

# UNIVERSIDAD AUTÓNOMA METROPOLITANA

## UNIDAD CUAJIMALPA

Licenciatura en Tecnologías y Sistemas de Información.

## Reporte de Proyecto Final

### Sistema de Gestión de Inventario

UNIDAD DE ENSEÑANZA  
**Programación de Web Dinámico**

Dr. Guillermo Monroy

Luis Antonio Salinas Mata  
2333030416

# 1. Descripción General

El proyecto consiste en una aplicación web Fullstack para la gestión y control de inventarios. El sistema permite administrar productos y categorías, así como registrar entradas y salidas de mercancía, manteniendo un historial preciso de los movimientos y actualizando el stock en tiempo real.

El objetivo principal fue implementar una arquitectura desacoplada utilizando Spring Boot para la lógica de negocio y Angular con Angular Material para una interfaz de usuario moderna y responsive.

## 2. Stack Tecnológico

### Backend

- Lenguaje: Java 21
- Framework: Spring Boot 3.5.3
- Base de Datos: MySQL 8.0
- Spring Data JPA
- Documentación: OpenAPI (Swagger UI)

### Frontend

- Framework: Angular 19+
- Diseño: Angular Material
- Comunicación: HttpClient

## 3. Arquitectura del Sistema

Se implementó una Arquitectura Limpia y modular para asegurar la mantenibilidad y escalabilidad del código.

### 3.1 Estructura del Backend

El backend sigue el patrón de capas:

1. **Controller:** Expone los endpoints REST (/productos, /categorias, /movimientos).
2. **Service:** Contiene la lógica de negocio (validaciones, cálculos de stock).
3. **Repository:** Interfaz con la base de datos mediante JPA.
4. **Entity/DTO:** Separación entre el modelo de base de datos y los objetos de transferencia de datos para seguridad.

### 3.2 Estructura del Frontend

El frontend se organizó por módulos funcionales:

- Servicios: ProductoService, CategoriaService, StockMovementService .
- Componentes: Dashboard, ProductoList, ProductoForm, CategoriaList, CategoriaForm, MovimientoList.

## 4. Funcionalidades Desarrolladas

### 4.1 Gestión de Productos y Categorías (CRUD)

El sistema permite crear, leer, actualizar y eliminar registros.

- **Validaciones:** Se implementaron validaciones tanto en Frontend (Reactive Forms) como en Backend para impedir precios negativos o nombres vacíos.
- **Integridad Referencial:** Se configuró el borrado en cascada (CascadeType.ALL) para asegurar que al eliminar un producto, se elimine su historial de movimientos, evitando errores de inconsistencia en la base de datos.

The screenshot shows a web application interface titled 'Sistema de Gestión de Inventario Básico'. The main section is titled 'Inventario de Productos'. At the top right is a blue button labeled '+ Nuevo Producto'. Below it is a search bar with the placeholder 'Buscar producto...'. A magnifying glass icon is to the right of the search bar. The main area contains a table with two rows of data. The columns are: ID, Producto, Precio, Stock, Descripción, ID de Categoría, and Acciones. The first row (ID 2) has a product named 'celular iphone 16' with a price of '\$15000' and stock '1 u.'. The second row (ID 6) has a product named 'laptop gamer es una lap asus' with a price of '\$4000' and stock '5 u.'. Both rows have an 'Editar Stock' link, an 'Editar Producto' link, and a red 'Eliminar Producto' link. At the bottom of the table, there is a pagination control with 'Items per page: 5' and '1 - 2 of 2'.

ID	Producto	Precio	Stock	Descripción	ID de Categoría	Acciones
2	celular iphone 16	\$15000	1 u.	iphone 16	5	<a href="#">Editar Stock</a> <a href="#">Editar Producto</a> <a href="#">Eliminar Producto</a>
6	laptop gamer es una lap asus	\$4000	5 u.	es una lap asus	2	<a href="#">Editar Stock</a> <a href="#">Editar Producto</a> <a href="#">Eliminar Producto</a>

Imagen 1. Ventana de tabla de productos

## 4.2 Control de Stock y Movimientos

En lugar de editar el stock manualmente, se implementó un módulo de Movimientos:

- El usuario registra una "Entrada" o "Salida".
- El sistema calcula automáticamente el nuevo stock.
- Se guarda un registro histórico con fecha, tipo y cantidad.

Historial de Movimientos de Stock				
ID	ID Producto	Cantidad	Tipo	Fecha
1	2	1	ENTRADA	12/7/25, 8:18 PM
2	2	1	ENTRADA	12/7/25, 8:18 PM
3	2	1	ENTRADA	12/7/25, 8:26 PM
6	2	1	ENTRADA	12/7/25, 10:33 PM
7	2	1	ENTRADA	12/7/25, 10:33 PM
10	2	14	SALIDA	12/8/25, 5:23 PM

Imagen 2. Ventana del historial de movimientos (Entradas/Salidas)

## 4.3 Dashboard y Métricas

Se desarrolló un panel de control visual que consume endpoints optimizados para mostrar:

- Total de productos y categorías.
- **Alerta de Stock Crítico:** Lista filtrada desde el backend (`findByStockActualLessThan`) que muestra productos con menos de 5 unidades.

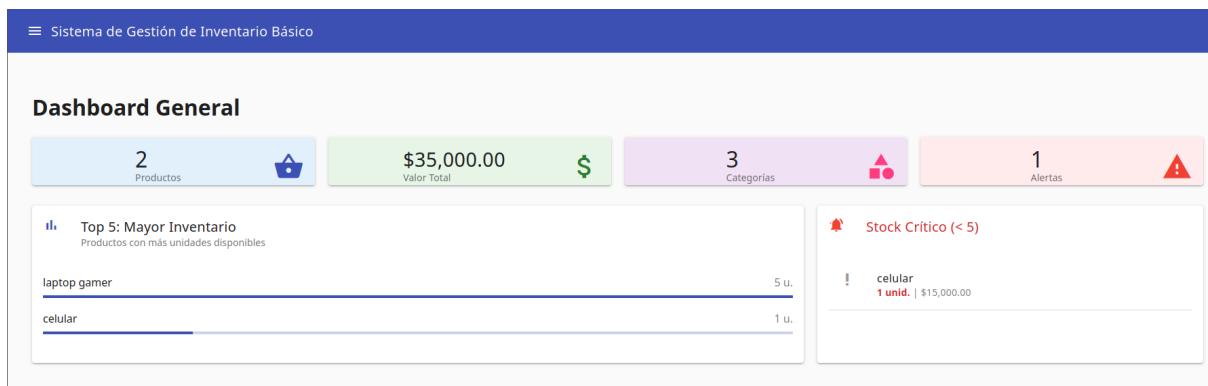


Imagen 3. Ventana del DashBoard

## 4.4 UX/UI Avanzada

Para cumplir con el criterio de "Calidad de UI", se implementaron:

- **Tablas Inteligentes:** Uso de MatTableDataSource con paginación, ordenamiento (Sort) y filtrado en tiempo real.
- **Diálogos Modales:** Los formularios de creación/edición se abren en ventanas flotantes (MatDialog) para no perder el contexto de navegación.
- **Feedback Visual:** Colores condicionales (Rojo/Verde) para el estado del stock.

## 5. Modelo de Base de Datos

El sistema utiliza tres entidades principales relacionadas:

1. **Categoría (1) ---- (N) Producto:** Un producto pertenece a una categoría.
2. **Producto (1) ---- (N) StockMovement:** Un producto tiene múltiples movimientos históricos.

## 6. Documentación de API (Swagger)

Todos los endpoints están documentados y son probables mediante Swagger UI, accesible en la <http://localhost:8080/swagger-ui/index.html>.

The screenshot shows the Swagger UI interface with three main sections: Stock Movements, Productos, and Categorías. Each section contains a list of API operations with their corresponding HTTP methods and URLs. The operations are color-coded: blue for GET, orange for PUT, red for DELETE, and green for POST. The Stock Movements section has one GET operation and one POST operation. The Productos section has five operations: GET /productos/{id}, PUT /productos/{id}, DELETE /productos/{id}, GET /productos, and POST /productos. The Categorías section also has five operations: GET /categorias/{id}, PUT /categorias/{id}, DELETE /categorias/{id}, GET /categorias, and POST /categorias.

Section	Operation	HTTP Method	Description
Movimientos de Stock	/stockMovement	GET	Listar todos los movimientos
	/stockMovement	POST	Registrar Movimiento
Producto	/productos/{id}	GET	Obtener producto por ID
	/productos/{id}	PUT	Actualizar producto
	/productos/{id}	DELETE	Eliminar producto
	/productos	GET	Obtener todos los productos
	/productos	POST	Crear producto
	/productos/bajo-stock	GET	Productos con bajo stock
Categoría	/categorias/{id}	GET	Obtener categoría por ID
	/categorias/{id}	PUT	Actualizar categoría
	/categorias/{id}	DELETE	Eliminar categoría
	/categorias	GET	Obtener todos los categorías
	/categorias	POST	Crear categoría

Imagen 4. Interfaz de Swagger.

## 7. Conclusiones

El desarrollo de este proyecto permitió consolidar conocimientos sobre la integración Fullstack. Los mayores retos técnicos fueron la sincronización de las entidades JPA para el borrado en cascada y la implementación de la comunicación reactiva entre

los componentes de Angular (tablas y diálogos). El resultado es un sistema robusto, funcional y escalable.