Documento de Requisitos de Producto (PRD)

Aplicación de Gestión de Inventario de Cómics (Comic Inventory Management App) [1]

1. Visión del Producto

Sistema modular para coleccionistas o tiendas de cómics para gestionar el inventario de números individuales, colecciones, series y editoriales, controlar el stock, órdenes de compra/venta y traders (proveedores/clientes especializados), utilizando Spring Boot + Spring Modulith. [2]

2. Objetivos

- Controlar el stock de cómics y coleccionables en tiempo real. [3]
- Gestionar traders (distribuidores, tiendas, particulares) y sus productos. [4]
- Procesar órdenes de adquisición (Purchase) y venta/intercambio (Sale/Trade). [5]
- Generar alertas de ejemplares con bajo stock o de alto valor. [6]
- Mantener un historial detallado de movimientos de inventario y cambios de estado (grading). [7]

3. Módulos Principales (4 módulos básicos) [8]

3.1 Módulo inventory [9]

- **Responsabilidad:** Gestión de cómics/items coleccionables y control de stock. [10]
- **Funcionalidades:** CRUD de items (Título, Número, Editorial, Artista, Fecha, Valor estimado, SKU)[11]. Control de stock actual por ejemplar (incluyendo el estado físico o grading)[12]. Alertas de stock mínimo o de alta demanda[13].

3.2 Módulo traders [14]

- **Responsabilidad:** Gestión de Traders (proveedores y clientes de cómics). [15]
- **Funcionalidades:** CRUD de Traders (nombre, contacto, especialidad/foco)[16]. Catálogo de ítems ofrecidos/buscados por Trader[17]. Precios de compra/venta/intercambio registrados por Trader[18].

3.3 Módulo orders [19]

- **Responsabilidad:** Gestión de transacciones (adquisición y venta/intercambio). [20]
- **Funcionalidades:** Órdenes de adquisición (Purchase) a Traders[21]. Órdenes de venta/intercambio (Sale/Trade) a Traders/Clientes[22]. Estados de órdenes (pendiente, en tránsito, completada)[23].

3.4 Módulo notifications (ACTUALIZADO)

- **Responsabilidad:** Sistema de notificaciones, alertas especializadas y **persistencia desacoplada de logs**. [24]
- **Funcionalidades:**
  - Alertas de stock bajo o de escasez[25].
  - Notificaciones de órdenes completadas[26].
  - Alertas de ejemplares que alcanzan cierto valor de mercado[27].
  - **Registro Histórico (Logs):** Todos los eventos de dominio (StockQuantityUpdated, GradingUpdated, etc.) serán persistidos en una base de datos **NoSQL (MongoDB)** para garantizar **rapidez de escritura y aislamiento** del *core* transaccional (MySQL).

4. Eventos del Sistema [28]

- **ComicAdded**: Nuevo cómic/ítem agregado[29].

- **StockQuantityUpdated**: Cambio en la cantidad de un ejemplar[30].
- **GradingUpdated**: Cambio en el estado físico (grading) de un cómic[31].
- **LowStockAlert**: Stock por debajo del mínimo definido[32].
- **OrderCompleted**: Transacción procesada completamente[33].

5. APIs REST Principales [34]

Módulo	Endpoint Base	Método	Descripción
inventory	/api/inventary	GET /all-comics	Obtener todos los cómics.
inventory	/api/inventary	POST /new-comic	Crear un nuevo cómic.
inventory	/api/inventary	POST /{id}/reduce-stock	Reducir stock (venta).
inventory	/api/inventary	POST /{id}/updateGrading	Actualizar estado físico (grading).
traders	/api/traders	POST	Crear Trader.
orders	/api/orders	POST	Crear orden (Adquisición/Venta) .

6. Modelo de Datos Básico [35]

- **ComicItem:** id, sku, title, issueNumber, publisher, estimatedValue, currentStock, minimalStock, grading, traderId[36].
- **Trader:** id, name, email, specialty, address[37].

- **Order:** id, type (ACQUISITION/SALE_TRADE), status, totalAmount, createdDate, traderId[38].

- **OrderIItem:** id, orderId, comicId, quantity, unitPrice, itemGrading[39].

7. Stack Tecnológico (ACTUALIZADO)

Se mantiene el stack propuesto y se añaden las nuevas tecnologías para la gestión avanzada y el historial:

Componente	Tecnología	Propósito
Framework Principal	Spring Boot 3.5+ [40]	Desarrollo rápido y arquitectura.
Modularidad	Spring Modulith 1.4+ [41]	Definición de límites entre módulos.
Lenguaje	Java 21 [42]	Estándar de desarrollo.
BD Relacional (Principal)	MySQL 8.0 / JPA/Hibernate [43]	**Inventario, Órdenes, Traders** (Datos transaccionales).
BD NoSQL (Secundaria)	**MongoDB**	**Logs de Notificaciones y Eventos** (Datos históricos y de alta escritura).
Procesos Pesados	**Spring Batch**	Ejecución de tareas programadas (Reportes, Auditorías).
Herramientas	Lombok [44], Maven [45]	Reducción de boilerplate, gestión de dependencias.

11. Fase 2 - Funcionalidades Avanzadas (ACTUALIZADO)

Esta fase se centra en la madurez del sistema y la calidad del código, aprovechando las nuevas integraciones.

- Módulo orders completo (Acquisición/Venta/Intercambio)[46].

- Sistema de Notificaciones avanzadas (valor, grading, escasez), respaldado por la base de datos
  MongoDB[47].

- **Reportes de valor total de la colección y rotación de ítems, implementados mediante Spring Batch**[48].

- Validaciones de negocio avanzadas (ej. validación de formato grading)[49].

- **Pruebas unitarias robustas: Objetivo de cobertura de código superior al 90%**, utilizando **Mockito** para simular todas las dependencias (Repositorios, Publishers y Servicios) y asegurar el cumplimiento de la lógica de negocio y la publicación de eventos.
- Integración y validación de límites de módulos con Spring Modulith[50].