# Requisitos Esenciales para un Sistema de Inventario Robusto

El objetivo principal es pasar de un inventario basado en "saldos copiados" (propenso a errores) a un inventario basado en "transacciones" (preciso y en tiempo real).

## I. Estructura de la Base de Datos (Modelos y Relaciones)

Para Flask, esto se traduce en los modelos de tu base de datos (por ejemplo, usando SQLAlchemy).

| **Elemento** | **Descripción y Justificación** |
| --- | --- |
| **1. Unidades de Medida (UoM)** | **MANDATORIO**. Como notaste, esto es clave. Debe ser una tabla dedicada (UND, KG, LT, ROLLOS, CANECA, CC, etc.) que se relaciona con la tabla de productos. Esto garantiza que un producto solo pueda tener movimientos en su unidad base. |
| **2. Categorías/Clases** | Una tabla dedicada (QUIMICOS, ALMACEN, POSCOSECHA, etc.) para agrupar productos. Permite reportes y búsquedas rápidas. |
| **3. Maestro de Productos** | La lista central de todos los ítems. Debe incluir: Nombre, Referencia/Código Único (SKU), Relación con UoM, Relación con Categoría, Stock Mínimo (para alertas). |
| **4. Entidades (Proveedores/Clientes)** | Dos tablas separadas (o una tabla de "Contactos" con un campo de tipo). Es fundamental saber **quién** te vendió (Entrada) y **a dónde/quién** le entregaste (Salida). |
| **5. Tabla de Movimientos (Transacciones)** | **El corazón del sistema.** Aquí se registra cada entrada y salida individualmente, eliminando la necesidad de las filas por día (Día 1, Día 2...). Campos esenciales: Fecha, Tipo (ENTRADA o SALIDA), ID Producto, Cantidad, ID de la Entidad (Proveedor/Cliente), No. Factura/Referencia, Valor Unitario. |
| **6. Cálculo de Saldo** | **IMPORTANTE:** No se almacena como un campo. El saldo se debe **calcular** sumando todas las Entradas y restando todas las Salidas para ese producto. |

## II. Lógica de Negocio y Cálculos

Estas son las funciones que tu aplicación Flask deberá implementar en Python para el *backend*.

1. **Valoración del Inventario (Costo):** Debes definir cómo valoras el inventario. El método más común es el **Costo Promedio Ponderado** o **FIFO (First-In, First-Out)**. Esto es crucial para calcular correctamente el VALOR TOTAL de tu inventario.
2. **Validación de Stock Negativo:** Antes de registrar una SALIDA, el sistema debe verificar que la cantidad solicitada no dejará el saldo por debajo de cero. Si sucede, debe mostrar una alerta e impedir la transacción.
3. **Trazabilidad y Costos:** Cada ENTRADA debe registrar su costo unitario exacto (VALOR C/U). Esto permite al sistema calcular el costo de la mercancía vendida (costo de cada SALIDA).
4. **Transferencia de Saldo Inicial:** Una función para migrar la hoja del mes anterior al nuevo. Esto solo se hace **una vez** al inicio, luego el sistema opera con transacciones.

## III. Funcionalidad de la Aplicación Web (Flask/UI/UX)

Estos son los *endpoints* y las vistas que necesitarás en tu aplicación.

| **Módulo** | **Funcionalidad Clave** |
| --- | --- |
| **Dashboard** | Vista general del stock. Alertas de stock bajo o vencimiento. |
| **Gestión de Maestros** | Vistas CRUD (Crear, Leer, Actualizar, Borrar) para **Productos**, **Categorías**, **UoM**, **Proveedores** y **Clientes**. |
| **Registro de Entradas** | Formulario simple para registrar nuevas compras. Debe buscar el producto, y pedir Fecha, No. Factura, Proveedor, Cantidad y Costo Unitario. |
| **Registro de Salidas** | Formulario para registrar despachos/consumos. Debe buscar el producto, y pedir Fecha, Destino/Cliente, y Cantidad. **Debe calcular el costo automáticamente** según el método de valoración elegido. |
| **Kardex (Historial)** | Vista de movimientos por producto. Permite ver el historial completo de entradas y salidas de un ítem, junto con sus saldos y costos en el tiempo. |

## IV. Mejoras Adicionales (Para una Versión 2.0)

* **Alertas de Stock Mínimo:** Notificaciones (visuales o por correo) cuando un producto cae por debajo del nivel de stock mínimo definido.
* **Manejo de Lotes/Fechas de Vencimiento:** Si manejas químicos o poscosecha, es esencial registrar el **Lote** y la **Fecha de Vencimiento** al registrar la entrada. Al hacer una salida, el sistema debe priorizar el lote que vence primero (lógica FEFO - First Expiry, First Out).
* **Reportes de Consumo:** Reportes filtrables por período, cliente o categoría para analizar qué se consume más o menos.
* **Roles de Usuario:** Implementar roles de usuario (ej. Almacenista solo registra movimientos, Gerente ve reportes de valor).