HISTORIA DE USUARIO

Sigue la siguiente estructura y agrega tareas de acuerdo con el número de semanas de cada módulo.

Nombre de la HU:	Mini-Tienda con **Interfaces + JDBC básico** (JOptionPane)
Objetivo de la HU	• Desarrollar una mini-aplicación en Java que gestione un inventario de productos mediante **JOptionPane**, aplicando **Interfaces** para desacoplar la lógica y **JDBC básico** para realizar **CRUD** en una base de datos relacional. Esta HU aporta valor al incorporar persistencia sencilla (INSERT/SELECT/UPDATE/DELETE con PreparedStatement) manteniendo la misma experiencia de la Mini-Tienda
TASK 1	**Descripción de la Tarea:** 1. Modelo de datos e interfaces: • Clase Producto { id, nombre, precio, stock }. • Interface Repositorio<T> con métodos: crear(T), buscarPorId(int), buscarTodos(), actualizar(T), eliminar(int). • Interface ServicioInventario con operaciones: agregarProducto, actualizarPrecio, actualizarStock, eliminarProducto, buscarPorNombre. • Configuración JDBC básica: ConnectionFactory (DriverManager) y db.properties (url, user, password, driver). • **Esquema** mínimo: tabla productos(id PK AI, nombre UNIQUE, precio DECIMAL(10,2), stock INT).

TAS	Menú principal con JOptionPane: Mostrar un menú repetitivo con las opciones: • Agregar producto • Listar inventario • Actualizar precio • Actualizar stock • Eliminar producto • Buscar producto por nombre • Salir con resumen (cantidad de operaciones realizadas)
TAS	3. Flujo de cada opción (CRUD con JDBC básico): 1. **Agregar producto**: pedir nombre, precio, stock → INSERT INTO productos(nombre, precio, stock) VALUES (?,?,?). 2. **Listar inventario**: SELECT id, nombre, precio, stock FROM productos y mostrar en JOptionPane. 3. **Actualizar precio**: solicitar id y nuevo precio → UPDATE productos SET precio=? WHERE id=?. 4. **Actualizar stock**: solicitar id y nuevo stock → UPDATE productos SET stock=? WHERE id=?. 5. **Eliminar producto**: solicitar id → DELETE FROM productos WHERE id=?. 6. **Buscar por nombre**: solicitar texto → SELECT ... FROM productos WHERE nombre LIKE ? (usar %texto%). 7. **Salir**: mostrar conteo de operaciones (altas, bajas, actualizaciones). a.
TAS	4. Validaciones y Mensajes • Manejo de NumberFormatException y entradas vacías. • Validación de duplicados por nombre (mensaje si BD retorna error de UNIQUE). • Manejo de SQLException con mensajes claros. • Uso de **try-with-resources** para cerrar Connection, PreparedStatement y ResultSet. • Mostrar errores y confirmaciones con showMessageDialog.

	Criterios de aceptación: • El programa compila y se ejecuta sin errores. • Todas las opciones del menú funcionan usando exclusivamente **JOptionPane**. • Se emplean **Interfaces** (Repositorio, ServicioInventario) y sus implementaciones. • Las operaciones **CRUD** funcionan en BD con **JDBC básico** usando PreparedStatement. • Se valida entrada de datos (no vacíos, números válidos, stock ≥ 0) y se comunica cualquier violación de UNIQUE(nombre). • Se muestra un **resumen final** al salir. **Story Points: 20**
	Cierre de la actividad

P – J – H - B	B	F
S1 - TASK	S1 - TASK	S1 – TASK
S2 – SPIKE - RETORSPECTIVE	S2 - TASK	S2 – TASK
S3 – HU - CALIFICABLE	S3 – SPIKE RETRO	S3 – SPIKE RETRO
	S4 - TASK	S4 – TASK
	S5 – HU - NOTA	S5 – SPIKE RETRO
		S5 – TASK
		S7 – HU - NOTA

criterio	recuerda	comprende	practica	analiza	Evalua
Tarea 1	2	3	7	8	10
Tarea 2					