Product Requirements Document (PRD)

Aplicación de Gestión de Inventario de Cómics (Comic Inventory Management App)

---

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. Objetivos

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

---

3. Módulos Principales (4 módulos básicos)

3.1 Módulo inventory

- **Responsabilidad:** Gestión de cómics/ítems coleccionables y control de *stock*.
- **Funcionalidades:**
    - CRUD de ítems (**Título, Número, Editorial, Artista, Fecha, Valor estimado, SKU**).
    - Control de *stock* actual por ejemplar (incluyendo el estado físico o *grading*).
    - Actualizaciones de inventario (entrada/salida/cambio de estado).
    - Alertas de *stock* mínimo o de alta demanda.

3.2 Módulo traders

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

3.3 Módulo orders

- **Responsabilidad:** Gestión de transacciones (adquisición y venta/intercambio).
- **Funcionalidades:**
    - Órdenes de adquisición (*Purchase*) a *Traders*.
    - Órdenes de venta/intercambio (*Sale/Trade*) a *Traders*/Clientes.
    - Estados de órdenes (pendiente, en tránsito, completada).
    - Registro de recepción y verificación de ítems.

3.4 Módulo notifications

- **Responsabilidad:** Sistema de notificaciones y alertas especializadas.
- **Funcionalidades:**
    - Alertas de *stock* bajo (para ítems comunes) o de escasez (para ítems raros).
    - Notificaciones de órdenes completadas.
    - Alertas de ejemplares que alcanzan cierto valor de mercado.
    - Reportes de movimientos y valor total de la colección.

---

4. Eventos del Sistema

- ComicAdded - Nuevo cómic/ítem agregado.
- StockQuantityUpdated - Cambio en la cantidad de un ejemplar.
- GradingUpdated - Cambio en el estado físico (*grading*) de un cómic.
- LowStockAlert - *Stock* por debajo del mínimo definido.
- AcquisitionOrderPlaced - Orden de adquisición creada.
- SaleTradeOrderPlaced - Orden de venta/intercambio creada.
- OrderCompleted - Transacción procesada completamente.

---

5. APIs REST Principales

/api/comics

POST	-	Registrar nuevo cómic
GET	-	Listar cómics (con filtros)
PUT	-	Actualizar datos de cómic
DELETE	-	Eliminar cómic

/api/inventory

POST	/update-quantity -	Actualizar cantidad de stock
POST	/update-grading -	Actualizar estado físico (grading)
GET	/low-stock -	Cómics con stock bajo o raros

/api/traders

POST	-	Crear Trader
GET	-	Listar Traders
PUT	-	Actualizar Trader

/api/orders

POST	/acquisition -	Crear orden de adquisición
POST	/sale-trade -	Crear orden de venta/intercambio
PUT	/complete -	Completar transacción

---

6. Modelo de Datos Básico

ComicItem: id, sku, title, issueNumber, publisher, releaseDate, estimatedValue, currentStock, minStock, grading, traderId

Trader: id, name, email, specialty, address

Order: id, type(ACQUISITION/SALE_TRADE), status, totalAmount, createdDate, traderId

OrderItem: id, orderId, comicId, quantity, unitPrice, itemGrading

---

7. Stack Tecnológico

Se mantiene el *stack* propuesto, ideal para la modularización:

- **Spring Boot 3.5+**
- **Spring Modulith 1.4+**
- **Java 21**
- **MySQL 8.0**
- **JPA/Hibernate**
- **Lombok**
- **SpringDoc OpenAPI**
- **Maven**

---

8. Criterios de Aceptación

- Cada módulo debe ser **independiente** (cohesión alta, acoplamiento bajo).
- Comunicación entre módulos solo por **eventos asíncronos**.
- APIs REST documentadas con **OpenAPI**.
- Base de datos normalizada para evitar redundancia de datos de cómics.
- Validaciones de negocio específicas para cómics (ej. formato de *issue number* o *grading*).
- Manejo de errores consistente en todas las APIs.

---

9. Casos de Uso Principales

9.1 Gestión de Cómics

1. **Registrar Cómic:** Coleccionista/Administrador agrega un nuevo título al inventario.
2. **Actualizar Stock y Grading:** Registrar la adquisición de un ejemplar y su estado físico (ej. *Near Mint*).
3. **Alerta de Valor:** Notificación cuando un ítem en inventario excede un valor predefinido.

9.2 Gestión de Traders

1. **Registrar Trader:** Agregar nuevo contacto de compra/venta al sistema.
2. **Asociar Ítems de Interés:** Vincular los cómics que el *Trader* típicamente compra o vende.

9.3 Gestión de Transacciones

1. **Orden de Adquisición:** Solicitar o registrar la compra de un cómic a un *Trader*.
2. **Orden de Venta/Intercambio:** Procesar la venta o intercambio de un ejemplar a un tercero.
3. **Recepción/Verificación:** Confirmar la recepción, verificar el estado (*grading*) y actualizar el inventario.

---

10. Fase 1 - MVP (Minimum Viable Product)

- Módulo inventory con CRUD básico de cómics.
- Módulo traders con gestión básica.
- Eventos básicos (ComicAdded, StockQuantityUpdated).
- APIs REST fundamentales (/api/comics, /api/traders).
- Base de datos con tablas ComicItem y Trader.

---

11. Fase 2 - Funcionalidades Avanzadas

- Módulo orders completo (Acquisición/Venta/Intercambio).
- Sistema de **Notificaciones** avanzadas (valor, *grading*, escasez).
- **Reportes** de valor total de la colección y rotación de ítems.
- Validaciones de negocio avanzadas (ej. validación de formato *grading*).
- Pruebas unitarias e integración con Spring Modulith.