizado en el diseño de archivos de configuración [33].

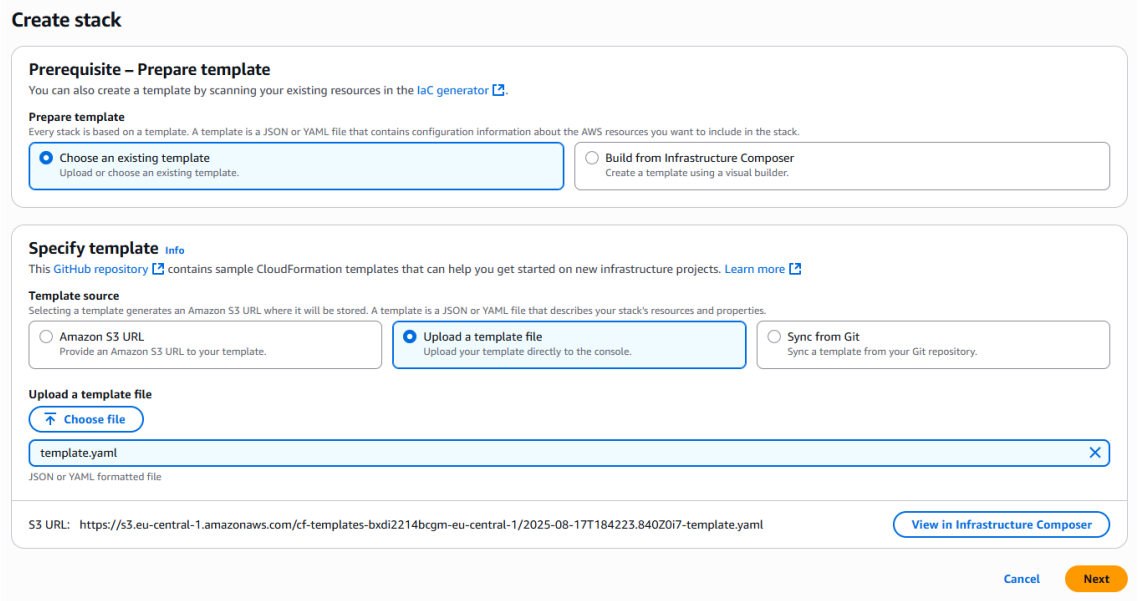


Figura 4.2. Creación de una pila en CloudFormation

El archivo yaml para un plantilla de AWS CloudFormation tiene la estructura.

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: "Descripcion del uso de plantilla"
Metadata:
  Version: "1.0"
  Author: "Nombre del autor"

Parameters:
  InstanceType:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t3.micro
      - t3.small
    Description: "Tipo de instancia EC2"

Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0abcdef1234567890
    eu-west-1:
      AMI: ami-0123456789abcdef0

Conditions:
  CreateProdResources: !Equals [ !Ref EnvType, prod ]

Resources:
  MyEC2Instance:
```

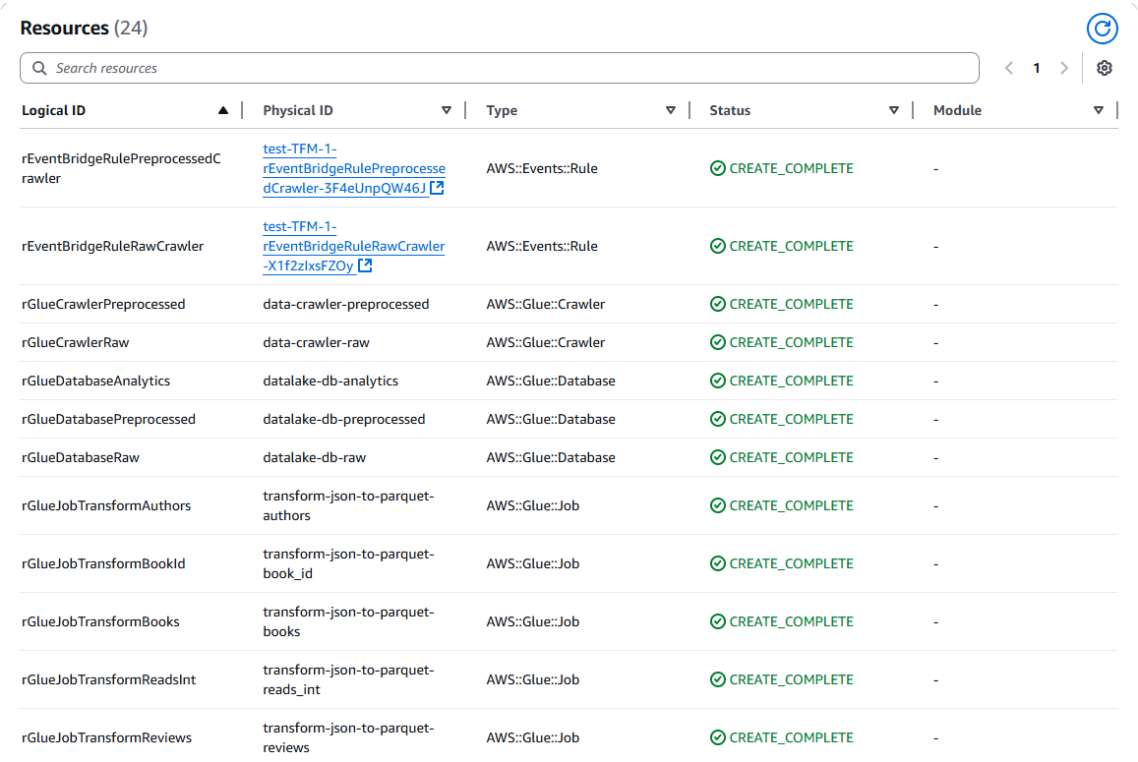
```
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref InstanceType
      ImageId: !FindInMap [ RegionMap, !Ref "AWS::Region", AMI ]
      Tags:
        - Key: Name
          Value: "MiInstanciaEC2"

Outputs:
  InstanceId:
    Description: "ID de la instancia EC2 creada"
    Value: !Ref MyEC2Instance
```

La estructura para la creación de los servicios tiene principalmente los siguientes apartados:

- AWSTemplateFormatVersion: Define la versión del formato (recomendado usar "2010-09-09").
- Description: Breve descripción de la plantilla
- Metadata: Información adicional (autor, versión, etc.)
- Parameters: Variables de entrada para que el usuario pueda personalizar la plantilla al crear el stack.
- Mappings: Tablas de mapeo (AMIs por región)
- Conditions: Condiciones para decidir si se crean ciertos recursos.
- Resources (obligatorio): Los recursos de AWS que CloudFormation debe crear (EC2, S3, Lambda, entre otros).
- Outputs: Valores de salida tras desplegar el stack (ID de instancia, nombre de bucket, URL de carga balanceada).

Toda la infraestructura como código de la arquitectura se observa en el anexo 8.0.1.

Todos los recursos se van creando uno a uno de acuerdo al archivo de configuración, hasta completarse.



Resources (24)

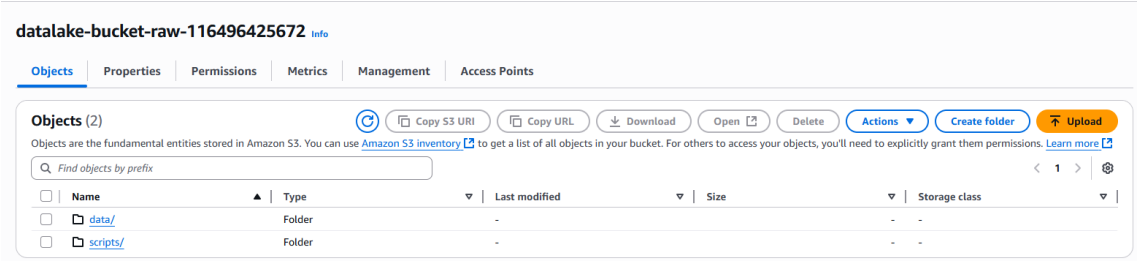
Search resources

Logical ID	Physical ID	Type	Status	Module
rEventBridgeRulePreprocessedCrawler	test-TFM-1-rEventBridgeRulePreprocessedCrawler-3F4eUnpQW46J	AWS::Events::Rule	CREATE_COMPLETE	-
rEventBridgeRuleRawCrawler	test-TFM-1-rEventBridgeRuleRawCrawler-X1f2zIxsFZOy	AWS::Events::Rule	CREATE_COMPLETE	-
rGlueCrawlerPreprocessed	data-crawler-preprocessed	AWS::Glue::Crawler	CREATE_COMPLETE	-
rGlueCrawlerRaw	data-crawler-raw	AWS::Glue::Crawler	CREATE_COMPLETE	-
rGlueDatabaseAnalytics	datalake-db-analytics	AWS::Glue::Database	CREATE_COMPLETE	-
rGlueDatabasePreprocessed	datalake-db-preprocessed	AWS::Glue::Database	CREATE_COMPLETE	-
rGlueDatabaseRaw	datalake-db-raw	AWS::Glue::Database	CREATE_COMPLETE	-
rGlueJobTransformAuthors	transform-json-to-parquet-authors	AWS::Glue::Job	CREATE_COMPLETE	-
rGlueJobTransformBookId	transform-json-to-parquet-book_id	AWS::Glue::Job	CREATE_COMPLETE	-
rGlueJobTransformBooks	transform-json-to-parquet-books	AWS::Glue::Job	CREATE_COMPLETE	-
rGlueJobTransformReadsInt	transform-json-to-parquet-reads_int	AWS::Glue::Job	CREATE_COMPLETE	-
rGlueJobTransformReviews	transform-json-to-parquet-reviews	AWS::Glue::Job	CREATE_COMPLETE	-

Figura 4.3. Finalización de stack de recursos de AWS

Esto creará parámetros, recursos tanto buckets, bases de datos, roles IAM y eventos como Lambdas y activadores. Todo basado en la arquitectura necesaria mostrada anteriormente.

En el bucket S3 donde se almacenan los datos en crudo, tiene la siguiente estructura de carpetas mostrada en la figura 4.4.



datalake-bucket-raw-116496425672 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (2)

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

Name	Type	Last modified	Size	Storage class
data/	Folder	-	-	-
scripts/	Folder	-	-	-

Figura 4.4. Estructura de carpetas S3 raw

Dentro de la ruta /data se encuentra la estructura mostrada en la figura 4.5.

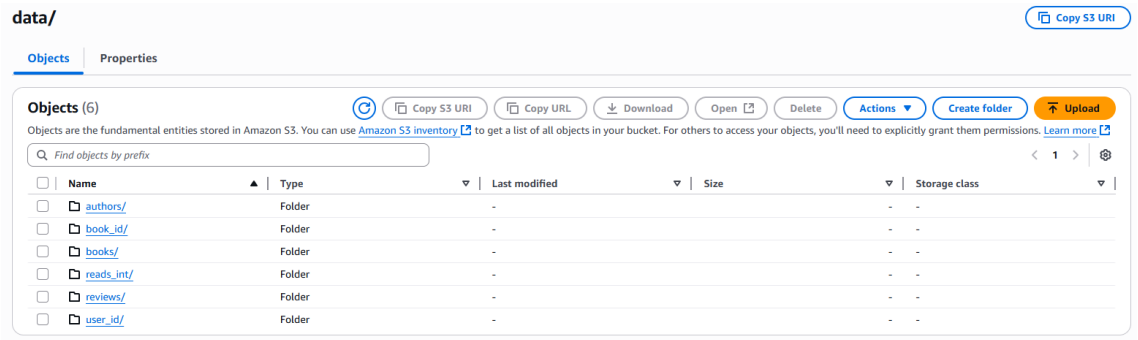


Figura 4.5. Estructura de carpetas ruta /data

Dentro de cada uno de las rutas establecidas se cargan los distintos datasets a utilizar.

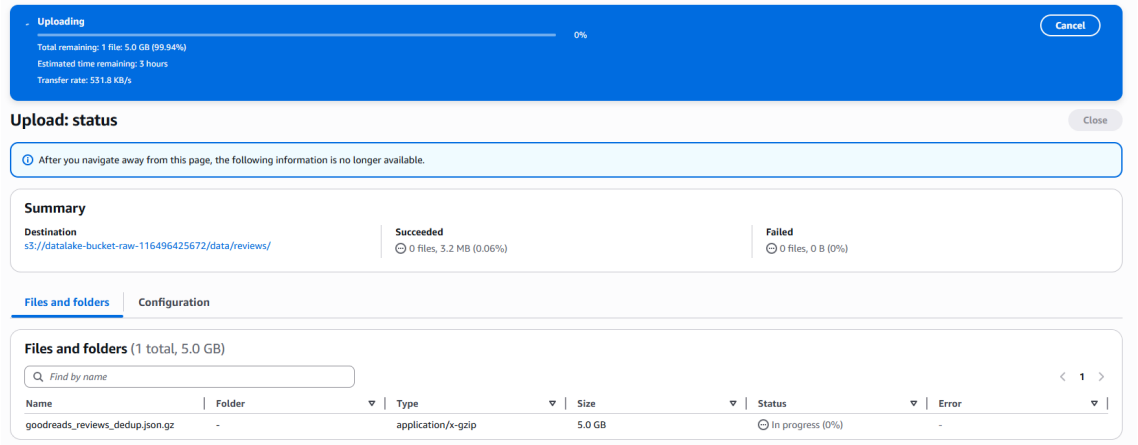


Figura 4.6. Subida de datos a S3

En la ruta /scripts se encuentran los scripts de pyspark para los distintos datasets, en este caso los scripts son para la transformación de los formatos de JSON y CSV a Parquet.

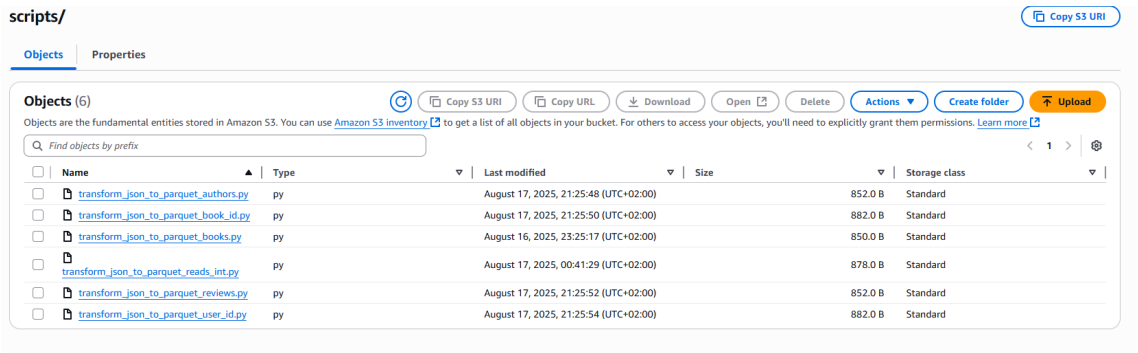


Figura 4.7. Archivos en ruta Scripts

Ahora que los datos en crudo han sido cargados, están listos para aplicar los distintos servicios de gobernaza de datos.

AWS Glue y Crawler

Dentro del servicio de AWS Glue se encuentra la herramienta de crawler, el objetivo principal de esta herramienta es el descubrimiento y catalogización automática de los datos.

Crawler trabaja de la siguiente manera:

Después de cargar los datos en el Bucket, crawler es capaz de recorrer todos los S3 (según su configuración) detectando esquemas, tipos de datos y particiones, y los registra en el Glue Data Catalog.

La configuración de los crawler creados está refleja en el archivo yaml de configuración, dado que se despliega de manera automática, estos ya estan configurados y listos para utilizar .

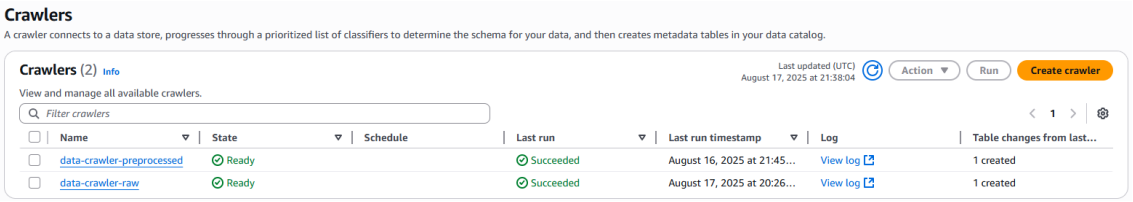


Figura 4.8. Crawlers creados para la catalogización

Cada vez que se cargue un archivo al bucket S3 raw, el crawler lo recorrerá y detectará su esquema; este se activa gracias a un desencadenador que activa un servicio de Lambda.

Figura 4.9. Lambdas de ejecución para crawlers

La lambda se ejecuta a través de un EventBridge, y su contexto es el siguiente.

```
1    import boto3
2    def handler(event, context):
3    glue = boto3.client('glue')
4    glue.start_crawler(Name='data-crawler-analytics')
5    return {"status": "started"}
```

Acto seguido se almacenará la información en la base de datos de Glue. Para esto se han creado tres bases de datos en Glue: una para los datos en crudo, otra para los datos procesados y otra para los datos listos para su uso y análisis.

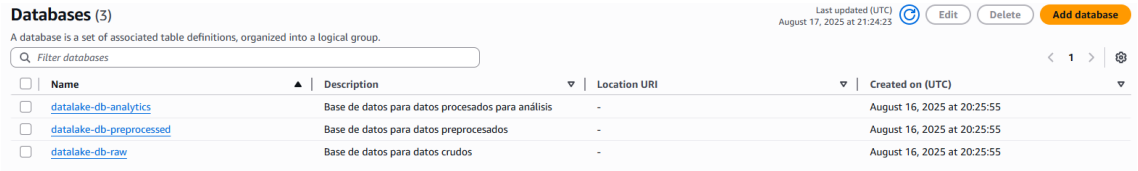


Figura 4.10. Bases de datos en AWS Glue

Una vez ejecutada la herramienta de crawler, este almacena los datos en tablas dentro de las bases de datos establecidas.

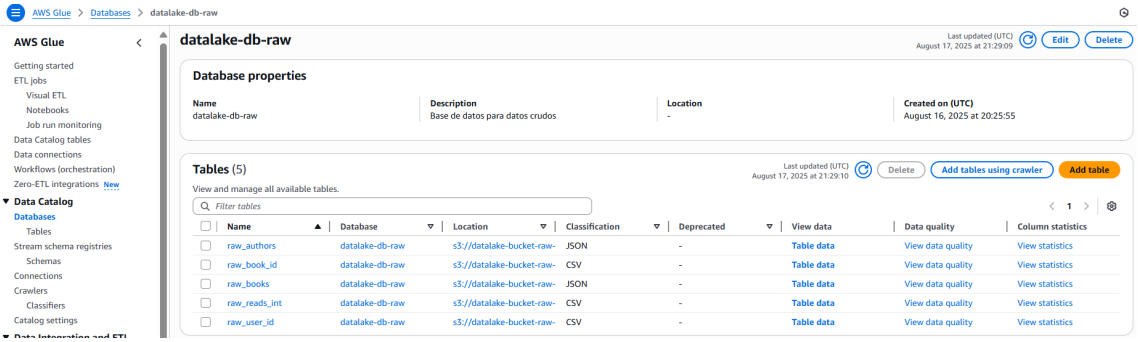


Figura 4.11. Tablas generadas por la herramienta crawler

Como resultado se obtiene el esquema total del dataset de manera automática.

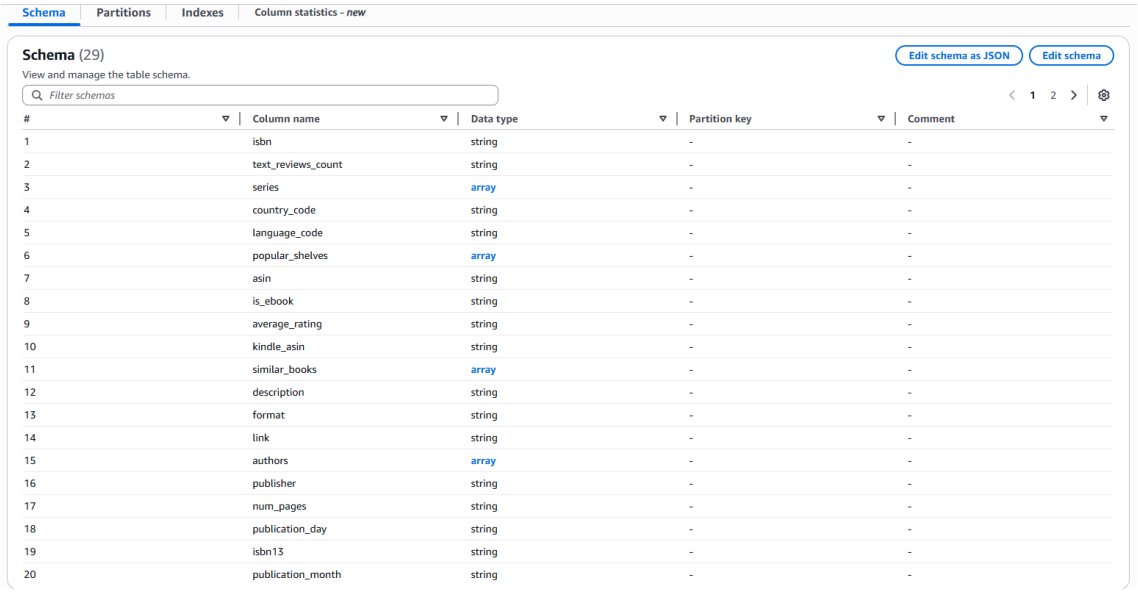


Figura 4.12. Esquema generado por la herramienta crawler

Transformaciones y ETL JOBS

El procesamiento de los datos en bruto constituye una fase importante dentro de la arquitectura de gobernanza del dato, ya que garantiza que la información almacenada en el Data

Lake sea consistente, fiable y lista para su consumo posterior.

En este proyecto, los procesos de Extracción, Transformación y Carga (ETL) se implementan mediante AWS Glue, un servicio serverless que permite ejecutar flujos de trabajo distribuidos escritos en PySpark, reduciendo la complejidad operativa y facilitando la escalabilidad.

AWS Glue tiene la herramienta ETL Jobs. Los ETL Jobs de AWS Glue constituyen el núcleo del preprocesamiento y se encargan de la transformación de los datos. Estos jobs funcionan de manera serverless, utilizando Apache Spark en segundo plano, lo que permite procesar grandes volúmenes de información de forma paralela y eficiente.

Los ETL Jobs pueden ejecutarse de tres maneras distintas:

- **Visual ETL:** Edición en una interfaz visual centrada en el flujo de datos [34].
- **Scritp:** A traves de Scripts ETL basados en Spark, selccionando el motor (shell de Python, Ray, Spark (Python) o Spark (Scala) se carga un script existente desde un archivo local. Si se elige usar el script, no se puede usar el editor visual de trabajos para diseñar o editar el trabajo [34].
- **Notebook:** Creación interactiva de trabajo basada en una interfaz de cuaderno de Jupyter Notebook [34].

Limpieza y transformación

En este trabajo se han usado los métodos de ETL Jobs mediante scripts cargados en una ruta de bucket S3 y el cuaderno de Jupyter Notebook.

Las transformaciones necesarias y la limpieza son lanzadas a través de Scripts, y la unión de dataset para el consumo del modelo de machine learning mediante el cuaderno de Jupyter Notebook.

Los Jobs son creados mediante el archivo de configuración yaml, por lo que simplemente deben ser ejecutados, sin antes haber cargado en la ruta del bucket S3 correspondiente que contiene los scripts de cada Job.

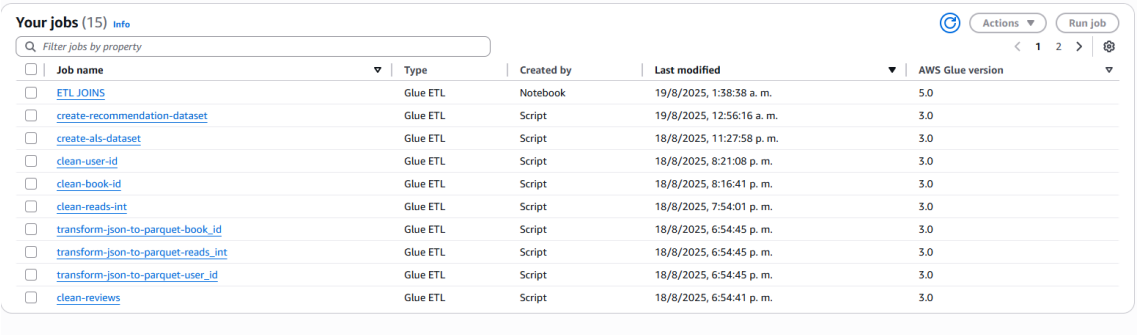


Figura 4.13. Jobs a ejecutar

Cada Job creado lee la ruta del bucket donde se encuentra el script de limpieza correspondiente para el dataset.

Si la ruta es correcta, el Job mostrará el script que se va a ejecutar.

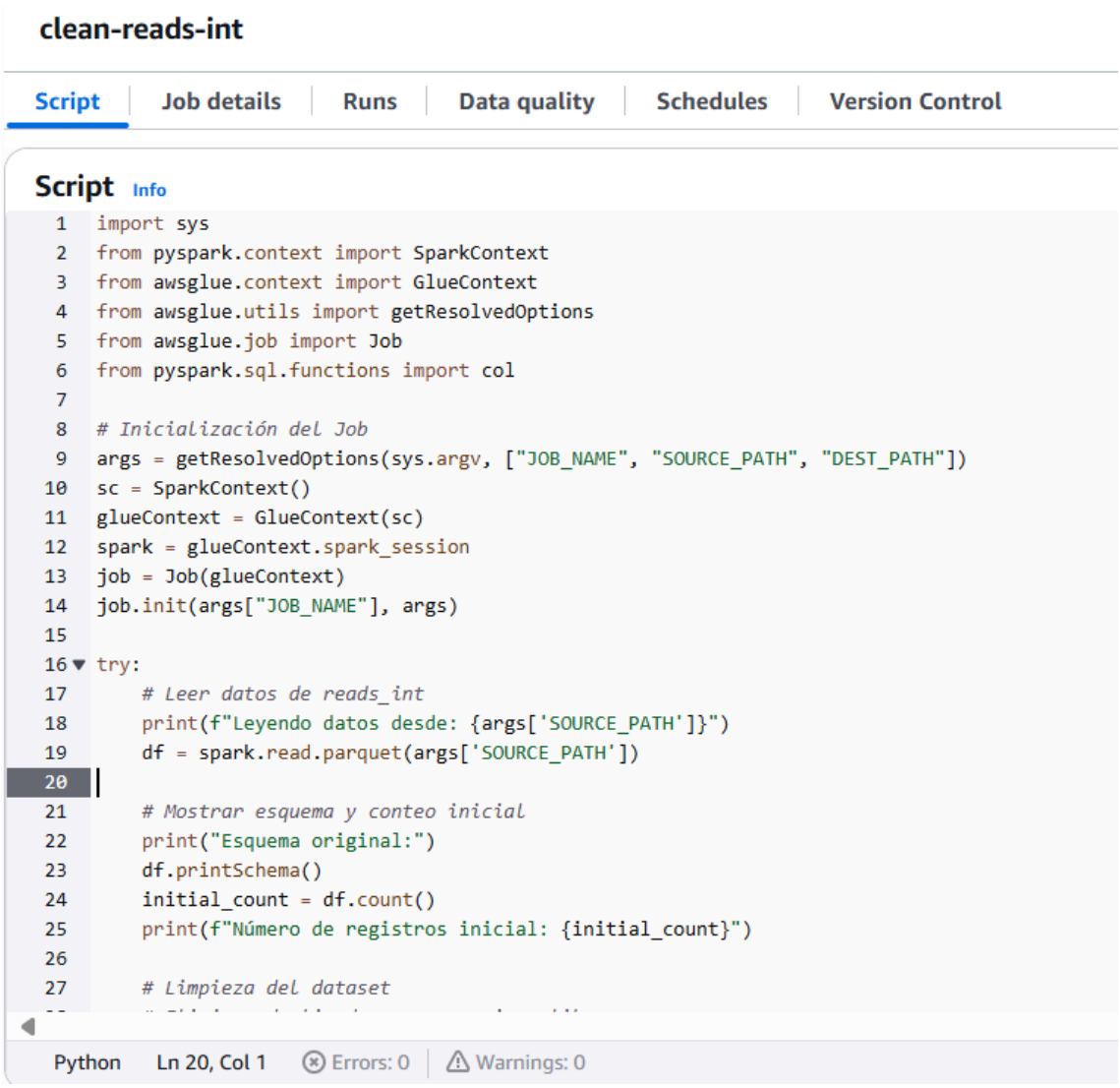


Figura 4.14. Fragmento de script de Job

Los scripts que se van a ejecutar en el Job siguen una estructura necesaria para el correcto funcionamiento.

Importación de librerias necesarias

Es necesario importar las principales clases y funciones de AWS Glue y PySpark.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
```

Definir los parámetros del job

La función *getResolvedOptions* permite acceder a los parámetros proporcionados por el job de Glue.

```
    args = getResolvedOptions(sys.argv, [
    "JOB_NAME",
    "SOURCE_PATH",
    "DEST_PATH"
    ])
```

Inicializar el contexto de Spark y Glue

Se crean los contextos necesarios para la ejecución del Job.

```
    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
```

Arrancar el Job

La clase Job maneja el estado y la configuración. Inicializa el Job con su nombre y argumentos:

```
    job = Job(glueContext)
    job.init(args["JOB_NAME"], args)
```

Definir el flujo ETL

Lee los datos, aplica las transformaciones y escribe el resultado. Ejemplo de script de dataset

```
    try:
    # Leer parquet desde la ruta especificada
    print(f"Leyendo datos desde: {args['SOURCE_PATH']}")
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Mostrar esquema y conteo inicial
    print("Esquema original:")
    df.printSchema()
    print(f"Registros iniciales: {df.count()}")
    # ...
    # resto de codigo
```

Finalizar el job

Commit al job al final del script

```
    job.commit()
```

Esta es una estructura recomendada que puede variar según la complejidad del código y del Job que se vaya a ejecutar. Esto se puede apreciar en el código aplicado para las uniones de los datasets.

A partir del momento de ejecución del script, este puede ser monitoreado, obteniendo el estado dependiendo de si el Job se ejecutó de manera exitosa o tuvo algún error.

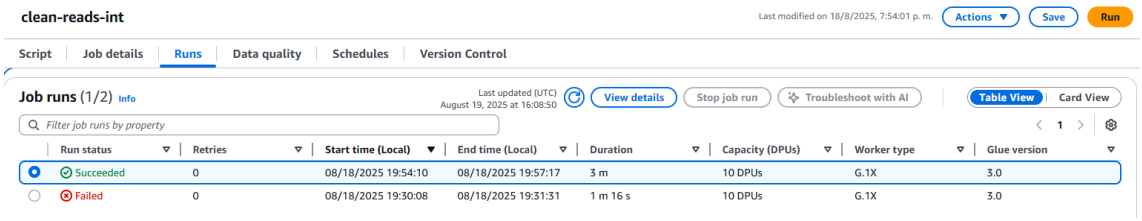


Figura 4.15. Ejecución y posibles estados de Job

De esta manera se ejecuta la limpieza y transformación de todos los datasets.

La tabla 4.9 muestra las transformaciones y limpiezas aplicadas a cada uno de los datasets.

Tabla 4.9. Transformaciones y limpieza aplicadas a los datasets

Dataset	Transformaciones / Limpieza
Books	• Elimina duplicados por `book_id`. • Convierte columnas a tipos correctos: `average_rating` a `double`, `ratings_count`, `text_reviews_count`, `num_pages` a `integer`. • Filtra libros con al menos 5 ratings. • Rellena valores nulos con promedios o ceros.
Authors	• Elimina duplicados por `author_id`. • Convierte columnas a tipos correctos: `average_rating` a `double`, `ratings_count`, `text_reviews_count` a `integer`. • Filtra autores con al menos 2 ratings. • Rellena valores nulos con promedios o ceros.

Reviews	• Convierte valores nulos en `comments_count` a 0.
	• Elimina duplicados.
	• Crea columna `engagement_level` según número de comentarios:
	  • `no_engagement` (0)
	  • `low_engagement` (1–5)
	  • `medium_engagement` (6–15)
	  • `high_engagement` (>15)

Reads_int	• Elimina duplicados por combinación de `user_id` y `book_id`. • Elimina registros con valores nulos en columnas principales. • Filtra valores fuera de rango: `rating` entre 1 y 5, `is_read` y `is_reviewed` deben ser 0 o 1.
Book_id	• Elimina duplicados por `book_id`. • Elimina registros con `book_id` nulo. • Convierte `book_id` a `integer`. • Filtra registros donde `book_id` >0.
User_id	• Elimina espacios en blanco en `user_id`. • Elimina duplicados por `user_id`. • Elimina registros con `user_id` nulo o vacío.

Los scripts aplicados en cada uno de los Jobs se pueden observar en el anexo 8.0.2 y anexo 8.0.3.

Unión y creación de dataset para consumo de modelo

La otra forma de utilizar los Jobs de AWS Glue es mediante Jupyter Notebooks, como se mencionó anteriormente.

Para la creación del dataset unificado que servirá para el consumo del modelo de machine learning, se ha usado este método, dado que al ser la ejecución mediante celdas, es po-

sible mantener un mejor control de errores, controlar las operaciones y obtener feedbacks necesarios para corrección de fallos.

El flujo de trabajo es el mismo que los scripts, y la base de su estructura sigue el mismo concepto mencionado anteriormente.

Es necesario implementar las configuraciones para su funcionamiento.

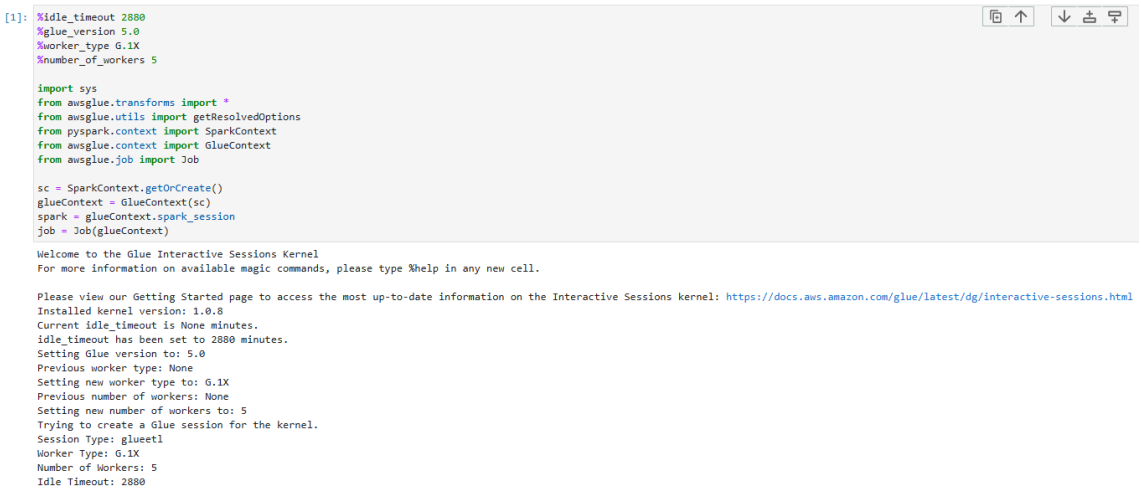


Figura 4.16. Inicio de configuración de Job Notebook

Y a continuación se aplica el resto del flujo del Job.

```python
import sys
import traceback
from pyspark.sql import functions as F
from pyspark.sql.types import IntegerType, StringType, DoubleType

# Configuracion
SOURCE_PATH = 's3://datalake-bucket-preprocessed-116496425672'
DEST_PATH = 's3://datalake-bucket-analytics-116496425672'

# funciones log
def log_error(message, error):
    """Log error details"""
    error_msg = f"""
    ERROR: {message}
    Type: {type(error).__name__}
    Message: {str(error)}
    Traceback:
    {traceback.format_exc()}
    """
    print(error_msg)

try:
    # carga de datos
    print("\nLoading datasets...")
    try:
        reads = spark.read.parquet(f"{SOURCE_PATH}/data_clean/reads_int_clean/")
        print(f"- Reads count: {reads.count()}")

        users = spark.read.parquet(f"{SOURCE_PATH}/data_clean/user_id_clean/")
        print(f"- Users count: {users.count()}")

        book_id_map = spark.read.parquet(f"{SOURCE_PATH}/data_clean/book_id_clean/")
        print(f"- Book ID map count: {book_id_map.count()}")
```

Figura 4.17. Fragmento de código de join en Notebook

Cabe recalcar que el script completo se encuentra en el anexo 8.0.4.

Una vez finalizada la transformación y limpieza, los resultados son catalogados automáticamente mediante los Glue Crawlers, que actualizan el Glue Data Catalog. Esto permite que los datasets procesados estén disponibles para su consulta inmediata desde Amazon Athena, su integración en Amazon Redshift, y su visualización en Amazon QuickSight para casos de uso analíticos y de machine learning.

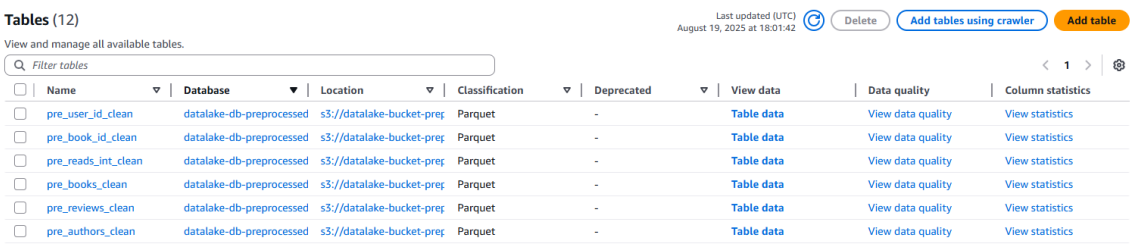


Figura 4.18. Resultado de las ejecuciones de los Jobs y Crawler

La figura 4.19 muestra el resultado de los Jobs de limpieza ejecutados, el cual se encargaba de revisar puntos como el correcto tipo de variables; se observa cómo ahora los campos corresponden a su tipo correcto.

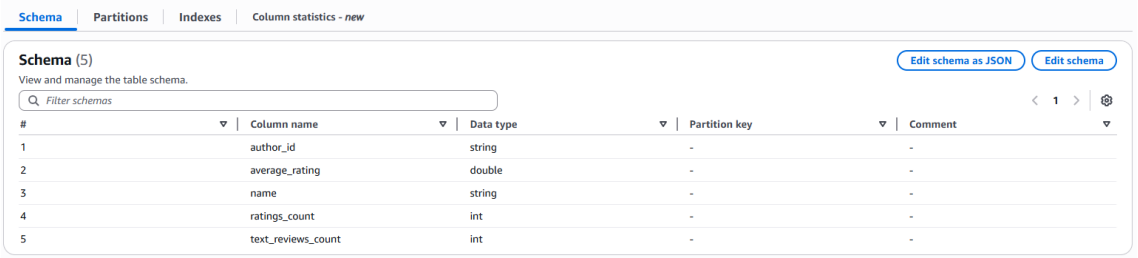


Figura 4.19. Correcion de tipos de variables realizado por script

También se obtuvo el dataset unido ejecutado por el Job en el Notebook.

Este Job estableció la unión y mapeo de los distintos datasets, con el objetivo de adquirir la información más relevante de cada uno de ellos, este dataset conformará la base para el consumo y construcción del modelo de recomendaciones.

El dataset resultante tiene el siguiente esquema mostrado en la tabla 4.10

Tabla 4.10. Descripción de los campos de la entidad *analytics_recommendation_dataset*

Campo	Tipo	Descripción
user_id	int	Identificador único del usuario
book_id	string	Identificador único del libro
rating	int	Valoración explícita del usuario al libro (1-5)
is_read	int	Indicador binario de si el usuario ha leído el libro (0 = no, 1 = sí)
is_reviewed	int	Indicador binario de si el usuario ha dejado reseña del libro (0 = no, 1 = sí)
book_title	string	Título completo del libro
title_without_series	string	Título del libro sin información de la serie
book_avg_rating	double	Promedio de calificaciones del libro en la base de datos
book_ratings_count	int	Número total de calificaciones recibidas por el libro
num_pages	int	Número de páginas del libro
language_code	string	Código del idioma del libro (ejemplo: en, es, fr)
publication_year	string	Año de publicación del libro
publisher	string	Editorial del libro
author_id	string	Identificador único del autor
author_name	string	Nombre del autor principal del libro
author_avg_rating	double	Promedio de calificaciones de las obras del autor
author_rating_count	int	Número total de calificaciones recibidas por el autor

Este dataset generado proviene del crawler ejecutado, por lo que su almacenamiento se encuentra en el bucket S3 correspondiente para el cosumo.

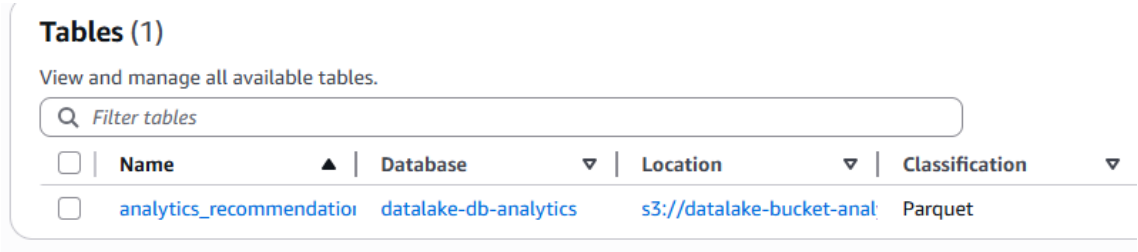


Figura 4.20. Tabla resultado del Job para consumo del modelo

AWS LakeFormation para la gobernanza del dato

El Data Lake de este proyecto se construye sobre Amazon S3, donde se almacenan los datasets en bruto (RAW), transformados (Parquet) e integrados (DataJoin). Sin embargo, la mera existencia del Data Lake no garantiza por sí misma un control adecuado sobre el acceso, seguridad y trazabilidad de los datos.

AWS LakeFormation actúa como un control de acceso unificado sobre los datos almacenados en S3. En lugar de definir permisos a nivel de bucket u objeto, los permisos se asignan en el Glue Data Catalog, permitiendo autorizar o restringir acceso a tablas, columnas o filas específicas.

La configuración de AWS LakeFormation es la siguiente.

Mediante la consola de LakeFormation asignamos un Data Lake Administrator, este corresponde al usuario con capacidad de administrar todo el entorno de Lake Formation.

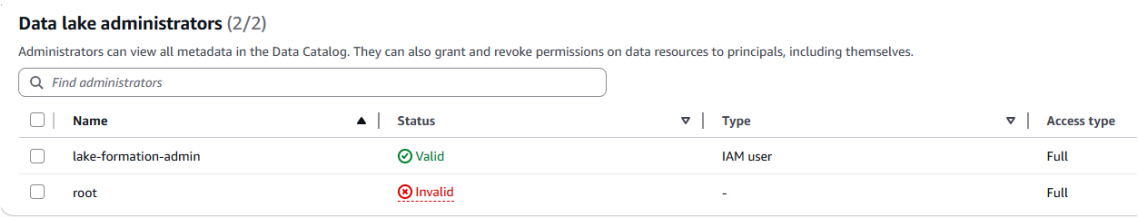


Figura 4.21. Usuarios administradores del Data Lake

Nótese, que el LakeFormation marca como usuario no válido al usuario root, para esto es necesario crear un usuario en específico que tenga acceso a los recursos que se van a administrar AWS Glue y S3.

Los roles, usuarios y políticas pueden verse en el anexo 8.0.5. A continuación, en la opción *Data Lake Locations* registramos todos los directorios que contienen los datos.

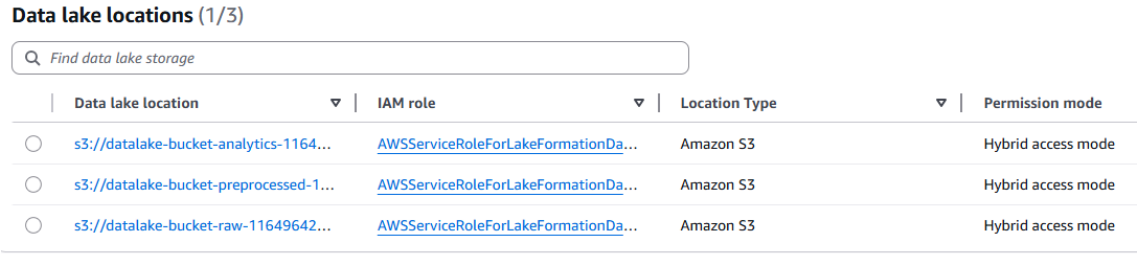


Figura 4.22. Registro de directorios en Lake Formation

Integración con AWS Glue Data Catalog

LakeFormation usa Glue Data Catalog como metadatos, una vez configurado LakeFormation detecta las bases de datos creadas, tablas y crawlers ejecutados, esto permite la administración desde un solo espacio.

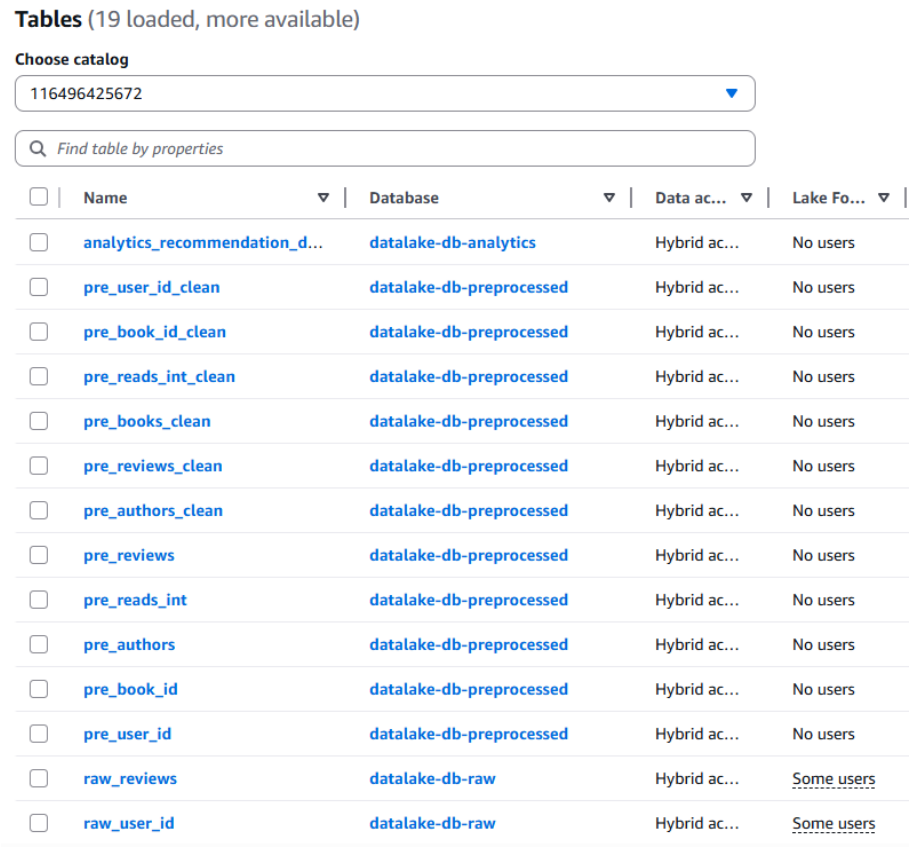


Figura 4.23. Tablas administradas desde Lake Formation

LakeFormation tomará el control del catálogo generado por si misma para administrar y gobernar permisos.

Políticas de acceso y usuarios

LakeFormation permite aplicar permisos granulares:

- Por base de datos / tabla
- Por columnas (Column-level security)
- Por filas (Row-level security)

Cada permiso granular puede ser aplicado a un usuario en específico, dependiendo del rol en el proyecto, Lake Formation aplica los permisos de acceso a los datos.

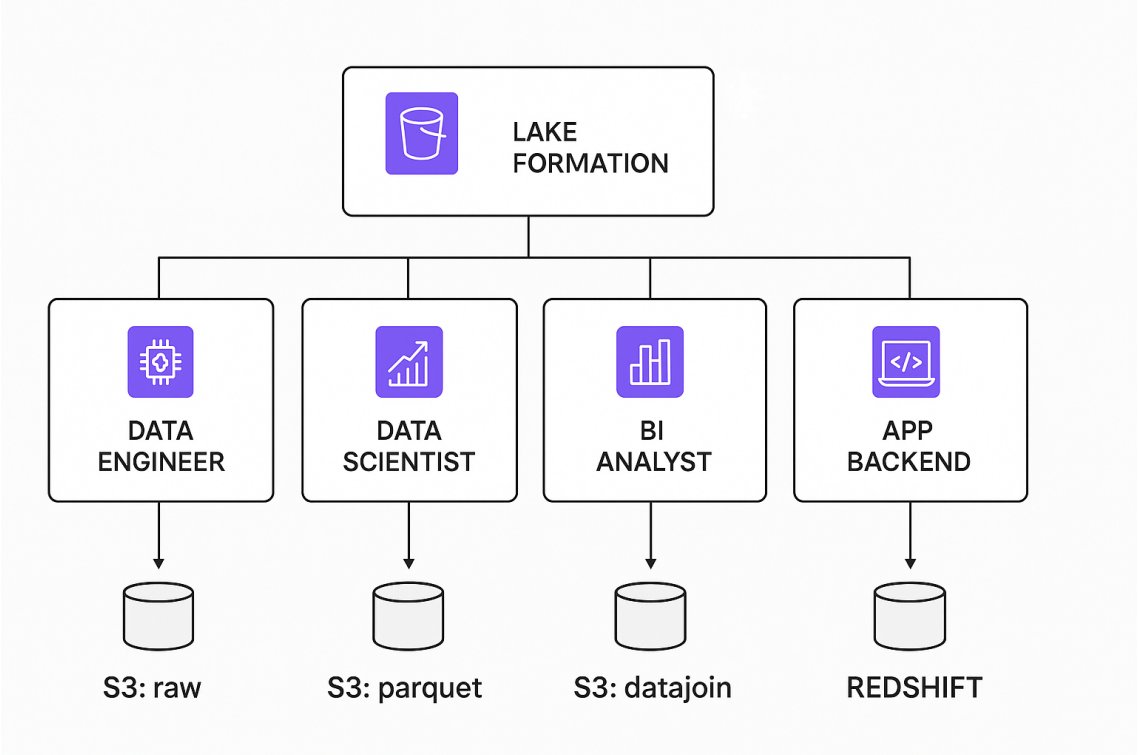


Figura 4.24. Usuarios y acceso a datos dependiendo del rol

La figura 4.24 muestra la idea de como cada usuario accede a los datos.

La tabla 4.11 muestra los usuarios que puede haber en este proyecto, roles y accesos a los datos.

Tabla 4.11. Usuarios, roles y permisos en Lake Formation para la arquitectura propuesta

Usuario	**Rol**	**Permisos / Acceso en Lake Formation**
DataEngineer	Ingesta y Transformación	- Acceso total a `S3 Raw`, `S3 Parquet Format` y `S3 DataJoin`. - Permiso para ejecutar ETL Jobs en Glue. - Crear y modificar catálogos de datos en Glue Crawler.
DataScientist	Modelado ML	- Acceso de lectura a `S3 DataJoin` y Redshift. - Permiso de consulta en Athena. - Lectura de datasets procesados para entrenamiento de modelos.
Analyst	Business Analytics	- Acceso de lectura a Redshift. - Acceso de sólo lectura a vistas preprocesadas de S3 Parquet Format en Athena. - Integración con QuickSight para dashboards.
Admin	Gobernanza del dato	- Permisos completos en Lake Formation. - Definición y asignación de políticas de seguridad y acceso. - Monitoreo de uso y cumplimiento normativo.

Estos roles se crean en ‘User‘ de IAM.

Figura 4.25. Creaciom de usuarios para administración en Lake Formation

Asignación de permisos en LakeFormation

En el apartado de *tablas* en LakeFormation podemos asignar los permisos a los distintos usuarios y el acceso a las tablas que necesita según su rol.

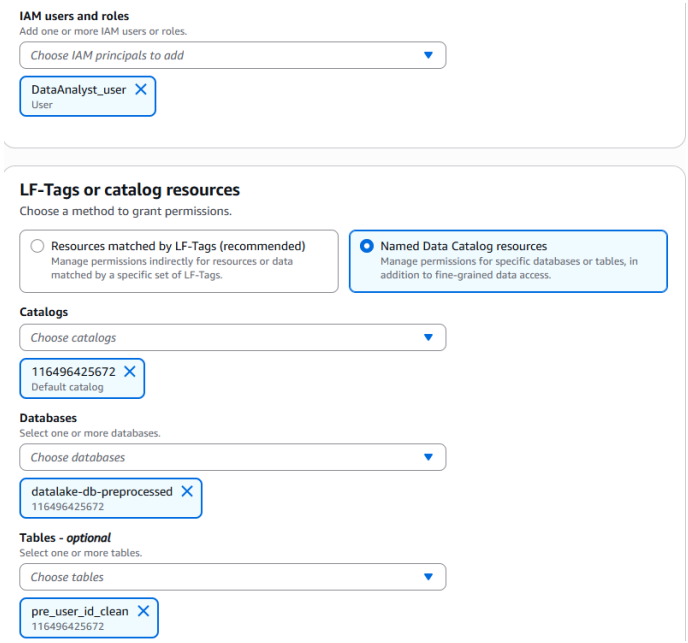


Figura 4.26. Asiganción permiso a tabla pre_user_id_clean a usuario data analyst

Los permisos pueden ser gestionados de acuerdo al nivel de tabla o columna. Un nivel tabla permite el acceso a todos los datos, mientras que un nivel de columna permite incluir o excluir columnas, esto es muy útil al momento de garantizar la seguridad en datos sensibles.

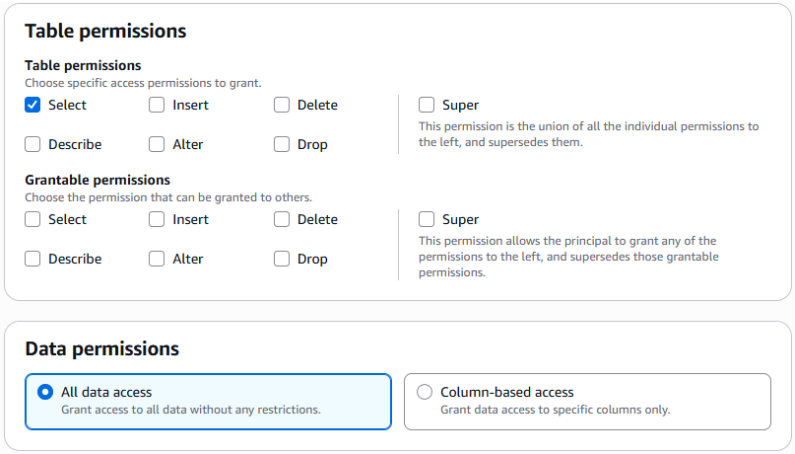


Figura 4.27. Selección de permisos a usuario

Una vez asignado los permisos a los usuarios, comprobamos que esto se este gestionando de manera correcta. Para ello ingresamos como usuario a uno de los creados, en este caso se ingresa al usuario del Data Scient.

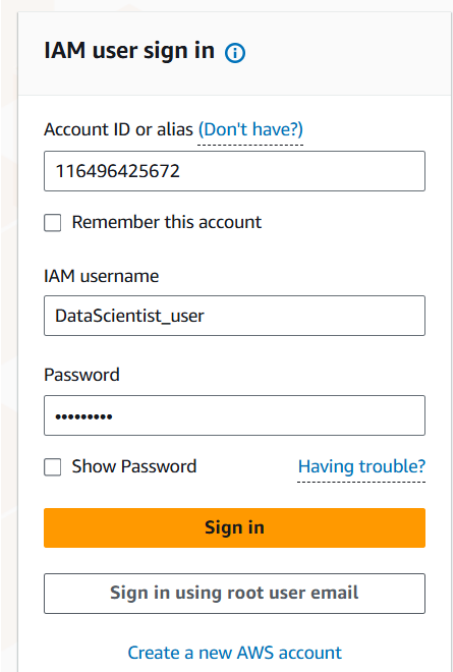


Figura 4.28. Ingeso al proyecto como usuario Data Scient

Este usuario tiene permisos de AWS Athena por lo que se le permite explorar los datos a través del mismo.

En AWS Athena se observa que el usuario puede leer los datos del dataset.

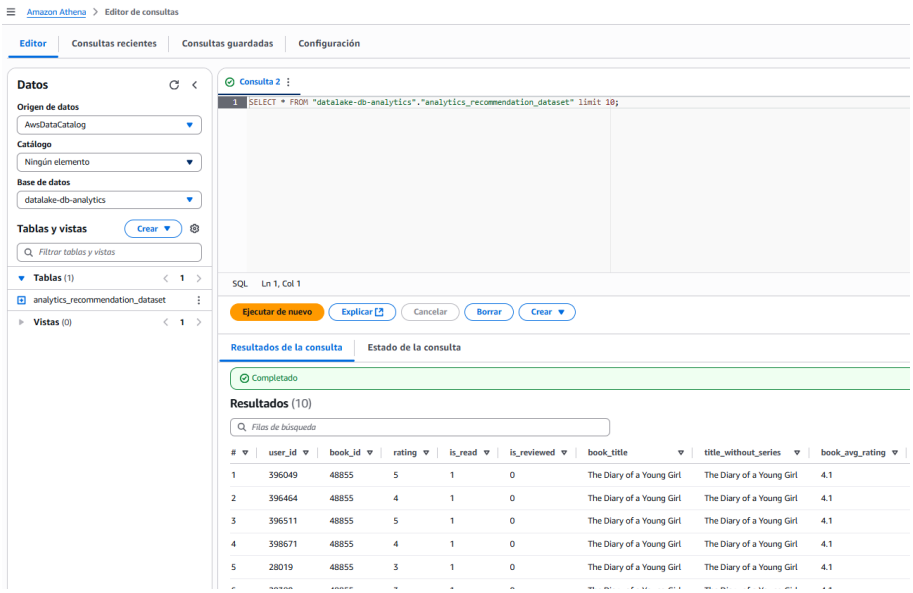


Figura 4.29. Lectura de datos por parte de usuario Data Scient

Si trata de consultar otra tabla dentro del catálogo y este no se encuentra dentro de los permisos asignados, no podrá leerlos.

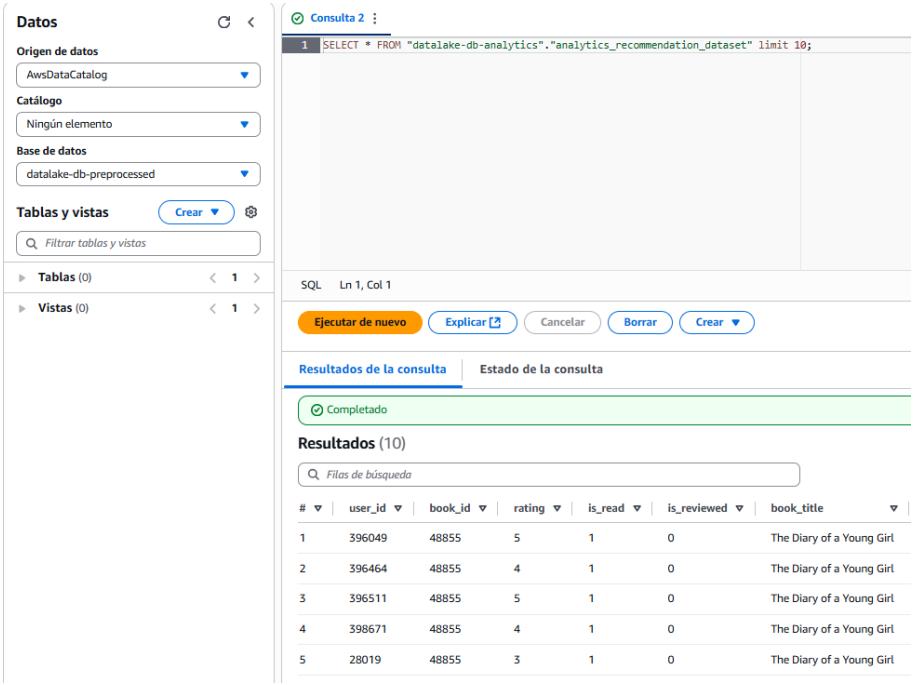


Figura 4.30. No acceso a tablas fuera del permiso asignado por LakeFormation

La figura 4.31 muestra los usuarios asignados y los accesos necesarios a cada tabla del DataLake.

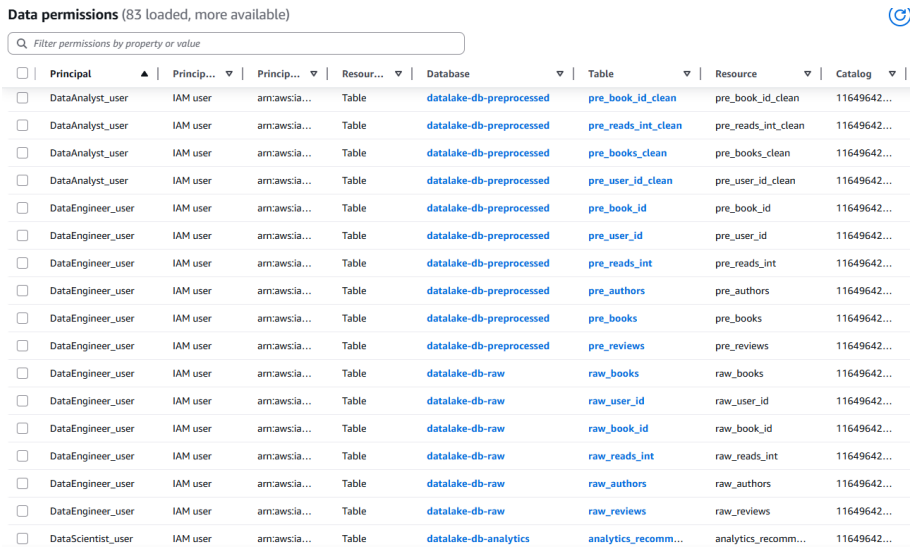


Figura 4.31. Asiganción de permisos a usuarios en LakeFormation

De esta forma se asegura que cada perfil de usuario (analistas, científicos de datos, administradores, entre otros) tenga acceso únicamente a los datos estrictamente necesarios para sus funciones, cumpliendo con el principio de mínimo privilegio. Además, al integrar permisos con servicios como Athena, Redshift y Glue, se garantiza que los datos se consulten y transformen bajo un marco común de control y auditoría, facilitando la trazabilidad y el cumplimiento

normativo.

Implementación a AWS Redshift

En el marco de la arquitectura propuesta, Amazon Redshift constituye el pilar central del almacén de datos. Este servicio de AWS está diseñado específicamente para el procesamiento analítico de grandes volúmenes de información, ofreciendo un rendimiento óptimo en la ejecución de consultas complejas y en la integración con herramientas de inteligencia de negocio.

El papel de Redshift dentro de la solución radica en recibir, consolidar y almacenar los datos ya transformados y gobernados, previamente procesados mediante AWS Glue y organizados en Amazon S3. A partir de este punto, los datos quedan disponibles para su explotación a través de consultas SQL de alto rendimiento, la creación de modelos predictivos con Redshift ML, y su consumo en herramientas de visualización como Amazon QuickSight.

Amazon Redshift incluye dos modalidades de uso;

- Redshift Aprovisionado (Provisioned): Administrar manualmente un clúster, eligiendo el tipo y la cantidad de nodos [35].
- Redshift Serverless: No se debe administrar ningún clúster. AWS aprovisiona y escala automáticamente los recursos según la demanda [35].

La configuración de Redshift es la siguiente:

En la consola de Amazon Redshift iniciamos un nuevo servicio serveless.

Establecemos un nombre al espacio.

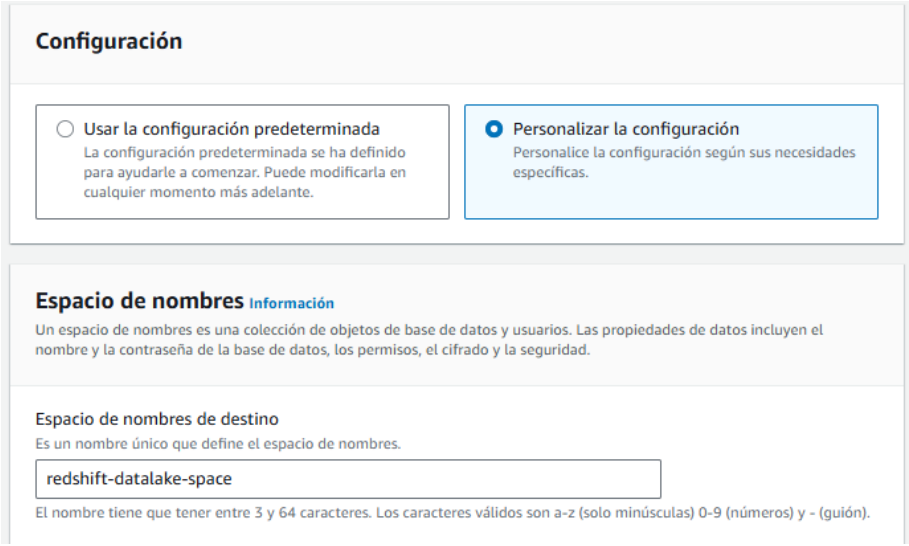


Figura 4.32. Nombre de espacio de trabajo de Amazon Redshift

Asignamos un permiso y un rol asociado.

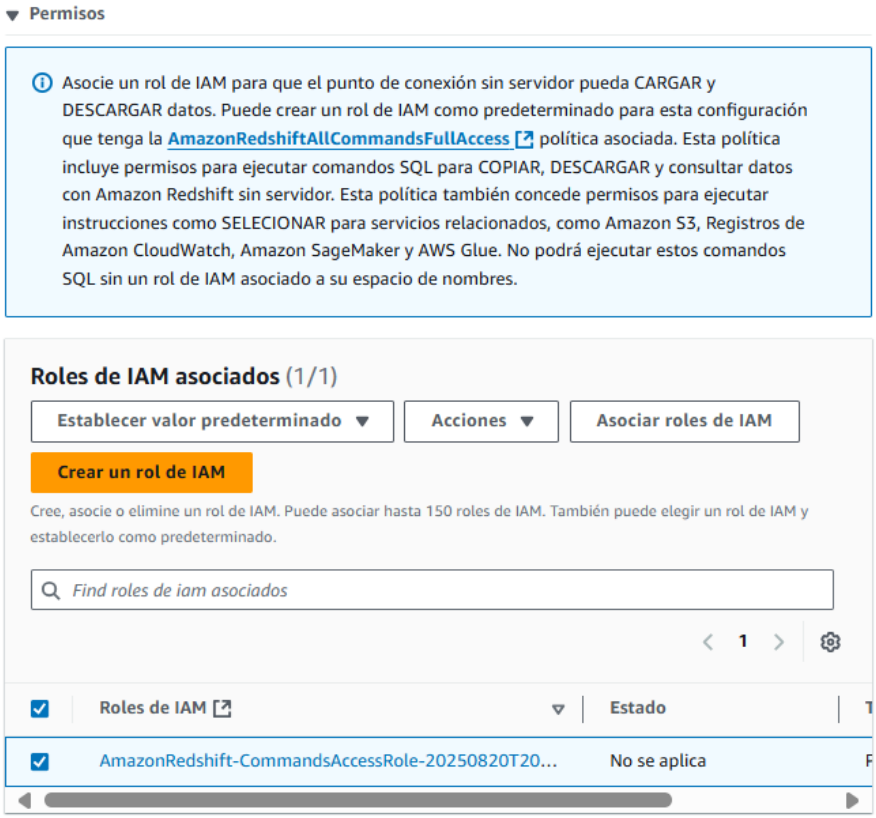


Figura 4.33. Permiso y rol asociados a Redshift

Configuramos la capacidad de procesamiento.

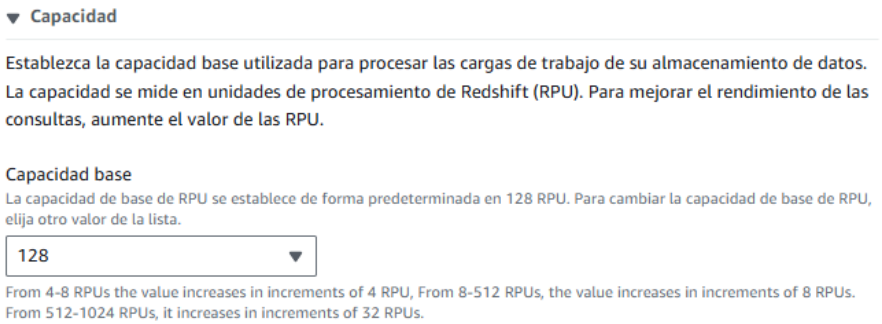


Figura 4.34. Capacidad de procesamiento de Redshift

La red y conexiones VPN, Redshift por defecto crea las conexiones VPN generadas en dos subnets privadas.

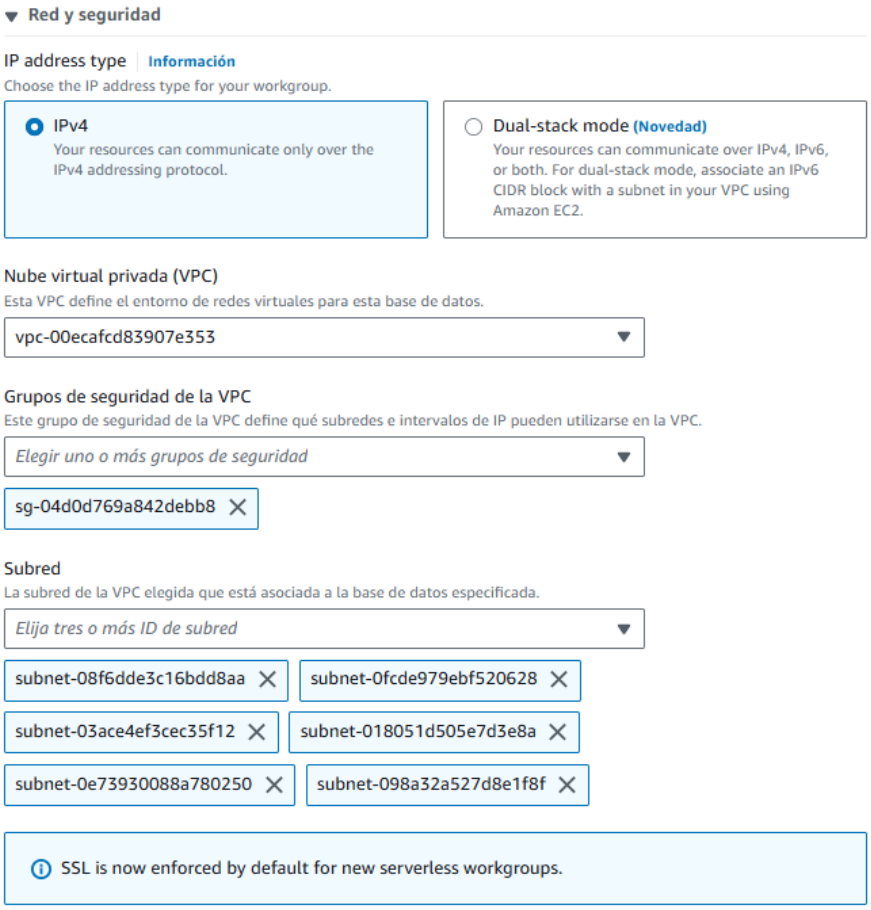


Figura 4.35. Conexiones y subnets VPN

Una vez configurado, iniciará la creación.

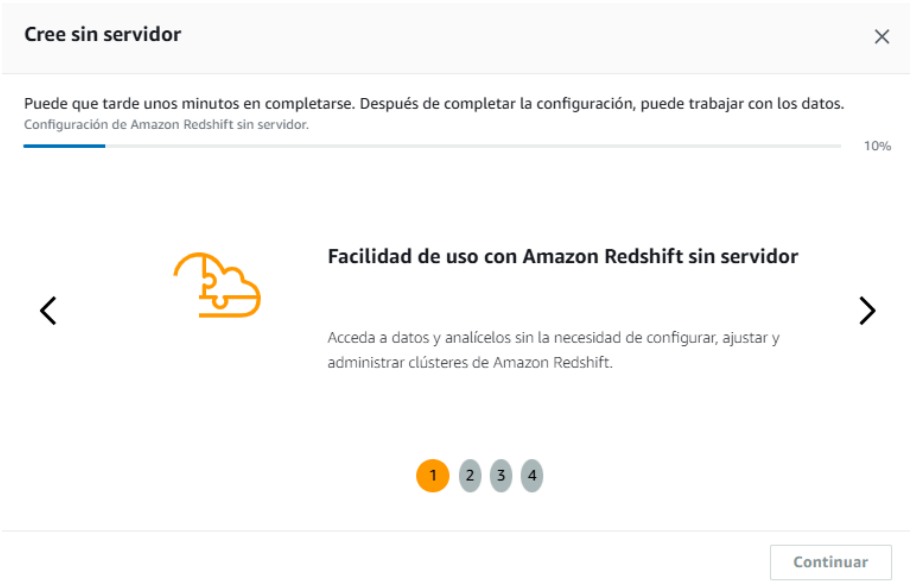


Figura 4.36. Inicio de creación de Redshift

Con esto, Amazon Redshift está listo para ejecutar consultas.

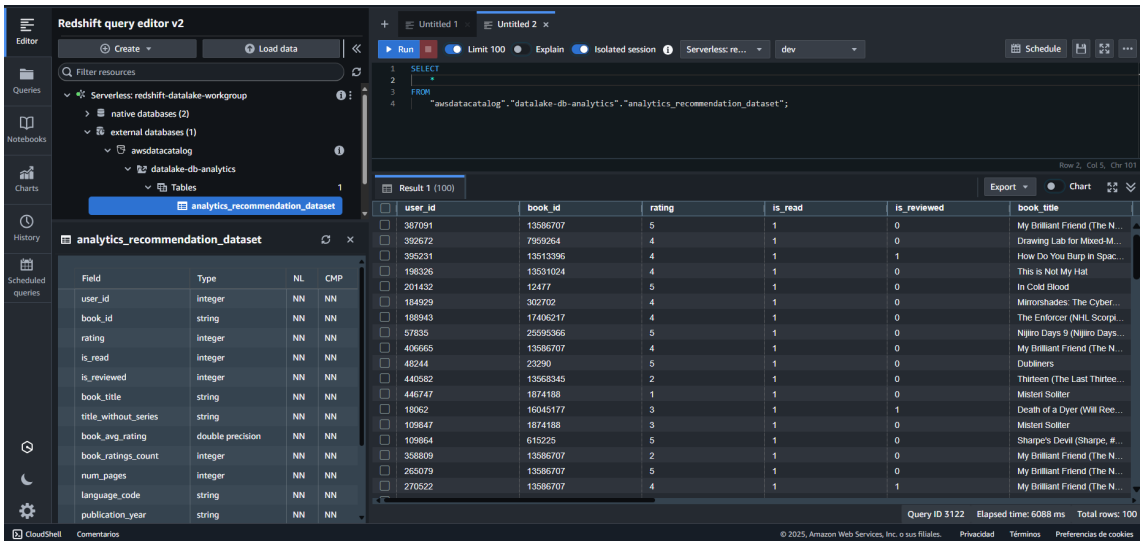


Figura 4.37. Ejecución de consultas mediate Amazon Redshift

De esta forma, Redshift no solo actúa como el destino final del ciclo de integración y calidad de los datos, sino también como el núcleo analítico que habilita tanto la generación de insights estratégicos para el negocio.

Análisis de negocio a través de QuickSight

Amazon QuickSight corresponde a un servicio de BI(Business Intelligence) serveless, diseñado para conectarse directamente a tus fuentes de datos (S3, Athena, Redshift, Glue Data Catalog) y generar dashboards interactivos y reportes analíticos en tiempo real [36].

QuickSight mantiene una integración nativa con Redshift, Athena y Glue Catalog. También cuenta con el control de seguridad de Amazon LakeFormation, solo muestra las tablas que los usuarios tienen permiso de consultar.

Perfecto para métricas como popularidad de libros, tendencias de reseñas, patrones de interacción, etc. Integra Machine Learning Insights para la detección automática de anomalías o predicciones básicas en los dashboards sin necesidad de entrenar modelos adicionales.

Para empezar a usar QuickSight es necesario establecer una conexión a una base de datos, en este caso será a través de Redshift, pero también es posible una conexión mediante bases de datos como: PostgresSQL, SQL Server, Amazon RDS, MySQL, Amazon Aurora, Buckets S3, entre otros.

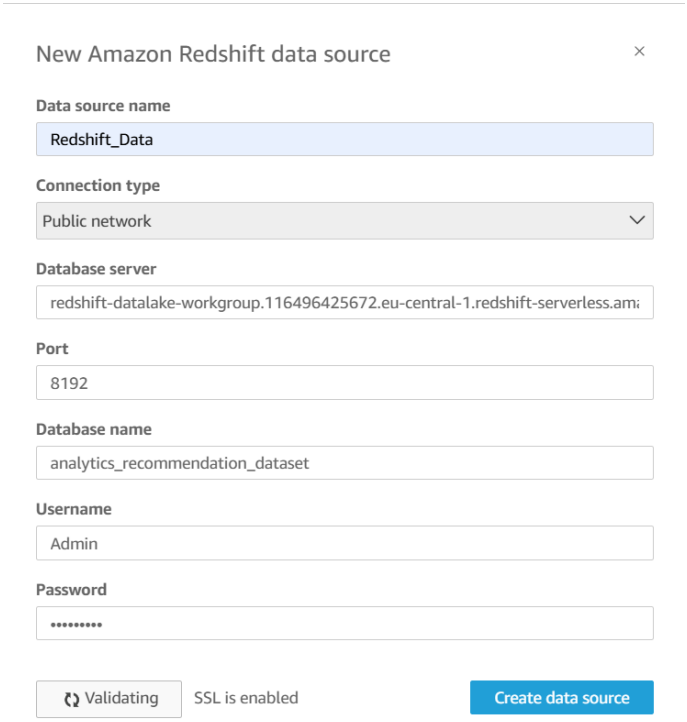


Figura 4.38. Conexión de de AWS Redshift a Amazon QuickSight

Una vez establecida la conexión al lugar donde QuickInsight consumirá los datos, podemos empezar a realizar dashboards concentrados en el análisis de negocio.

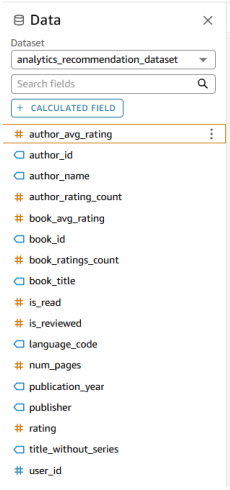


Figura 4.39. Esquema mostrado en QuickSight

El siguiente conjunto de paneles de Business Intelligence (BI) ha sido desarrollado en Amazon QuickSight como parte del sistema de gobernanza del dato y analítica avanzada aplicada al dominio de reseñas y recomendaciones de libros.

El objetivo principal de estas visualizaciones es proporcionar una visión integral y comprensible del ecosistema de lectura, combinando información proveniente de múltiples fuentes:

usuarios, libros, autores, reseñas y metadatos editoriales.

A través del proceso de ingestión, transformación y limpieza de datos en AWS Glue, se consolidó un dataset analítico optimizado para consultas exploratorias y generación de conocimiento.

Estos tipos de gráficas permiten responder preguntas de negocio, dado que están diseñadas para un propósito en específico.

A continuación, se presentan los dashboards creados a partir del dataset *recomendation_dataset*.

Por usuario

Este dashboard busca responder la cantidad de autores y libros que generan mayor interés en los lectores.

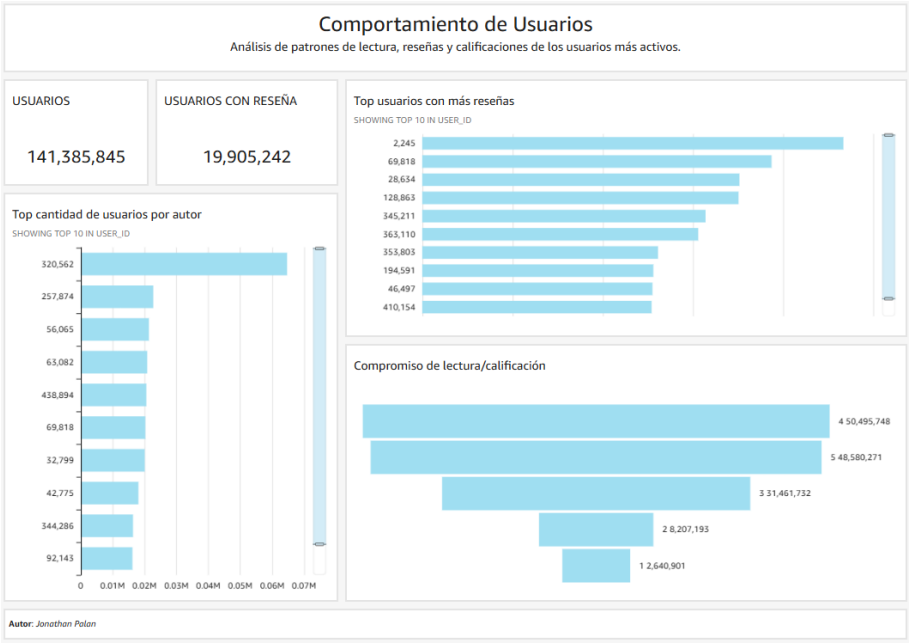


Figura 4.40. Dashboard de usuarios generado en QuickSight

Por libros

El siguiente dashboard busca la influencia de las calificaciones y reseñas en la percepción de calidad de los libros.

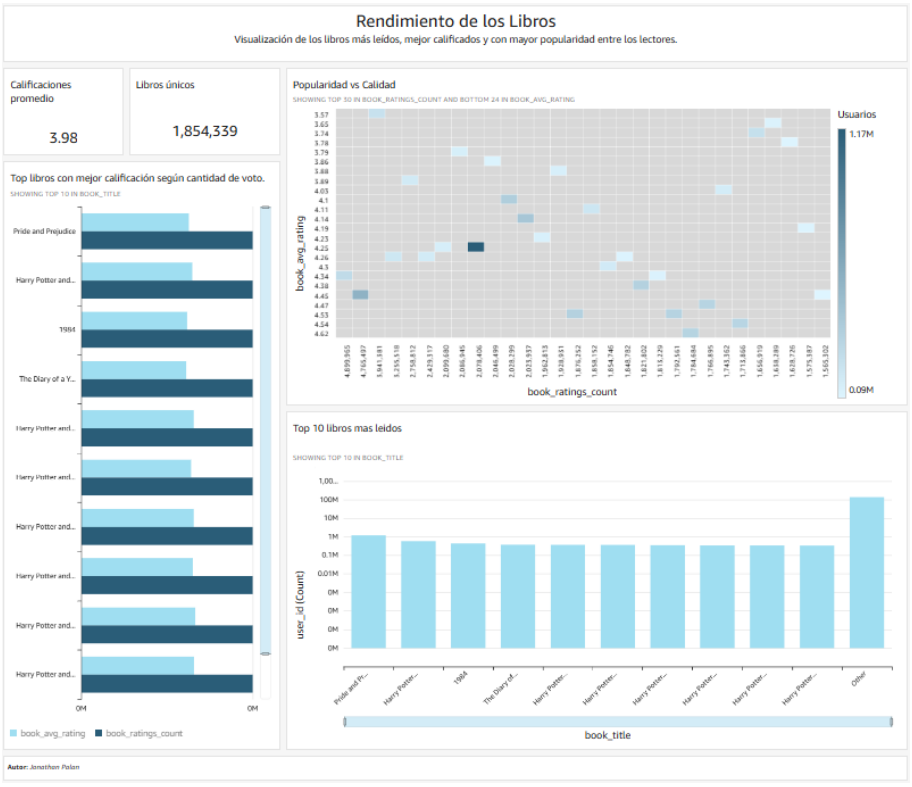


Figura 4.41. Dashboard de libros generado en QuickSight

Por autores y editoriales

Este dashboard busca las tendencias que se observan en las publicaciones de las editoriales más populares.

Figura 4.42. Dashboard de autores generado en QuickSight

El rol de analistas de datos es el principal usuario del dashboard en QuickSight. Su función no es entrenar modelos de Machine Learning, sino interpretar y comunicar hallazgos a partir de los datos ya transformados y curados.

Gracias a la arquitectura, los analistas acceden a datos limpios, integrados y gobernados, lo que evita que tengan que gastar tiempo en:

- Buscar datasets dispersos.
- Resolver inconsistencias (IDs duplicados, nulos, formatos distintos).
- Construir queries manuales para obtener métricas básicas.

La construcción de dashboards ayuda a la exploración rápida de insights clave identificando en segundos qué libros son los más populares, qué autores tienen mayor impacto o qué editoriales concentran más títulos.

Impacto de QuickSight en el ámbito de negocio

QuickSight permite que usuarios de negocio sin perfil técnico (marketing, ventas, dirección, producto) accedan a visualizaciones e insights sin necesidad de saber SQL ni programar.

Esto ayuda a la toma de decisiones responder preguntas como: ¿Qué géneros o autores deberían ser recomendados con mayor frecuencia?, ¿Qué segmentos de usuarios son más activos y merecen campañas específicas?

Las gráficas interactivas ayudan a que las decisiones se fundamenten en datos reales y actualizados, no solo en intuición.

También permiten que se identifiquen sesgos de oportunidades y riesgos, detectar tendencias y alertas al mercado.

El uso de herramientas de visualización como QuickSight nativa de AWS o externas como PowerBI o Tableu, ayuda a conectar la capa de negocios con la técnica, lo que permite una mejor culturización en la organización (data-driven) [37].

4.2.4. Desarrollo del modelo de recomendaciones

Amazon SageMaker se integra como la capa de Machine Learning (ML) encargada de entrenar, validar, desplegar y monitorear modelos predictivos basados en los datos procesados en el Data Lake y el Data Warehouse.

SageMaker es capaz de consumir datos de distintos sitios:

- Desde S3: para entrenar modelos con los datos originales o transformados.
- Desde Glue Catalog/Athena: permitiendo consultas SQL sobre datos en S3 antes de enviarlos a SageMaker.
- Desde Redshift: ideal para modelos que necesiten datos ya integrados y optimizados para analítica.

Para empezar, creamos un nuevo espacio de trabajo en SageMaker, configurando los parámetros necesarios para el funcionamiento y creación del modelo.

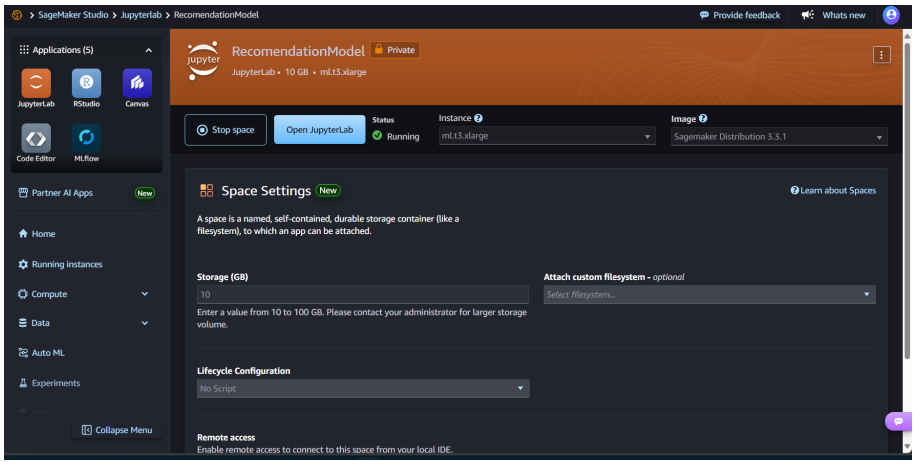


Figura 4.43. Espacio de trabajo de SageMaker

Modelo de recomendaciones híbrido: filtrado colabortivo (ALS) y filtrado basado en contenido

El modelo va a ser entrenado usando un notebook de Sagemaker; para la aplicación, ocuparemos los datos limpios y generados anteriormente.

Filtrado colaborativo

Este tipo de modelos se basa en la interacción y relación de los usuarios. Las técnicas de Filtrado Colaborativo obtienen predicciones automáticas (filtrado) sobre los intereses de cada usuario a partir de información de múltiples usuarios (colaborativo).

En particular, los algoritmos basados en la factorización implícita de la matriz de interacciones usuarios-ítems permiten desarrollar sistemas de recomendación basados en conjuntos de datos que, en el caso más básico, se componen de usuarios, ítems y valoraciones (ratings) realizadas por los usuarios sobre múltiples ítems [38].

Filtrado basado en contenido

Los sistemas de recomendación basados en contenido incorporan algoritmos de aprendizaje automático y técnicas de ciencia de datos para recomendar nuevos elementos, el motor de recomendaciones compara esencialmente un perfil de usuario y un perfil de artículo para predecir la interacción usuario-artículo [39].

La similitud entre número de elementos se calcula a través de distintas métricas que pueden ser: similitud del coseno, distancia euclidiana o producto punto.

Ambos filtrados se basan en el uso de la similitud del coseno, este genera valores normalizados entre -1 y 1, donde, cuanto mayor sea el puntaje del coseno, más similares se considerarán dos elementos [39].

$$\text{Cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{4.1}$$

Etapas del modelo

A continuación se muestra las partes más importantes que conforman el modelo de recomendaciones híbrido. El código completo utilizado se encuentra en el anexo 8.0.7.

Librerías necesarias

Es necesario cargar las librerías necesarias para AWS, PySpark, ML y análisis de datos para preparar un entorno de machine learning.

```
# Librerias
import boto3
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql import SparkSession
from pyspark.sql.functions import (
    col, concat_ws, udf, lower, regexp_replace,
    explode, collect_list, when, count
)
from pyspark.ml.feature import Tokenizer, StopWordsRemover,
   HashingTF, IDF
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.sql.types import ArrayType, StringType, DoubleType
from scipy.spatial.distance import cosine
from sklearn.model_selection import train_test_split
```

Las librerías corresponden:

- **AWS:**
  - boto3: Para interacción con servicios de AWS.
  - time: Control del tiempo.
- **Ciencia de datos:**
  - numpy: Control de vectores y matrices.
  - pandas: Manejo de datos tabulares.
- **Apache Spark:**
  - pyspark.sql.SparkSession: Entrada para trabajar con Spark.
  - pyspark.sql.functions: Funciones para manipulación de datos.
- **NLP y feature engineering en Spark:**
  - Tokenizer : Divide texto en palabras (tokens).
  - StopWordsRemover: Elimina palabras irrelevantes.
  - HashingTF:Transforma tokens en vectores numéricos usando hashing.
  - IDF: Ajusta los vectores con TF-IDF.
- **Algoritmo de recomendación**
  - ALS: Algoritmo de factorización de matrices
  - RegressionEvaluator: Para evalución del modelo.

TF-IDF Features

El siguiente paso corresponde a la creación del vector para el filtrado basado en contenido.

```
def create_tfidf_features(df):
    """Create TF-IDF features"""
    tokenizer = Tokenizer(inputCol="text_features_clean", outputCol
        ="words")
    wordsData = tokenizer.transform(df)

    remover = StopWordsRemover(inputCol="words", outputCol="
        filtered")
    filtered = remover.transform(wordsData)

    hashingTF = HashingTF(inputCol="filtered", outputCol="
        raw_features", numFeatures=10000)
    featurizedData = hashingTF.transform(filtered)

    idf = IDF(inputCol="raw_features", outputCol="features")
    idfModel = idf.fit(featurizedData)
    return idfModel.transform(featurizedData)
```

Esta función hace que cada libro quede representado por un vector de características semánticas.

Realiza una limpieza de tokenización dividiendo texto de las columnas en palabras individuales(Tokens). Luego elimina palabras vacías (stopwords) que no aportan valor semántico. A continuación, convierte la lista de palabras en un vector de frecuencia de términos (TF) usando hashing trick para mapear cada palabra a un índice en un vector. Y aplica TF-IDF, que ajusta los valores de TF teniendo en cuenta la frecuencia de las palabras en todo el corpus.

```
def load_sample_dataset(spark, input_path, sample_size=10000):
    """Load a sample of the dataset"""
    full_df = spark.read.parquet(input_path)
    return full_df.sample(False, sample_size/full_df.count(), seed
        =42)

def preprocess_data(df):
    """Preprocess text data"""
    # Extract shelf names
    df_shelves = df.select(
        "book_id",
        explode("popular_shelves").alias("shelf")
    ).select(
        "book_id",
        col("shelf.name").alias("shelf_name")
    ).groupBy("book_id").agg(
        concat_ws(" ", collect_list("shelf_name")).alias("
            shelf_features")
```

```
    )

    # Join and create text features
    df = df.join(df_shelves, "book_id").withColumn(
        "text_features_raw",
        concat_ws(" ",
            col("title"),
            col("title_without_series"),
            col("description"),
            col("author_name"),
            col("publisher"),
            col("shelf_features")
        )
    )

    return df.withColumn(
        "text_features_clean",
        lower(regexp_replace(col("text_features_raw"), "[^a-zA-Z\\s]", ""))
    )
```

La siguiente sección corresponde a dos funciones que se encargan de la carga y preprocesamiento de los datos, respectivamente.

La primera función lee el dataset completo desde un archivo en formato Parquet, este corresponde al archivo almacenado en el bucket de analítica. Luego, toma una muestra aleatoria y sa seed=42 para que siempre devuelva la misma muestra.

La siguiente función consta de varias partes:

- Extraer popular shelves almacenados en *popular_shelves*
- Crea una columna *text_features_raw* concatenando varios atributos, lo que crea una *bolsa* de palabras para representar un libro.
- Aplica limpieza eliminando caracteres no alfabéticos.

```
Sample processed data:
+-------+----------------+--------------+--------------------+
|book_id|           title|   author_name|            features|
+-------+----------------+--------------+--------------------+
|    412|Gravity's Rainbow|Thomas Pynchon|(10000,[20,31,327...|
|    412|Gravity's Rainbow|Thomas Pynchon|(10000,[20,31,327...|
+-------+----------------+--------------+--------------------+
```

Figura 4.44. Resultado del preprocesamiento y carga de datos

Creación del modelo híbrido

La primera función asigna el peso que tendrá cada método de recomendación; así, content-based (30 %) y collaborative filtering (70 %).

Luego, crea diccionarios para mapear los book_id a números (necesarios para Spark ALS)

```
def __init__(self, content_weight=0.3):
    self.content_weight = content_weight
    self.collaborative_weight = 1 - content_weight
    self.book_id_mapping = {}
    self.reverse_book_mapping = {}
```

La siguiente función:

```
def train(self, df, df_tfidf):
....
    self.book_id_mapping = { str(row['book_id']): float(idx) ... }
    self.reverse_book_mapping = { float(idx): str(book_id) ... }
....
    ratings_df = df.filter(...).select("user_id", "book_id", "
       rating")
....
    ratings_mapped = ratings_df.withColumn("book_id_mapped",
       safe_map_udf(col("book_id")))
....
     self.als = ALS(
            maxIter=10,
            regParam=0.01,
            userCol="user_id",
            itemCol="book_id",
            ratingCol="rating",
            coldStartStrategy="drop",
            nonnegative=True
        )
        self.model_als = self.als.fit(ratings_mapped)
....
    self.features_map = { str(row['book_id']): row['features'].
       toArray() ... }
```

Cumple varios objetivos:

- Crea un mapping de *book_id* necesario para ALS ya que no acepta strings en columnas como IDs de ítems.
- Prepara el dataset de ratings filtrando solo datos válidos (*user_id*, *book_id*, rating no nulos) y los convierte a tipos correctos.
- Mapea *book_id* a numérico con un UDF seguro para que ALS pueda procesarlos.
- Entrena un modelo colaborativo(ALS)
- Crea el mapa de features TF-IDF usando TF-IDF donde ya están los embedings de libros en formato vector.

Luego se crean tres funciones, una para recomendaciones basadas en contenido, otra basadas en usuario y otra basadas en ambas.

```
def recommend_by_book(self, book_id, n=5):
....
```

```
def recommend_by_user(self, user_id, n=5, spark_session=None):
....
def recommend_hybrid(self, user_id, book_id, n=5, spark_session=
    None):
....
final_score = 0.7 * cf_score + 0.3 * cb_score
```

Para la prueba del modelo se escoje un `user_id` y un `book_id` de ejemplo.

```
    test_sample = df.select("user_id", "book_id", "title", "
        author_name").first()
    print(f"Testing recommendations for book: {test_sample.title}
        by    {test_sample.author_name}")
```

La primera función llama a *recommend_by_book* con el *book_id* del libro de prueba. Encuentra libros similares al libro escogido usando similitud coseno sobre TF-IDF y devuelve los top libros parecidos.

```
Testing recommendations for book: Gravity's Rainbow by Thomas Pynchon

1. Content-based recommendations:
+-------+--------------------+--------------------+---------------+
|book_id|               title|         author_name|book_avg_rating|
+-------+--------------------+--------------------+---------------+
|  19482|The New York Tril...|         Paul Auster|           3.92|
|  19482|The New York Tril...|         Paul Auster|           3.92|
|    416|Slow Learner: Ear...|      Thomas Pynchon|           3.48|
|  23943|         Naked Lunch|        Barry  Miles|           3.46|
|  23943|         Naked Lunch|        Barry  Miles|           3.46|
|  23943|         Naked Lunch|        Barry  Miles|           3.46|
|    410|                  V.|      Thomas Pynchon|           3.97|
|    410|                  V.|      Thomas Pynchon|           3.97|
|   5815|           Vineland|      Thomas Pynchon|           3.66|
|   5815|           Vineland|      Thomas Pynchon|           3.66|
|   5815|           Vineland|Manuel Saenz de H...|           3.66|
+-------+--------------------+--------------------+---------------+
```

Figura 4.45. Resultado libros basados en el contenido

La segunda función llama a *recommend_by_user* con el *user_id* del ejemplo. Usa el modelo ALS entrenado para predecir qué libros le gustarían a ese usuario en función de su historial y el de otros usuarios. Devuelve los top 5 libros recomendados.

```
2. Collaborative filtering recommendations:
+-------+--------------------+---------------+---------------+
|book_id|               title|    author_name|book_avg_rating|
+-------+--------------------+---------------+---------------+
|   6925|Welcome to the Mo...|   Tony Roberts|           4.14|
|   6925|Welcome to the Mo...|    Maria Tucci|           4.14|
|   6925|Welcome to the Mo...|     Bill Irwin|           4.14|
|  62075|My Daddy Is a Pre...| Baron Baptiste|           4.19|
|  12937|See You Around, S...|     Lois Lowry|           3.66|
|   1032|Trump: The Art of...|Donald J. Trump|           3.66|
|   1032|Trump: The Art of...|Donald J. Trump|           3.66|
|  48248|Early Novels and ...|   Willa Cather|           4.22|
|  48248|Early Novels and ...| Sharon O'Brien|           4.22|
+-------+--------------------+---------------+---------------+
```

Figura 4.46. Resultado del modelo libros basados en el usuario

Llama a *recommend_hybrid*, que mezcla ALS (colaborativo) y TF-IDF (contenido). Combina ambos scores con los pesos definidos en el modelo. Devuelve los top 5 libros con mayor score combinado.

```
3. Hybrid recommendations:
+-------+--------------------+---------------+---------------+
|book_id|               title|    author_name|book_avg_rating|
+-------+--------------------+---------------+---------------+
|   6925|Welcome to the Mo...|   Tony Roberts|           4.14|
|   6925|Welcome to the Mo...|    Maria Tucci|           4.14|
|   6925|Welcome to the Mo...|     Bill Irwin|           4.14|
|  62075|My Daddy Is a Pre...| Baron Baptiste|           4.19|
|  12937|See You Around, S...|     Lois Lowry|           3.66|
|   1032|Trump: The Art of...|Donald J. Trump|           3.66|
|   1032|Trump: The Art of...|Donald J. Trump|           3.66|
|  48248|Early Novels and ...|   Willa Cather|           4.22|
|  48248|Early Novels and ...| Sharon O'Brien|           4.22|
+-------+--------------------+---------------+---------------+
```

Figura 4.47. Resultado del modelo, libros basados en usuario y contenido

Finalmente, el modelo y las features son guardadas en un bucket de s3 para su posterior consumo.

Métricas y evaluación del modelo

La evaluación del modelo es importante ya que esta nos indica que también está funcionando el modelo.

Las métricas utilizadas en modelos de recomendaciones suelen ser:

- **MAE(Mean Absolute Error):** Métrica que mide el error promedio que comete un modelo al hacer predicciones. Se calcula tomando la diferencia absoluta entre el valor real

y el valor predicho y luego promediando esas diferencias [40].

- **RMSE(Root Mean Squared Error):** Esta métrica mide también la diferencia entre valores reales y predichos, pero antes de promediarlos eleva los errores al cuadrado, lo que hace que los errores grandes tengan un impacto mayor [40].

La siguiente función se encarga de dividir los datos y medir las métricas correspondientes al modelo.

```
def evaluate_recommendations_optimized(recommender, test_data,
    spark_session):
....
# Split data for evaluation
print("Splitting data for evaluation...")

# Split data into train and test sets
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)

print(f"Training data: {train_data.count():,} records")
print(f"Test data: {test_data.count():,} records")
```

La tabla 4.12 refleja los resultados del modelo.

Tabla 4.12. Resultados de la evaluación del modelo de recomendación.

Métrica	Valor	Descripción
MAE	0.155	En promedio, las predicciones del modelo difieren en apenas 0.155 puntos respecto al rating real, lo que refleja un error bajo.
RMSE	0.255	Penaliza más los errores grandes; el valor obtenido indica que la mayoría de predicciones se mantienen cercanas al valor real, con pocos errores significativos.
Cobertura (Coverage)	99.69 %	El modelo logró generar recomendaciones para prácticamente todos los usuarios del conjunto de prueba, mostrando una alta capacidad de cobertura.

Finalmente el modelo se almacena en un directorio de S3 junto todos sus artefactos para el posterior consumo de la aplicación.

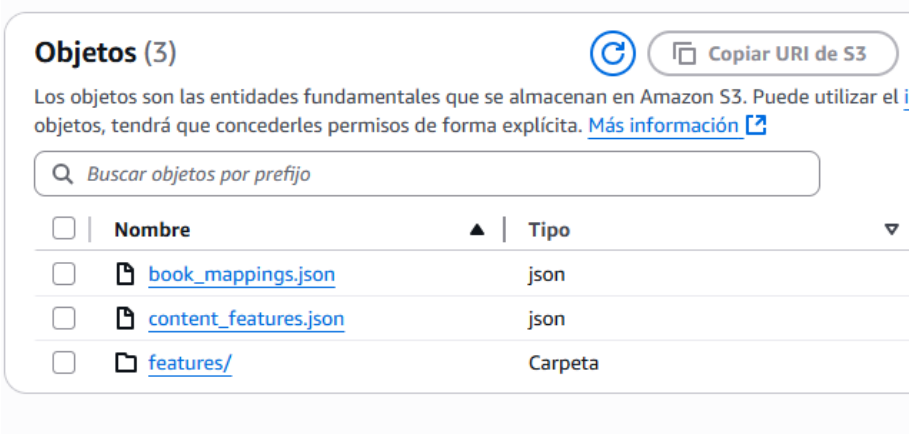


Figura 4.48. Artefactos del modelo

Análisis de sesgos SageMaker Clarify

En el desarrollo de sistemas de recomendación y modelos de aprendizaje automático, el análisis de sesgos ayuda a garantizar la equidad, transparencia y la confiabilidad de los resultados. Un modelo sesgado no solo reduce la calidad de las recomendaciones, sino que también puede amplificar desigualdades existentes, reforzar patrones no deseados en los datos o excluir a ciertos grupos de usuarios.

En el proyecto se han usado distintas funciones para analizar el sesgo que puede afectar al rendimiento de las recomendaciones,

La siguientes gráficas presentan distintos análisis realizados.

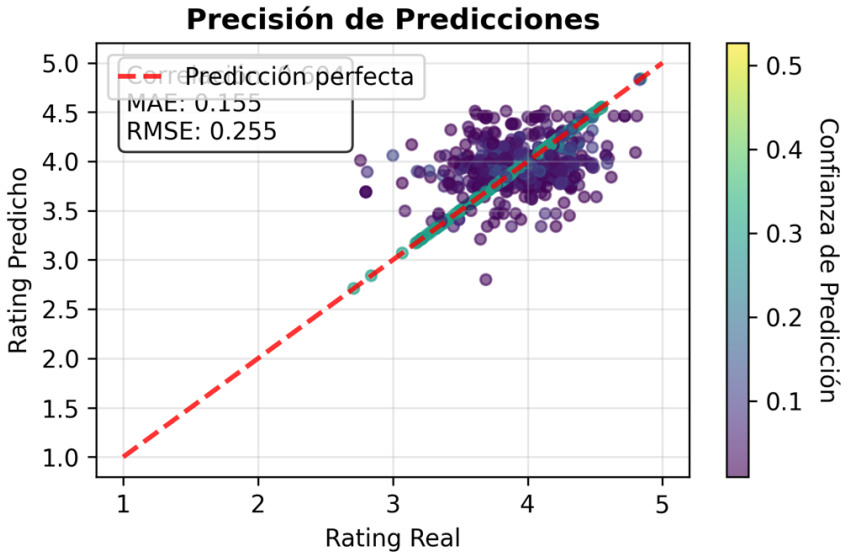


Figura 4.49. Precisión de las predicciones del modelo

En la gráfica 4.49 se puede observar cómo el modelo predice ratings con buena precisión y

baja dispersión respecto al valor real. Los errores absolutos y cuadráticos son bajos, indicando que las predicciones son confiables.

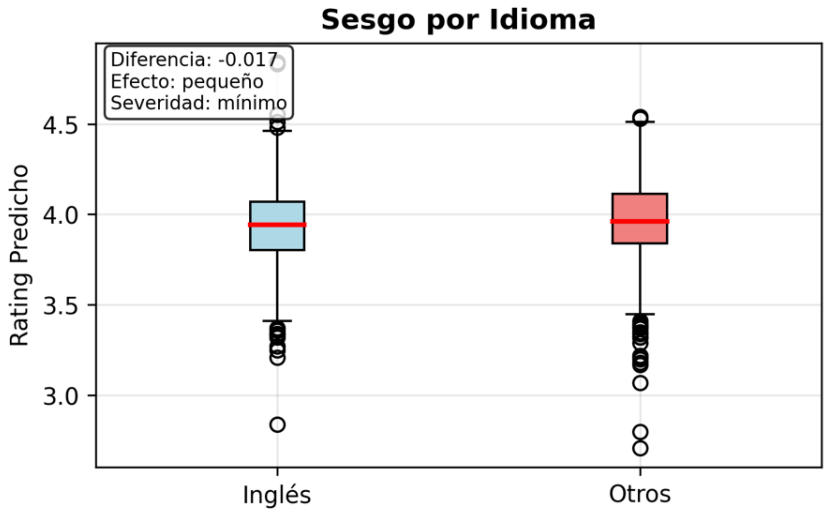


Figura 4.50. Análsis de sesgo por idioma

La gráfica 4.50 de boxplot indica que el modelo de predicción es prácticamente imparcial respecto al idioma, con una diferencia mínima en el rating predicho entre textos en inglés y otros idiomas.

El sesgo podría interpretarse como inexistente o tan pequeño que se considera insignificante y no representaría un problema relevante.

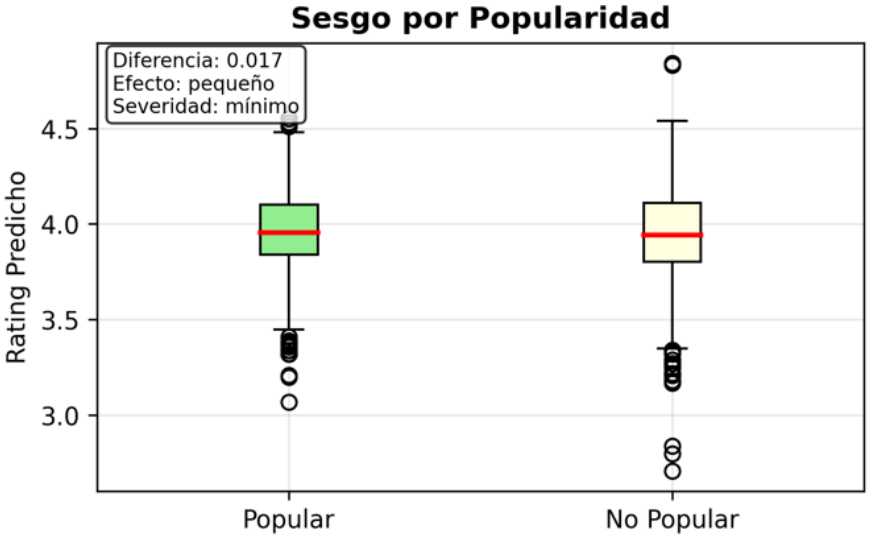


Figura 4.51. Análisis de sesgo por popularidad

La gráfica 4.51 muestra que el modelo mantiene una buena imparcialidad respecto a la popularidad de los elementos, evitando favorecer o perjudicar a los menos o más populares en la predicción de ratings.

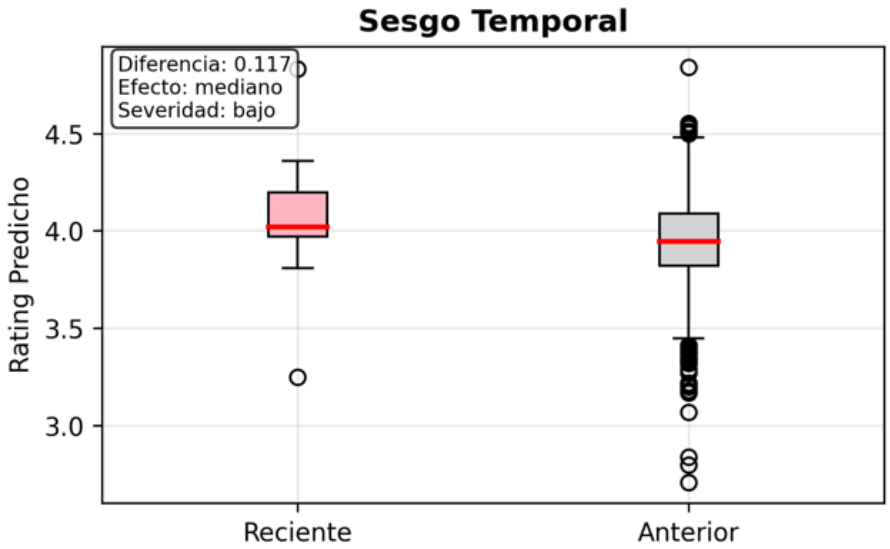


Figura 4.52. Análisis de sesgo por antiguedad

En la gráfica 4.52 existe un sesgo temporal. El modelo predice ratings mejor puntuados para el grupo “Reciente” frente al “Anterior”, con una diferencia moderada y severidad baja, pero no despreciable. Hay un sesgo temporal a favor de ejemplares más recientes.

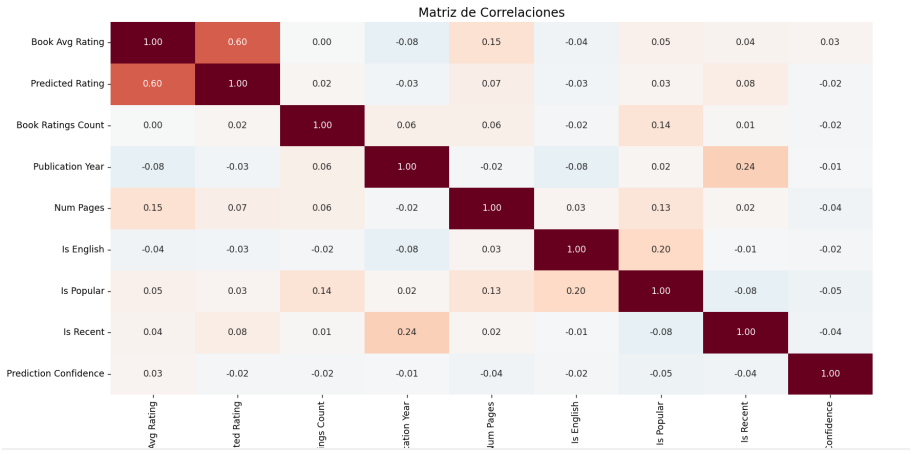


Figura 4.53. Matriz de correlación

La gráfica 4.53 indica que las variables de mayor correlación son las relacionadas directamente con el rating. Las variables como popularidad, idioma y confianza no influyen significativamente en los ratings ni entre sí.

SageMaker Clarify

Amazon SageMaker ofrece la herramienta de detección de sesgos conocido como Clarify. Este genera un reporte con toda la información obtenida de los resultados de un modelo de machine learning.

```
    from sagemaker import clarify
    clarify_processor = clarify.SageMakerClarifyProcessor(
    role=role, instance_count=1, instance_type="ml.m5.xlarge",
        sagemaker_session=session
)
```

Amazon Clarify contribuye a la gobernanza de datos, debido a la transparencia que permite que los modelos sean explicables, ayudando a la auditoría de toma de decisiones automatizadas.

La detección de sesgos permite que la equidad entre grupos sea lo menos discriminatoria. Muchos sectores requieren demostrar que las decisiones de IA son justas y explicables.

Figura 4.54. Monitoreo de sesgos

Clarify genera un reporte que contribuye al análisis de BIAS del modelo.

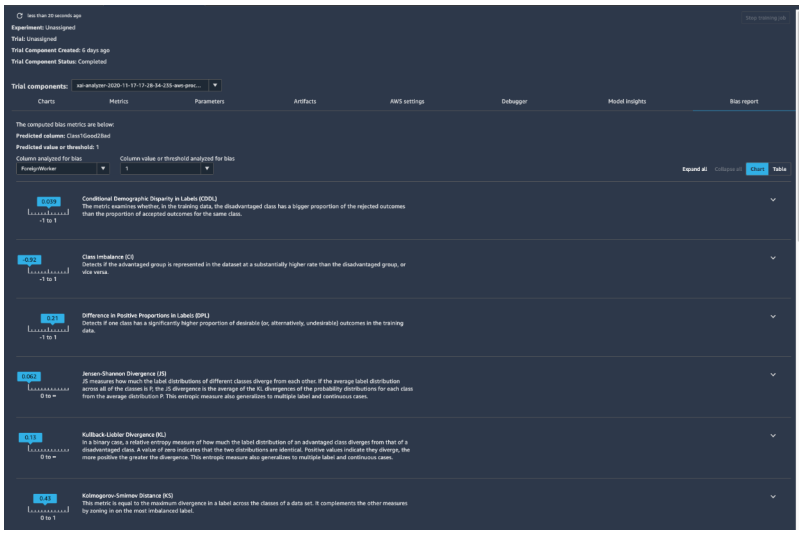


Figura 4.55. Reporte de Sesgos Amazon Clarify

Clarify puede generarse como JSON, a continuación se muestra el resultado:

```
    {
  "analysis_metadata": {
    "timestamp": "2025-08-23T17:37:08.493354",
    "framework": "Bias Analysis",
    "dataset_size": 109,
    "model_type": "Hybrid Recommendation System"
  },
  "data_quality": {
    "csv_reading_method": "Robust parsing with error handling",
    "original_size": "Variable (dependent on export)",
    "cleaned_size": 109,
    "columns_analyzed": [
      "book_id",
      "user_id",
      "title",
      "author_name",
      "actual_rating",
      "book_avg_rating",
      "book_ratings_count",
      "publication_year",
      "num_pages",
      "language_code",
      "is_english",
      "is_popular",
      "is_recent",
      "is_long_book",
      "is_highly_rated",
      "user_liked_book",
      "is_very_popular",
      "predicted_rating",
```

```
      "prediction_confidence",
      "prediction_method"
    ]
  },
  "bias_analysis_results": {
    "is_english": {
      "sensitive_attribute": "is_english",
      "privileged_group_size": 48,
      "unprivileged_group_size": 61,
      "privileged_mean_prediction": 3.942028173839178,
      "unprivileged_mean_prediction": 3.9367036919814042,
      "mean_difference": 0.005324481857773566,
      "demographic_parity_difference": -0.13012295081967212,
      "equalized_odds_difference": 0.09875776397515529,
      "calibration_difference": 0.008171756530964913,
      "privileged_positive_rate": 0.3125,
      "unprivileged_positive_rate": 0.4426229508196721,
      "bias_magnitude": 0.13012295081967212,
      "bias_level": "Medium"
    },
    "is_popular": {
      "sensitive_attribute": "is_popular",
      "privileged_group_size": 57,
      "unprivileged_group_size": 52,
      "privileged_mean_prediction": 3.9892982456140347,
      "unprivileged_mean_prediction": 3.8839668760605033,
      "mean_difference": 0.1053313695535314,
      "demographic_parity_difference": 0.11167341430499322,
      "equalized_odds_difference": 0.10792682926829267,
      "calibration_difference": 0.021471720430725316,
      "privileged_positive_rate": 0.43859649122807015,
      "unprivileged_positive_rate": 0.3269230769230769,
      "bias_magnitude": 0.11167341430499322,
      "bias_level": "Medium"
    },
    "is_recent": {
      "sensitive_attribute": "is_recent",
      "privileged_group_size": 2,
      "unprivileged_group_size": 107,
      "privileged_mean_prediction": 3.555,
      "unprivileged_mean_prediction": 3.9462268930387503,
      "mean_difference": -0.39122689303875013,
      "demographic_parity_difference": -0.3925233644859813,
      "equalized_odds_difference": 0.375,
      "calibration_difference": 0.04795586500136739,
      "privileged_positive_rate": 0.0,
      "unprivileged_positive_rate": 0.3925233644859813,
      "bias_magnitude": 0.3925233644859813,
      "bias_level": "High"
    },
```

```
    "is_long_book": {
      "sensitive_attribute": "is_long_book",
      "privileged_group_size": 47,
      "unprivileged_group_size": 62,
      "privileged_mean_prediction": 3.9159345789545887,
      "unprivileged_mean_prediction": 3.9565701991012983,
      "mean_difference": -0.04063562014670952,
      "demographic_parity_difference": -0.04152367879203839,
      "equalized_odds_difference": 0.03501228501228498,
      "calibration_difference": 0.015278337935876962,
      "privileged_positive_rate": 0.3617021276595745,
      "unprivileged_positive_rate": 0.4032258064516129,
      "bias_magnitude": 0.04152367879203839,
      "bias_level": "Low"
    }
  },
  "summary": {
    "total_facets_analyzed": 4,
    "high_bias_facets": 1,
    "medium_bias_facets": 2,
    "low_bias_facets": 1
  },
  "recommendations": [
    {
      "facet": "is_english",
      "bias_level": "Medium",
      "demographic_parity": -0.13012295081967212,
      "recommendation": "Review is_english bias - Medium level
         detected"
    },
    {
      "facet": "is_popular",
      "bias_level": "Medium",
      "demographic_parity": 0.11167341430499322,
      "recommendation": "Review is_popular bias - Medium level
         detected"
    },
    {
      "facet": "is_recent",
      "bias_level": "High",
      "demographic_parity": -0.3925233644859813,
      "recommendation": "Review is_recent bias - High level
         detected"
    }
  ]
}
```


El informe es explicado en la tabla 4.13.

Tabla 4.13. Resumen del análisis de sesgo por atributo sensible en el sistema de recomendación de libros.

Atributo	Nivel de Sesgo	Descripción
is_english	Medio	El modelo muestra una ligera diferencia en las predicciones entre libros en inglés (48 registros) y no en inglés (61 registros). La paridad demográfica es -0.13, indicando sesgo medio.
is_popular	Medio	Existe un sesgo medio en la predicción según la popularidad de los libros: 57 libros populares frente a 52 no populares. La paridad demográfica es 0.11.
is_recent	Alto	Se detecta un sesgo alto en libros recientes (2 registros) frente a no recientes (107 registros). La paridad demográfica es -0.39, indicando que los libros recientes podrían estar siendo desfavorecidos.
is_long_book	Bajo	El sesgo según la longitud del libro es bajo. 47 libros largos frente a 62 cortos, con una diferencia de paridad demográfica de -0.04.

Y visualmente, mediante la gráfica 4.56, que ha sido realizada a partir de los resultados obtenidos.

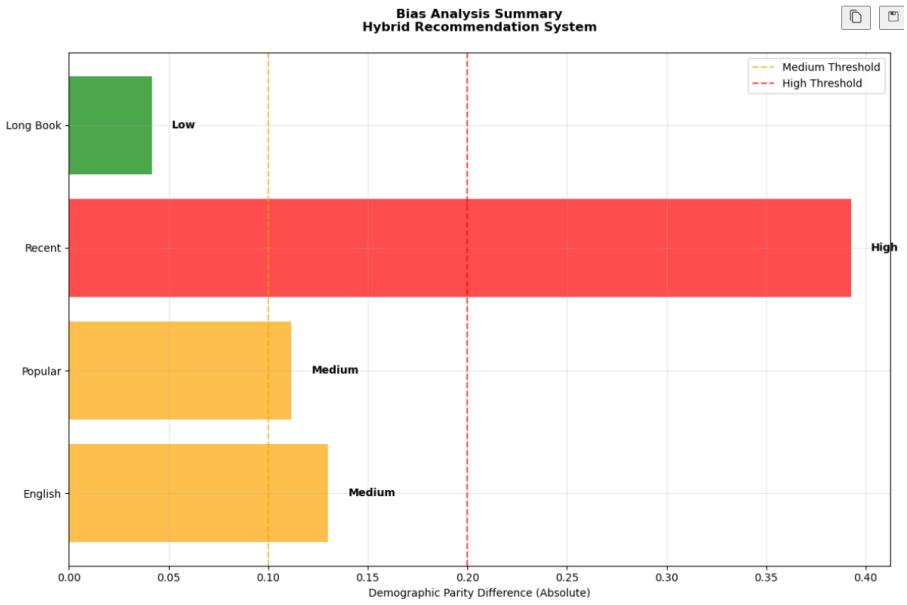


Figura 4.56. Representación gráfica de resultado de análisis de sesgos Clarify

Este JSON cumple como un informe de auditoría de sesgo y calidad, mostrando dónde el modelo podría discriminar entre libros o usuarios según ciertas características.

Cabe recalcar que los análisis de sesgos han sido realizados con un conjunto de ejemplo del dataset original de libros, esto para mitigar el costo de uso que tiene el servicio de SageMaker, ya que es un servicio que se encuentra fuera de los servicios de capa gratuita; el costo de uso se puede observar mejor en el apartado de presupuesto.

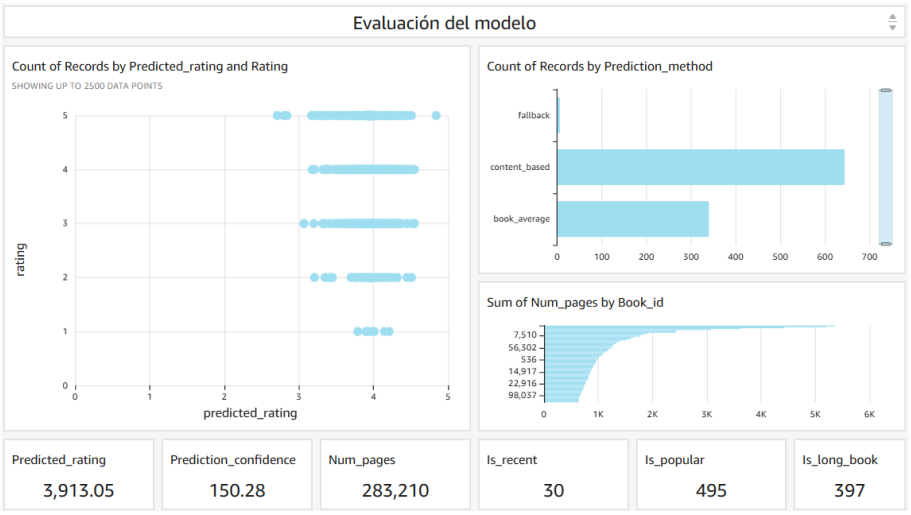


Figura 4.57. Análisis de resultdos de modelo en Amazon QuickSight

Sagmeaker model monitor

Amazon SageMaker Model Monitor es un servicio administrado que permite supervisar de manera continua los modelos de machine learning en producción. Su objetivo es detectar desviaciones entre los datos que recibe el modelo en tiempo real y los datos utilizados durante su entrenamiento, así como identificar problemas de calidad en las predicciones. Model Monitor genera reportes automáticos y puede integrarse con Amazon CloudWatch para generar alarmas, lo que facilita una gobernanza del ciclo de vida del modelo. De esta forma, las organizaciones aseguran que sus modelos desplegados sean precisos, robustos, justos y confiables a lo largo del tiempo [41].

El uso de SageMaker Model Monitor se justifica principalmente en escenarios donde las predicciones del modelo tienen un impacto directo en decisiones críticas, como en finanzas (detección de fraude, aprobación de créditos), salud (diagnóstico automático, monitoreo de pacientes), e-commerce (pricing dinámico, recomendaciones principales) y manufactura (control de calidad, mantenimiento predictivo).

Áreas donde un error del modelo puede traducirse en pérdidas económicas significativas, riesgos regulatorios o incluso consecuencias de seguridad. En contraste, aplicaciones no críticas como estudios académicos, análisis históricos o segmentaciones exploratorias no requieren monitoreo constante, ya que el impacto de posibles desvíos es mínimo y no compensa el costo operativo.

Tabla 4.14. Casos de uso específicos para SageMaker Model Monitor.

Sector	Uso de monitor	No necesario	Descripción
Finanzas	• Detección de fraude en tiempo real • Aprobación automática de créditos • Trading algorítmico	• Análisis histórico de riesgo • Reportes mensuales de compliance • Modelos de investigación	Alto impacto económico y regulatorio
E-commerce	• Sistemas de recomendación principales • Pricing dinámico • Detección de bots	• Análisis de sentimientos • Segmentación de clientes • A/B testing de features	Impacto directo en revenue y UX
Salud	• Diagnóstico automático • Monitoreo de pacientes críticos • Dosificación de medicamentos	• Análisis epidemiológico • Investigación clínica • Estudios retrospectivos	Seguridad del paciente es crítica
Manufactura	• Control de calidad automático • Mantenimiento predictivo crítico • Optimización de producción	• Análisis de eficiencia • Planificación de inventario • Estudios de mejora	Paradas no planificadas son costosas

En términos de costo-beneficio, aunque la implementación inicial implica gastos en instancias de cómputo, almacenamiento en S3 y configuración, los beneficios de detectar desviaciones tempranas, garantizar compliance y evitar pérdidas financieras justifican la inversión en la mayoría de los contextos empresariales.

El retorno de la inversión (ROI) suele lograrse rápidamente (de 2 a 3 meses) especialmente cuando el costo de un error supera con creces el gasto de monitoreo. A medida que el sistema escala y se aplican políticas centralizadas para múltiples modelos, la relación costo/beneficio se vuelve aún más favorable, reforzando la conveniencia de usarlo en entornos productivos con alto impacto.

Trazabilidad de modelos a tráves de SageMaker Pipelines

Amazon SageMaker Pipelines es el servicio de orquestación de flujos de trabajo de machine learning de AWS, diseñado para automatizar todo el ciclo de vida de los modelos, desde el procesamiento de datos hasta el entrenamiento, la evaluación y el despliegue del modelo [42].

Tabla 4.15. Trazabilidad y reproducibilidad en SageMaker Pipelines

Aspecto	Trazabilidad	Reproducibilidad
Versionado automático	Los artefactos, datos y modelos se almacenan y versionan en S3 y Model Registry.	Permite reejecutar el pipeline con los mismos datos y parámetros, manteniendo versiones idénticas.
Registro de linaje	Se documentan los datos, parámetros y código usados en cada etapa para auditoría.	Scripts y configuraciones quedan guardados junto a cada ejecución y pueden reutilizarse en futuras pruebas.
Metadatos y logs	Se generan automáticamente para cada ejecución y etapa, facilitando el seguimiento histórico.	Posibilita relanzar experimentos y comparar resultados pasados de forma fiable.
Definición modular	El pipeline se construye como una secuencia declarativa, con dependencias explícitas entre pasos.	Al ser modular, cada paso es reproducible y puede ejecutarse igual en cualquier entorno.

IA Responsable

Amazon SageMaker Clarify permite detectar y mitigar sesgos en los datos y en los modelos, además de ofrecer explicabilidad de predicciones. Importante para cumplir con principios de transparencia y equidad, ya que ayuda a identificar si ciertos grupos están siendo desfavorecidos y a proporcionar explicaciones claras de cómo el modelo toma decisiones.

Algunos sectores como la banca, donde es crítico explicar decisiones de crédito, hasta la sanidad, automatización industrial, recursos humanos y seguros, ámbitos donde la equidad y la transparencia son obligatorias, es necesaria una manera de interpretar los resultados y decisiones aportadas por el modelo, por lo que, Clarify evalúa el modelo y cuantifica el impacto de cada atributo sobre el resultado final, mejorando la confianza de usuarios y reguladores.

SageMaker Model Monitor garantiza que los modelos se mantengan bajo control una vez desplegados en producción, supervisando continuamente la calidad de datos, métricas de deriva, sesgos emergentes y cumplimiento normativo. Esta supervisión es importante dentro de la gobernanza del dato, ya que establece un ciclo de control y mejora continua, reduciendo riesgos de degradación o decisiones incorrectas. Clarify y Model Monitor permiten a las organizaciones implementar una IA responsable bajo un marco de gobernanza sólido, donde los modelos no solo entregan valor de negocio, sino que lo hacen de manera ética, transparente y sostenible.

Monitoreo, Observabilidad y Auditoría

Un aspecto esencial dentro de la gobernanza del dato y del ciclo de vida de un sistema analítico con inteligencia artificial es garantizar la observabilidad y trazabilidad de los procesos. En este contexto, los servicios Amazon CloudWatch y AWS CloudTrail cumplen un rol transversal a la arquitectura, asegurando que las operaciones no solo sean eficientes, sino también auditables y responsables.

Amazon CloudWatch

Por un lado, Amazon CloudWatch permite recopilar métricas y registros en tiempo real sobre el comportamiento de los diferentes componentes de la solución, tales como AWS Lambda, Amazon API Gateway, Amazon S3 y Amazon SageMaker.

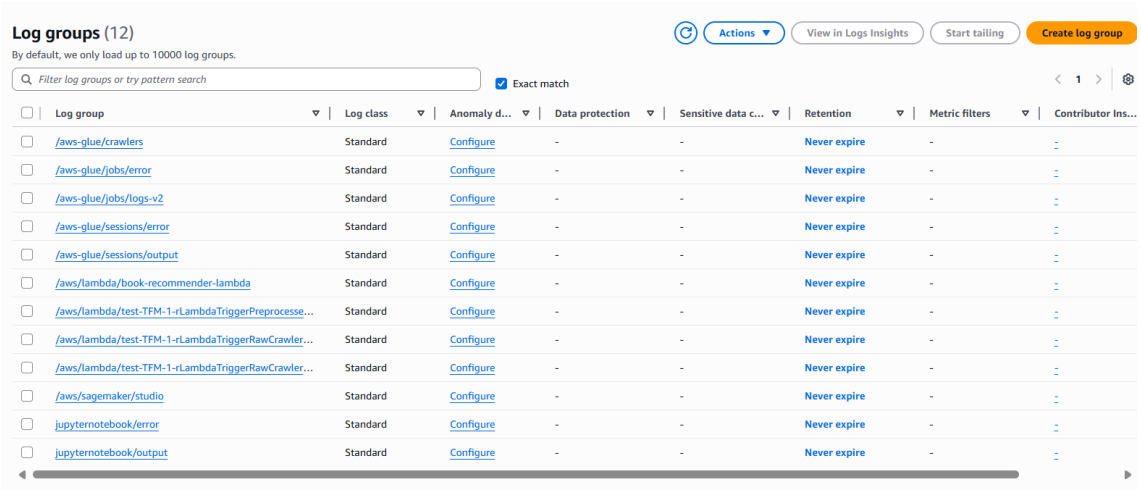


Figura 4.58. Grupos creados para logs en CloudWatch

A través de los logs se puede obtener métricas operativas, diagnósticas y de negocio, dependiendo del tipo de servicio que se requiera.

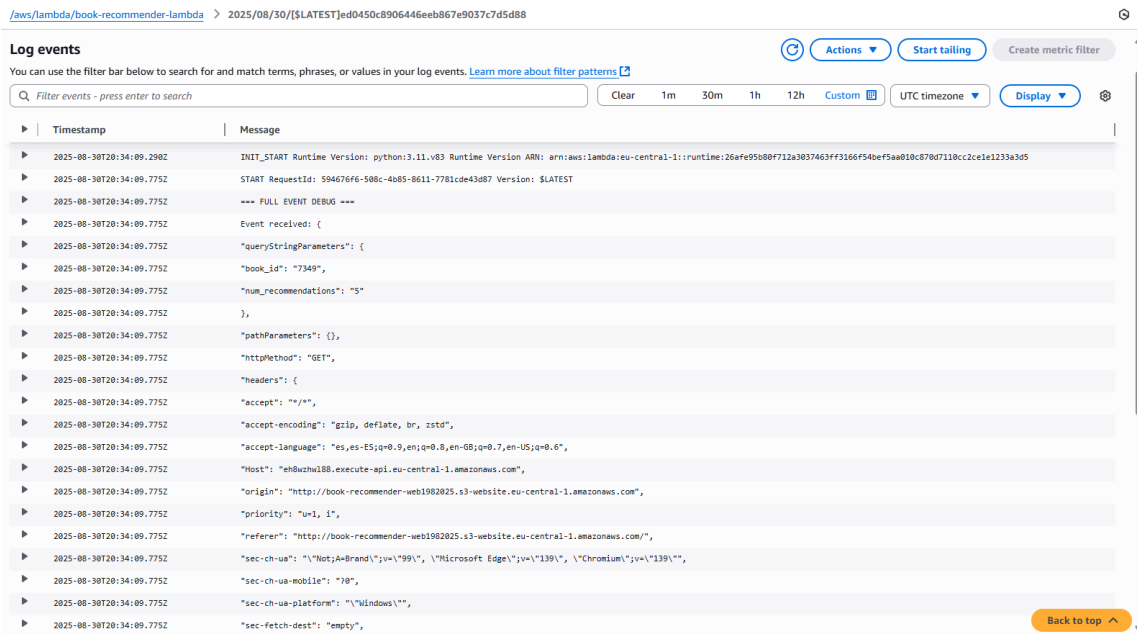


Figura 4.59. Logs Events de servicio Lambda

La figura 4.59 muestra los ejecuciones realizadas por un evento lambda. La tabla 4.16 muestra los usos por servicio de Amazon CloudWatch.

Tabla 4.16. Información registrada en CloudWatch Logs por servicio de AWS.

Servicio	Información en logs	Uso
AWS Lambda	Ejecuciones exitosas y fallidas; mensajes de error y excepciones (stack trace); tiempo de ejecución (detección de timeouts); memoria utilizada vs. asignada; eventos de entrada y salida.	Optimización de rendimiento; depuración de errores; validación del modelo de recomendaciones.
Amazon API Gateway	Solicitudes recibidas (método HTTP, endpoint, timestamp); latencia de respuestas; códigos de estado HTTP (200, 400, 500); identificación de usuarios (API Keys o JWT).	Detección de cuellos de botella; auditoría de consumo de la API; visibilidad de errores de clientes y servidor.
Amazon S3	Accesos a objetos (quién, cuándo, desde dónde); errores de acceso (denegado por políticas o bucket privado); eventos de escritura (nuevos datos o resultados de modelos).	Seguridad y control de accesos; auditoría de cambios en datos; validación de cargas de datos.
Amazon SageMaker	Latencia de inferencia; logs de predicciones y entradas mal formadas; errores en contenedor (memoria insuficiente, dependencias rotas).	Monitorización del rendimiento del modelo; depuración de fallos en predicciones; optimización de endpoints de inferencia.

Las métricas constituyen también la posibilidad de analizar el consumo de recursos, establecer alarmas de rendimiento y generar paneles de control que faciliten la supervisión continua.

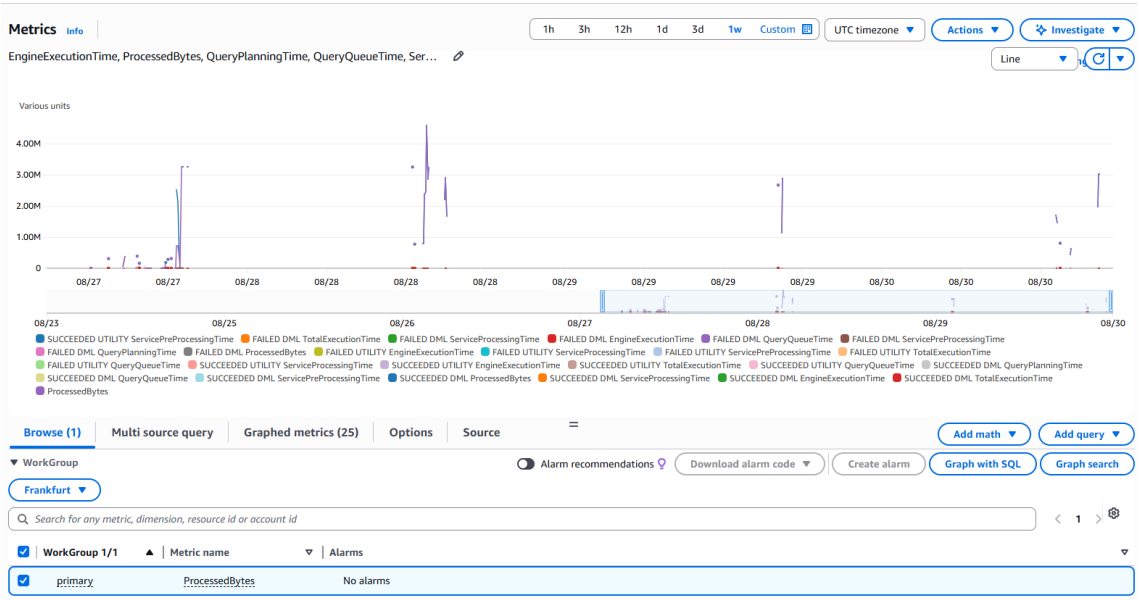


Figura 4.60. Métricas registradas por uso de Amazon Athena

Esto garantiza que la aplicación mantenga niveles de calidad del servicio acordes a los objetivos planteados.

Amazon CloudTrail

Por otro lado, AWS CloudTrail registra cada llamada a la API realizada dentro de la cuenta de AWS, incluyendo información sobre el usuario, la hora, el servicio invocado y los recursos implicados [21].

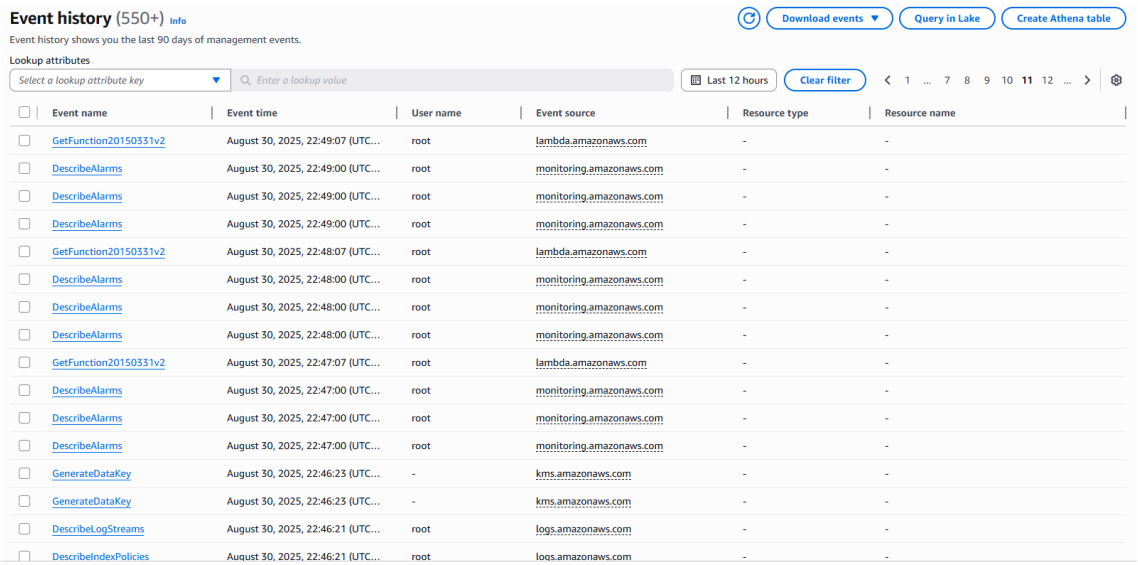


Figura 4.61. Historial de eventos registrados por CloudTrail

Cada evento se registra en CloudTrail, como resultado, se genera un JSON que registra todas las actividades realizadas en la cuenta.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "Root",
        "principalId": "116496425672",
        "arn": "arn:aws:iam::116496425672:root",
        "accountId": "116496425672",
        "accessKeyId": "ASIARWH52MLEB24GANZJ",
        "sessionContext": {
            "attributes": {
                "creationDate": "2025-08-30T16:57:14Z",
                "mfaAuthenticated": "true"
            }
        }
    },
    "eventTime": "2025-08-30T20:49:07Z",
    "eventSource": "lambda.amazonaws.com",
    "eventName": "GetFunction20150331v2",
    "awsRegion": "eu-central-1",
```

```
20      "sourceIPAddress": "5.40.161.229",
21      "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
           AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0
           Safari/537.36 Edg/139.0.0.0",
22      "requestParameters": {
23          "functionName": "search_books_api"
24      },
25      "responseElements": null,
26      "requestID": "9eb28dfc-5b16-463b-8b21-a667d6e8ff04",
27      "eventID": "65b0687b-3e61-4f57-acc1-daa2c0b03362",
28      "readOnly": true,
29      "eventType": "AwsApiCall",
30      "managementEvent": true,
31      "recipientAccountId": "116496425672",
32      "eventCategory": "Management",
33      "tlsDetails": {
34          "tlsVersion": "TLSv1.3",
35          "cipherSuite": "TLS_AES_128_GCM_SHA256",
36          "clientProvidedHostHeader": "lambda.eu-central-1.amazonaws.
               com"
37      },
38      "sessionCredentialFromConsole": "true"
39  }
```

El JSON presenta el registro de actividades generado para el recurso Lambda al generar una función. La estructura del evento es explicado en la tabla 4.17.

Tabla 4.17. Ejemplo de evento registrado por AWS CloudTrail.

Campo	Valor	Descripción
eventTime	2025-08-30T20:49:07Z	Fecha y hora en que ocurrió el evento.
eventSource	lambda.amazonaws.com	Servicio de AWS afectado (Lambda).
eventName	GetFunction20150331v2	Operación realizada, en este caso obtener la configuración de una función Lambda.
functionName	search_books_api	Nombre de la función Lambda consultada.
userIdentity	Root user con MFA habilitado	Usuario que realizó la acción, con acceso desde la consola de AWS.
awsRegion	eu-central-1	Región de AWS donde ocurrió la acción (Frankfurt).
sourceIPAddress	5.40.161.229	Dirección IP pública desde donde se ejecutó la acción.
userAgent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Edge/139.0.0.0	Navegador/cliente utilizado para acceder a la consola de AWS.
readOnly	true	Indica que el evento no modificó recursos.
tlsDetails	TLSv1.3, AES_128_GCM_SHA256	Protocolo y cifrado de la conexión segura.

Esta trazabilidad resulta fundamental en un entorno donde se gestionan datos de usuarios y modelos de aprendizaje automático, ya que asegura un control detallado de los accesos y modificaciones, lo que contribuye al cumplimiento de políticas de seguridad y requisitos regulatorios. En el marco de la gobernanza del dato, CloudTrail aporta una capa de auditoría y responsabilidad indispensable para garantizar la confianza y transparencia en el uso de la arquitectura.

Ambos servicios fortalecen la solución al un entorno seguro, auditable y monitorizable, lo que permite detectar incidencias de manera temprana, demostrar cumplimiento con buenas prácticas de gobernanza de datos y uso responsable de la inteligencia artificial.

4.2.5. Desarrollo de la aplicación

Una vez que los datos han sido procesados, transformados y almacenados en el Data Warehouse (Amazon Redshift) y catalogados en el Data Lake de AWS Glue, se habilita la capa de consumo que materializa el objetivo principal del sistema: alimentar un modelo de recomendación y ponerlo a disposición de los usuarios finales mediante una aplicación web.

Microservicios Serveless AWS Lambda

El modelo de recomendación entrenado se despliega en AWS de manera serverless.

Se crean 3 microservicios Lambda.

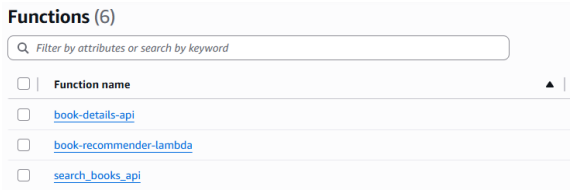


Figura 4.62. Lambdas creados para aplicación

El primer microservicio *book-recommender-lambda* se encarga del llamado al modelo y tabla 'features' conseguidas mediante SageMaker. Devolviendo a la salida los k libros recomendados según el libro ingresado.

El script usa TF-IDF + Cosine Similarity con algoritmos híbridos. Para ello, el microservicio consta de:

- Entrada: Recibe eventos de API Gateway o pruebas directas
- Salida: JSON con recomendaciones rankeadas por similitud

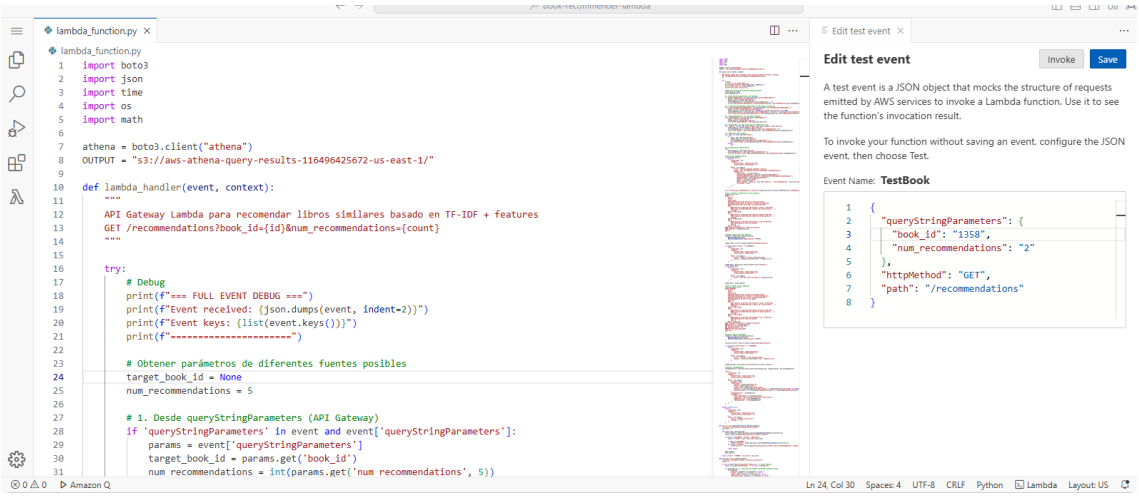


Figura 4.63. Script y test de prueba de microservicio de recomendaciones

la salida es un JSON de la siguiente manera.

```
    {
  "statusCode": 200,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
  "body": "{\"success\": true, \"target_book\": {\"book_id\":
     \"1358\", \"title\": \"Gorgias: Encomium of Helen\", \"
     author_name\": \"Gorgias of Leontini\", \"rating\": 3.63, \"
     publication_year\": 1991}, \"recommendations\": [{\"book_id\":
      \"96163\", \"title\": \"Keep It Simple: Daily Meditations For
```

```
 Twelve-Step Beginnings And Renewal\", \"author_name\": \"
Anonymous\", \"rating\": 4.55, \"ratings_count\": 56, \"
publication_year\": 1989, \"pages\": 416, \"publisher\": \"
Hazelden Publishing\", \"similarity_score\": 0.258, \"
algorithm_used\": \"TF-IDF + Multi-factor\"}, {\"book_id\":
\"54747\", \"title\": \"Mafalda 10 (Mafalda, #10)\", \"
author_name\": \"Quino\", \"rating\": 4.59, \"ratings_count\":
 408, \"publication_year\": 1991, \"pages\": 86, \"publisher
\": \"De La Flor\", \"similarity_score\": 0.256, \"
algorithm_used\": \"TF-IDF + Multi-factor\"}], \"metadata\":
{\"total_candidates\": 100, \"algorithm\": \"TF-IDF Cosine
Similarity + Multi-factor\", \"requested_count\": 2, \"
returned_count\": 2}}"
}
```

Se obtiene la información del libro objetivo y los k libros recomendados, junto a algunos metadatos como: puntuación de similitud y el algoritmo usado.

El segundo microservicio *lambda_books_details* proprociona la información detallada de los libros para el frontend. Este microservicio es complemetario para el recomendador.

El propósito es obtener información completa de uno o múltiples libros, para ello recibe:

- Entrada: *book_id* (individual) o book_ids (múltiples)
- Salida: JSON con todos los detalles del libro

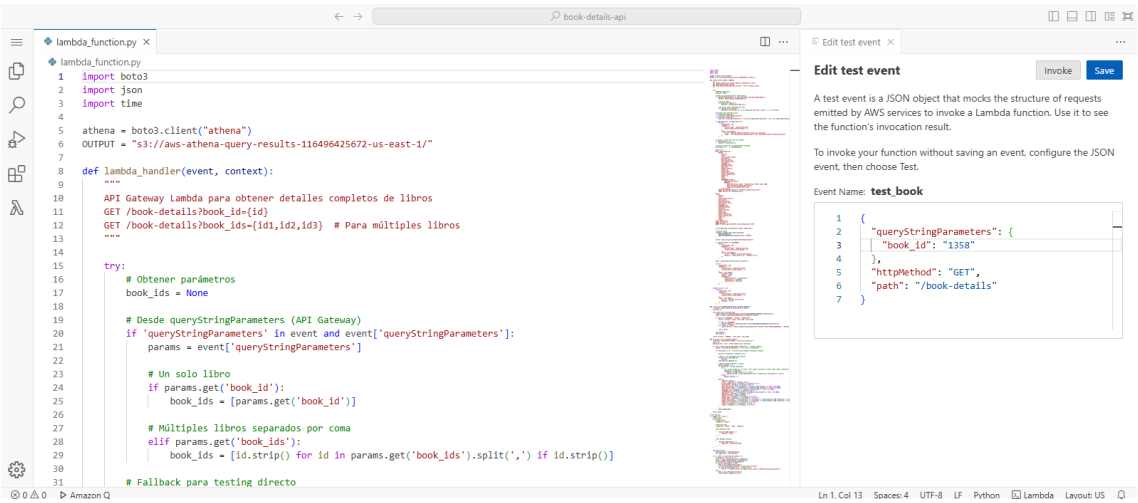


Figura 4.64. Script y test de prueba de microservicio para obteción de información de libros

La salida es un JSON de la siguiente manera:

```
    {
  "statusCode": 200,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
```

```
  "body": "{\"success\": true, \"books\": [{\"book_id\": \"1358\",
     \"title\": \"Gorgias: Encomium of Helen\", \"
     title_without_series\": \"Gorgias: Encomium of Helen\", \"
     description\": \"The Encomium of Helenis thought to have been
     the demonstration piece of the Ancient Greek sophist,
     Presocratic philosopher and rhetorician, Gorgias. In this
     edition Malcolm MacDowell provides a useful introduction, the
     Greek text, his own English translation, and commentary.\", \"
     book_avg_rating\": 3.63, \"book_ratings_count\": 86, \"
     num_pages\": 48, \"language_code\": \"\", \"publication_year
     \": 1991, \"publisher\": \"Bristol Classical Press\", \"
     popular_shelves\": [], \"author_id\": \"5622822\", \"
     author_name\": \"Gorgias of Leontini\", \"author_avg_rating\":
      3.68, \"author_rating_count\": 114, \"image_url\": \"https://
     s.gr-assets.com/assets/nophoto/book/111x148-
     bcc042a9c91a29c1d680899eff700a03.png\", \"isbn\":
     \"0862920531\", \"isbn13\": \"9780862920531\"}], \"metadata\":
      {\"requested_count\": 1, \"found_count\": 1, \"requested_ids
     \": [\"1358\"]}}"
}
```


Se observa que se obtienen aspectos importantes del libro como su id, titulo, descripción, autor, imagen, código SBN, entre otros. Información útil para el frontend de la aplicación.

El tercer microservicio *lambda_search_books* busca reemplazar la busqueda por id a busqueda por título, esto proporciona una experiencia de usuario más natural.

- Entrada: title (término de búsqueda)
- Salida: Lista de libros ordenados por relevancia y rating

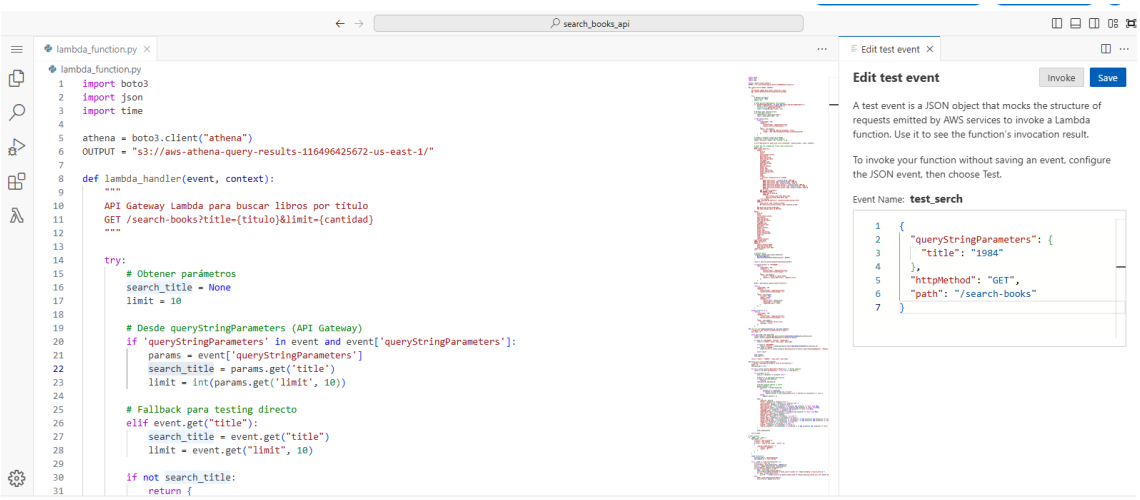


Figura 4.65. Script y test de prueba de microservicio para búsqueda de libro por título

La salida es un JSON con la siguiente estructura:

```
{
```

```
  "statusCode": 200,
  "headers": {
    "Content-Type": "application/json",
    "Access-Control-Allow-Origin": "*"
  },
  "body": "{\"success\": true, \"books\": [{\"book_id\": \"12716\",
      \"title\": \"George Orwell's 1984: A Guide to Understanding
     the Classics\", \"title_without_series\": \"George Orwell's
     1984: A Guide to Understanding the Classics\", \"description
     \": \"\", \"book_avg_rating\": 4.15, \"book_ratings_count\":
     114, \"num_pages\": 0, \"language_code\": \"eng\", \"
     publication_year\": null, \"publisher\": \"\", \"
     popular_shelves\": [], \"author_id\": \"206832\", \"
     author_name\": \"Ralph A. Ranald\", \"author_avg_rating\":
     4.11, \"author_rating_count\": 124, \"image_url\": \"https://s
     .gr-assets.com/assets/nophoto/book/111x148-
     bcc042a9c91a29c1d680899eff700a03.png\", \"isbn\": \"067100719X
     \", \"isbn13\": \"9780671007195\", \"search_relevance\": 80}],
      \"metadata\": {\"search_term\": \"1984\", \"found_count\": 1,
      \"requested_limit\": 10}}"
}
```


Este Lambda es importante para la UX porque proporciona toda la información necesaria que el usuario necesita para tomar decisiones sobre qué libros leer.

En resumen los microservicios funcionan de la forma establecida en la tabla 4.18.

Tabla 4.18. Comparativa de funcionalidades en los módulos de búsqueda, detalles y recomendación de libros.

Aspecto	Search Books	Book Details	Book Recommender
Input	Texto libre	book_id(s) específicos	1 book_id
Algoritmo	Relevancia textual	Simple lookup	TF-IDF + ML
Ordering	Relevancia + Rating	Rating DESC	Similarity score
Dedup	Por book_id	No necesaria	No necesaria
Uso	“Buscar libros”	“Detalles del libro”	“Similar a este”

Endpoint AWS API Gateway

En la capa de consumo de la arquitectura, se incorpora Amazon API Gateway como servicio central para exponer de forma segura y escalable el acceso al sistema de recomendaciones.

API Gateway permite la creación, la publicación, el mantenimiento, el monitoreo y la protección de las API REST, HTTP y de WebSocket a cualquier escala [43]. Este actúa como la puerta entre los usuarios de la aplicación web y los microservicios desplegados en la nube.

El sistema de recomendación de libros funciona con 3 endpoints REST en API Gateway que invocan Lambdas específicos:

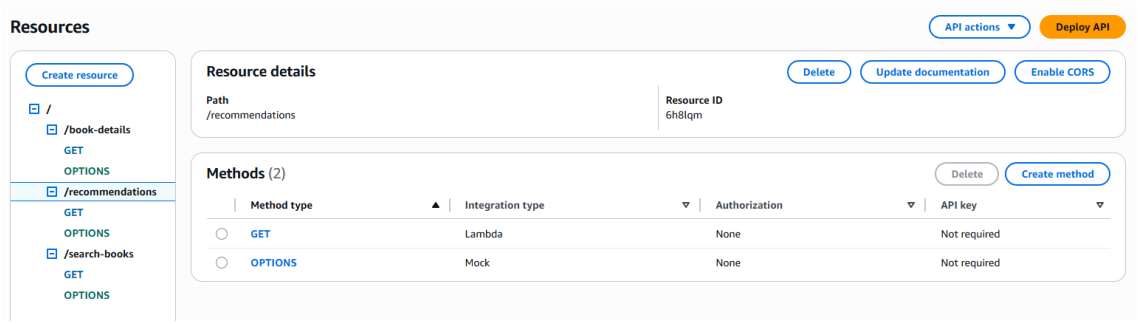


Figura 4.66. Endpoints de la aplicación en API Gateway

- recommendations ejecuta lambda_book_recommender.py para generar recomendaciones usando TF-IDF + cosine similarity a partir de book_id
- book-details ejecuta lambda_book_details.py para obtener información completa de libros (portadas, descripciones, autor) usando book_id o book_ids
- search-books ejecuta lambda_search_books.py para buscar libros por título con ranking de relevancia.

Cada Lambda consulta diferentes tablas Athena y retorna JSON con CORS habilitado. CORS permite al servidor compartir recursos con otros orígenes mediante cabeceras HTTP [44].

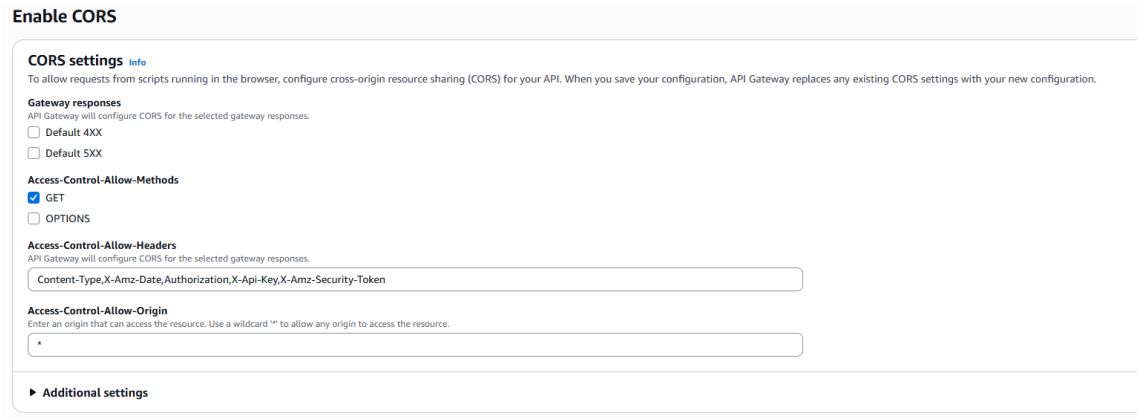


Figura 4.67. Configuración de CORS en API Gateway

El frontend JavaScript consume estos endpoints secuencialmente: primero busca libros por título, luego obtiene detalles completos, y finalmente, genera recomendaciones basadas en el libro seleccionado, creando una experiencia de usuario fluida desde búsqueda hasta recomendaciones personalizadas.

S3 Static website hosting

Además de la exposición de servicios mediante API Gateway, se implementa el alojamiento del frontend de la aplicación web en Amazon S3 como sitio web estático. Este enfoque permite desplegar de forma sencilla y escalable una aplicación web desarrollada en JavaScript, almacenando directamente en un bucket los archivos estáticos generados.

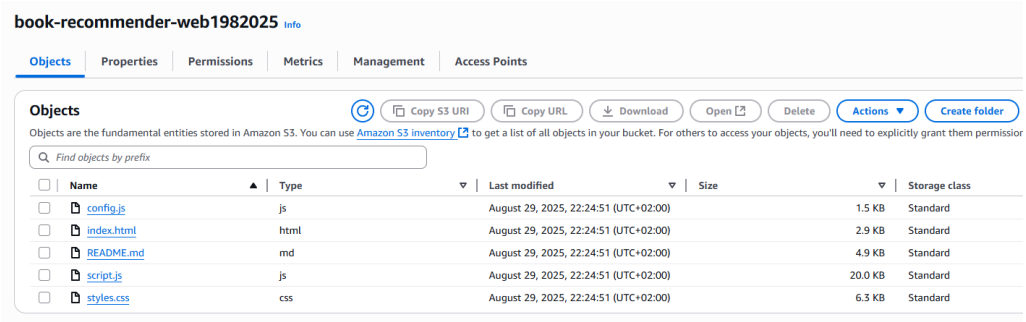


Figura 4.68. Archivos estáticos en S3

Para que S3 funcione como alojamiento estático es necesario habilitar la opción disponible en su configuración.

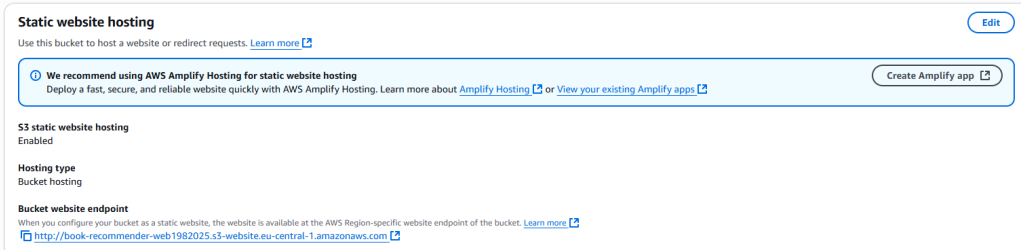


Figura 4.69. Habilitación como bucket de contenido estático

Al habilitar la opción de static website hosting en S3, la aplicación queda disponible a través de una URL pública.

Y también configurar su política:

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Sid": "PublicReadGetObject",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::book-recommender-web1982025
            /*"
    }
]
}
```

Otra opción disponible también es habilitar un dominio personalizado gestionado mediante Amazon Route 53 el cual pude integrarse junto a Amazon CloudFront [45] como CDN, el cual distribuye el contenido de manera global y habilita soporte HTTPS mediante AWS Certificate Manager, para este trabajo simplemente se dejará como contenido estático mediante una URL pública.

Aplicación web y funcionamiento

La figura 4.70 muestra el frontend desplegado.

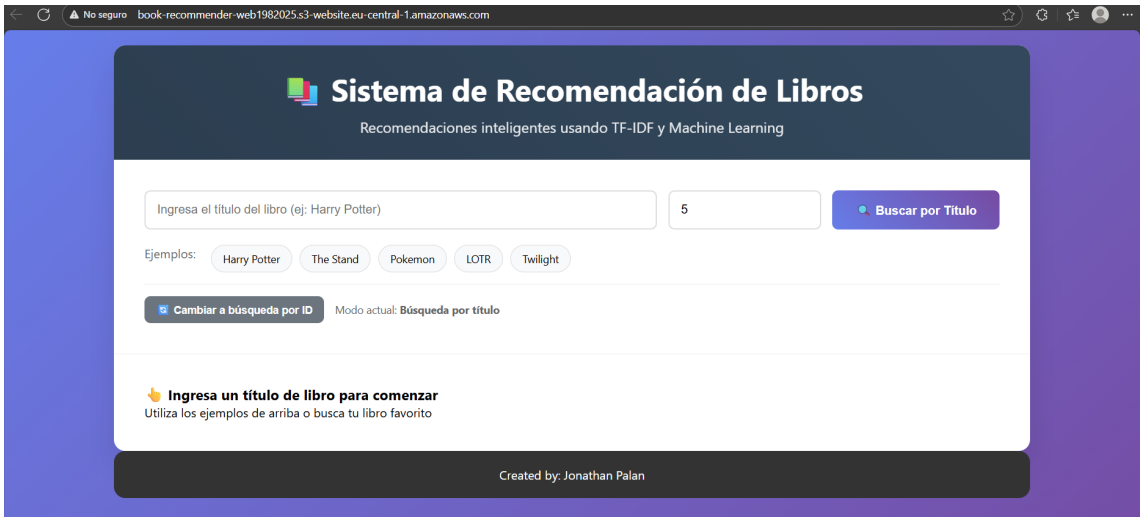


Figura 4.70. Frontend de aplicación

Primero buscamos dentro del catálogo el libros alojado en AWS, el libro que interesa generar recomendaciones.

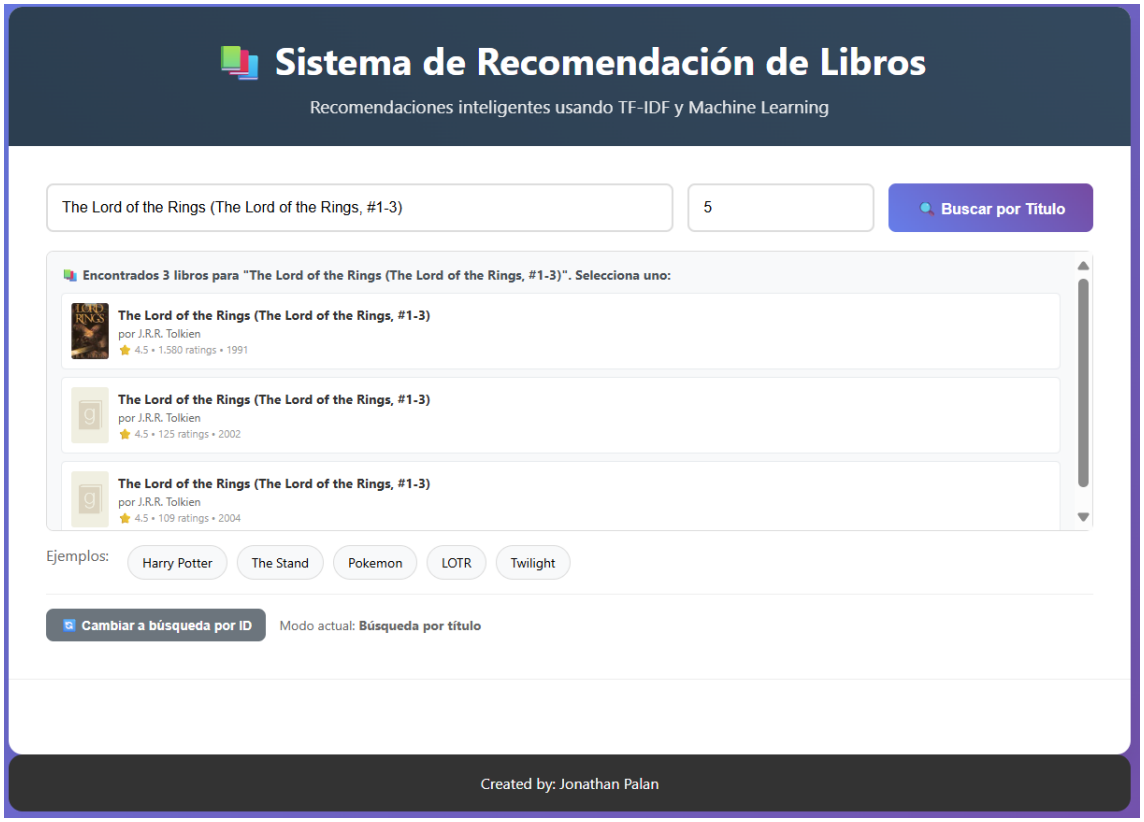


Figura 4.71. Selección de libro en catálogo

Una vez seleccionado generamos las recomendaciones.

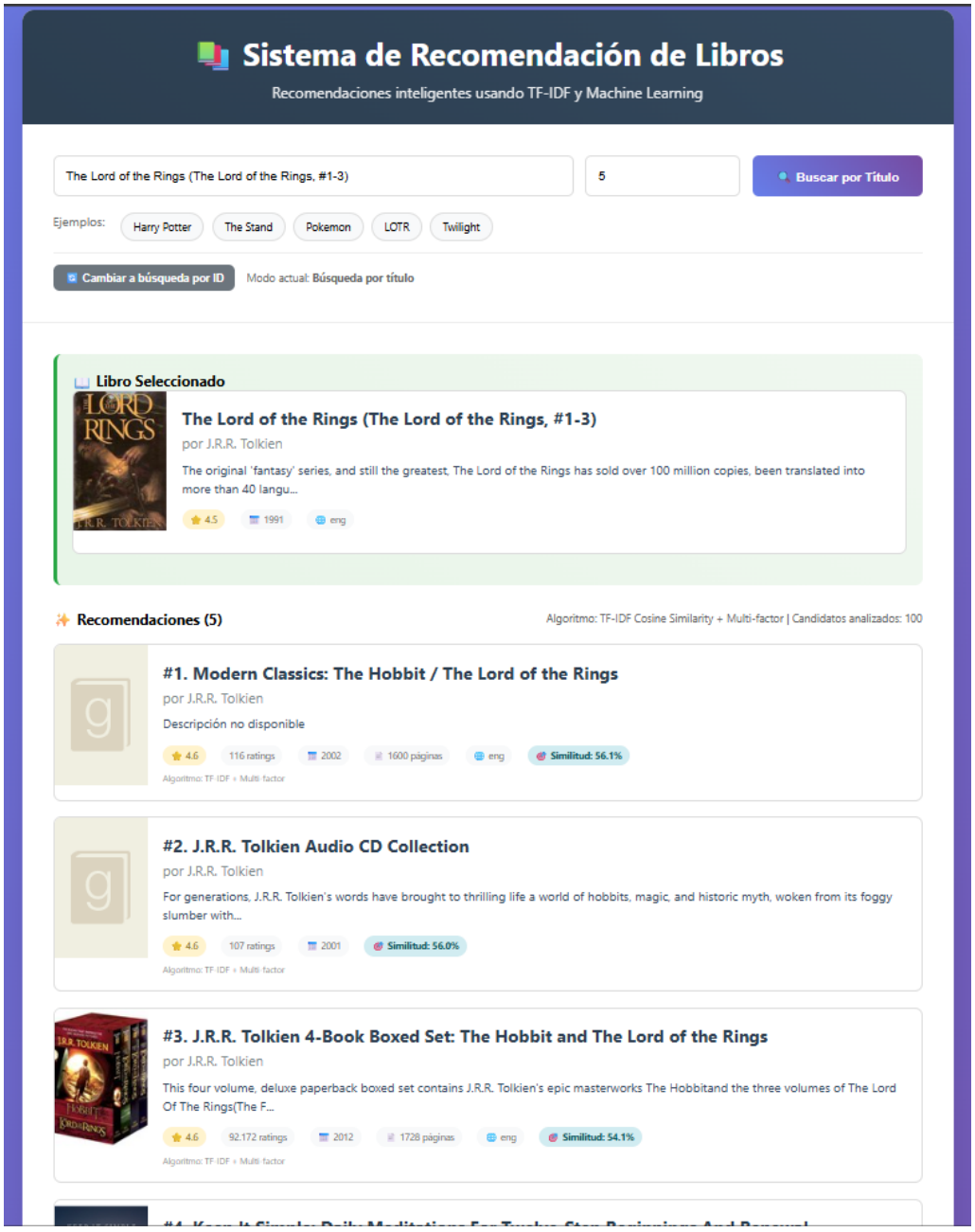


Figura 4.72. Recomendaciones generadas por aplicación web

Al buscar recomendaciones del libro *The Lord of the Rings (The Lord of the Rings, #13)* vemos que el algortimo recomienda libros continuados de la misma saga del autor como primeras opciones, el algoritmo calcula las similitud junto a los demás opiciones y las presenta.

Con esta última capa, la arquitectura queda en estado end-to-end: desde la ingesta y gobernanza del dato en AWS, pasando por el almacenamiento en S3/Redshift, análisis de negocio con QuickSight, creación de modelo de machine learning, hasta el consumo mediante un sistema de recomendaciones accesible vía aplicación web.

4.3 Recursos requeridos

Se presentan los recursos necesarios para el desarrollo.

- **Ordenador portátil:**
  - Procesador: Intel i7-11800H
  - Memoria RAM: 24 GB
  - Tarjeta gráfica: Nvidia RTX 3050 TI
- **Periféricos:**
  - Teclado
  - Ratón
  - Auriculares
- **Servicios en la nube:**
  - Cuenta AWS personal
- **Software:**
  - IDE Visual Studio Code
  - Overlaf para la redacción de memoria en Latex
  - Power Point

4.4 Presupuesto

En esta sección se muestra el presupuesto previsto para el desarrollo del proyecto este refleja la evaluación económica total.

La tabla 4.19 muestra un desglose del presupuesto, incluyendo horas de trabajo y gastos.

Tabla 4.19. Tabla de costes del proyecto

Tipo de coste	Valor	Comentarios
Horas de trabajo en el proyecto	140 horas	Cantidad de horas usadas para el desarrollo del proyecto, desde la planificación hasta la redacción de la memoria final. Se incluye el tiempo en investigación, pruebas y documentación.
Horas de trabajo en colaboración	5 horas	Horas dedicadas a revisión y rectificación del proyecto junto al tutor.
Ordenador portátil	€1500	Valor del portátil personal usado para el desarrollo del proyecto.
Periféricos	€100	Valor estimado de periféricos correpondientes al PC.
IDE Visual Studio Code	€0	IDE de desarrollo gratuito
Cuenta AWS	€0	Free Tier 12 meses capa gratuita de AWS
Overleaf	€8	Suscripción para estudiantes universitarios usada para la redacción de la memoria.
Power Point	€0	Uso gratuito de Power Point
Materiales empleados	€0	No se ha usado materiales físicos para el desarrollo del proyecto.

4.5 Viabilidad

Se estima que el gasto en materiales incluidos equipos y software es de alrededor de 1608€ que podrían ser redondeados a 1700€ para cubrir improvistos.

Es necesario tener en cuenta que el proyecto refleja gastos fijos en los servicios de AWS, a pesar de que se utiliza la capa gratuita para el desarrollo de este proyecto, se estimula un aproximado sobre el consumo de distintos servicios que no están incluidos en el uso de free tier de AWS o que por el uso exceden el límite de gratuidad.

Tabla 4.20. Cálculo de costes dependiendo del servicio.

Servicio	**Costo por uso**
Amazon S3 (almacenamiento)	$\text{Cost}_{S3} = \text{size}_{\text{GB}} \times 0{,}023\text{EUR}\ /\text{GB·mes}$
AWS Glue (ETL)	$\text{Cost}_{Glue} = \text{runs}_m \times \left(\text{DPU} \times \frac{\text{min\_per\_run}}{60}\right) \times 0{,}44\text{EUR}\ /\text{DPU·h}$
Redshift Serverless	$\text{Cost}_{RS} = \text{RPUh}_m \times 0{,}375\text{EUR}\ /\text{RPU·h}$
Lambda — invocaciones	$\text{Cost}_{\text{inv}} = \frac{\text{inv}_m}{10^6} \times 0{,}20\text{EUR}\ /\text{1M inv}$
Lambda — cómputo	$\text{Cost}_{\text{comp}} = \text{inv}_m \times (\text{mem}_{\text{GB}} \times \text{sec}) \times 0{,}000016667\text{EUR}\ /\text{GB·s}$
API Gateway (REST)	$\text{Cost}_{API} = \frac{\text{inv}_m}{10^6} \times 3{,}50\text{EUR}\ /\text{1M calls}$
SageMaker (entrenamiento)	$\text{Cost}_{SM} = \text{hrs}_m \times \text{price}_h\text{EUR}\ /\text{h}$
CloudWatch + CloudTrail (logs)	$\text{Cost}_{logs} = \text{ingest}_{\text{GB}} \times 0{,}50\text{EUR} + \text{ret}_{\text{GB}} \times 0{,}03\text{EUR}$
QuickSight (Standard)	$\text{Cost}_{QS} = \text{users} \times 12\text{EUR}\ /\text{usuario·mes}$

Servicios como SageMaker y sus herramientas estan fuera de los usos con servicio gratuito, por lo que el costo dependerá de la capacidad de la máquina a usar el tiempo de ejecución y el número de trabajos paralelos, esto se conoce como DPU (Data Processing Unit, Unidad de Procesamiento de Datos) la cual es la medida de los recursos de cómputo asignados a los procesos ETL [46].

La tabla 4.21 muestra la viabilidad económica aproximada mensualmente por el uso de los distintos servicios.

Tabla 4.21. Estimación mensual de costes por escenario.

Concepto	**Demo**	**Produción ligera**
Amazon S3 (almacenamiento)	€2.53	€11.50
AWS Glue (ETL + Crawler)	€4.70	€26.40
Redshift Serverless (consumo)	€37.50	€225.00
Lambda + cómputo (512MB, 200ms)	€0.02	€1.67
API Gateway (REST)	€0.04	€3.50
SageMaker (entreno esporádico)	€7.36	€25.00
CloudWatch y CloudTrail (monitoreo)	€2.59	€21.00
QuickSight (un usuario)	€12.00	€24.00
Total mensual aprox.	**€66.7**	**€338.1**

Los valores calculados corresponden a aproximaciones dependientes de los servicios a usar y las regiones de los servidores de AWS.

4.6 Resultados del proyecto

El desarrollo del proyecto ha permitido implementar un sistema integral de gestión y análisis de datos con enfoque en gobernanza del dato en entornos de analítica y machine learning sobre AWS. Los principales resultados alcanzados se basan en los objetivos establecidos al inicio del proyecto.

El primer objetivo estableció describir las funcionalidades de los distintos servicios de AWS para la gobernanza de datos. Para esto se ha repasado el contexto de todos los servicios disponibles útiles para el objetivo y aplicarlos. Se ha implementado AWS Lake Formation que se utilizó para gestionar permisos centralizados sobre los datos almacenados en S3, garantizando control granular a nivel de tablas, columnas y filas. AWS Glue Data Catalog permitió centralizar la catalogación de metadatos, facilitando la búsqueda, clasificación y control de calidad de los datasets. AWS IAM proporcionó la base para la gestión de identidades y roles, definiendo accesos diferenciados para administradores, analistas de datos y científicos de datos. AWS CloudTrail y AWS CloudWatch se integraron para asegurar un monitoreo continuo y auditoría de accesos, generando trazabilidad completa sobre el uso de datos y servicios.

El segundo objetivo planteaba establecer procesos para extracción, transformación y carga, catalogación y clasificación de datos que faciliten la calidad, integridad y disponibilidad de los datos para el entrenamiento de modelos de inteligencia artificial. Los datos fueron transformados a formato Parquet mediante AWS Glue y almacenados en capas (raw, transformed, join) en Amazon S3, facilitando su posterior análisis. Glue Data Catalog permitió clasificar datasets según categorías y aplicar gobernanza en Lake Formation. Los datos gobernados alimentaron tanto el entrenamiento del modelo de recomendación como distintos dashboards de análisis.

El cumplimiento del tercer objetivo que mencionaba definir políticas, roles y responsabilidades necesarias para implementar una gobernanza de datos, incluyendo el control de acceso y la gestión de identidades mediante servicios como AWS Lake Formation Y AWS IAM. Para esto se definieron políticas de IAM para restringir accesos según el rol del usuario: un administrador de datos que tiene el control total de metadatos y permisos en Lake Formation. Analista de negocio con acceso de solo lectura a tablas catalogadas y mediante el cual está a disposición para la creación de dashboards en QuickSight tal como se vió en la sección 4.2.3 Análisis de negocio a través de QuickSight. Científico de datos con acceso a datasets anonimizados para entrenamiento de modelos en SageMaker. Permitiendo demostrar la correcta aplicación del principio de mínimo privilegio en un entorno real.

El cuarto objetivo estipulaba el diseñar y aplicar un caso práctico que utilice servicios analíticos e inteligencia artificial en AWS para demostrar la integración efectiva de servicios de gobernanza del dato, garantizando el acceso controlado a datos y su posterior uso y despliegue en modelos de inteligencia artificial. Se construyó un servicio completo de datos en Amazon S3 junto a AWS Glue y Redshift que alimenta un modelo de recomendación de libros basado en filtrado en contenido y filtrado colaborativo desplegado en AWS Lambda. El

modelo es accesible mediante un endpoint en Amazon API Gateway, integrando analítica, ML y gobernanza del dato en un mismo ecosistema.

El último objetivo era implementar mecanismos de monitoreo, auditoría y cumplimiento normativo automatizados utilizando herramientas de AWS como AWS CloudTrail y AWS CloudWatch para asegurar la protección. Esto como se puede observar en la sección 4.2.4 Monitoreo, observabilidad y auditoría se ha implementado CloudTrail donde se obtuvieron registros detallados de accesos a S3, Redshift y Lake Formation y demás servicios, garantizando evidencia para auditorías. Mientras que con CloudWatch, se recopilaron métricas de rendimiento de Lambda, API Gateway y SageMaker, facilitando la detección temprana de errores y optimización de recursos.

Capítulo 5. DISCUSIÓN

La integración de AWS Lake Formation, AWS Glue Data Catalog, y AWS IAM resultó vital para aplicar los principios de gobernanza del marco DAMA-DMBOK. En particular, se trataron eficientemente las áreas de seguridad de datos, gestión de metadatos y gobernanza de datos. Lake Formation facilitó la administración centralizada y detallada de políticas de seguridad, asegurando el cumplimiento del principio de .ªcceso controlado"del DAMA. Por otro lado, Glue Data Catalog sirvió como un depósito confiable de metadatos, en línea con la gestión de metadatos, lo que mejoró la accesibilidad y comprensión de los conjuntos de datos. Finalmente, IAM apoyó el componente de Roles y Responsabilidades, permitiendo una definición clara de identidades, permisos y niveles de acceso.

No obstante, durante el desarrollo surgieron importantes desafíos prácticos. Entre ellos, la dificultad de establecer políticas de acceso en Lake Formation, que sean lo bastante estrictas para asegurar la protección de datos y lo suficientemente adaptables para no obstaculizar el análisis y el entrenamiento de modelos. También se destacó el reto de integrar de manera eficiente diversas fuentes de datos heterogéneas, lo que exigió un esfuerzo adicional en la fase de ETL utilizando AWS Glue. Asimismo, la configuración detallada de IAM y la administración de múltiples roles con permisos específicos incrementó la carga administrativa y demandó un nivel avanzado de conocimiento para evitar configuraciones incorrectas que pudieran comprometer la seguridad.

En cuanto al impacto directo de la gobernanza del dato sobre el modelo de IA, los beneficios fueron tangibles. La calidad y consistencia de los datos transformados y catalogados mejoraron la fiabilidad de las predicciones generadas por el sistema de recomendación. Gracias a la trazabilidad y auditoría habilitadas con CloudTrail y CloudWatch, fue posible detectar anomalías en las invocaciones al modelo y en el flujo de datos, lo que redujo errores y permitió corregir rápidamente situaciones que podrían haber afectado los resultados. Esto se tradujo en un modelo no solo más preciso, sino también más confiable desde una perspectiva de negocio y de seguridad.

Tanto la experiencia como la práctica evidencian que la gobernanza de datos no es simplemente un añadido, sino un factor esencial para el éxito de la inteligencia artificial en el ámbito empresarial. En la ausencia de un marco de control adecuado, la calidad de los datos se deteriora, aumenta el riesgo de accesos no autorizados y los modelos pueden volverse ineficaces o perjudiciales para la empresa. Por el contrario, al implementar prácticas de gobernanza basadas en DAMA dentro de la infraestructura de AWS, se logró desarrollar un sistema de recomendación escalable, seguro y transparente, que puede satisfacer tanto las exigencias analíticas como las de negocio. Esto ratifica que la teoría y la práctica se encuentran entrelazadas al diseñar arquitecturas en la nube con la gobernanza como fundamento clave.

En el mundo empresarial actual, adoptar modelos de gobernanza en AWS es esencial para mantener la seguridad, asegurar el cumplimiento con normativas y optimizar los procesos de análisis. Por ejemplo, herramientas automatizadas de auditoría y monitoreo, como CloudTrail

y CloudWatch, permiten a las empresas demostrar cumplimiento durante auditorías internas y externas, lo cual es crucial para evitar multas vinculadas al RGPD. Definir con precisión los roles y políticas de acceso (utilizando IAM y Lake Formation) minimiza el riesgo de fuga de datos o accesos no autorizados, mientras que el uso de Glue para catalogar y transformar datos asegura la calidad y trazabilidad necesaria según la normativa. Industrias como la financiera y la de salud ya emplean técnicas avanzadas para mitigar sesgos en modelos de IA (como SageMaker Clarify y Model Monitor) con el fin de garantizar justicia y transparencia en decisiones críticas, siguiendo recomendaciones éticas y legales. Los análisis de costos revelan que, aunque la inversión inicial puede ser alta, los beneficios se manifiestan rápidamente al reducir pérdidas por errores, mejorar la calidad de los modelos y cumplir con los requerimientos regulatorios.

Capítulo 6. CONCLUSIONES

En este apartado se presentan las concluciones del proyecto, enfocadas personalmente al emplear los servicios tratados y también las obtenidas por el desarrollo del trabajo.

6.1 Conclusiones del trabajo

El desarrollo de un sistema integral de gobernanza y análisis de datos sobre AWS demuestra que es posible alcanzar elevados estándares de seguridad, control y calidad en la gestión de datos para entornos de análisis avanzados y aprendizaje automático. Esto se consigue mediante la integración coordinada de servicios como Lake Formation, Glue, IAM, junto con herramientas de monitoreo. Este sistema centraliza la gestión de permisos, asegura el cumplimiento del principio de mínimo privilegio y mejora la trazabilidad, lo que facilita tanto la protección como la disponibilidad de los datos para usuarios con diferentes perfiles y requerimientos.

Del mismo modo, adoptar procesos ETL automáticos, un sistema de catalogación y reglas de acceso bien definidas, junto con la implementación de un modelo de recomendación, demuestra que una adecuada gobernanza de datos no solo reduce riesgos de seguridad, sino que también mejora el aprovechamiento y reutilización de datos. Esto permite desarrollar soluciones analíticas e inteligentes que son robustas, escalables y auditables, estableciendo una fuerte base para tomar decisiones estratégicas y cumplir con regulaciones en organizaciones centradas en datos.

En resumen, se consiguió examinar y utilizar satisfactoriamente los servicios de AWS para el manejo de datos en un caso práctico de inteligencia artificial, demostrando la capacidad para integrar seguridad, catalogación y gestión de accesos dentro de un entorno de analítica avanzada. Entre los aprendizajes clave, se destaca la importancia de automatizar el ciclo de vida del dato, centralizar el control de permisos y mantener una supervisión constante para asegurar la fiabilidad de los modelos de IA en contextos empresariales.

6.2 Conclusiones personales

La realización de este proyecto ha permitido aplicar de forma integral los conocimientos adquiridos en las diversas áreas abordadas a lo largo del ciclo académico del máster, abarcando desde servicios de cloud computing en AWS hasta la gobernanza de datos y la implementación de técnicas de machine learning. Este trabajo constituye una síntesis práctica donde convergen conceptos fundamentales de gestión, procesamiento y análisis avanzado de datos, evidenciando la competencia para diseñar soluciones técnicas robustas en entornos de nube.

A nivel personal, el desarrollo del proyecto ha fortalecido las capacidades de comprensión y resolución en el ámbito de tratamiento de datos, destacando la aplicación eficiente de distintas técnicas y metodologías para abordar problemáticas complejas. Todo ello se integra en un entorno de trabajo completo que demuestra la adquisición y puesta en práctica de conocimientos avanzados, así como la capacidad para integrar y desplegar soluciones analíticas bajo principios de gobernanza y seguridad.

Capítulo 7. FUTURAS LÍNEAS DE TRABAJO

Si bien el proyecto desarrollado alcanzó los objetivos planteados en cuanto a la implementación de un sistema de recomendación de libros con un marco de gobernanza de datos en AWS, se identificaron diversas líneas de trabajo que podrían potenciar su alcance y generar mayor valor en el futuro.

Primero, resultaría pertinente expandir la capa de análisis avanzado e inteligencia artificial, integrando modelos de aprendizaje profundo y técnicas de procesamiento de lenguaje natural para analizar reseñas textuales y potenciar el sistema de recomendación. Esta expansión podría elevar la personalización y exactitud de las recomendaciones, alineándose con las tendencias modernas de sistemas inteligentes de recomendación. Es importante considerar que tales mejoras también implican un aumento en recursos y gastos.

Otro aspecto de evolución se enfoca en perfeccionar la gestión de datos, mediante la integración de herramientas de trazabilidad de datos y evaluación de calidad de datos que posibiliten evaluar automáticamente la confiabilidad de los conjuntos de datos empleados en los modelos. Además, se podría considerar el uso de AWS Macie para la identificación y protección de datos confidenciales, reforzando aún más la dimensión de seguridad y cumplimiento regulatorio.

En términos de despliegue, resultaría valioso avanzar hacia una arquitectura multi-región y altamente disponible, que asegure resiliencia y reduzca la latencia para usuarios globales. Además, la inclusión de CI/CD para pipelines de datos y modelos de IA mediante servicios como AWS CodePipeline o CodeBuild permitiría acelerar la iteración y el mantenimiento continuo del sistema.

Finalmente, desde la perspectiva de negocio, una futura línea de trabajo consistiría en integrar métricas de uso y feedback de usuarios finales para retroalimentar tanto el modelo de recomendación como los dashboards de negocio en QuickSight. De esta forma, el sistema no solo respondería a criterios técnicos, sino que también evolucionaría en función de las necesidades reales de los usuarios, logrando un ciclo virtuoso de mejora continua.

Bibliografía

[1] M. Alabi, «Data Governance and Quality: Ensuring Data Reliability and Trustworthiness,» oct. de 2023.

[2] R. O. Santos, «Modelo de Integración de datos, metadatos y conocimiento geograficos,» feb. de 2013.

[3] *¿Qué es el linaje de datos?* ibm.com, 2022. visitado 31 de mayo de 2025. dirección: https://www.ibm.com/es%E2%88%92es/think/topics/data%E2%88%92lineage.

[4] E. Commision, «Data governance and data policies,» jul. de 2020.

[5] *The Data Management Body of Knowledge*, dama.org. visitado 31 de mayo de 2025. dirección: https://dama.org/.

[6] A. M. Wibowo, «COBIT Control Objectives for IT Related Technology,» feb. de 2008.

[7] *Normas técnicas para un correcto gobierno del dato*, datos.gob.es. visitado 31 de mayo de 2025. dirección: https://datos.gob.es/es/documentacion/normas-tecnicas-para-un-correcto-gobierno-del-dato.

[8] *DAMA en la gestión de datos abiertos.* datos.gob.es. visitado 31 de mayo de 2025. dirección: https://datos.gob.es/es/blog/el-potencial-uso-de-la-metodologia-de-dama-en-la-gestion-de-los-datos-abiertos.

[9] *Data warehouse vs data lake—Complete guide for data engineers*, redpanda.com. visitado 31 de mayo de 2025. dirección: https://www.redpanda.com/guides/fundamentals-of-data-engineering-data-warehouse-vs-data-lake.

[10] M. S. Serna, «Inteligencia artificial y gobernanza de datos en las administraciones públicas: reflexiones y evidencias para su desarrollo,» pág. 32, feb. de 2021.

[11] J. L. F. de Córdoba, «Diseño y Despliegue de un Pipeline de Datos con Amazon Web Services para el Análisis de Datos de Clickstream,» pág. 75, jun. de 2024.

[12] *Herramientas de Gobierno de Datos en Amazon Web Services*, aws.amazon.com. visitado 31 de mayo de 2025. dirección: https://aws.amazon.com/es/blogs/aws-spanish/herramientas-de-gobierno-de-datos-en-amazon-web-services/.

[13] *Amazon S3*, aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://aws.amazon.com/es/s3/.

[14] *¿Qué es AWS Lake Formation?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/lake-formation/latest/dg/what-is-lake-formation.html.

[15] *Using crawlers to populate the Data Catalog*, docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/glue/latest/dg/add-crawler.html.

[16] *¿Qué es AWS Glue?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/glue/latest/dg/what-is-glue.html.

[17] *What is AWS Glue DataBrew?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/databrew/latest/dg/what-is.html.

[18] *¿Qué es Amazon Athena?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/athena/latest/ug/what-is.html.

[19] *¿Qué es Amazon Redshift?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/redshift/latest/mgmt/welcome.html.

[20] *¿Qué es Amazon SageMaker AI?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/sagemaker/latest/dg/whatis.html.

[21] *What Is AWS CloudTrail?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html.

[22] *¿Qué es AWS Config?* docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/es_es/config/latest/developerguide/WhatIsConfig.html.

[23] IDC, «The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things,» 2014.

[24] J. S. Esteve, «La regulación del uso de los datos en la Administración Pública I. Aspectos generales.,» *Universidad Nacional de Educación a Distancia, UNED*, 2023.

[25] *¿Cómo implementar una estrategia de data governance efectiva?* www.obsbusiness.school, 2025. visitado 2 de ago. de 2025. dirección: https://www.obsbusiness.school/blog/como-implementar-una-estrategia-de-data-governance-efectiva.

[26] *What is AWS CloudFormation?* amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.

[27] *Trabajando con JSON*, developer.mozilla.org, 2024. visitado 27 de jul. de 2025. dirección: https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/JSON.

[28] *Apache Spark - A Unified engine for large-scale data analytics*, spark.apache.org, 2024. visitado 27 de jul. de 2025. dirección: https://spark.apache.org/docs/4.0.0/.

[29] *Streaming ETL jobs in AWS Glue*, docs.aws.amazon.com, 2024. visitado 27 de jul. de 2025. dirección: https://docs.aws.amazon.com/glue/latest/dg/add-job-streaming.html.

[30] *¿Por qué deberías de usar ficheros Parquet si procesas muchos datos?* datos.gob.es, 2024. visitado 27 de jul. de 2025. dirección: https://datos.gob.es/es/blog/por-que-deberias-de-usar-ficheros-parquet-si-procesas-muchos-datos.

[31] *goodreads*, cseweb.ucsd.edu, 2024. visitado 27 de jul. de 2025. dirección: https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html.

[32] *collaborative-filtering*, ibm.com, 2024. visitado 2 de ago. de 2025. dirección: https://www.ibm.com/es-es/think/topics/collaborative-filtering.

[33] *YAML*, www.redhat.com, 2024. visitado 2 de ago. de 2025. dirección: https://www.redhat.com/es/topics/automation/what-is-yaml.

[34] *Starting visual ETL jobs in AWS Glue Studio*, docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/glue/latest/dg/edit-nodes-chapter.html#:~:text=You%20can%20use%20the%20simple%20visual%20interface%20in,in%20the%20AWS%20Glue%20Studio%20ETL%20job%20script..

[35] *Amazon Redshift*, aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://aws.amazon.com/es/redshift/free-trial/.

[36] *Amazon QuickSigth*, docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/quicksight/.

[37] *¿Qué significa ser data driven?* universidadeuropea.com, 2024. visitado 2 de ago. de 2025. dirección: https://universidadeuropea.com/blog/data-driven/.

[38] *Introducción a los sistemas de recomendación basados en filtrado colaborativo con PySpark*, medium.com, 2024. visitado 2 de ago. de 2025. dirección: https://medium.com/datos-y-ciencia/intro-als-pyspark-7de7f3ba3b0a.

[39] *¿Qué es el filtrado basado en contenido?* www.ibm.com, 2024. visitado 2 de ago. de 2025. dirección: https://www.ibm.com/mx-es/think/topics/content-based-filtering.

[40] T. Hodson, «Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not,» *Geoscientific Model Development*, vol. 15, págs. 5481-5487, jul. de 2022. DOI: 10.5194/gmd-15-5481-2022.

[41] *SageMaker Model Monitor*, docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor-mlops.html.

[42] *Pipelines*, docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines.html.

[43] *What is Amazon API Gateway?* docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html.

[44] *Intercambio de recursos de origen cruzado (CORS)*, developer.mozilla.org, 2024. visitado 2 de ago. de 2025. dirección: https://developer.mozilla.org/es/docs/Web/HTTP/Guides/CORS.

[45] *¿Qué es Amazon CloudFront?* docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: https://docs.aws.amazon.com/es_es/AmazonCloudFront/latest/DeveloperGuide/Introduction.html.

[46] *¿Qué es Amazon CloudFront?* docs.aws.amazon.com, 2024. visitado 2 de ago. de 2025. dirección: Configuraci%C3%B3n%20de%20las%20propiedades%20de%20trabajos%20para%20trabajos%20de%20Spark%20en%20AWS%20Glue.

Capítulo 8. ANEXOS

8.0.1. Template JSON de CloudFormation para despliegue de arquitectura en la nube.

```
   AWSTemplateFormatVersion: "2010-09-09"
Description: AWS ETL

Parameters:
  pNameBucketRaw:
    Type: String
    Description: Nombre del bucket donde se encuentran los datos
       crudos
    Default: datalake-bucket-raw

  pNameBucketPreprocessed:
    Type: String
    Description: Nombre del bucket donde se encuentran los datos
       preprocesados
    Default: datalake-bucket-preprocessed

  pNameBucketAnalytics:
    Type: String
    Description: Nombre del bucket donde se encuentran los datos
       procesados para an lisis
    Default: datalake-bucket-analytics

  pNameBucketAthenaResults:
    Type: String
    Description: Nombre del bucket donde se encuentran los
       resultados de las consultas de Athena
    Default: datalake-bucket-athena-results

  pGlueDatabaseRaw:
    Type: String
    Description: Nombre de la base de datos en Glue para datos
       crudos
    Default: datalake-db-raw

  pGlueDatabasePreprocessed:
    Type: String
    Description: Nombre de la base de datos en Glue para datos
       preprocesados
    Default: datalake-db-preprocessed

  pGlueDatabaseAnalytics:
    Type: String
```

```
37      Description: Nombre de la base de datos en Glue para datos
            procesados para an lisis
38      Default: datalake-db-analytics
39
40    pWebAppBucketName:
41      Type: String
42      Description: Nombre del bucket para la aplicaci n web
43      Default: tfm-book-recommender-web
44
45    pLambdaCodeBucket:
46      Type: String
47      Description: Nombre del bucket donde est n los scripts de las
            lambdas
48      Default: lambda-code-bucket
49
50  Resources:
51    # Buckets
52    rs3BucketRaw:
53      Type: AWS::S3::Bucket
54      Properties:
55        BucketName: !Join [ "-", [!Ref pNameBucketRaw, !Ref AWS::
              AccountId] ]
56
57    rs3BucketPreprocessed:
58      Type: AWS::S3::Bucket
59      Properties:
60        BucketName: !Join [ "-", [!Ref pNameBucketPreprocessed, !Ref
              AWS::AccountId] ]
61
62    rs3BucketAnalytics:
63      Type: AWS::S3::Bucket
64      Properties:
65        BucketName: !Join [ "-", [!Ref pNameBucketAnalytics, !Ref AWS
              ::AccountId] ]
66
67    rs3BucketAthenaResults:
68      Type: AWS::S3::Bucket
69      Properties:
70        BucketName: !Join [ "-", [!Ref pNameBucketAthenaResults, !Ref
               AWS::AccountId] ]
71
72    # Glue databases
73    rGlueDatabaseRaw:
74        Type: 'AWS::Glue::Database'
75        Properties:
76          DatabaseInput:
77            Name: !Ref pGlueDatabaseRaw
78            Description: 'Base de datos para datos crudos'
79          CatalogId: !Ref AWS::AccountId
80
```

```
81    rGlueDatabasePreprocessed:
82        Type: 'AWS::Glue::Database'
83        Properties:
84          DatabaseInput:
85            Name: !Ref pGlueDatabasePreprocessed
86            Description: 'Base de datos para datos preprocesados'
87          CatalogId: !Ref AWS::AccountId
88
89    rGlueDatabaseAnalytics:
90        Type: 'AWS::Glue::Database'
91        Properties:
92          DatabaseInput:
93            Name: !Ref pGlueDatabaseAnalytics
94            Description: 'Base de datos para datos procesados para
                  an lisis'
95          CatalogId: !Ref AWS::AccountId
96
97    # Glue crawlers
98    rGlueCrawlerRaw:
99      Type: AWS::Glue::Crawler
100     Properties:
101       Name: data-crawler-raw
102       Role: !GetAtt rRoleGlueCrawler.Arn
103       DatabaseName: !Ref pGlueDatabaseRaw
104       Targets:
105         S3Targets:
106           - Path: !Sub 's3://${rs3BucketRaw}/data/books/'
107           - Path: !Sub 's3://${rs3BucketRaw}/data/reviews/'
108           - Path: !Sub 's3://${rs3BucketRaw}/data/authors/'
109           - Path: !Sub 's3://${rs3BucketRaw}/data/book_id/'
110           - Path: !Sub 's3://${rs3BucketRaw}/data/user_id/'
111           - Path: !Sub 's3://${rs3BucketRaw}/data/reads_int/'
112       TablePrefix: raw_
113       SchemaChangePolicy:
114         UpdateBehavior: UPDATE_IN_DATABASE
115         DeleteBehavior: LOG
116
117   rGlueCrawlerPreprocessed:
118     Type: AWS::Glue::Crawler
119     Properties:
120       Name: data-crawler-preprocessed
121       Role: !GetAtt rRoleGlueCrawler.Arn
122       DatabaseName: !Ref pGlueDatabasePreprocessed
123       Targets:
124         S3Targets:
125           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/books/'
126           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/reviews
                  /'
127           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/authors
                  /'
```

```
128           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/book_id
                /'
129           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/user_id
                /'
130           - Path: !Sub 's3://${rs3BucketPreprocessed}/data/
                reads_int/'
131           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                books_clean/'
132           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                authors_clean/'
133           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                reviews_clean/'
134           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                book_id_clean/'
135           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                user_id_clean/'
136           - Path: !Sub 's3://${rs3BucketPreprocessed}/data_clean/
                reads_int_clean/'
137       TablePrefix: pre_
138       SchemaChangePolicy:
139         UpdateBehavior: UPDATE_IN_DATABASE
140         DeleteBehavior: LOG
141
142   rGlueCrawlerAnalytics:
143     Type: AWS::Glue::Crawler
144     Properties:
145       Name: data-crawler-analytics
146       Role: !GetAtt rRoleGlueCrawler.Arn
147       DatabaseName: !Ref pGlueDatabaseAnalytics
148       Targets:
149         S3Targets:
150           - Path: !Sub 's3://${rs3BucketAnalytics}/
                recommendation_dataset/'
151           - Path: !Sub 's3://${rs3BucketAnalytics}/
                content_based_dataset/'
152       TablePrefix: analytics_
153       SchemaChangePolicy:
154         UpdateBehavior: UPDATE_IN_DATABASE
155         DeleteBehavior: LOG
156
157   # Roles
158   rRoleGlueCrawler:
159     Type: AWS::IAM::Role
160     Properties:
161       AssumeRolePolicyDocument:
162         Version: '2012-10-17'
163         Statement:
164           - Effect: Allow
165             Principal:
166               Service: glue.amazonaws.com
```

```
167             Action:
168               - sts:AssumeRole
169       Path: '/'
170       ManagedPolicyArns:
171         - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
172       Policies:
173         - PolicyName: GlueCrawlerPolicy
174           PolicyDocument:
175             Version: '2012-10-17'
176             Statement:
177               - Effect: Allow
178                 Action:
179                   - s3:GetObject
180                   - s3:ListBucket
181                   - s3:GetBucketLocation
182                 Resource:
183                   - !Sub 'arn:aws:s3:::${rs3BucketRaw}'
184                   - !Sub 'arn:aws:s3:::${rs3BucketRaw}/*'
185               - Effect: Allow
186                 Action:
187                   - s3:GetObject
188                   - s3:ListBucket
189                   - s3:GetBucketLocation
190                 Resource:
191                   - !Sub 'arn:aws:s3:::${rs3BucketPreprocessed}'
192                   - !Sub 'arn:aws:s3:::${rs3BucketPreprocessed}/*'
193               - Effect: Allow
194                 Action:
195                   - s3:GetObject
196                   - s3:ListBucket
197                   - s3:GetBucketLocation
198                 Resource:
199                   - !Sub 'arn:aws:s3:::${rs3BucketAnalytics}'
200                   - !Sub 'arn:aws:s3:::${rs3BucketAnalytics}/*'
201               - Effect: Allow
202                 Action:
203                   - glue:GetDatabase
204                   - glue:CreateTable
205                   - glue:UpdateTable
206                   - glue:GetTable
207                   - glue:GetTables
208                   - glue:BatchCreatePartition
209                   - glue:BatchDeletePartition
210                 Resource:
211                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                        AccountId}:catalog'
212                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                        AccountId}:database/${pGlueDatabaseRaw}'
213                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                        AccountId}:table/${pGlueDatabaseRaw}/*'
```

```
214                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:database/${
                            pGlueDatabasePreprocessed}'
215                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:table/${pGlueDatabasePreprocessed
                            }/*'
216                 - Effect: Allow
217                   Action:
218                     - glue:GetDatabase
219                     - glue:CreateTable
220                     - glue:UpdateTable
221                     - glue:GetTable
222                     - glue:GetTables
223                     - glue:BatchCreatePartition
224                     - glue:BatchDeletePartition
225                   Resource:
226                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:database/${pGlueDatabaseAnalytics}'
227                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:table/${pGlueDatabaseAnalytics}/*'
228
229   rRoleGlueJobs:
230     Type: AWS::IAM::Role
231     Properties:
232       AssumeRolePolicyDocument:
233         Version: '2012-10-17'
234         Statement:
235           - Effect: Allow
236             Principal:
237               Service: glue.amazonaws.com
238             Action:
239               - sts:AssumeRole
240       Path: '/'
241       Policies:
242         - PolicyName: GlueJobsPolicy
243           PolicyDocument:
244             Version: '2012-10-17'
245             Statement:
246               - Effect: Allow
247                 Action:
248                   - s3:GetObject
249                   - s3:ListBucket
250                   - s3:GetBucketLocation
251                   - s3:PutObject
252                   - s3:DeleteObject
253                 Resource:
254                   - !Sub 'arn:aws:s3:::${rs3BucketRaw}'
255                   - !Sub 'arn:aws:s3:::${rs3BucketRaw}/*'
256                   - !Sub 'arn:aws:s3:::${rs3BucketPreprocessed}'
257                   - !Sub 'arn:aws:s3:::${rs3BucketPreprocessed}/*'
```

```
258                 - Effect: Allow
259                   Action:
260                     - glue:GetDatabase
261                     - glue:CreateTable
262                     - glue:UpdateTable
263                     - glue:GetTable
264                     - glue:GetTables
265                     - glue:CreateSession
266                     - glue:GetSession
267                     - glue:DeleteSession
268                     - glue:RunStatement
269                     - glue:CancelStatement
270                     - glue:GetStatement
271                     - glue:TagResource
272                     - glue:UntagResource
273                   Resource:
274                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:catalog'
275                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:database/${pGlueDatabaseRaw}'
276                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:database/${
                            pGlueDatabasePreprocessed}'
277                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:table/${pGlueDatabaseRaw}/*'
278                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:table/${pGlueDatabasePreprocessed
                            }/*'
279                     - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                            AccountId}:session/*'
280
281   # Glue Jobs
282   rGlueJobTransformBooks:
283     Type: AWS::Glue::Job
284     Properties:
285       Name: transform-json-to-parquet-books
286       Role: !GetAtt rRoleGlueJobs.Arn
287       Command:
288         Name: glueetl
289         #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
              transform_to_parquet.py'
290         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
              transform_json_to_parquet_books.py'
291         PythonVersion: 3
292       DefaultArguments:
293         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
294         "--job-language": "python"
295         "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/books'
296         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
              books'
```

```
297         MaxCapacity: 10
298         GlueVersion: "3.0"
299         Timeout: 60
300
301     rGlueJobTransformAuthors:
302       Type: AWS::Glue::Job
303       Properties:
304         Name: transform-json-to-parquet-authors
305         Role: !GetAtt rRoleGlueJobs.Arn
306         Command:
307           Name: glueetl
308           #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_to_parquet.py'
309           ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_json_to_parquet_authors.py'
310           PythonVersion: 3
311         DefaultArguments:
312           "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
313           "--job-language": "python"
314           "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/authors'
315           "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                authors'
316         MaxCapacity: 10
317         GlueVersion: "3.0"
318         Timeout: 60
319
320     rGlueJobTransformReviews:
321       Type: AWS::Glue::Job
322       Properties:
323         Name: transform-json-to-parquet-reviews
324         Role: !GetAtt rRoleGlueJobs.Arn
325         Command:
326           Name: glueetl
327           #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_to_parquet.py'
328           ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_json_to_parquet_reviews.py'
329           PythonVersion: 3
330         DefaultArguments:
331           "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
332           "--job-language": "python"
333           "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/reviews'
334           "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                reviews'
335         MaxCapacity: 10
336         GlueVersion: "3.0"
337         Timeout: 60
338
339     rGlueJobTransformBookId:
340       Type: AWS::Glue::Job
```

```
    Properties:
      Name: transform-json-to-parquet-book_id
      Role: !GetAtt rRoleGlueJobs.Arn
      Command:
        Name: glueetl
        #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
            transform_to_parquet.py'
        ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
            transform_csv_to_parquet_book_id.py'
        PythonVersion: 3
      DefaultArguments:
        "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
        "--job-language": "python"
        "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/book_id'
        "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
            book_id'
      MaxCapacity: 10
      GlueVersion: "3.0"
      Timeout: 60

  rGlueJobTransformUserId:
    Type: AWS::Glue::Job
    Properties:
      Name: transform-json-to-parquet-user_id
      Role: !GetAtt rRoleGlueJobs.Arn
      Command:
        Name: glueetl
        #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
            transform_to_parquet.py'
        ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
            transform_csv_to_parquet_user_id.py'
        PythonVersion: 3
      DefaultArguments:
        "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
        "--job-language": "python"
        "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/user_id'
        "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
            user_id'
      MaxCapacity: 10
      GlueVersion: "3.0"
      Timeout: 60

  rGlueJobTransformReadsInt:
    Type: AWS::Glue::Job
    Properties:
      Name: transform-json-to-parquet-reads_int
      Role: !GetAtt rRoleGlueJobs.Arn
      Command:
        Name: glueetl
```

```
384         #ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_to_parquet.py'
385         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts/
                transform_csv_to_parquet_reads_int.py'
386         PythonVersion: 3
387       DefaultArguments:
388         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
389         "--job-language": "python"
390         "--SOURCE_PATH": !Sub 's3://${rs3BucketRaw}/data/reads_int'
391         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                reads_int'
392       MaxCapacity: 10
393       GlueVersion: "3.0"
394       Timeout: 60
395
396   # Glue Clean Jobs
397   rGlueJobCleanBooks:
398     Type: AWS::Glue::Job
399     Properties:
400       Name: clean-books
401       Role: !GetAtt rRoleGlueJobs.Arn
402       Command:
403         Name: glueetl
404         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
                clean_books.py'
405         PythonVersion: 3
406       DefaultArguments:
407         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
408         "--job-language": "python"
409         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                books'
410         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
                data_clean/books_clean'
411       MaxCapacity: 10
412       GlueVersion: "3.0"
413       Timeout: 60
414
415   rGlueJobCleanAuthors:
416     Type: AWS::Glue::Job
417     Properties:
418       Name: clean-authors
419       Role: !GetAtt rRoleGlueJobs.Arn
420       Command:
421         Name: glueetl
422         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
                clean_authors.py'
423         PythonVersion: 3
424       DefaultArguments:
425         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
426         "--job-language": "python"
```

```
427         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                authors'
428         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
                data_clean/authors_clean'
429       MaxCapacity: 10
430       GlueVersion: "3.0"
431       Timeout: 60
432
433   rGlueJobCleanReviews:
434     Type: AWS::Glue::Job
435     Properties:
436       Name: clean-reviews
437       Role: !GetAtt rRoleGlueJobs.Arn
438       Command:
439         Name: glueetl
440         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
                clean_reviews.py'
441         PythonVersion: 3
442       DefaultArguments:
443         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
444         "--job-language": "python"
445         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                reviews'
446         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
                data_clean/reviews_clean'
447       MaxCapacity: 10
448       GlueVersion: "3.0"
449       Timeout: 60
450
451   rGlueJobCleanBookId:
452     Type: AWS::Glue::Job
453     Properties:
454       Name: clean-book-id
455       Role: !GetAtt rRoleGlueJobs.Arn
456       Command:
457         Name: glueetl
458         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
                clean_book_id.py'
459         PythonVersion: 3
460       DefaultArguments:
461         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
462         "--job-language": "python"
463         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
                book_id'
464         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
                data_clean/book_id_clean'
465       MaxCapacity: 10
466       GlueVersion: "3.0"
467       Timeout: 60
468
```

```
469   rGlueJobCleanUserId:
470     Type: AWS::Glue::Job
471     Properties:
472       Name: clean-user-id
473       Role: !GetAtt rRoleGlueJobs.Arn
474       Command:
475         Name: glueetl
476         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
              clean_user_id.py'
477         PythonVersion: 3
478       DefaultArguments:
479         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
480         "--job-language": "python"
481         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
              user_id'
482         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
              data_clean/user_id_clean'
483       MaxCapacity: 10
484       GlueVersion: "3.0"
485       Timeout: 60
486
487   rGlueJobCleanReadsInt:
488     Type: AWS::Glue::Job
489     Properties:
490       Name: clean-reads-int
491       Role: !GetAtt rRoleGlueJobs.Arn
492       Command:
493         Name: glueetl
494         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_clean/
              clean_reads_int.py'
495         PythonVersion: 3
496       DefaultArguments:
497         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
498         "--job-language": "python"
499         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}/data/
              reads_int'
500         "--DEST_PATH": !Sub 's3://${rs3BucketPreprocessed}/
              data_clean/reads_int_clean'
501       MaxCapacity: 10
502       GlueVersion: "3.0"
503       Timeout: 60
504
505   rGlueJobRecommendationDataset:
506     Type: AWS::Glue::Job
507     Properties:
508       Name: create-recommendation-dataset
509       Role: !GetAtt rRoleGlueJobs.Arn
510       Command:
511         Name: glueetl
```

```
512         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_models/
                recommendation_dataset.py'
513         PythonVersion: 3
514       DefaultArguments:
515         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
516         "--job-language": "python"
517         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}'
518         "--DEST_PATH": !Sub 's3://${rs3BucketAnalytics}'
519       MaxCapacity: 10
520       GlueVersion: "3.0"
521       Timeout: 60
522
523   rGlueJobContentBasedDataset:
524     Type: AWS::Glue::Job
525     Properties:
526       Name: create-content-based-dataset
527       Role: !GetAtt rRoleGlueJobs.Arn
528       Command:
529         Name: glueetl
530         ScriptLocation: !Sub 's3://${rs3BucketRaw}/scripts_models/
                content_based_dataset.py'
531         PythonVersion: 3
532       DefaultArguments:
533         "--TempDir": !Sub 's3://${rs3BucketPreprocessed}/temp/'
534         "--job-language": "python"
535         "--SOURCE_PATH": !Sub 's3://${rs3BucketPreprocessed}'
536         "--DEST_PATH": !Sub 's3://${rs3BucketAnalytics}'
537       MaxCapacity: 10
538       GlueVersion: "3.0"
539       Timeout: 60
540
541   # Lambda Role
542   rRoleLambdaCrawlerTrigger:
543     Type: AWS::IAM::Role
544     Properties:
545       AssumeRolePolicyDocument:
546         Version: '2012-10-17'
547         Statement:
548           - Effect: Allow
549             Principal:
550               Service: lambda.amazonaws.com
551             Action:
552               - sts:AssumeRole
553       Path: '/'
554       Policies:
555         - PolicyName: LambdaCrawlerTriggerPolicy
556           PolicyDocument:
557             Version: '2012-10-17'
558             Statement:
559               - Effect: Allow
```

```
560                 Action:
561                   - glue:StartCrawler
562                 Resource:
563                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                          AccountId}:crawler/data-crawler-raw'
564                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                          AccountId}:crawler/data-crawler-preprocessed'
565                   - !Sub 'arn:aws:glue:${AWS::Region}:${AWS::
                          AccountId}:crawler/data-crawler-analytics'
566               - Effect: Allow
567                 Action:
568                   - logs:CreateLogGroup
569                   - logs:CreateLogStream
570                   - logs:PutLogEvents
571                 Resource: "*"
572
573   # Lambda Function to trigger raw crawler
574   rLambdaTriggerRawCrawler:
575     Type: AWS::Lambda::Function
576     Properties:
577       Handler: index.handler
578       Role: !GetAtt rRoleLambdaCrawlerTrigger.Arn
579       Runtime: python3.9
580       Timeout: 30
581       Code:
582         ZipFile: |
583           import boto3
584           def handler(event, context):
585               glue = boto3.client('glue')
586               glue.start_crawler(Name='data-crawler-raw')
587               return {"status": "started"}
588       Environment:
589         Variables: {}
590
591   # Lambda Function to trigger preprocessed crawler
592   rLambdaTriggerPreprocessedCrawler:
593     Type: AWS::Lambda::Function
594     Properties:
595       Handler: index.handler
596       Role: !GetAtt rRoleLambdaCrawlerTrigger.Arn
597       Runtime: python3.9
598       Timeout: 30
599       Code:
600         ZipFile: |
601           import boto3
602           def handler(event, context):
603               glue = boto3.client('glue')
604               glue.start_crawler(Name='data-crawler-preprocessed')
605               return {"status": "started"}
606       Environment:
```

```
607         Variables: {}
608
609   # Lambda Function to trigger analytics crawler
610   rLambdaTriggerAnalyticsCrawler:
611     Type: AWS::Lambda::Function
612     Properties:
613       Handler: index.handler
614       Role: !GetAtt rRoleLambdaCrawlerTrigger.Arn
615       Runtime: python3.9
616       Timeout: 30
617       Code:
618         ZipFile: |
619           import boto3
620           def handler(event, context):
621               glue = boto3.client('glue')
622               glue.start_crawler(Name='data-crawler-analytics')
623               return {"status": "started"}
624       Environment:
625         Variables: {}
626
627   # EventBridge Rule for raw bucket
628   rEventBridgeRuleRawCrawler:
629     Type: AWS::Events::Rule
630     Properties:
631       EventPattern:
632         source:
633           - "aws.s3"
634         detail-type:
635           - "Object Created"
636         resources:
637           - !GetAtt rs3BucketRaw.Arn
638       Targets:
639         - Arn: !GetAtt rLambdaTriggerRawCrawler.Arn
640           Id: "TargetFunctionV1"
641
642   # EventBridge Rule for preprocessed bucket
643   rEventBridgeRulePreprocessedCrawler:
644     Type: AWS::Events::Rule
645     Properties:
646       EventPattern:
647         source:
648           - "aws.s3"
649         detail-type:
650           - "Object Created"
651         resources:
652           - !GetAtt rs3BucketPreprocessed.Arn
653       Targets:
654         - Arn: !GetAtt rLambdaTriggerPreprocessedCrawler.Arn
655           Id: "TargetFunctionV1"
656
```

```
  # EventBridge Rule for analytics bucket
  rEventBridgeRuleAnalyticsCrawler:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
        source:
          - "aws.s3"
        detail-type:
          - "Object Created"
        resources:
          - !GetAtt rs3BucketAnalytics.Arn
      Targets:
        - Arn: !GetAtt rLambdaTriggerAnalyticsCrawler.Arn
          Id: "TargetFunctionV1"

  # Permission for EventBridge to invoke Lambda (raw)
  rLambdaInvokePermissionRaw:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref rLambdaTriggerRawCrawler
      Action: 'lambda:InvokeFunction'
      Principal: 'events.amazonaws.com'
      SourceArn: !GetAtt rEventBridgeRuleRawCrawler.Arn

  # Permission for EventBridge to invoke Lambda (preprocessed)
  rLambdaInvokePermissionPreprocessed:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref rLambdaTriggerPreprocessedCrawler
      Action: 'lambda:InvokeFunction'
      Principal: 'events.amazonaws.com'
      SourceArn: !GetAtt rEventBridgeRulePreprocessedCrawler.Arn

  # Permission for EventBridge to invoke Lambda (analytics)
  rLambdaInvokePermissionAnalytics:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref rLambdaTriggerAnalyticsCrawler
      Action: 'lambda:InvokeFunction'
      Principal: 'events.amazonaws.com'
      SourceArn: !GetAtt rEventBridgeRuleAnalyticsCrawler.Arn

  # ===== APLICACI N WEB DE RECOMENDACIONES =====

  # Bucket S3 para la aplicaci n web
  rS3BucketWebApp:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Join ["-", [!Ref pWebAppBucketName, !Ref AWS::
          AccountId]]
```

```
706       WebsiteConfiguration:
707         IndexDocument: index.html
708         ErrorDocument: index.html
709       PublicAccessBlockConfiguration:
710         BlockPublicAcls: false
711         BlockPublicPolicy: false
712         IgnorePublicAcls: false
713         RestrictPublicBuckets: false
714
715   # Pol tica del bucket para acceso p blico
716   rS3BucketPolicyWebApp:
717     Type: AWS::S3::BucketPolicy
718     Properties:
719       Bucket: !Ref rS3BucketWebApp
720       PolicyDocument:
721         Version: "2012-10-17"
722         Statement:
723           - Sid: PublicReadGetObject
724             Effect: Allow
725             Principal: "*"
726             Action: s3:GetObject
727             Resource: !Sub "${rS3BucketWebApp}/*"
728
729   # Bucket S3 para el c digo de las Lambdas
730   rS3BucketLambdaCode:
731     Type: AWS::S3::Bucket
732     Properties:
733       BucketName: !Join ["-", [!Ref pLambdaCodeBucket, !Ref AWS::
              AccountId]]
734
735   # Rol para las Lambdas de API
736   rRoleLambdaBookAPI:
737     Type: AWS::IAM::Role
738     Properties:
739       AssumeRolePolicyDocument:
740         Version: "2012-10-17"
741         Statement:
742           - Effect: Allow
743             Principal:
744               Service: lambda.amazonaws.com
745             Action: sts:AssumeRole
746       ManagedPolicyArns:
747         - arn:aws:iam::aws:policy/service-role/
              AWSLambdaBasicExecutionRole
748       Policies:
749         - PolicyName: AthenaAndS3Access
750           PolicyDocument:
751             Version: "2012-10-17"
752             Statement:
753               - Effect: Allow
```

```
                Action:
                  - athena:StartQueryExecution
                  - athena:GetQueryExecution
                  - athena:GetQueryResults
                  - athena:StopQueryExecution
                  - athena:GetWorkGroup
                Resource: "*"
              - Effect: Allow
                Action:
                  - s3:GetObject
                  - s3:ListBucket
                  - s3:PutObject
                  - s3:GetBucketLocation
                Resource:
                  - !Sub "${rs3BucketAthenaResults}/*"
                  - !GetAtt rs3BucketAthenaResults.Arn
                  - !Sub "${rs3BucketAnalytics}/*"
                  - !GetAtt rs3BucketAnalytics.Arn
              - Effect: Allow
                Action:
                  - glue:GetDatabase
                  - glue:GetTable
                  - glue:GetPartitions
                Resource: "*"

  # Lambda para recomendaciones de libros
  rLambdaBookRecommender:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: book-recommender-api
      Runtime: python3.9
      Handler: lambda_book_recommender.lambda_handler
      Role: !GetAtt rRoleLambdaBookAPI.Arn
      Timeout: 30
      MemorySize: 512
      Environment:
        Variables:
          ATHENA_DATABASE: !Ref pGlueDatabaseAnalytics
          ATHENA_RESULTS_BUCKET: !Ref rs3BucketAthenaResults
      Code:
        S3Bucket: !Ref rS3BucketLambdaCode
        S3Key: lambda_book_recommender.zip

  # Lambda para detalles de libros
  rLambdaBookDetails:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: book-details-api
      Runtime: python3.9
      Handler: lambda_book_details.lambda_handler
```

```
      Role: !GetAtt rRoleLambdaBookAPI.Arn
      Timeout: 30
      MemorySize: 256
      Environment:
        Variables:
          ATHENA_DATABASE: !Ref pGlueDatabaseAnalytics
          ATHENA_RESULTS_BUCKET: !Ref rs3BucketAthenaResults
      Code:
        S3Bucket: !Ref rS3BucketLambdaCode
        S3Key: lambda_book_details.zip

  # Lambda para b squeda de libros
  rLambdaSearchBooks:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: search-books-api
      Runtime: python3.9
      Handler: lambda_search_books.lambda_handler
      Role: !GetAtt rRoleLambdaBookAPI.Arn
      Timeout: 30
      MemorySize: 256
      Environment:
        Variables:
          ATHENA_DATABASE: !Ref pGlueDatabaseAnalytics
          ATHENA_RESULTS_BUCKET: !Ref rs3BucketAthenaResults
      Code:
        S3Bucket: !Ref rS3BucketLambdaCode
        S3Key: lambda_search_books.zip

  # API Gateway
  rAPIGateway:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: book-recommendation-api
      Description: API para el sistema de recomendaci n de libros
      EndpointConfiguration:
        Types:
          - REGIONAL

  # Recurso /recommendations
  rAPIGatewayResourceRecommendations:
    Type: AWS::ApiGateway::Resource
    Properties:
      RestApiId: !Ref rAPIGateway
      ParentId: !GetAtt rAPIGateway.RootResourceId
      PathPart: recommendations

  # M todo GET para /recommendations
  rAPIGatewayMethodRecommendationsGET:
    Type: AWS::ApiGateway::Method
```

```
    Properties:
      RestApiId: !Ref rAPIGateway
      ResourceId: !Ref rAPIGatewayResourceRecommendations
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: AWS_PROXY
        IntegrationHttpMethod: POST
        Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path
            /2015-03-31/functions/${rLambdaBookRecommender.Arn}/
            invocations"
      MethodResponses:
        - StatusCode: 200
          ResponseHeaders:
            Access-Control-Allow-Origin: "'*'"

  # Recurso /book-details
  rAPIGatewayResourceBookDetails:
    Type: AWS::ApiGateway::Resource
    Properties:
      RestApiId: !Ref rAPIGateway
      ParentId: !GetAtt rAPIGateway.RootResourceId
      PathPart: book-details

  # M todo GET para /book-details
  rAPIGatewayMethodBookDetailsGET:
    Type: AWS::ApiGateway::Method
    Properties:
      RestApiId: !Ref rAPIGateway
      ResourceId: !Ref rAPIGatewayResourceBookDetails
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: AWS_PROXY
        IntegrationHttpMethod: POST
        Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path
            /2015-03-31/functions/${rLambdaBookDetails.Arn}/
            invocations"
      MethodResponses:
        - StatusCode: 200
          ResponseHeaders:
            Access-Control-Allow-Origin: "'*'"

  # Recurso /search-books
  rAPIGatewayResourceSearchBooks:
    Type: AWS::ApiGateway::Resource
    Properties:
      RestApiId: !Ref rAPIGateway
      ParentId: !GetAtt rAPIGateway.RootResourceId
      PathPart: search-books
```

```
900
901   # M todo GET para /search-books
902   rAPIGatewayMethodSearchBooksGET:
903     Type: AWS::ApiGateway::Method
904     Properties:
905       RestApiId: !Ref rAPIGateway
906       ResourceId: !Ref rAPIGatewayResourceSearchBooks
907       HttpMethod: GET
908       AuthorizationType: NONE
909       Integration:
910         Type: AWS_PROXY
911         IntegrationHttpMethod: POST
912         Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path
              /2015-03-31/functions/${rLambdaSearchBooks.Arn}/
              invocations"
913       MethodResponses:
914         - StatusCode: 200
915           ResponseHeaders:
916             Access-Control-Allow-Origin: "'*'"
917
918   # CORS para /recommendations
919   rAPIGatewayMethodRecommendationsOPTIONS:
920     Type: AWS::ApiGateway::Method
921     Properties:
922       RestApiId: !Ref rAPIGateway
923       ResourceId: !Ref rAPIGatewayResourceRecommendations
924       HttpMethod: OPTIONS
925       AuthorizationType: NONE
926       Integration:
927         Type: MOCK
928         IntegrationResponses:
929           - StatusCode: 200
930             ResponseHeaders:
931               Access-Control-Allow-Headers: "'Content-Type,X-Amz-
                    Date,Authorization,X-Api-Key,X-Amz-Security-Token
                    '"
932               Access-Control-Allow-Methods: "'GET,OPTIONS'"
933               Access-Control-Allow-Origin: "'*'"
934             ResponseTemplates:
935               application/json: ''
936         RequestTemplates:
937           application/json: '{"statusCode": 200}'
938       MethodResponses:
939         - StatusCode: 200
940           ResponseHeaders:
941             Access-Control-Allow-Headers: true
942             Access-Control-Allow-Methods: true
943             Access-Control-Allow-Origin: true
944
945   # CORS para /book-details
```

```
  rAPIGatewayMethodBookDetailsOPTIONS:
    Type: AWS::ApiGateway::Method
    Properties:
      RestApiId: !Ref rAPIGateway
      ResourceId: !Ref rAPIGatewayResourceBookDetails
      HttpMethod: OPTIONS
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        IntegrationResponses:
          - StatusCode: 200
            ResponseHeaders:
              Access-Control-Allow-Headers: "'Content-Type,X-Amz-
                  Date,Authorization,X-Api-Key,X-Amz-Security-Token
                  '"
              Access-Control-Allow-Methods: "'GET,OPTIONS'"
              Access-Control-Allow-Origin: "'*'"
            ResponseTemplates:
              application/json: ''
        RequestTemplates:
          application/json: '{"statusCode": 200}'
      MethodResponses:
        - StatusCode: 200
          ResponseHeaders:
            Access-Control-Allow-Headers: true
            Access-Control-Allow-Methods: true
            Access-Control-Allow-Origin: true

  # CORS para /search-books
  rAPIGatewayMethodSearchBooksOPTIONS:
    Type: AWS::ApiGateway::Method
    Properties:
      RestApiId: !Ref rAPIGateway
      ResourceId: !Ref rAPIGatewayResourceSearchBooks
      HttpMethod: OPTIONS
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        IntegrationResponses:
          - StatusCode: 200
            ResponseHeaders:
              Access-Control-Allow-Headers: "'Content-Type,X-Amz-
                  Date,Authorization,X-Api-Key,X-Amz-Security-Token
                  '"
              Access-Control-Allow-Methods: "'GET,OPTIONS'"
              Access-Control-Allow-Origin: "'*'"
            ResponseTemplates:
              application/json: ''
        RequestTemplates:
          application/json: '{"statusCode": 200}'
```

```
992         MethodResponses:
993           - StatusCode: 200
994             ResponseHeaders:
995               Access-Control-Allow-Headers: true
996               Access-Control-Allow-Methods: true
997               Access-Control-Allow-Origin: true
998
999     # Permisos para que API Gateway invoque las Lambdas
1000    rLambdaPermissionRecommendations:
1001      Type: AWS::Lambda::Permission
1002      Properties:
1003        FunctionName: !Ref rLambdaBookRecommender
1004        Action: lambda:InvokeFunction
1005        Principal: apigateway.amazonaws.com
1006        SourceArn: !Sub "${rAPIGateway}/*/GET/recommendations"
1007
1008    rLambdaPermissionBookDetails:
1009      Type: AWS::Lambda::Permission
1010      Properties:
1011        FunctionName: !Ref rLambdaBookDetails
1012        Action: lambda:InvokeFunction
1013        Principal: apigateway.amazonaws.com
1014        SourceArn: !Sub "${rAPIGateway}/*/GET/book-details"
1015
1016    rLambdaPermissionSearchBooks:
1017      Type: AWS::Lambda::Permission
1018      Properties:
1019        FunctionName: !Ref rLambdaSearchBooks
1020        Action: lambda:InvokeFunction
1021        Principal: apigateway.amazonaws.com
1022        SourceArn: !Sub "${rAPIGateway}/*/GET/search-books"
1023
1024    # Deployment de la API
1025    rAPIGatewayDeployment:
1026      Type: AWS::ApiGateway::Deployment
1027      DependsOn:
1028        - rAPIGatewayMethodRecommendationsGET
1029        - rAPIGatewayMethodRecommendationsOPTIONS
1030        - rAPIGatewayMethodBookDetailsGET
1031        - rAPIGatewayMethodBookDetailsOPTIONS
1032        - rAPIGatewayMethodSearchBooksGET
1033        - rAPIGatewayMethodSearchBooksOPTIONS
1034      Properties:
1035        RestApiId: !Ref rAPIGateway
1036        StageName: prod
1037
1038  Outputs:
1039    # Outputs de buckets
1040    RawBucketName:
1041      Description: Nombre del bucket de datos crudos
```

```
    Value: !Ref rs3BucketRaw
    Export:
      Name: !Sub "${AWS::StackName}-RawBucket"

  PreprocessedBucketName:
    Description: Nombre del bucket de datos preprocesados
    Value: !Ref rs3BucketPreprocessed
    Export:
      Name: !Sub "${AWS::StackName}-PreprocessedBucket"

  AnalyticsBucketName:
    Description: Nombre del bucket de analytics
    Value: !Ref rs3BucketAnalytics
    Export:
      Name: !Sub "${AWS::StackName}-AnalyticsBucket"

  AthenaBucketName:
    Description: Nombre del bucket de resultados de Athena
    Value: !Ref rs3BucketAthenaResults
    Export:
      Name: !Sub "${AWS::StackName}-AthenaBucket"

  # Outputs de la aplicaci n web
  WebAppBucketName:
    Description: Nombre del bucket S3 para la aplicaci n web
    Value: !Ref rS3BucketWebApp
    Export:
      Name: !Sub "${AWS::StackName}-WebAppBucket"

  WebAppBucketURL:
    Description: URL del sitio web est tico en S3
    Value: !GetAtt rS3BucketWebApp.WebsiteURL
    Export:
      Name: !Sub "${AWS::StackName}-WebAppURL"

  LambdaCodeBucketName:
    Description: Nombre del bucket para c digo de lambdas
    Value: !Ref rS3BucketLambdaCode
    Export:
      Name: !Sub "${AWS::StackName}-LambdaCodeBucket"

  # Outputs de API
  APIGatewayURL:
    Description: URL base de la API
    Value: !Sub "https://${rAPIGateway}.execute-api.${AWS::Region}.
        amazonaws.com/prod"
    Export:
      Name: !Sub "${AWS::StackName}-APIURL"

  RecommendationsEndpoint:
```

```
    Description: Endpoint para recomendaciones
    Value: !Sub "https://${rAPIGateway}.execute-api.${AWS::Region}.
        amazonaws.com/prod/recommendations"

  BookDetailsEndpoint:
    Description: Endpoint para detalles de libros
    Value: !Sub "https://${rAPIGateway}.execute-api.${AWS::Region}.
        amazonaws.com/prod/book-details"

  SearchBooksEndpoint:
    Description: Endpoint para b squeda de libros
    Value: !Sub "https://${rAPIGateway}.execute-api.${AWS::Region}.
        amazonaws.com/prod/search-books"
```

8.0.2. ETL transformaciones de tipo de archivo

Transformación de JSON a Parquet

```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# ========================
# Lectura de datasets en bruto
# ========================
authors_raw = spark.read.json(f"{args['SOURCE_PATH']}")

# ========================
# Escritura en formato Parquet en el bucket de preprocesamiento
# ========================
authors_raw.write.mode("overwrite").parquet(f"{args['DEST_PATH']}")

# ========================
# Finalizaci n del Job
# ========================
job.commit()
```

Transformación de CSV a Parquet

```
 import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# ========================
# Lectura de datasets en bruto
# ========================
book_id_raw = spark.read.csv(f"{args['SOURCE_PATH']}", header=True,
     inferSchema=True)

# ========================
# Escritura en formato Parquet en el bucket de preprocesamiento
# ========================
book_id_raw.write.mode("overwrite").parquet(f"{args['DEST_PATH']}")

# ========================
# Finalizaci n del Job
# ========================
job.commit()
```

8.0.3. ETL JOBS Limpieza

ETL clean_authors

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job
from pyspark.sql.functions import col, avg, count

# Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
```

```
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

try:
    # Leer datos de autores
    print(f"Leyendo datos desde: {args['SOURCE_PATH']}")
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Limpieza de datos
    df_clean = df.dropDuplicates(["author_id"]) \
        .withColumn("average_rating", col("average_rating").cast("
            double")) \
        .withColumn("ratings_count", col("ratings_count").cast("
            integer")) \
        .withColumn("text_reviews_count", col("text_reviews_count")
            .cast("integer"))

    # Filtrar autores con al menos 2 ratings
    df_clean = df_clean.filter(col("ratings_count") >= 2)

    # Calcular estad sticas
    stats = df_clean.agg(
        avg("average_rating").alias("avg_rating"),
        avg("ratings_count").alias("avg_ratings"),
        count("*").alias("total_authors")
    ).collect()[0]

    # Rellenar valores nulos
    df_clean = df_clean.na.fill({
        "average_rating": stats["avg_rating"],
        "ratings_count": 0,
        "text_reviews_count": 0
    })

    # Guardar datos limpios
    print(f"\nGuardando datos en: {args['DEST_PATH']}")
    df_clean.write.mode("overwrite").parquet(args['DEST_PATH'])

    # Finalizar job exitosamente
    job.commit()
    print("Proceso completado exitosamente")

except Exception as e:
    print(f"Error en el proceso: {str(e)}")
    job.commit()
    sys.exit(1)
```

ETL clean_books

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job
from pyspark.sql.functions import col, avg

# Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

try:
    # Leer parquet desde la ruta especificada
    print(f"Leyendo datos desde: {args['SOURCE_PATH']}")
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Mostrar esquema y conteo inicial
    print("Esquema original:")
    df.printSchema()
    print(f"Registros iniciales: {df.count()}")

    # Limpieza de datos
    df_clean = df.dropDuplicates(["book_id"]) \
        .withColumn("average_rating", col("average_rating").cast("
            double")) \
        .withColumn("ratings_count", col("ratings_count").cast("
            integer")) \
        .withColumn("text_reviews_count", col("text_reviews_count")
            .cast("integer")) \
        .withColumn("num_pages", col("num_pages").cast("integer"))

    # Filtrar libros con al menos 5 ratings
    df_clean = df_clean.filter(col("ratings_count") >= 5)

    # Calcular promedio para valores nulos
    avg_rating = df_clean.agg(avg("average_rating")).first()[0]

    # Rellenar valores nulos
    df_clean = df_clean.na.fill({
        "average_rating": avg_rating,
        "ratings_count": 0,
        "text_reviews_count": 0,
        "num_pages": 0
```

```
    })

    # Mostrar estad sticas finales
    print(f"Registros despu s de limpieza: {df_clean.count()}")

    # Guardar datos limpios
    print(f"Guardando datos en: {args['DEST_PATH']}")
    df_clean.write.mode("overwrite").parquet(args['DEST_PATH'])

    # Finalizar job exitosamente
    job.commit()
    print("Proceso completado exitosamente")

except Exception as e:
    print(f"Error en el proceso: {str(e)}")
    job.commit()
    sys.exit(1)
```

ETL clean_book_id

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job
from pyspark.sql.functions import col

# Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

try:
    # Leer datos
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Limpieza
    df_clean = df.dropDuplicates(["book_id"]) \
        .filter(col("book_id").isNotNull()) \
        .withColumn("book_id", col("book_id").cast("integer")) \
        .filter(col("book_id") > 0)

    # Guardar datos limpios
    df_clean.write.mode("overwrite").parquet(args['DEST_PATH'])

```

```
    job.commit()
    print("Proceso completado exitosamente")

except Exception as e:
    print(f"Error en el proceso: {str(e)}")
    job.commit()
    sys.exit(1)
```

ETL clean_reads_int

```
 import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job
from pyspark.sql.functions import col

# Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

try:
    # Leer datos de reads_int
    print(f"Leyendo datos desde: {args['SOURCE_PATH']}")
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Mostrar esquema y conteo inicial
    print("Esquema original:")
    df.printSchema()
    initial_count = df.count()
    print(f"N mero de registros inicial: {initial_count}")

    # Limpieza b sica para el dataset mostrado en la imagen
    # Eliminar duplicados por usuario y libro
    df_clean = df.dropDuplicates(["user_id", "book_id"])

    # Eliminar registros con valores nulos en las columnas
        principales
    df_clean = df_clean.dropna(subset=["user_id", "book_id", "
        is_read", "rating", "is_reviewed"])

    # Filtrar valores fuera de rango
    df_clean = df_clean \
        .filter((col("rating") >= 1) & (col("rating") <= 5)) \
```

```
        .filter((col("is_read").isin(0, 1))) \
        .filter((col("is_reviewed").isin(0, 1)))

    # Estad sticas finales
    final_count = df_clean.count()
    print(f"\nRegistros despu s de limpieza: {final_count}")
    print(f"Registros eliminados: {initial_count - final_count}")
    df_clean.show(5, truncate=False)

    # Guardar datos limpios
    print(f"\nGuardando datos en: {args['DEST_PATH']}")
    df_clean.write.mode("overwrite").parquet(args['DEST_PATH'])

    job.commit()
    print("Proceso completado exitosamente")

except Exception as e:
    print(f"Error en el proceso: {str(e)}")
    job.commit()
    sys.exit(1)
```

ETL clean_reviews

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job
from pyspark.sql.functions import col, avg, when, count, desc

# Inicializaci n del Job
args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
    DEST_PATH"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

try:
    # Leer los datos desde Parquet
    print(f"Leyendo datos desde: {args['SOURCE_PATH']}")
    df = spark.read.parquet(args['SOURCE_PATH'])

    # Mostrar estad sticas iniciales
    print("Esquema original:")
    df.printSchema()
    print(f"N mero de reviews inicial: {df.count()}")

```

```
    # Calcular estadísticas de comentarios
    comment_stats = df.select(
        avg("n_comments").alias("avg_comments"),
        count(when(col("n_comments") > 0, True)).alias("reviews_with_comments"),
        count(when(col("n_comments") == 0, True)).alias("reviews_without_comments")
    ).collect()[0]

    print("\nEstadísticas de comentarios:")
    print(f"- Promedio de comentarios por review: {comment_stats['avg_comments']:.2f}")
    print(f"- Reviews con comentarios: {comment_stats['reviews_with_comments']}")
    print(f"- Reviews sin comentarios: {comment_stats['reviews_without_comments']}")

    # Limpiar y transformar datos
    df_clean = df.select(
        "*",
        when(col("n_comments").isNull(), 0).otherwise(col("n_comments")).alias("comments_count")
    )

    # Eliminar duplicados si los hay
    df_clean = df_clean.dropDuplicates()

    # Categorizar reviews por número de comentarios
    df_clean = df_clean.withColumn(
        "engagement_level",
        when(col("comments_count") == 0, "no_engagement")
        .when(col("comments_count").between(1, 5), "low_engagement")
        .when(col("comments_count").between(6, 15), "medium_engagement")
        .otherwise("high_engagement")
    )

    # Mostrar distribución de niveles de engagement
    print("\nDistribución de niveles de engagement:")
    df_clean.groupBy("engagement_level").agg(
        count("*").alias("count")
    ).orderBy(desc("count")).show()

    # Mostrar estadísticas finales
    print(f"\nReviews después de limpieza: {df_clean.count()}")

    # Guardar datos limpios
    print(f"Guardando datos en: {args['DEST_PATH']}")
    df_clean.write.mode("overwrite").parquet(args['DEST_PATH'])
```

```
68
69      # Finalizar job exitosamente
70      job.commit()
71      print("Proceso completado exitosamente")
72
73  except Exception as e:
74      print(f"Error en el proceso: {str(e)}")
75      job.commit()
76      sys.exit(1)
```

ETL clean_user_id

```
1   import sys
2   from pyspark.context import SparkContext
3   from awsglue.context import GlueContext
4   from awsglue.utils import getResolvedOptions
5   from awsglue.job import Job
6   from pyspark.sql.functions import col, trim
7
8   # Inicializaci n del Job
9   args = getResolvedOptions(sys.argv, ["JOB_NAME", "SOURCE_PATH", "
        DEST_PATH"])
10  sc = SparkContext()
11  glueContext = GlueContext(sc)
12  spark = glueContext.spark_session
13  job = Job(glueContext)
14  job.init(args["JOB_NAME"], args)
15
16  try:
17      # Leer datos
18      df = spark.read.parquet(args['SOURCE_PATH'])
19
20      # Limpieza
21      df_clean = df.withColumn("user_id", trim(col("user_id"))) \
22          .dropDuplicates(["user_id"]) \
23          .filter(col("user_id").isNotNull()) \
24          .filter(col("user_id") != "")
25
26      # Guardar datos limpios
27      df_clean.write.mode("overwrite").parquet(args["DEST_PATH"])
28      print(f"Datos limpios guardados en: {args['DEST_PATH']}")
29
30  except Exception as e:
31      print(f"Ocurri  un error: {str(e)}")
32
33  finally:
34      job.commit()
```

8.0.4. ETL Joins

```
# Detener cualquier sesion existente
%stop_session

# Configuracion inicial
%idle_timeout 2880
%glue_version 5.0
%worker_type G.1X
%number_of_workers 5

import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# Implementaci n de dataset para modelo Filtrado Colaborativo ALS
import sys
import traceback
from pyspark.sql import functions as F
from pyspark.sql.types import IntegerType, StringType, DoubleType

# Configuration
SOURCE_PATH = 's3://datalake-bucket-preprocessed-116496425672'
DEST_PATH = 's3://datalake-bucket-analytics-116496425672'

# Add logging function
def log_error(message, error):
    """Log error details"""
    error_msg = f"""
    ERROR: {message}
    Type: {type(error).__name__}
    Message: {str(error)}
    Traceback:
    {traceback.format_exc()}
    """
    print(error_msg)

try:
    # Load datasets with verification
    print("\nLoading datasets...")
    try:
```

```
        reads = spark.read.parquet(f"{SOURCE_PATH}/data_clean/reads_int_clean/")
        print(f"- Reads count: {reads.count()}")

        users = spark.read.parquet(f"{SOURCE_PATH}/data_clean/user_id_clean/")
        print(f"- Users count: {users.count()}")

        book_id_map = spark.read.parquet(f"{SOURCE_PATH}/data_clean/book_id_clean/")
        print(f"- Book ID map count: {book_id_map.count()}")

        books = spark.read.parquet(f"{SOURCE_PATH}/data_clean/books_clean/")
        print(f"- Books count: {books.count()}")

        authors = spark.read.parquet(f"{SOURCE_PATH}/data_clean/authors_clean/")
        print(f"- Authors count: {authors.count()}")
    except Exception as e:
        log_error("Error loading datasets", e)
        raise

    # Print schemas for debugging
    print("\nVerifying schemas...")
    reads.printSchema()
    users.printSchema()
    book_id_map.printSchema()
    books.printSchema()
    authors.printSchema()

    # Process data with correct types
    print("\nProcessing data with correct types...")

    # user_id_clean: cast user_id_csv to int
    users = users.withColumn("user_id_csv", F.col("user_id_csv").cast(IntegerType()))

    # book_id_clean: book_id_csv to int, book_id to string for later join
    book_id_map = book_id_map \
        .withColumn("book_id_csv", F.col("book_id_csv").cast(IntegerType())) \
        .withColumn("book_id", F.col("book_id").cast(StringType()))

    # books_clean: ensure correct types
    books = books \
        .withColumn("book_id", F.col("book_id").cast(StringType())) \
```

```
        .withColumn("average_rating", F.col("average_rating").cast(DoubleType())) \
        .withColumn("ratings_count", F.col("ratings_count").cast(IntegerType())) \
        .withColumn("num_pages", F.col("num_pages").cast(IntegerType())) \
        .withColumn("text_reviews_count", F.col("text_reviews_count").cast(IntegerType()))

    # Process joins with correct types
    print("\nProcessing joins...")
    # 1. Join reads with users
    reads_users = reads.join(
        users,
        reads.user_id == users.user_id_csv,
        "inner"
    ).select(
        reads.user_id,
        reads.book_id,
        reads.rating,
        reads.is_read,
        reads.is_reviewed
    )
    print(f"- Records after user join: {reads_users.count()}")

    # 2. Join with book_id_map
    reads_books = reads_users.join(
        book_id_map,
        reads_users.book_id == book_id_map.book_id_csv,
        "inner"
    ).select(
        reads_users["*"],
        book_id_map.book_id.alias("mapped_book_id")
    )
    print(f"- Records after book mapping join: {reads_books.count()}")

    # 3. Join with books
    reads_meta = reads_books.join(
        books,
        reads_books.mapped_book_id == books.book_id,
        "inner"
    )
    print(f"- Records after books metadata join: {reads_meta.count()}")

    # 4. Process authors array
    reads_authors = reads_meta \
        .select("*", F.explode("authors").alias("author_struct")) \
        .select(
```

```
            reads_meta.user_id.cast(IntegerType()),
            reads_meta.mapped_book_id.alias("book_id"),
            reads_meta.rating.cast(IntegerType()),
            reads_meta.is_read.cast(IntegerType()),
            reads_meta.is_reviewed.cast(IntegerType()),
            books.title,
            books.title_without_series,
            books.average_rating.cast(DoubleType()),
            books.ratings_count.cast(IntegerType()),
            books.num_pages.cast(IntegerType()),
            books.language_code,
            books.publication_year,
            books.publisher,
            F.col("author_struct.author_id").alias("author_id")
        )
    print(f"- Records after author processing: {reads_authors.count()}")

    # 5. Join with authors
    reads_full = reads_authors.join(
        authors.select(
            "author_id",
            "name",
            F.col("average_rating").alias("author_avg_rating"),
            F.col("ratings_count").alias("author_rating_count")
        ),
        "author_id",
        "left"
    )
    print(f"- Records after author join: {reads_full.count()}")

    # Create final dataset
    final_dataset = reads_full.select(
        F.col("user_id").cast(IntegerType()),
        F.col("book_id").cast(StringType()),
        F.col("rating").cast(IntegerType()),
        F.col("is_read").cast(IntegerType()),
        F.col("is_reviewed").cast(IntegerType()),
        F.col("title").alias("book_title"),
        F.col("title_without_series"),
        F.col("average_rating").cast(DoubleType()).alias("book_avg_rating"),
        F.col("ratings_count").cast(IntegerType()).alias("book_ratings_count"),
        F.col("num_pages").cast(IntegerType()),
        F.col("language_code"),
        F.col("publication_year"),
        F.col("publisher"),
        F.col("author_id"),
        F.col("name").alias("author_name"),
```

```
        F.col("author_avg_rating").cast(DoubleType()),
        F.col("author_rating_count").cast(IntegerType())
    )

    print(f"\nFinal dataset count: {final_dataset.count()}")

    # Save dataset
    print(f"\nSaving dataset to: {DEST_PATH}/recommendation_dataset/")
    final_dataset.write.mode("overwrite").parquet(f"{DEST_PATH}/recommendation_dataset/")
    final_dataset.show(5)
    print("Processing completed successfully")

except Exception as e:
    log_error("Fatal error in processing", e)
    raise

# Implementaci n de dataset para modelo filtrado basado en contenido
import sys
import traceback
from pyspark.sql import functions as F
from pyspark.sql.types import IntegerType, StringType, DoubleType

# Configuration
SOURCE_PATH = 's3://datalake-bucket-preprocessed-116496425672'
DEST_PATH = 's3://datalake-bucket-analytics-116496425672'

# Logging
def log_error(message, error):
    error_msg = f"""
    ERROR: {message}
    Type: {type(error).__name__}
    Message: {str(error)}
    Traceback:
    {traceback.format_exc()}
    """
    print(error_msg)

try:
    print("\nLoading datasets...")
    books = spark.read.parquet(f"{SOURCE_PATH}/data_clean/books_clean/")
    authors = spark.read.parquet(f"{SOURCE_PATH}/data_clean/authors_clean/")
    interactions = spark.read.parquet(f"{SOURCE_PATH}/data_clean/reads_int_clean/")

    print("- Data loaded successfully")
```

```
223
224     # Cast and cleanup
225     books = books \
226         .withColumn("book_id", F.col("book_id").cast(StringType())) \
227         .withColumn("average_rating", F.col("average_rating").cast(DoubleType())) \
228         .withColumn("ratings_count", F.col("ratings_count").cast(IntegerType())) \
229         .withColumn("num_pages", F.col("num_pages").cast(IntegerType())) \
230         .withColumn("publication_year", F.col("publication_year").cast(IntegerType()))
231
232     authors = authors \
233         .withColumn("author_id", F.col("author_id").cast(StringType())) \
234         .withColumn("average_rating", F.col("average_rating").cast(DoubleType())) \
235         .withColumn("ratings_count", F.col("ratings_count").cast(IntegerType()))
236
237     interactions = interactions \
238         .withColumn("user_id", F.col("user_id").cast(StringType())) \
239         .withColumn("rating", F.col("rating").cast(DoubleType()))
240
241     # Explode authors from books
242     print("\nProcessing joins...")
243     books_meta = books.select(
244         F.col("book_id"),
245         F.col("title"),
246         F.col("title_without_series"),
247         F.col("description"),
248         F.col("average_rating").alias("book_avg_rating"),
249         F.col("ratings_count").alias("book_ratings_count"),
250         F.col("num_pages"),
251         F.col("language_code"),
252         F.col("publication_year"),
253         F.col("publisher"),
254         F.col("popular_shelves"),
255         F.explode("authors").alias("author_struct")
256     )
257
258     print(f"- Records after books explode: {books_meta.count()}")
259
260     # Add author info
261     books_authors = books_meta.join(
262         authors.select(
263             "author_id",
```

```
            "name",
            F.col("average_rating").alias("author_avg_rating"),
            F.col("ratings_count").alias("author_rating_count")
        ),
        books_meta.author_struct.author_id == authors.author_id,
        "left"
    ).select(
        F.col("book_id"),
        F.col("title"),
        F.col("title_without_series"),
        F.col("description"),
        F.col("book_avg_rating"),
        F.col("book_ratings_count"),
        F.col("num_pages"),
        F.col("language_code"),
        F.col("publication_year"),
        F.col("publisher"),
        F.col("popular_shelves"),
        F.col("author_id"),
        F.col("name").alias("author_name"),
        F.col("author_avg_rating"),
        F.col("author_rating_count")
    )

    print(f"- Records after author join: {books_authors.count()}")

    # Add user ratings
    final_dataset = books_authors.join(
        interactions.select("user_id", "book_id", "rating"),
        "book_id",
        "left"
    ).select(
        F.col("book_id"),
        F.col("title"),
        F.col("title_without_series"),
        F.col("description"),
        F.col("book_avg_rating").cast(DoubleType()),
        F.col("book_ratings_count").cast(IntegerType()),
        F.col("num_pages").cast(IntegerType()),
        F.col("language_code"),
        F.col("publication_year"),
        F.col("publisher"),
        F.col("popular_shelves"),
        F.col("author_id"),
        F.col("author_name"),
        F.col("author_avg_rating").cast(DoubleType()),
        F.col("author_rating_count").cast(IntegerType()),
        F.col("user_id"),
        F.col("rating").cast(DoubleType())
    )
```

```
314
315     print(f"\nFinal dataset count: {final_dataset.count()}")
316
317     # Save dataset
318     print(f"\nSaving dataset to: {DEST_PATH}/content_based_dataset/
            ")
319     final_dataset.write.mode("overwrite").parquet(f"{DEST_PATH}/
            content_based_dataset/")
320     final_dataset.show(5)
321     print("Content-based dataset created successfully")
322
323 except Exception as e:
324     log_error("Fatal error in processing", e)
325     raise
326
327 # Sample Data
328
329 import sys
330 import traceback
331 from pyspark.sql import functions as F
332 from pyspark.sql.types import IntegerType, StringType, DoubleType
333
334 # Configuration
335 SOURCE_PATH = 's3://datalake-bucket-analytics-116496425672/
        content_based_dataset'
336 DEST_PATH = 's3://datalake-bucket-analytics-116496425672/
        content_based_dataset_sample'
337 SAMPLE_SIZE = 10000
338 MIN_RATING = 3.0
339
340 # Logging
341 def log_error(message, error):
342     error_msg = f"""
343     ERROR: {message}
344     Type: {type(error).__name__}
345     Message: {str(error)}
346     Traceback:
347     {traceback.format_exc()}
348     """
349     print(error_msg)
350
351 try:
352     print("\nLoading content-based dataset...")
353     df = spark.read.parquet(SOURCE_PATH)
354     initial_count = df.count()
355     print(f"- Initial records: {initial_count:,}")
356
357     # Filter by rating and create sample
358     print("\nFiltering and sampling data...")
359     filtered_df = df.filter(
```

```
        (F.col("book_avg_rating") >= MIN_RATING) &
        (F.col("book_ratings_count") > 100)  # Add minimum ratings threshold
    )

    filtered_count = filtered_df.count()
    print(f"- Records after filtering: {filtered_count:,}")

    # Sample records
    sampled_df = filtered_df.orderBy(
        F.desc("book_ratings_count"),  # Prioritize popular books
        F.desc("book_avg_rating")      # Then by rating
    ).limit(SAMPLE_SIZE)

    final_count = sampled_df.count()
    print(f"- Final sample size: {final_count:,}")

    # Calculate statistics
    stats = sampled_df.agg(
        F.round(F.avg("book_avg_rating"), 2).alias("avg_rating"),
        F.round(F.avg("book_ratings_count"), 0).alias("avg_ratings_count"),
        F.countDistinct("author_id").alias("unique_authors")
    ).collect()[0]

    print("\nSample Statistics:")
    print(f"- Average Rating: {stats['avg_rating']}")
    print(f"- Average Ratings Count: {stats['avg_ratings_count']:,.0f}")
    print(f"- Unique Authors: {stats['unique_authors']:,}")

    # Save sampled dataset
    print(f"\nSaving sampled dataset to: {DEST_PATH}")
    sampled_df.write.mode("overwrite").parquet(DEST_PATH)

    # Show sample of results
    print("\nSample of selected books:")
    sampled_df.select(
        "book_id", "title", "author_name",
        "book_avg_rating", "book_ratings_count"
    ).orderBy(
        F.desc("book_avg_rating")
    ).show(5, truncate=False)

    print("Sample dataset created successfully")

except Exception as e:
    log_error("Fatal error in processing", e)
    raise
```

```
# Catalogo de libros para frontend

import sys
import traceback
from pyspark.sql import functions as F
from pyspark.sql.types import IntegerType, StringType, DoubleType

# Configuration
SOURCE_PATH = 's3://datalake-bucket-preprocessed-116496425672'
DEST_PATH = 's3://datalake-bucket-analytics-116496425672'

# Logging
def log_error(message, error):
    error_msg = f"""
    ERROR: {message}
    Type: {type(error).__name__}
    Message: {str(error)}
    Traceback:
    {traceback.format_exc()}
    """
    print(error_msg)

def log_step(message, df=None):
    print(f"\n{message}")
    if df is not None:
        try:
            print(f"Records: {df.count():,}")
        except:
            print("Error getting count")

try:
    log_step("Loading existing catalog...")

    # Load existing catalog
    existing_catalog = spark.read.parquet(f"{DEST_PATH}/
        book_catalog/")
    log_step("Existing catalog loaded", existing_catalog)

    log_step("Loading books dataset for image_url and isbn...")

    # Load books dataset to get image_url and isbn
    books = spark.read.parquet(f"{SOURCE_PATH}/data_clean/
        books_clean/")

    # Select only the fields we need
    books_additional = books.select(
        F.col("book_id").cast(StringType()),
        F.col("image_url"),
        F.col("isbn"),
        F.col("isbn13")
```

```
    ).filter(F.col("book_id").isNotNull())

    log_step("Additional book fields selected", books_additional)

    log_step("Joining catalog with additional fields...")

    # Join existing catalog with additional fields
    updated_catalog = existing_catalog.join(
        F.broadcast(books_additional),
        "book_id",
        "left"
    ).select(
        # Existing fields in order
        F.col("book_id"),
        F.col("title"),
        F.col("title_without_series"),
        F.col("description"),
        F.col("book_avg_rating"),
        F.col("book_ratings_count"),
        F.col("num_pages"),
        F.col("language_code"),
        F.col("publication_year"),
        F.col("publisher"),
        F.col("popular_shelves"),
        F.col("author_id"),
        F.col("author_name"),
        F.col("author_avg_rating"),
        F.col("author_rating_count"),
        F.col("user_id"),
        F.col("rating"),

        # New fields
        F.col("image_url"),
        F.col("isbn"),
        F.col("isbn13")
    )

    log_step("Catalog updated with new fields", updated_catalog)

    # Show schema to verify
    print("\nUpdated schema:")
    updated_catalog.printSchema()

    # Save updated catalog
    log_step(f"Saving updated catalog to: {DEST_PATH}/book_catalog/
        ")
    updated_catalog.write.mode("overwrite").parquet(f"{DEST_PATH}/
        book_catalog_front/")

    # Show sample with new fields
```

```
    log_step("Sample data with new fields:")
    updated_catalog.select(
        "book_id", "title", "author_name", "image_url", "isbn"
    ).show(5, truncate=False)

    # Show statistics
    total_books = updated_catalog.select("book_id").distinct().
        count()
    books_with_images = updated_catalog.filter(
        F.col("image_url").isNotNull() & (F.length(F.col("image_url
            ")) > 10)
    ).select("book_id").distinct().count()
    books_with_isbn = updated_catalog.filter(
        F.col("isbn").isNotNull() & (F.length(F.col("isbn")) > 5)
    ).select("book_id").distinct().count()

    print(f"\  n     STATISTICS:")
    print(f"Total books: {total_books:,}")
    print(f"Books with images: {books_with_images:,} ({
        books_with_images/total_books*100:.1f}%)")
    print(f"Books with ISBN: {books_with_isbn:,} ({books_with_isbn/
        total_books*100:.1f}%)")

    print("\nCatalog updated successfully with image_url")

except Exception as e:
    log_error("Fatal error updating catalog", e)
    raise
```

8.0.5. IAM Roles

AWSServiceRoleForLakeFormationDataAccess

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LakeFormationDataAccessPermissionsForS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::datalake-bucket-preprocessed
                    -116496425672/*",
                "arn:aws:s3:::lake-formation-18082025/*",
```

```
                "arn:aws:s3:::datalake-bucket-raw-116496425672/*"
            ]
        },
        {
            "Sid": "
               LakeFormationDataAccessPermissionsForS3ListBucket",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::datalake-bucket-preprocessed
                   -116496425672",
                "arn:aws:s3:::lake-formation-18082025",
                "arn:aws:s3:::datalake-bucket-raw-116496425672"
            ]
        }
    ]
}
```

Glue Role

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*/*",
                "arn:aws:s3:::*/*aws-glue-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::crawler-public*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Condition": {
```

```
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws-glue-service-resource"
                    ]
                }
            },
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        }
    ]
}
```

Crawler Role

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::datalake-bucket-raw-116496425672",
                "arn:aws:s3:::datalake-bucket-raw-116496425672/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::datalake-bucket-preprocessed
                   -116496425672",
                "arn:aws:s3:::datalake-bucket-preprocessed
                   -116496425672/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
```

```
31                  "s3:ListBucket",
32                  "s3:GetBucketLocation"
33              ],
34              "Resource": [
35                  "arn:aws:s3:::datalake-bucket-analytics
                        -116496425672",
36                  "arn:aws:s3:::datalake-bucket-analytics
                        -116496425672/*"
37              ],
38              "Effect": "Allow"
39          },
40          {
41              "Action": [
42                  "glue:GetDatabase",
43                  "glue:CreateTable",
44                  "glue:UpdateTable",
45                  "glue:GetTable",
46                  "glue:GetTables",
47                  "glue:BatchCreatePartition",
48                  "glue:BatchDeletePartition"
49              ],
50              "Resource": [
51                  "arn:aws:glue:eu-central-1:116496425672:catalog",
52                  "arn:aws:glue:eu-central-1:116496425672:database/
                        datalake-db-raw",
53                  "arn:aws:glue:eu-central-1:116496425672:table/
                        datalake-db-raw/*",
54                  "arn:aws:glue:eu-central-1:116496425672:database/
                        datalake-db-preprocessed",
55                  "arn:aws:glue:eu-central-1:116496425672:table/
                        datalake-db-preprocessed/*"
56              ],
57              "Effect": "Allow"
58          },
59          {
60              "Action": [
61                  "glue:GetDatabase",
62                  "glue:CreateTable",
63                  "glue:UpdateTable",
64                  "glue:GetTable",
65                  "glue:GetTables",
66                  "glue:BatchCreatePartition",
67                  "glue:BatchDeletePartition"
68              ],
69              "Resource": [
70                  "arn:aws:glue:eu-central-1:116496425672:database/
                        datalake-db-analytics",
71                  "arn:aws:glue:eu-central-1:116496425672:table/
                        datalake-db-analytics/*"
72              ],
```

```
            "Effect": "Allow"
        }
    ]
}
```

QuickSigth Role

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:BatchGetQueryExecution",
                "athena:CancelQueryExecution",
                "athena:GetCatalogs",
                "athena:GetExecutionEngine",
                "athena:GetExecutionEngines",
                "athena:GetNamespace",
                "athena:GetNamespaces",
                "athena:GetQueryExecution",
                "athena:GetQueryExecutions",
                "athena:GetQueryResults",
                "athena:GetQueryResultsStream",
                "athena:GetTable",
                "athena:GetTables",
                "athena:ListQueryExecutions",
                "athena:RunQuery",
                "athena:StartQueryExecution",
                "athena:StopQueryExecution",
                "athena:ListWorkGroups",
                "athena:ListEngineVersions",
                "athena:GetWorkGroup",
                "athena:GetDataCatalog",
                "athena:GetDatabase",
                "athena:GetTableMetadata",
                "athena:ListDataCatalogs",
                "athena:ListDatabases",
                "athena:ListTableMetadata"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
```

```
                "glue:GetCatalog",
                "glue:GetCatalogs",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-athena-query-results-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

Lambda Role

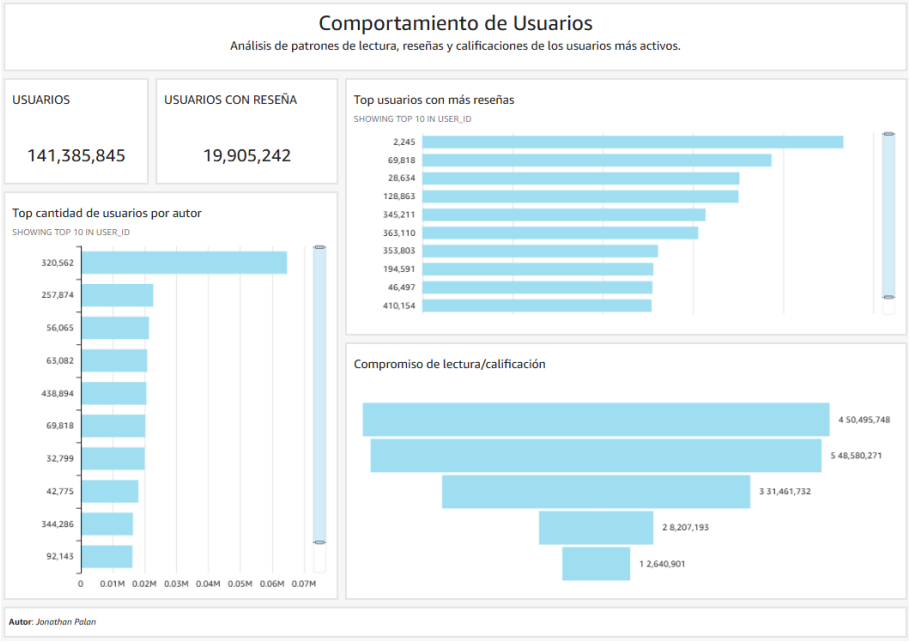
```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:eu-central-1:116496425672:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:eu-central-1:116496425672:log-group:/
                   aws/lambda/book-recommender-lambda:*"
            ]
        }
    ]
}
```

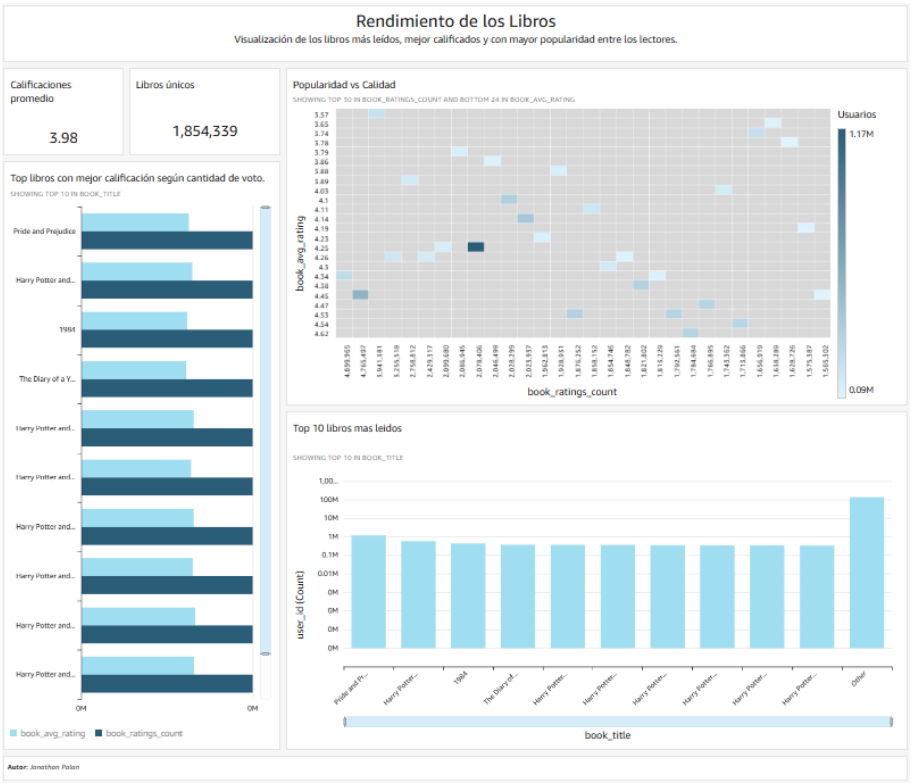
Redshift Role

```
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetBucketAcl",
                "s3:GetBucketCors",
                "s3:GetEncryptionConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutBucketAcl",
                "s3:PutBucketCors",
                "s3:DeleteObject",
```

```
                "s3:AbortMultipartUpload",
                "s3:CreateBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::datalake-bucket-analytics-116496425672/*",
                "arn:aws:s3:::datalake-bucket-analytics-116496425672"
            ]
        }
    ]
}
```

8.0.6. QuickSight dashboards




Rendimiento de los Libros
Visualización de los libros más leídos, mejor calificados y con mayor popularidad entre los lectores.
Calificaciones promedio
3.98
Libros únicos
1,854,339
Top libros con mejor calificación según cantidad de voto.
SHOWING TOP 10 IN BOOK_TITLE
Pride and Prejudice
Harry Potter and...
1984
The Diary of a Y...
Harry Potter and...
Harry Potter and...
Harry Potter and...
Harry Potter and...
Harry Potter and...
Harry Potter and...
0M
0M
book_avg_rating
book_ratings_count
Popularidad vs Calidad
SHOWING TOP 50 IN BOOK_RATINGS_COUNT AND BOTTOM 24 IN BOOK_AVG_RATING
Usuarios
1.17M
0.09M
book_avg_rating
book_ratings_count
Top 10 libros mas leidos
SHOWING TOP 10 IN BOOK_TITLE
user_id (Count)
book_title
Autor: Jonathan Palan

8.0.7. Script usado en SageMaker Notebook para modelo de recomendaciones

```
# Detener cualquier sesion existente
%stop_session
# Configuracion Glue session
%%configure
{
    "glue_version": "3.0",
    "worker_type": "G.1X",
    "number_of_workers": 10,
    "idle_timeout": 2880
}
# Librerias
import boto3
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import (
    col, concat_ws, udf, lower, regexp_replace,
    explode, collect_list, when, count
)
from pyspark.ml.feature import Tokenizer, StopWordsRemover,
   HashingTF, IDF
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.sql.types import ArrayType, StringType, DoubleType
from scipy.spatial.distance import cosine
from sklearn.model_selection import train_test_split

print("Librerias importadas correctamente")

# TF-IDF Features
def create_tfidf_features(df):
    """Create TF-IDF features"""
    tokenizer = Tokenizer(inputCol="text_features_clean", outputCol
       ="words")
    wordsData = tokenizer.transform(df)

    remover = StopWordsRemover(inputCol="words", outputCol="
       filtered")
    filtered = remover.transform(wordsData)

    hashingTF = HashingTF(inputCol="filtered", outputCol="
       raw_features", numFeatures=10000)
    featurizedData = hashingTF.transform(filtered)

    idf = IDF(inputCol="raw_features", outputCol="features")
    idfModel = idf.fit(featurizedData)
    return idfModel.transform(featurizedData)

# Carga de datos y Funciones de preprocesamiento

def load_sample_dataset(spark, input_path, sample_size=10000):
    """Load a sample of the dataset"""
    full_df = spark.read.parquet(input_path)
    return full_df.sample(False, sample_size/full_df.count(), seed
       =42)

def preprocess_data(df):
    """Preprocess text data"""
    # Extract shelf names
    df_shelves = df.select(
        "book_id",
        explode("popular_shelves").alias("shelf")
    ).select(
        "book_id",
```

```
        col("shelf.name").alias("shelf_name")
    ).groupBy("book_id").agg(
        concat_ws(" ", collect_list("shelf_name")).alias("
            shelf_features")
    )

    # Join and create text features
    df = df.join(df_shelves, "book_id").withColumn(
        "text_features_raw",
        concat_ws(" ",
            col("title"),
            col("title_without_series"),
            col("description"),
            col("author_name"),
            col("publisher"),
            col("shelf_features")
        )
    )

    return df.withColumn(
        "text_features_clean",
        lower(regexp_replace(col("text_features_raw"), "[^a-zA-Z\\s
            ]", ""))
    )

# Load and process data
input_path = "s3://datalake-bucket-analytics-116496425672/
    content_based_dataset/"
df = load_sample_dataset(spark, input_path)
print(f"Loaded {df.count():,} records")

# Process data
df_preprocessed = preprocess_data(df)
df_tfidf = create_tfidf_features(df_preprocessed)
print("Data processing completed")

# Display sample data
print("\nSample processed data:")
df_tfidf.select("book_id", "title", "author_name", "features").show
    (2)

# Modelo hibrido
class HybridRecommender:
    def __init__(self, content_weight=0.3):
        self.content_weight = content_weight
        self.collaborative_weight = 1 - content_weight
        self.book_id_mapping = {}
        self.reverse_book_mapping = {}

    def train(self, df, df_tfidf):
```

```
        """Train both collaborative and content-based models"""
        # Create book ID mapping with validation
        distinct_books = df.select("book_id").filter(
            col("book_id").isNotNull()
        ).distinct().collect()

        print(f"Found {len(distinct_books)} distinct books")

        # Create safe mappings with string conversion
        self.book_id_mapping = {
            str(row['book_id']): float(idx)
            for idx, row in enumerate(distinct_books)
        }
        self.reverse_book_mapping = {
            float(idx): str(book_id)
            for book_id, idx in self.book_id_mapping.items()
        }

        # Prepare ratings with validation
        ratings_df = df.filter(
            col("user_id").isNotNull() &
            col("book_id").isNotNull() &
            col("rating").isNotNull()
        ).select(
            col("user_id").cast("integer"),
            "book_id",
            col("rating").cast("double")
        ).dropDuplicates()

        print(f"Found {ratings_df.count()} valid ratings")

        # Safe mapping function
        def safe_map_book_id(book_id):
            if book_id is None:
                return None
            return self.book_id_mapping.get(str(book_id))

        # Create UDF for mapping
        safe_map_udf = udf(safe_map_book_id, DoubleType())

        # Map book IDs safely
        ratings_mapped = ratings_df.withColumn(
            "book_id_mapped",
            safe_map_udf(col("book_id"))
        ).filter(
            col("book_id_mapped").isNotNull()
        ).select(
            "user_id",
            col("book_id_mapped").alias("book_id"),
            "rating"
```

```
        ).cache()  # Cache for better performance

        print(f"Mapping complete with {ratings_mapped.count()} ratings")

        # Train ALS
        self.als = ALS(
            maxIter=10,
            regParam=0.01,
            userCol="user_id",
            itemCol="book_id",
            ratingCol="rating",
            coldStartStrategy="drop",
            nonnegative=True
        )
        self.model_als = self.als.fit(ratings_mapped)

        # Create content features map with validation
        self.features_map = {
            str(row['book_id']): row['features'].toArray()
            for row in df_tfidf.select('book_id', 'features').collect()
            if row['book_id'] is not None and row['features'] is not None
        }

        print(f"Created features map with {len(self.features_map)} books")

        self.df = df
        return self

    def recommend_by_book(self, book_id, n=5):
        """Content-based recommendations"""
        book_id = str(book_id)
        if book_id not in self.features_map:
            print(f"Warning: Book {book_id} not found in features map")
            return self.df.limit(0)

        similarities = [
            (other_id, 1 - cosine(self.features_map[book_id], feat))
            for other_id, feat in self.features_map.items()
            if other_id != book_id
        ]

        top_books = sorted(similarities, key=lambda x: x[1], reverse=True)[:n]

```

```
202         return self.df.filter(
203             col("book_id").isin([b[0] for b in top_books])
204         ).select(
205             "book_id", "title", "author_name", "book_avg_rating"
206         )
207 
208     def recommend_by_user(self, user_id, n=5, spark_session=None):
209         """Collaborative filtering recommendations"""
210         if spark_session is None:
211             print("Warning: No Spark session provided")
212             return self.df.limit(0)
213 
214         try:
215             user_id = int(user_id)
216             user_df = spark_session.createDataFrame(
217                 [(user_id,)],
218                 ["user_id"]
219             )
220 
221             user_recs = self.model_als.recommendForUserSubset(
222                 user_df,
223                 n
224             ).collect()[0].recommendations
225 
226             book_ids = [
227                 self.reverse_book_mapping.get(float(rec.book_id))
228                 for rec in user_recs
229                 if rec.book_id is not None
230             ]
231 
232             return self.df.filter(
233                 col("book_id").isin(book_ids)
234             ).select(
235                 "book_id", "title", "author_name", "book_avg_rating
                        "
236             )
237         except Exception as e:
238             print(f"Error in user recommendations: {str(e)}")
239             return self.df.limit(0)
240 
241     def recommend_hybrid(self, user_id, book_id, n=5, spark_session
            =None):
242         """Hybrid recommendations"""
243         book_id = str(book_id)
244         try:
245             numeric_book_id = self.book_id_mapping.get(book_id)
246             if numeric_book_id is None:
247                 print(f"Warning: Book {book_id} not found in
                        mapping")
248                 return self.df.limit(0)
```

```
249
250             user_df = spark_session.createDataFrame(
251                 [(int(user_id),)],
252                 ["user_id"]
253             )
254
255             user_recs = self.model_als.recommendForUserSubset(
256                 user_df,
257                 20
258             ).collect()[0].recommendations
259
260             content_sims = []
261             if book_id in self.features_map:
262                 content_sims = [
263                     (bid, 1 - cosine(self.features_map[book_id],
                        feat))
264                     for bid, feat in self.features_map.items()
265                     if bid != book_id
266                 ]
267
268             final_scores = {}
269             for rec in user_recs:
270                 orig_book_id = self.reverse_book_mapping.get(float(
                    rec.book_id))
271                 if orig_book_id:
272                     cf_score = float(rec.rating)
273
274                     cb_score = 0.0
275                     for bid, sim in content_sims:
276                         if bid == orig_book_id:
277                             cb_score = sim
278                             break
279
280                     final_scores[orig_book_id] = (
281                         self.collaborative_weight * cf_score +
282                         self.content_weight * cb_score
283                     )
284
285             top_books = sorted(
286                 final_scores.items(),
287                 key=lambda x: x[1],
288                 reverse=True
289             )[:n]
290
291             return self.df.filter(
292                 col("book_id").isin([b[0] for b in top_books])
293             ).select(
294                 "book_id", "title", "author_name", "book_avg_rating
                    "
295             )
```

```
        except Exception as e:
            print(f"Error in hybrid recommendations: {str(e)}")
            return self.df.limit(0)

# Create and train recommender
print("Initializing recommender...")
recommender = HybridRecommender(content_weight=0.3)
recommender.train(df, df_tfidf)
print("Model trained successfully")

# Test
# Get test book and user
test_sample = df.select("user_id", "book_id", "title", "author_name
    ").first()
print(f"Testing recommendations for book: {test_sample.title} by {
    test_sample.author_name}")

# Test all recommendation types
print("\n1. Content-based recommendations:")
content_recs = recommender.recommend_by_book(test_sample.book_id)
content_recs.show()

print("\n2. Collaborative filtering recommendations:")
collab_recs = recommender.recommend_by_user(
    test_sample.user_id,
    n=5,
    spark_session=spark
)
collab_recs.show()

print("\n3. Hybrid recommendations:")
hybrid_recs = recommender.recommend_hybrid(
    test_sample.user_id,
    test_sample.book_id,
    n=5,
    spark_session=spark
)
hybrid_recs.show()
```

Evaluación del modelo

```
    def evaluate_recommendations_optimized(recommender, test_data,
        spark_session):
    """Optimized model evaluation with null value handling"""

    # Convert test data to DataFrame with validation
    valid_test_data = [
        row for row in test_data
```

```
        if row.user_id is not None and row.book_id is not None and
            row.rating is not None
    ]

    if not valid_test_data:
        print("No valid test data found")
        return None

    test_df = spark_session.createDataFrame(valid_test_data)

    # Batch process recommendations
    batch_size = 1000
    predictions = []

    # Process data in batches
    test_data_list = test_df.collect()

    for i in range(0, len(test_data_list), batch_size):
        batch = test_data_list[i:i + batch_size]
        batch_predictions = []

        # Filter out None values and convert to int safely
        valid_users = []
        user_mapping = {}  # Map original user_id to validated
            user_id

        for row in batch:
            try:
                if row.user_id is not None:
                    user_id_int = int(row.user_id)
                    valid_users.append(user_id_int)
                    user_mapping[user_id_int] = row
            except (ValueError, TypeError):
                continue

        if not valid_users:
            continue

        # Get recommendations for valid users
        try:
            user_df = spark_session.createDataFrame(
                [(uid,) for uid in valid_users],
                ["user_id"]
            )

            batch_recs = recommender.model_als.
                recommendForUserSubset(
                user_df,
                1
            ).collect()
```

```
54
55              # Process batch results
56              for user_rec in batch_recs:
57                  try:
58                      user_id = user_rec.user_id
59                      if user_id in user_mapping and user_rec.
                            recommendations:
60                          original_row = user_mapping[user_id]
61                          pred_rating = user_rec.recommendations[0].
                                rating
62                          actual_rating = float(original_row.rating)
63                          batch_predictions.append((actual_rating,
                                pred_rating))
64                  except Exception as e:
65                      continue
66
67          except Exception as e:
68              print(f"Error processing batch: {e}")
69              continue
70
71          predictions.extend(batch_predictions)
72
73          # Print progress
74          print(f"Processed {len(predictions)} predictions so far..."
                )
75
76      if predictions:
77          mae = np.mean([abs(actual - pred) for actual, pred in
                predictions])
78          rmse = np.sqrt(np.mean([(actual - pred) ** 2 for actual,
                pred in predictions]))
79          return {
80              'mae': mae,
81              'rmse': rmse,
82              'coverage': len(predictions) / len(valid_test_data)
83          }
84      return None
85
86  # Dividir datos
87  # Split data for evaluation
88  print("Splitting data for evaluation...")
89
90  # Split data into train and test sets
91  train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)
92
93  print(f"Training data: {train_data.count():,} records")
94  print(f"Test data: {test_data.count():,} records")
95
96  # Cache the test data for better performance
97  test_data = test_data.cache()
```

```
print("Data split completed")

# Evaluar modelo
print("Starting model evaluation...")
metrics = evaluate_recommendations_optimized(
    recommender,
    test_data.collect(),
    spark_session=spark
)

if metrics:
    print("\nModel Metrics:")
    print(f"MAE: {metrics['mae']:.4f}")
    print(f"RMSE: {metrics['rmse']:.4f}")
    print(f"Coverage: {metrics['coverage']:.2%}")
else:
    print("No metrics could be calculated")

# Guardar modelo
print("\nSaving model artifacts...")
output_path = "s3://datalake-bucket-analytics-116496425672/
    hybrid_model/"
df_tfidf.write.mode("overwrite").parquet(output_path + "features")

print("Features saved successfully")
```

Guardar artefactos y features del modelo

```
import pickle
import json

print("\nSaving complete model artifacts...")
output_path = "s3://datalake-bucket-analytics-116496425672/
    hybrid_model/"

# Save book mappings
mappings = {
    'book_id_mapping': recommender.book_id_mapping,
    'reverse_book_mapping': {str(k): v for k, v in recommender.
        reverse_book_mapping.items()},
    'features_map_keys': list(recommender.features_map.keys())
}

# Save to local first, then upload
import json
with open("/tmp/book_mappings.json", "w") as f:
    json.dump(mappings, f)

# Upload to S3
```

```
import boto3
s3_client = boto3.client('s3')
s3_client.upload_file(
    "/tmp/book_mappings.json",
    "datalake-bucket-analytics-116496425672",
    "hybrid_model/book_mappings.json"
)

# 3. Save content features for fast lookup
features_dict = {k: v.tolist() for k, v in recommender.features_map
    .items()}
with open("/tmp/content_features.json", "w") as f:
    json.dump(features_dict, f)

s3_client.upload_file(
    "/tmp/content_features.json",
    "datalake-bucket-analytics-116496425672",
    "hybrid_model/content_features.json"
)
```

8.0.8. Script para Amazon Clarify

```
# An lisis de Bias con SageMaker Clarify

import sagemaker
from sagemaker import clarify
import pandas as pd
import boto3
import json
from datetime import datetime
import numpy as np
import io

print("xx AN LISIS DE BIAS CON SAGEMAKER CLARIFY")
print("=" * 70)

# Configuraci n
session = sagemaker.Session()
role = sagemaker.get_execution_role()
region = session.boto_region_name
bucket = "datalake-bucket-analytics-116496425672"

TIMESTAMP = "20250823_172430"

print(f"xx Usando datos exportados con timestamp: {TIMESTAMP}")

# Rutas de datos
data_prefix = f"clarify_input_data/sample_{TIMESTAMP}"
```

```
results_prefix = f"clarify_reports/clarify_results_{TIMESTAMP}"

# Configurar paths
input_path = f"s3://{bucket}/{data_prefix}"
output_path = f"s3://{bucket}/{results_prefix}"

print(f"xx Datos de entrada: {input_path}")
print(f"xx Resultados: {output_path}")

# 1. Funci n robusta para leer CSV con errores
def read_csv_robust(file_path, max_attempts=3):
    """Leer CSV con manejo robusto de errores"""

    print(f"xx Intentando leer CSV: {file_path}")

    # Lectura normal
    try:
        df = pd.read_csv(file_path)
        print(f"Lectura exitosa - Estrategia 1")
        return df
    except Exception as e:
        print(f"Estrategia 1 fall : {str(e)[:100]}...")

    # Con error_bad_lines=False (versiones antiguas de pandas)
    try:
        df = pd.read_csv(file_path, on_bad_lines='skip')
        print(f"Lectura exitosa - Estrategia 2 (saltando l neas
            problem ticas)")
        return df
    except Exception as e:
        print(f"Estrategia 2 fall : {str(e)[:100]}...")

    # Leer como texto y limpiar
    try:
        print(f"xx Estrategia 3: Limpieza manual del CSV")

        with open(file_path, 'r', encoding='utf-8') as f:
            content = f.read()

        # Obtener header
        lines = content.split('\n')
        header = lines[0]
        expected_cols = len(header.split(','))

        print(f"xx Columnas esperadas: {expected_cols}")

        # Filtrar l neas v lidas
        valid_lines = [header]
        problematic_lines = 0

```

```
        for i, line in enumerate(lines[1:], 1):
            if line.strip():
                cols = line.split(',')
                if len(cols) == expected_cols:
                    valid_lines.append(line)
                else:
                    problematic_lines += 1
                    if problematic_lines <= 5:
                        print(f"L nea {i}: {len(cols)} columnas (
                            esperadas {expected_cols})")

        print(f"xx L neas v lidas: {len(valid_lines)-1},
            Problem ticas: {problematic_lines}")

        # Crear DataFrame desde l neas v lidas
        cleaned_content = '\n'.join(valid_lines)
        df = pd.read_csv(io.StringIO(cleaned_content))

        print(f"Lectura exitosa - Estrategia 3")
        return df

    except Exception as e:
        print(f"Estrategia 3 fall : {str(e)[:100]}...")

    # Leer solo las primeras N l neas para diagn stico
    try:
        print(f"xx Estrategia 4: Muestra diagn stica")

        df_sample = pd.read_csv(file_path, nrows=100)
        print(f"Solo se pudieron leer 100 l neas de muestra")
        return df_sample

    except Exception as e:
        print(f"Todas las estrategias fallaron: {e}")
        raise Exception("No se pudo leer el archivo CSV")

# 2. Leer datos para an lisis directo
print("\nxx Descargando y leyendo datos...")

s3_client = boto3.client('s3')

try:
    # Encontrar el archivo CSV
    response = s3_client.list_objects_v2(
        Bucket=bucket,
        Prefix=data_prefix
    )

    csv_files = [obj['Key'] for obj in response.get('Contents', [])
        if obj['Key'].endswith('.csv')]
```

```
123
124     if csv_files:
125         csv_file = csv_files[0]  # Tomar el primer CSV
126         print(f"xx Encontrado: {csv_file}")
127
128         # Descargar archivo
129         local_csv = '/tmp/clarify_data.csv'
130         s3_client.download_file(bucket, csv_file, local_csv)
131         print(f"    Descargado a: {local_csv}")
132
133         # Leer datos con funci n robusta
134         df = read_csv_robust(local_csv)
135         print(f"Datos cargados: {len(df):,} registros, {len(df.
                columns)} columnas")
136
137         # Mostrar informaci n b sica
138         print(f"\nxx INFORMACI N DEL DATASET:")
139         print(f"    Columnas: {list(df.columns)}")
140         print(f"    Tipos de datos:")
141         for col in df.columns:
142             print(f"      - {col}: {df[col].dtype}")
143
144     else:
145         print("No se encontraron archivos CSV")
146         # Crear datos sint ticos para demo
147         print("xx Creando datos sint ticos para demostraci n...")
148
149         np.random.seed(42)
150         n_samples = 1000
151
152         df = pd.DataFrame({
153             'book_id': range(n_samples),
154             'user_id': np.random.randint(1, 100, n_samples),
155             'title': [f'Book_{i}' for i in range(n_samples)],
156             'author_name': [f'Author_{i%50}' for i in range(
                    n_samples)],
157             'actual_rating': np.random.normal(3.5, 0.8, n_samples).
                    clip(1, 5),
158             'book_avg_rating': np.random.normal(3.5, 0.6, n_samples
                    ).clip(1, 5),
159             'book_ratings_count': np.random.lognormal(5, 1,
                    n_samples).astype(int),
160             'publication_year': np.random.randint(1990, 2024,
                    n_samples),
161             'num_pages': np.random.normal(300, 100, n_samples).clip
                    (50, 1000),
162             'language_code': np.random.choice(['eng', 'spa', 'fra',
                     'deu'], n_samples, p=[0.7, 0.15, 0.1, 0.05]),
163             'is_english': np.random.binomial(1, 0.7, n_samples),
164             'is_popular': np.random.binomial(1, 0.5, n_samples),
```

```
            'is_recent': np.random.binomial(1, 0.4, n_samples),
            'is_long_book': np.random.binomial(1, 0.5, n_samples),
            'predicted_rating': np.random.normal(3.5, 0.7,
               n_samples).clip(1, 5),
            'prediction_confidence': np.random.uniform(0.3, 0.9,
               n_samples)
        })

        print(f"Dataset sint tico creado: {len(df)} registros")

except Exception as e:
    print(f"Error general: {e}")
    raise

# 3. Validar y limpiar datos
print(f"\nxx VALIDANDO Y LIMPIANDO DATOS")
print("-" * 40)

# Verificar columnas requeridas
required_cols = ['predicted_rating', 'is_english', 'is_popular', '
   is_recent', 'is_long_book']
missing_cols = [col for col in required_cols if col not in df.
   columns]

if missing_cols:
    print(f"Columnas faltantes: {missing_cols}")

    # Crear columnas faltantes si es necesario
    for col in missing_cols:
        if col == 'predicted_rating':
            df['predicted_rating'] = df.get('book_avg_rating', np.
               random.normal(3.5, 0.7, len(df)))
        elif col.startswith('is_'):
            df[col] = np.random.binomial(1, 0.5, len(df))

    print(f"Columnas faltantes creadas")

# Limpiar datos
print(f"xx Datos antes de limpieza: {len(df)} registros")

# Eliminar filas con valores nulos en columnas cr ticas
df_clean = df.dropna(subset=['predicted_rating'] + [col for col in
   required_cols[1:] if col in df.columns])

# Asegurar tipos correctos
df_clean['predicted_rating'] = pd.to_numeric(df_clean['
   predicted_rating'], errors='coerce')
for col in ['is_english', 'is_popular', 'is_recent', 'is_long_book'
   ]:
    if col in df_clean.columns:
```

```
        df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce').fillna(0).astype(int)

# Filtrar valores v lidos
df_clean = df_clean[
    (df_clean['predicted_rating'] >= 1) &
    (df_clean['predicted_rating'] <= 5)
].reset_index(drop=True)

print(f"Datos despu s de limpieza: {len(df_clean)} registros")

# Usar el DataFrame limpio
df = df_clean

# 4. An lisis de bias directo con pandas
print("\nxx EJECUTANDO AN LISIS DE BIAS DIRECTO")
print("-" * 50)

def calculate_bias_metrics(df, sensitive_attr, target_col='predicted_rating', threshold=4.0):
    """Calcular m tricas de bias manualmente con validaci n"""

    if sensitive_attr not in df.columns:
        print(f"Columna {sensitive_attr} no encontrada")
        return None

    # Grupos
    privileged = df[df[sensitive_attr] == 1]
    unprivileged = df[df[sensitive_attr] == 0]

    if len(privileged) == 0 or len(unprivileged) == 0:
        print(f"Grupos vac os para {sensitive_attr}")
        return None

    # M tricas b sicas
    priv_mean = privileged[target_col].mean()
    unpriv_mean = unprivileged[target_col].mean()

    # Demographic Parity Difference
    priv_positive_rate = (privileged[target_col] >= threshold).mean()
    unpriv_positive_rate = (unprivileged[target_col] >= threshold).mean()
    demographic_parity_diff = priv_positive_rate - unpriv_positive_rate

    # Equalized Odds (si hay actual_rating)
    equalized_odds_diff = np.nan
    if 'actual_rating' in df.columns:
        try:
```

```
            priv_actual_pos = privileged[privileged['actual_rating'] >= threshold]
            unpriv_actual_pos = unprivileged[unprivileged['actual_rating'] >= threshold]

            if len(priv_actual_pos) > 0 and len(unpriv_actual_pos) > 0:
                priv_tpr = (priv_actual_pos[target_col] >= threshold).mean()
                unpriv_tpr = (unpriv_actual_pos[target_col] >= threshold).mean()
                equalized_odds_diff = abs(priv_tpr - unpriv_tpr)
        except:
            pass

    # Calibration
    calibration_diff = np.nan
    if 'actual_rating' in df.columns:
        try:
            priv_cal_error = abs(privileged[target_col].mean() - privileged['actual_rating'].mean())
            unpriv_cal_error = abs(unprivileged[target_col].mean() - unprivileged['actual_rating'].mean())
            calibration_diff = abs(priv_cal_error - unpriv_cal_error)
        except:
            pass

    return {
        'sensitive_attribute': sensitive_attr,
        'privileged_group_size': len(privileged),
        'unprivileged_group_size': len(unprivileged),
        'privileged_mean_prediction': float(priv_mean),
        'unprivileged_mean_prediction': float(unpriv_mean),
        'mean_difference': float(priv_mean - unpriv_mean),
        'demographic_parity_difference': float(demographic_parity_diff),
        'equalized_odds_difference': float(equalized_odds_diff) if not np.isnan(equalized_odds_diff) else None,
        'calibration_difference': float(calibration_diff) if not np.isnan(calibration_diff) else None,
        'privileged_positive_rate': float(priv_positive_rate),
        'unprivileged_positive_rate': float(unpriv_positive_rate),
        'bias_magnitude': float(abs(demographic_parity_diff)),
        'bias_level': 'High' if abs(demographic_parity_diff) > 0.2 else 'Medium' if abs(demographic_parity_diff) > 0.1 else 'Low'
    }

# Analizar cada atributo sensible
```

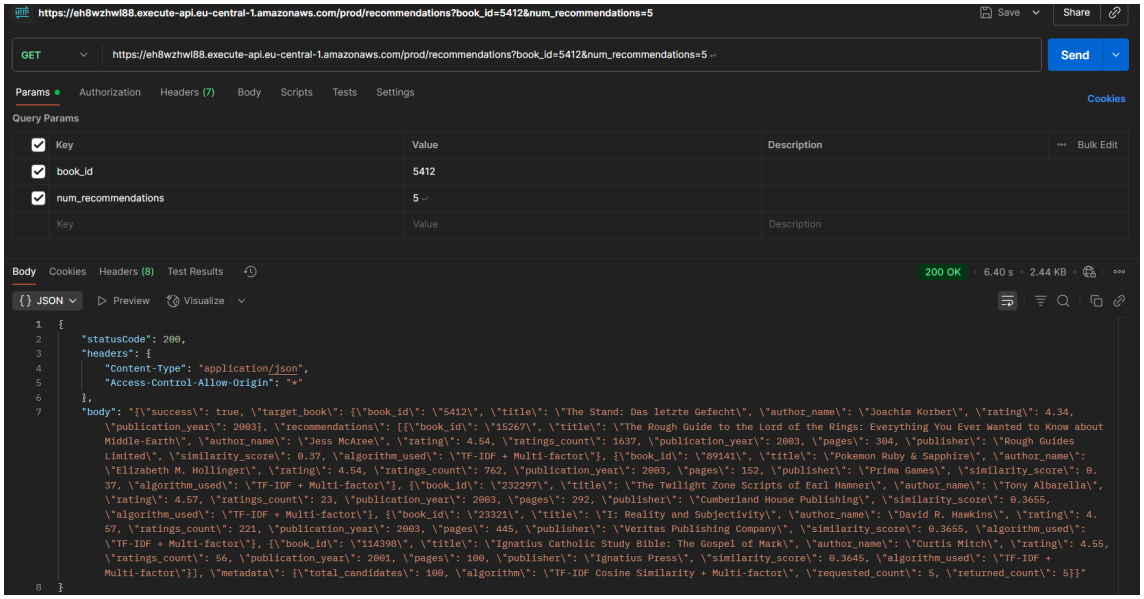
```
sensitive_attributes = ['is_english', 'is_popular', 'is_recent', 'is_long_book']
bias_results = {}

for attr in sensitive_attributes:
    print(f"xx Analizando {attr}...")
    result = calculate_bias_metrics(df, attr)

    if result:
        bias_results[attr] = result
        print(f"   xx Bias level: {result['bias_level']}")
        print(f"   xx Demographic Parity: {result['demographic_parity_difference']:.3f}")
        print(f"   xx Grupos: {result['privileged_group_size']} vs {result['unprivileged_group_size']}")
    else:
        print(f"No se pudo analizar {attr}")

# 5. Crear visualizaciones
print(f"\nxx CREANDO VISUALIZACIONES")
print("-" * 30)

import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('default')
sns.set_palette("Set2")

# Gr fica 1: Resumen de bias
if bias_results:
    fig, ax = plt.subplots(figsize=(12, 8))

    attrs = list(bias_results.keys())
    dp_values = [bias_results[attr]['demographic_parity_difference'] for attr in attrs]
    bias_levels = [bias_results[attr]['bias_level'] for attr in attrs]

    colors = {'Low': 'green', 'Medium': 'orange', 'High': 'red'}
    bar_colors = [colors[level] for level in bias_levels]

    bars = ax.barh([attr.replace('is_', '').replace('_', ' ').title() for attr in attrs],
                   [abs(dp) for dp in dp_values],
                   color=bar_colors, alpha=0.7)

    ax.axvline(x=0.1, color='orange', linestyle='--', alpha=0.7, label='Medium Threshold')
    ax.axvline(x=0.2, color='red', linestyle='--', alpha=0.7, label='High Threshold')
```

```
331
332     ax.set_xlabel('Demographic Parity Difference (Absolute)')
333     ax.set_title('Bias Analysis Summary\nHybrid Recommendation
            System', fontweight='bold', pad=20)
334     ax.legend()
335     ax.grid(True, alpha=0.3)
336
337     # A adir etiquetas
338     for bar, level in zip(bars, bias_levels):
339         ax.text(bar.get_width() + 0.01, bar.get_y() + bar.
                get_height()/2,
340                 level, ha='left', va='center', fontweight='bold')
341
342     plt.tight_layout()
343     plt.savefig('/tmp/bias_summary_clarify.png', dpi=300,
            bbox_inches='tight')
344     plt.show()
345
346 # 6. Crear reporte final
347 print(f"\nxx CREANDO REPORTE FINAL")
348 print("-" * 30)
349
350 comprehensive_report = {
351     "analysis_metadata": {
352         "timestamp": datetime.now().isoformat(),
353         "framework": "Custom Bias Analysis (Robust CSV)",
354         "dataset_size": len(df),
355         "model_type": "Hybrid Recommendation System"
356     },
357     "data_quality": {
358         "csv_reading_method": "Robust parsing with error handling",
359         "original_size": "Variable (dependent on export)",
360         "cleaned_size": len(df),
361         "columns_analyzed": list(df.columns)
362     },
363     "bias_analysis_results": bias_results,
364     "summary": {
365         "total_facets_analyzed": len(bias_results),
366         "high_bias_facets": len([r for r in bias_results.values()
                if r['bias_level'] == 'High']),
367         "medium_bias_facets": len([r for r in bias_results.values()
                if r['bias_level'] == 'Medium']),
368         "low_bias_facets": len([r for r in bias_results.values() if
                r['bias_level'] == 'Low'])
369     },
370     "recommendations": []
371 }
372
373 # A adir recomendaciones
374 for attr, result in bias_results.items():
```

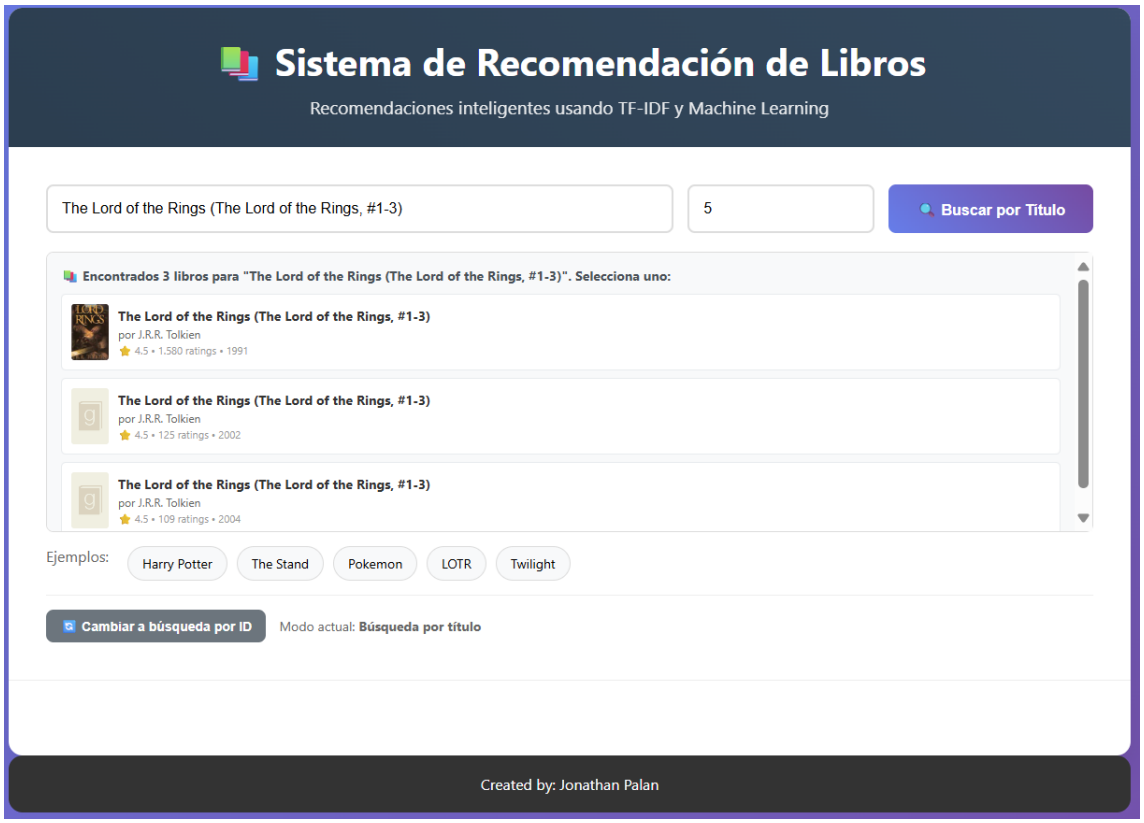
```
    if result['bias_level'] in ['High', 'Medium']:
        comprehensive_report["recommendations"].append({
            "facet": attr,
            "bias_level": result['bias_level'],
            "demographic_parity": result['
               demographic_parity_difference'],
            "recommendation": f"Review {attr} bias - {result['
               bias_level']} level detected"
        })

# Guardar reporte
with open('model/robust_bias_report.json', 'w') as f:
    json.dump(comprehensive_report, f, indent=2, default=str)

print(f"Reporte guardado: /tmp/robust_bias_report.json")

# 7. Resumen final
print(f"\n" + "="*70)
print("xx AN LISIS DE BIAS ROBUSTO COMPLETADO")
print("="*70)

print(f"\n RESUMEN DE BIAS ANALYSIS:")
if bias_results:
    for attr, result in bias_results.items():
        bias_icon = "" if result['bias_level'] == 'High' else "xx"
           if result['bias_level'] == 'Medium' else "xx"
        print(f"   {bias_icon} {attr}: {result['bias_level']} (DP:
           {result['demographic_parity_difference']:.3f})")
else:
    print("No se pudieron calcular m tricas de bias")

print(f"\nxx ESTAD STICAS DEL DATASET:")
print(f" Registros analizados: {len(df):,}")
print(f" Columnas disponibles: {len(df.columns)}")
print(f" Atributos sensibles: {len([attr for attr in
   sensitive_attributes if attr in df.columns])}")

print(f"\ARCHIVOS GENERADOS:")
print(f"/model/robust_bias_report.json")
print(f"/model/bias_summary_clarify.png")
```

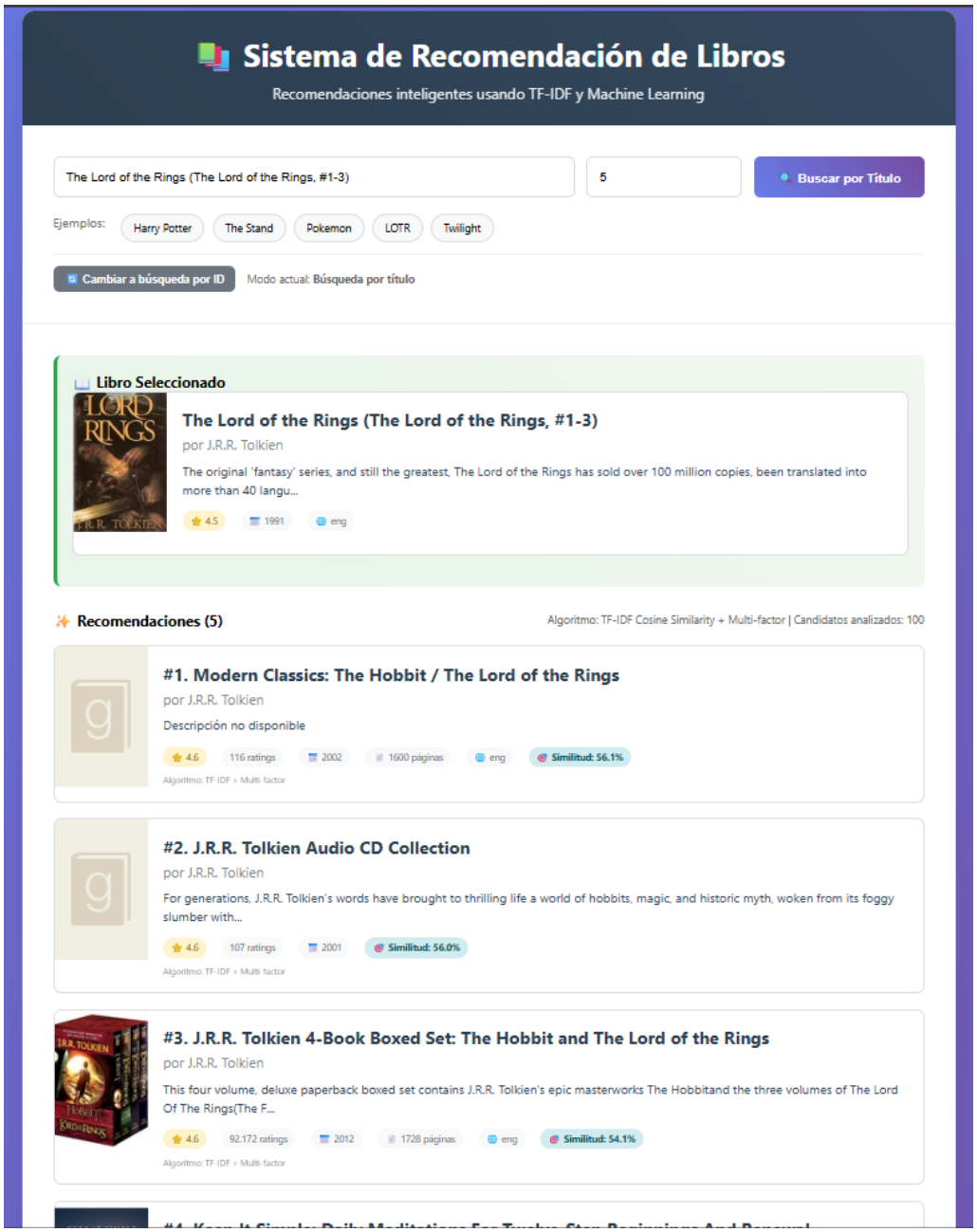
8.0.9. Pruebas de aplicación y API

llamada a la API



Interfaz y Prueba


Sistema de Recomendación de Libros
Recomendaciones inteligentes usando TF-IDF y Machine Learning
The Lord of the Rings (The Lord of the Rings, #1-3)
5
Buscar por Título
Encontrados 3 libros para "The Lord of the Rings (The Lord of the Rings, #1-3)". Selecciona uno:
The Lord of the Rings (The Lord of the Rings, #1-3)
por J.R.R. Tolkien
4.5 • 1.580 ratings • 1991
The Lord of the Rings (The Lord of the Rings, #1-3)
por J.R.R. Tolkien
4.5 • 125 ratings • 2002
The Lord of the Rings (The Lord of the Rings, #1-3)
por J.R.R. Tolkien
4.5 • 109 ratings • 2004
Ejemplos:
Harry Potter
The Stand
Pokemon
LOTR
Twilight
Cambiar a búsqueda por ID
Modo actual: Búsqueda por título
Created by: Jonathan Palan


Sistema de Recomendación de Libros
Recomendaciones inteligentes usando TF-IDF y Machine Learning
The Lord of the Rings (The Lord of the Rings, #1-3)
5
Buscar por Título
Ejemplos:
Harry Potter
The Stand
Pokemon
LOTR
Twilight
Cambiar a búsqueda por ID
Modo actual: Búsqueda por título
Libro Seleccionado
The Lord of the Rings (The Lord of the Rings, #1-3)
por J.R.R. Tolkien
The original 'fantasy' series, and still the greatest, The Lord of the Rings has sold over 100 million copies, been translated into more than 40 langu...
4.5
1991
eng
Recomendaciones (5)
Algoritmo: TF-IDF Cosine Similarity + Multi-factor | Candidatos analizados: 100
#1. Modern Classics: The Hobbit / The Lord of the Rings
por J.R.R. Tolkien
Descripción no disponible
4.6
116 ratings
2002
1600 páginas
eng
Similitud: 56.1%
Algoritmo: TF-IDF + Multi-factor
#2. J.R.R. Tolkien Audio CD Collection
por J.R.R. Tolkien
For generations, J.R.R. Tolkien's words have brought to thrilling life a world of hobbits, magic, and historic myth, woken from its foggy slumber with...
4.6
107 ratings
2001
Similitud: 56.0%
Algoritmo: TF-IDF + Multi-factor
#3. J.R.R. Tolkien 4-Book Boxed Set: The Hobbit and The Lord of the Rings
por J.R.R. Tolkien
This four volume, deluxe paperback boxed set contains J.R.R. Tolkien's epic masterworks The Hobbitand the three volumes of The Lord Of The Rings(The F...
4.6
92.172 ratings
2012
1728 páginas
eng
Similitud: 54.1%
Algoritmo: TF-IDF + Multi-factor