**HISTORIA DE USUARIO**

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

|  |  |  |
| --- | --- | --- |
| **Nombre de la feature:** | | **Mini-Tienda con JOptionPane (Arrays + ArrayList + HashMap)** |
| **Objetivo de la HU** | | Desarrollar una mini-aplicación en Java que permita **gestionar un inventario de productos** con entrada y salida de datos mediante JOptionPane. El sistema debe manejar nombres de productos con ArrayList, precios con Array, y stock con HashMap, además de incluir opciones de compra, búsqueda y estadísticas. Esta HU aporta valor porque integra estructuras de datos con interfaces gráficas simples, mejorando la experiencia de usuario y reforzando la lógica de programación. |
| **TASKS** | **Descripción de la Tarea:**  **Implementar la Mini-Tienda con menú completo y validaciones.**  **Pasos a realizar:**   1. **Modelo de datos**:    * Crear ArrayList<String> para nombres de productos.    * Crear double[] para precios, sincronizado con nombres.    * Crear HashMap<String, Integer> para stock.    * Implementar métodos utilitarios: addProducto, expandPrecios, indexOfNombre. 2. **Menú principal con JOptionPane**:    * Mostrar un menú repetitivo con las opciones:      1. Agregar producto.      2. Listar inventario.      3. Comprar producto.      4. Mostrar estadísticas (más barato y más caro).      5. Buscar producto por nombre.      6. Salir con ticket final. 3. **Flujo de cada opción**:    * **Agregar producto**: pedir nombre, precio, stock. Validar entradas, no duplicados.    * **Listar inventario**: recorrer estructuras y mostrar todos los productos con nombre, precio y stock.    * **Comprar**: solicitar producto y cantidad. Validar existencia y stock, confirmar compra y actualizar.    * **Estadísticas**: encontrar precio mínimo y máximo recorriendo el array precios.    * **Buscar producto**: permitir coincidencias parciales en el nombre.    * **Salir**: mostrar total acumulado de compras en la sesión. 4. **Validaciones y mensajes**:    * Manejo de excepciones (NumberFormatException).    * Mensajes de error y confirmación con showMessageDialog.    * Formatear salidas para claridad.    * Mensajes de error y confirmación con showMessageDialog.    * Formatear salidas para claridad. | |
| **Criterios de Aceptación:**  **El programa compila y se ejecuta sin errores.**   * Todas las opciones del menú funcionan correctamente usando exclusivamente JOptionPane. * Las estructuras (Array, ArrayList, HashMap) se usan de manera integrada y coherente. * Se valida entrada de datos (no vacíos, números válidos, stock suficiente). * Se muestra ticket final con el total acumulado al salir. * Entrega: archivo .java con comentarios explicativos. | |
| **Story Poins: 20** | |
|  | **Cierre de la actividad** | |