UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA, MANAGUA
UNAN-MANAGUA

CENTRO UNIVERSITARIO REGIONAL DE CHONTALES

"Cornelio Silva Argüello"
UNAN Managua, CUR Chontales

2025: "Eficiencia y Calidad para seguir en victoria"

DEPARTAMENTO DE CIENCIAS TECNOLÓGICAS Y SALUD
INGENIERÍA EN SISTEMAS DE INFORMACIÓN
I AÑO, II SEMESTRE

TEMA

Desarrollo de un sistema de escritorio para la Farmacia "Vida Sana" en Juigalpa.

GRUPO 1

DOCENTES:

- José Jonathan Moreno Castro
- Joel Isidro Solano Duartes
- Yesenia Sobeyda Téllez Gómez
- María Jorlenis Ríos Pérez
- Saira María Urbina Cienfuegos

INTEGRANTES:

- Jasser Josafaph Bermúdez Suazo
- Ana Gabriela Irías Murillo
- Eduardo Jesús Gudiel González
- Francis Nallely Ayala Calero

Septiembre, 2025
¡Universidad del Pueblo y para el Pueblo!

Índice

Contenido

1. Tema

Desarrollo de un sistema de escritorio para la Farmacia “Vida Sana” en Juigalpa

2. Objetivos

Identificar los procesos actuales de gestión en la farmacia “Vida Sana” que nos permita la identificación de requerimientos funcionales para el desarrollo de la aplicación de escritorio.

Diseñar un sistema de escritorio que integre los módulos de ventas y productos donde incluya interfaces de uso práctico, dando cumplimiento a las necesidades del cliente.

Desarrollar una aplicación de escritorio para la farmacia “Vida Sana”, que automatice el registro de ventas, control de medicamentos y generación de reportes, para hacer más eficiente las ventas, controlar mejor los medicamentos y mejorar la atención al cliente.

3. Descripción de la problemática

En la farmacia Vida Sana ubicada en Juigalpa, genera 30.000 C$ mensuales de venta la cual tiene los siguientes problemas: gestión manual de la empresa dedicada a la comercialización y distribución de medicamentos y productos de cuidado personal. Esta institución requiere una solución tecnológica que le permita mejorar el control de inventario, optimizar el proceso de ventas y facilitar la generación de reportes administrativos garantizando un servicio más eficiente para los clientes. El sistema propuesto consiste en una aplicación de escritorio para farmacias, diseñada para optimizar la gestión de medicamentos, control de inventario y atención al cliente.

3.1 Propuesta de Solución

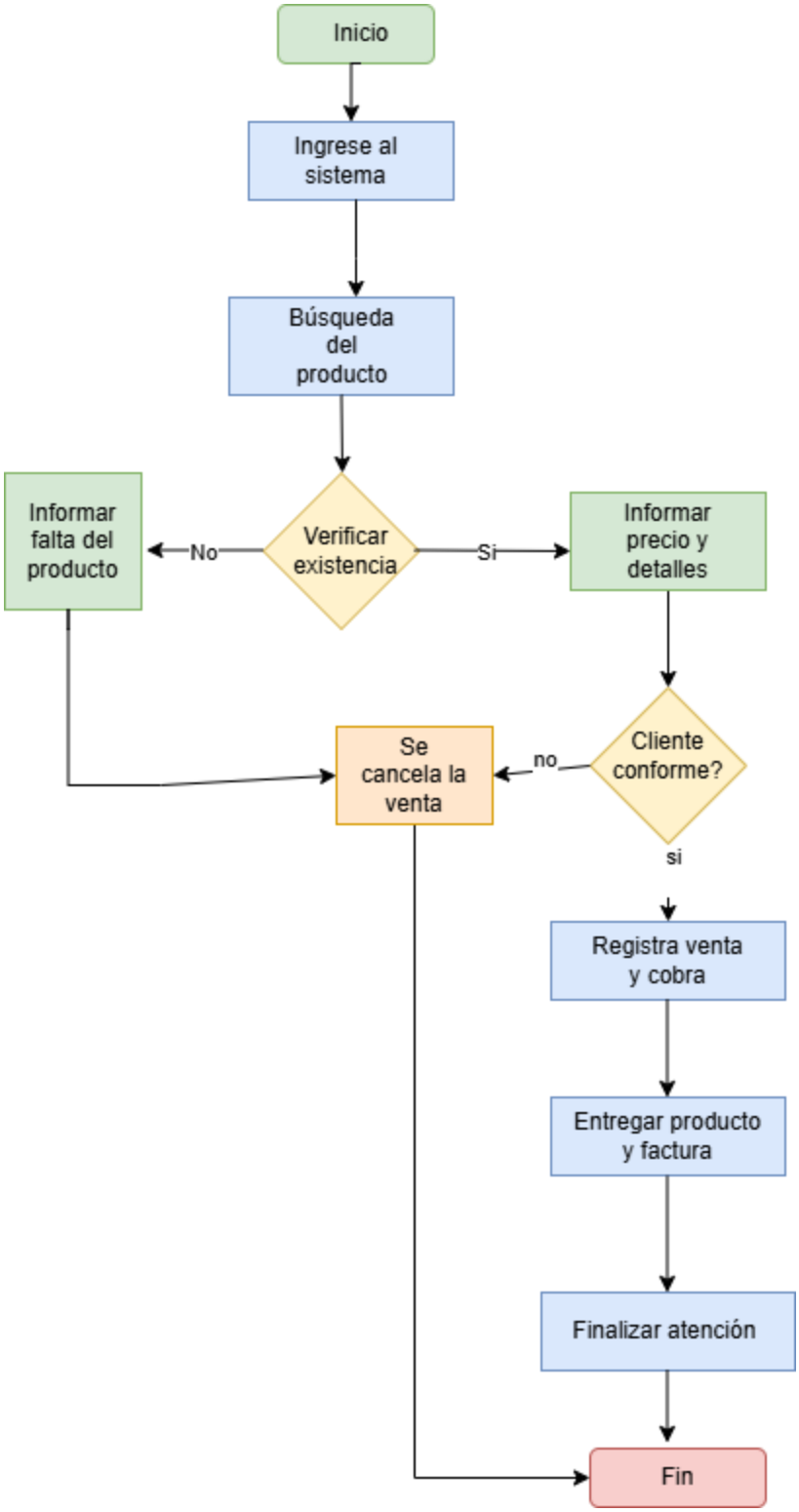
Este sistema permite a los administradores y empleados realizar registros de productos, verificar existencias, gestionar ventas, emitir facturas y recibos, así como llevar el control de fechas de caducidad. El objetivo principal es agilizar los procesos internos de la farmacia, reducir errores humanos y garantizar una mejor atención al usuario final. Además, el sistema contribuye a la toma de decisiones mediante reportes automatizados sobre movimientos de inventario y ventas.

El sistema integra tres áreas fundamentales: venta y atención al cliente, para registrar transacciones, emitir comprobantes y calcular totales; inventario y bodega, para controlar entradas, salidas y alertas sobre desabastecimiento; y administrativa, para generar reportes, gestionar proveedores. En conclusión, automatiza procesos claves que optimizan la gestión de la farmacia.

4. Funcionalidades del sistema

No.	Tipo de usuario	Funcionalidad	Descripción	Encargado
1	Administrador/ Vendedor	Gestión de productos	Registrar,consultar, actualizar y eliminar productos de la ED	Eduardo Guidiel
2	Administrador	Gestion de usuarios	Registrar y eliminar usuarios en el sistema	Ana Gabriela Irias
3	Empleado	Gestión de ventas de productos	Registrar o listar ventas y emitir facturas de ventas realizadas	Jasser Bermudez
4	Adminisreador/ Vendedor	Actualizar Stock	Actualizar nuevas cantidades de productos en el sistema	Jasser Bermudez
5	Empleado	Consultar productos	Consultar disponibilidad de producto en tiempo real	Francis Ayala
6	Administrador	Generar reportes	Los reportes incluyen parámetros como: el rango de fechas y por categoría.	Ana Gabriela Irias
7	Administrador/ Vendedor	Inicio de sesión	Iniciar sesión con un usuario creado dentro del sistema y que pueda acceder a los módulos que le corresponde.	Eduardo Gudiel

5. Mapeo del software de Vendedor



6. Planificación SCRUM

6.1. Tabla de roles del equipo de desarrollo

Rol	Nombre y Apellidos	Funciones
Product owner	Maria Aragón	Listar las necesidades y prioridades del producto y las toma decisiones sobre que se construye primero.
Scrum Master	Francis Ayala	Motivar al equipo de desarrollo para alcanzar los objetivos del proyecto.
DeveloPmet Team	Eduardo Gudiel Ana Irias Jasser Bermúdez	Desarrollar la lógica del sistema. Diseñar la interfaz del usuario, asegurando una experiencia visual y funcional adecuada. Probar las funcionalidades del sistema, identificar errores y verificar que el producto final cumpla con los requerimientos establecidos antes de su entrega.
Stakeholders	José Sequeira	Se encarga en observar los resultados de cada una de las acciones

6.2 Requerimientos funcionales

Requerimientos funcionales	
N°	**Descripción**
1	Permitir registrar nuevos medicamentos, actualizar precios, eliminar productos obsoletos y consultar inventario.
2	Registrar y eliminar usuarios en el sistema.
3	Registrar ventas, hacer cálculos automáticos de totales, cambios y emitir facturas de ventas realizadas
4	Realizar actualizaciones con la nueva cantidad del producto y especificar qué usuario la realizó.
5	consultar la disponibilidad y ver los detalles del producto.
6	Generar reportes de productos, de ventas y que puedan ser exportados a pdf.
7	Iniciar sesión con el nombre, correo, rol y contraseña para poder acceder a los módulos del sistema.

6.3 Historias de Usuario

HU01– Gestión de Productos			
Como	Administrador		
Quiero	Registrar, consultar, actualizar y eliminar medicamentos y productos disponibles		
Para	Mantener un control ordenado del inventario y resolver problemas relacionados con información incorrecta o desactualizada		
Validación		**Impacto**	32

(Opcional)	- Registrar un nuevo medicamento con sus datos principales. - Consultar la lista de productos y detalles. - Actualizar información (precio, vencimiento). - Eliminar productos descontinuados o vencidos.	**Prioridad**	5
		Estimación	14 horas

HU02-Agregar usuarios			
Como	Administrador		
Quiero	Gestionar nuevos usuarios en el sistema		
Para	Darles acceso con un rol definido y controlar quién puede operar en el sistema		
Validación **(Opcional)**	Puede agregar usuarios y visualizar cada uno de los detalles ingresados. El sistema permite ingresar nombre completo, correo y rol. Registra al usuario si los datos son válidos. El sistema evita registrar correos duplicados.	Impacto	16
		Prioridad	3
		Estimación	12 Hrs

HU03- Registrar ventas			
Como	Empleados		
Quiero	Registrar las ventas realizadas y emitir facturas a los clientes		
Para	Mantener control de las transacciones y entregar comprobantes válidos a los compradores		
Validación **(Opcional)**	El sistema permite registrar una venta con productos, cantidades y precios Se genera automáticamente la factura (impresa o digital). Las ventas quedan almacenadas para consultas y reportes. El inventario se actualiza de manera automática después de cada venta.	Impacto	32
		Prioridad	5
		Estimación	12hrs

HU04-Actualizar Stock			
Como	Administrador, vendedor		
Quiero	Actualizar el stock de los productos en el sistema		
Para	Mantener el inventario al día y evitar faltantes o excesos de productos		
Validación **(Opcional)**	Puede modificar la cantidad de stock disponible y visualizar el historial de cambios realizados en los productos.	Impacto	20
		Prioridad	3
		Estimación	12 hrs

HU05- Consultar Productos			
Como	Vendedor		
Quiero	Consultar la lista de productos registrados en el sistema de la farmacia.		
Para	Visualizar información importante como nombre, código, cantidad disponible, precio y fecha de vencimiento, con el fin de gestionar correctamente el inventario y realizar ventas precisas.		
Validación (Opcional)	El sistema debe mostrar correctamente todos los productos almacenados en la base de datos y permitir la búsqueda por nombre o código.	**Impacto**	20
		Prioridad	3
		Estimación	12 hrs

HU06- Reportes			
Como	Administrador		
Quiero	Recibir alertas de los medicamentos que están próximos a vencer		
Para	Retirar productos antes de que caduquen		
Validación (opcional)	El sistema notifica medicamentos con fecha de vencimiento El administrador puede consultar un listado de los próximos a vencer.	**Impacto**	16
		Prioridad	3
		Estimación	10 Hrs

	Las alertas se muestran en el sistema o por notificación (según configuración)		

HU07- Inicio de Sesión			
Como	Usuario		
Quiero	Iniciar sesión en el sistema de la farmacia mediante mi nombre de usuario y contraseña.		
Para	Acceder a mis funciones de trabajo de forma segura y proteger la información del sistema de usuarios no autorizados.		
Validación (Opcional)	El sistema debe verificar que el usuario y la contraseña coincidan con los datos registrados en la base de datos. En caso contrario, mostrar un mensaje de error.	**Impacto**	16
		Prioridad	3
		Estimación	9 Hrs

6.4 Formato Product Backlog Priorizado

ID	Historia de usuarios	Importancia	Spring programado	Fecha de inicio	Fecha fin
HU01	Gestión de productos	160	1	20/10/2025	23/10/2025
HU03	Registro de ventas	160	1	24/10/2025	27/10/2025
HU04	Actualizar stock	60	2	28/10/2025	31/10/2025
HU02	Agregar Usuarios	60	2	31/10/2025	02/11/2025
HU05	Consultar productos	48	3	02/11/2025	04/11/2025
HU06	Reportes	48	3	05/11/2025	10/10/2025
HU07	Inicio de sesión	48	3	11/11/2025	14/11/2025

6.5 Sprint Backlog 1

ID	Historia de Usuario	Tarea	Tiempo	Responsable
HU01	Agregar productos	Diseño de la interfaz de registro de productos	4hrs	Eduardo Gudiel

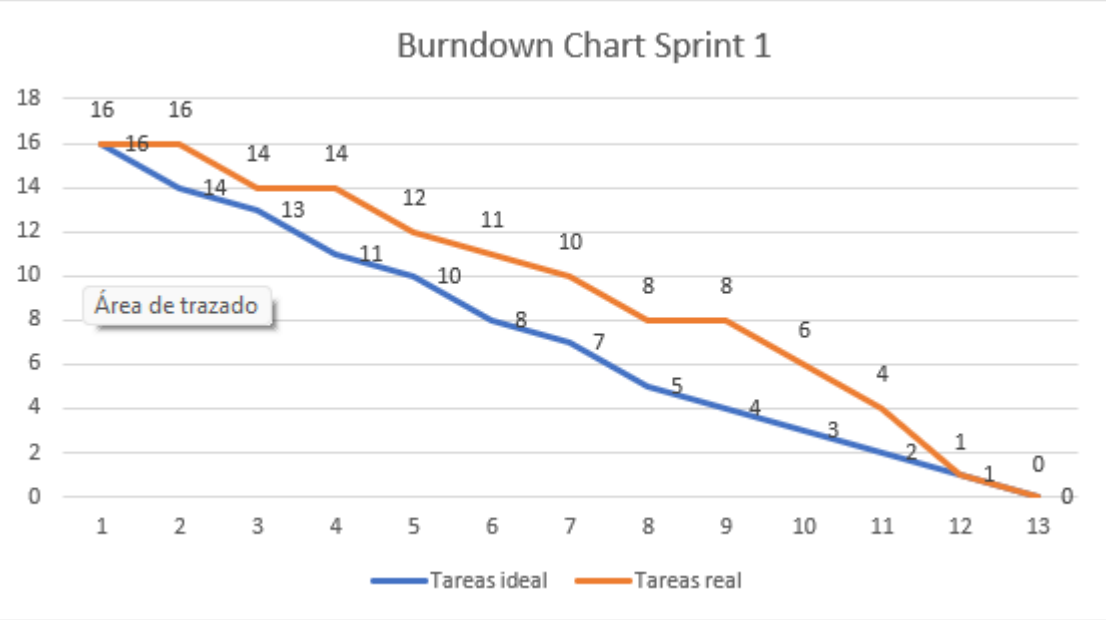
		Programar validación de campos obligatorios y del tipo correcto	2 hrs	Eduardo Gudiel
		Programar la estructura de datos: arreglos y clases	2hrs	Francis Ayala
		Programar la actualización de un producto	1hr	Jasser Bermúdez
		Programar el registro del producto en la estructura de datos.	2hrs	Francis Ayala
		Programar la eliminación del producto.	1hr	Jasser Bermúdez
		programar la visualización de productos.	1hr	Ana Irias
		Pruebas de funcionalidad.	1 hr	Jasser Bermúdez
HU03	Registrar Venta	Diseño de la interfaz de venta (formulario con buscador y detalle).	3 hrs	Ana Irias

		Programar validaciones de datos de entrada.	4 hrs	Ana Irias
		Programar la búsqueda de productos para agregar al detalle de venta.	2 hrs	Eduardo Gudiel
		Programar las validaciones de la venta (que la cantidad no supere al stock).	1hrs	Francis Ayala
		Programar actualización de inventario.	2 hrs	Jasser Bermúdez
		Programar registro de venta.	1hrs	Ana Irias
		Programar generación de factura .	2hr	Eduardo Gudiel
		Realizar pruebas de validación .	2 hrs	Francis Ayala

6.6 Burn down Chart del Sprint 1

En este Sprint, al cual en base una perspectiva optimista se le ha asignado una duración de 13 días, se considera un equipo dedicado al desarrollo de 3 miembros con un trabajo ideal de 5 horas al día.

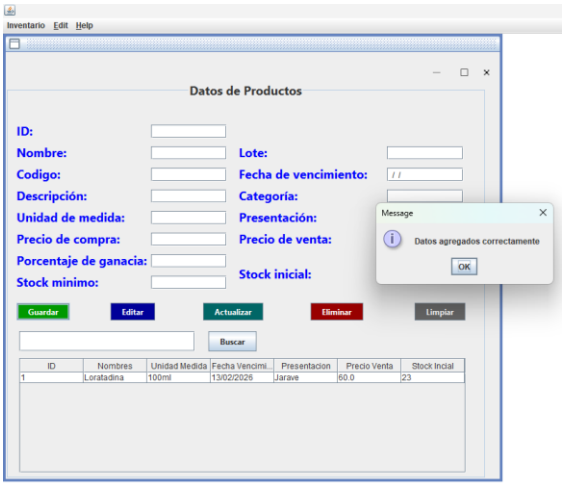
La figura muestra el contraste de la pendiente de trabajo ideal versus la real. Como se observa al inicio del sprint 1 se contabilizan 16 tareas por realizar y se lograron culminar en el tiempo establecido, nos indica que en el siguiente sprint lo podemos calcular de la misma manera asignando la misma cantidad de tareas.

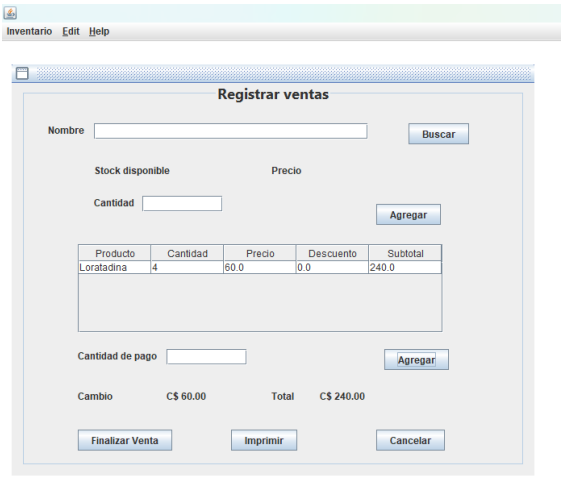


6.7 Resultados del Sprint 1

La figura permite observar la creación del formulario de agregar productos que muestra al usuario los productos agregados. La interfaz de productos es un módulo totalmente funcional a como se observa en la figura ya se pueden agregar productos en el sistema también se pueden eliminar, editar y actualizar.

La figura de registrar ventas muestra que se puede realizar una búsqueda de productos, agregarlos a la venta y hacer cálculos automáticos de totales y cambio.





6.8 Sprint Backlog 2

ID	Historia de usuario	Tarea	Tiempo	Responsable
HU04	Actualizar stock	Diseño de la interfaz de actualizar stock (campos, botones y navegación lateral).	3 hrs	Francis Ayala
		Programar la validación de los campos obligatorios y del tipo correcto.	2 hrs	Eduardo Gudiel
		Programar el registro del producto en la estructura de datos.	1 hrs	Ana Irias
		Programar la actualización de un producto.	1 hrs	Ana Irias

		Programar los botones de acción (actualizar y cancelar).	2 hrs	Jasser Bermúdez
		Prueba de funcionalidad.	1 hrs	Eduardo Gudiel
HU02	Agregar Usuarios	Diseño de la interfaz de usuarios agregados.	3 hrs	Ana Irias
		Programar la carga y visualización de la lista de usuarios (nombre, correo, rol, estado) en la tabla.	1 hrs	Ana Irias
		Programar la funcionalidad del botón Editar a los usuarios en la tabla.	1 hrs	Eduardo Gudiel
		Implementar la validación de los usuarios que pueden hacer uso del sistema.	1 hrs	Francis Ayala
		Programar la funcionalidad para registrar un nuevo usuario con validación de datos (incluyendo rol).	2 hrs	Jasser Bermúdez
		Programar la funcionalidad del botón “Editar”.	1 hrs	Eduardo Gudiel
		Programar la lógica para actualizar los datos y el “Estado” del usuario en la estructura de datos.	2 hrs	Jasser Bermúdez
		Programar el botón “Eliminar”.	1 hrs	Francis Ayala

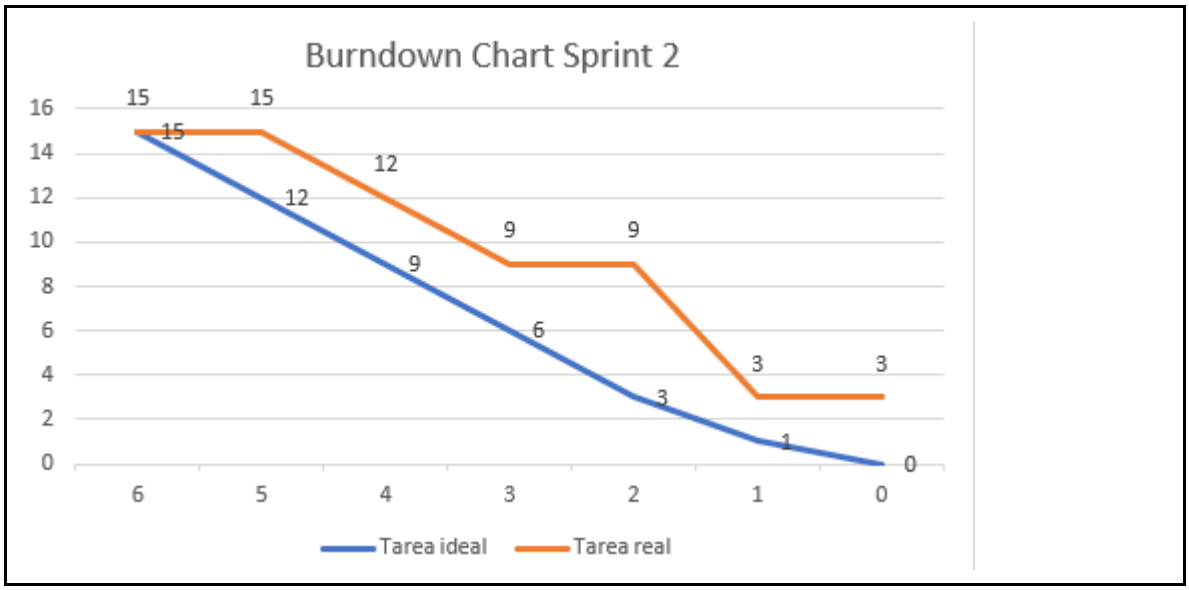
		Prueba de funcionalidad.	1 hrs	Eduardo Gudiel

6.6.1 Burn Down Chart del Sprint 2

Al Sprint se le ha asignado una duración de 6 días, se considera un equipo dedicado al desarrollo de 3 miembros con un trabajo ideal de 3 horas al día.

La figura muestra el contraste de la pendiente de trabajo ideal versus la real. Como se observa al inicio del sprint 2 se contabilizan 15 tareas por realizar, pero en términos reales 3 no llegan a culminarse en el tiempo establecido.

La funcionalidad que quedó inconclusa en este Sprint deberá desarrollarse fuera del sprint.

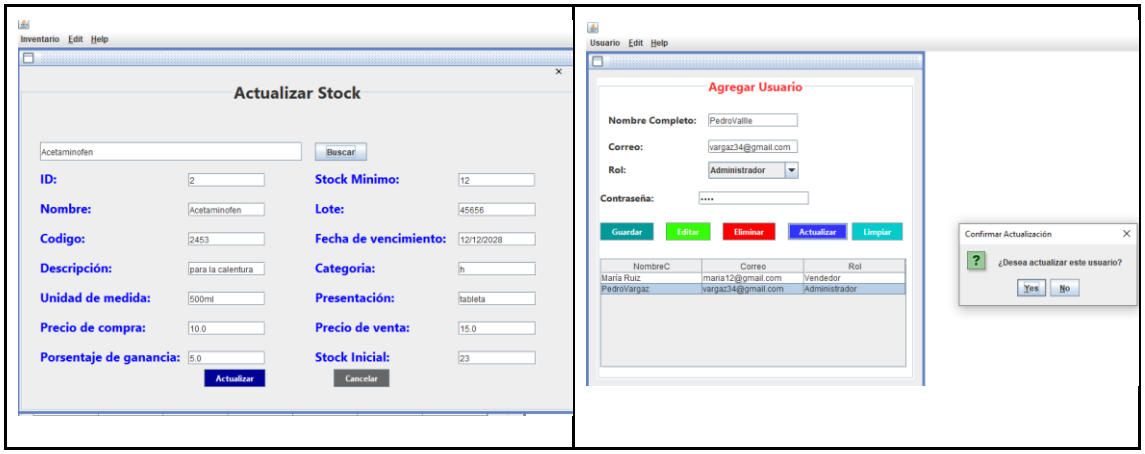


6.6.2 Resultados del Sprint 2

La figura permite observar la creación del formulario para actualizar el stock de productos, el cual muestra al usuario la cantidad disponible y permite realizar modificaciones de manera rápida. La interfaz de stock

es un módulo totalmente funcional; como se aprecia en la figura, es posible incrementar o disminuir existencias, así como guardar los cambios realizados en el sistema.

Asimismo, la figura correspondiente al módulo de agregar usuarios muestra un formulario en el que se pueden registrar nuevos usuarios ingresando datos esenciales como nombre, correo y rol dentro del sistema. Este módulo permite guardar, editar y actualizar información de los usuarios registrados, garantizando un control adecuado de quienes tienen acceso al sistema, la acción de los botones eliminar, limpiar y validar la creación de usuarios aun esta inconcluida la funcionalidad.



6.6 Sprint Backlog 3

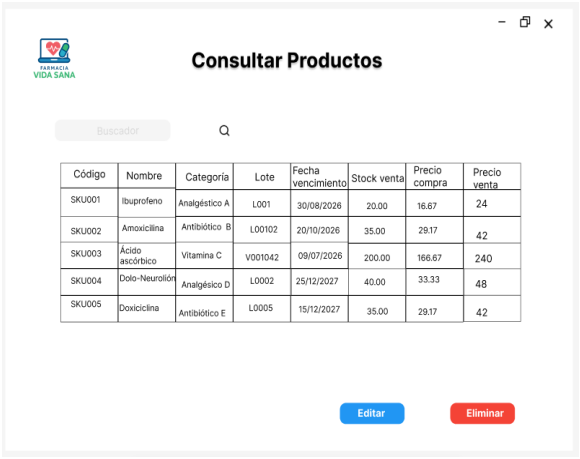
ID	Historia de Usuario	Tarea	Tiempo	Responsable
HU06	Reportes de venta	Diseñar la interfaz del reporte de ventas.	3 hrs	Francis Ayala
		Programar la generación automática del reporte (por fecha o producto).	hrs	Eduardo Gudiel

		Permitir exportar o imprimir el reporte.	2 hrs	Jasser Bermúdez
		Validar que solo usuarios autorizados puedan acceder.	1 hrs	Ana Irias
HU05	Consultar Producto	Diseño de la interfaz de consulta de productos	3 hrs	Eduardo Gudiel
		Programar la visualización de la lista de productos en la estructura de datos.	2 hrs	Eduardo Gudiel
		Programar el funcionamiento del buscador (búsqueda de productos por nombre o código).	1 hrs	Francis Ayala
		Programar el filtro y ordenamiento de productos en la tabla.	2 hrs	Ana Irias

		Programar los botones de acción Editar y Eliminar en las filas de productos.	2 hrs	Jasser Bermúdez
		Programar validación al realizar consultas (evitar campos vacíos).	1 hrs	Jasser Bermúdez
		Realizar pruebas de validación.	1 hrs	Francis Ayala
HU07	Inicio de Sesión	Diseñar la interfaz del formulario (Usuario contraseña y botón de ingreso).	3 hrs	Francis Ayala
		Redirigir al menú principal si el inicio es correcto.	2 hrs	Eduardo Gudiel
		Conectar la estructura de datos para verificar usuario y contraseña.	2 hrs	Jasser Bermúdez

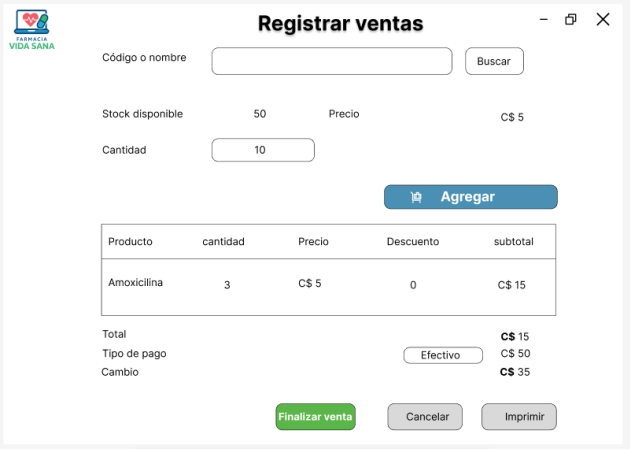
		Implementar opción para cerrar sesión para salir del sistema segura.	1 hrs	Ana Irias

7. Interfaces de usuario figma y enlace.




Agregar Usuarios
Nombre Completo:
Correo:
Rol:
Administrador
Contraseña:
Guardar
Editar
Actualizar
Eliminar
Limpiar
Nombre Completo
Correo
Rol
Juan Pérez
jperez12@gmail.com
Administrador
Ana Gómez
agomez78@gmail.com
Vendedor
Pedro Martinez
pmartinez9@gmail.com
Vendedor


Repote de ventas
Reportes por mes
Ventas totales
C$ 15,250.00
Número de ventas
235
Productos mas vedidos
Ibuprofeno, Antimicina
Detalle de transacciones
ID Fecha Producto Codigo Lote Cantidad Precio Metodo pago Total
01 02/04/25 Tapcin 70021625 250116 5 C$ 6 Efectivo C$ 18
02 02/04/25 Loratadina 89011385 E022502 2 C$ 4 Efectivo C$ 8
03 03/04/25 Amoxicilina 89011382 1124401 10 C$ 5 Efectivo C$ 50
Reporte de productos a vencer
Exportar
Salir


Registrar ventas
Código o nombre
Buscar
Stock disponible
50
Precio
C$ 5
Cantidad
10
Agregar
Producto
cantidad
Precio
Descuento
subtotal
Amoxicilina
3
C$ 5
0
C$ 15
Total
C$ 15
Tipo de pago
Efectivo
C$ 50
Cambio
C$ 35
Finalizar venta
Cancelar
Imprimir

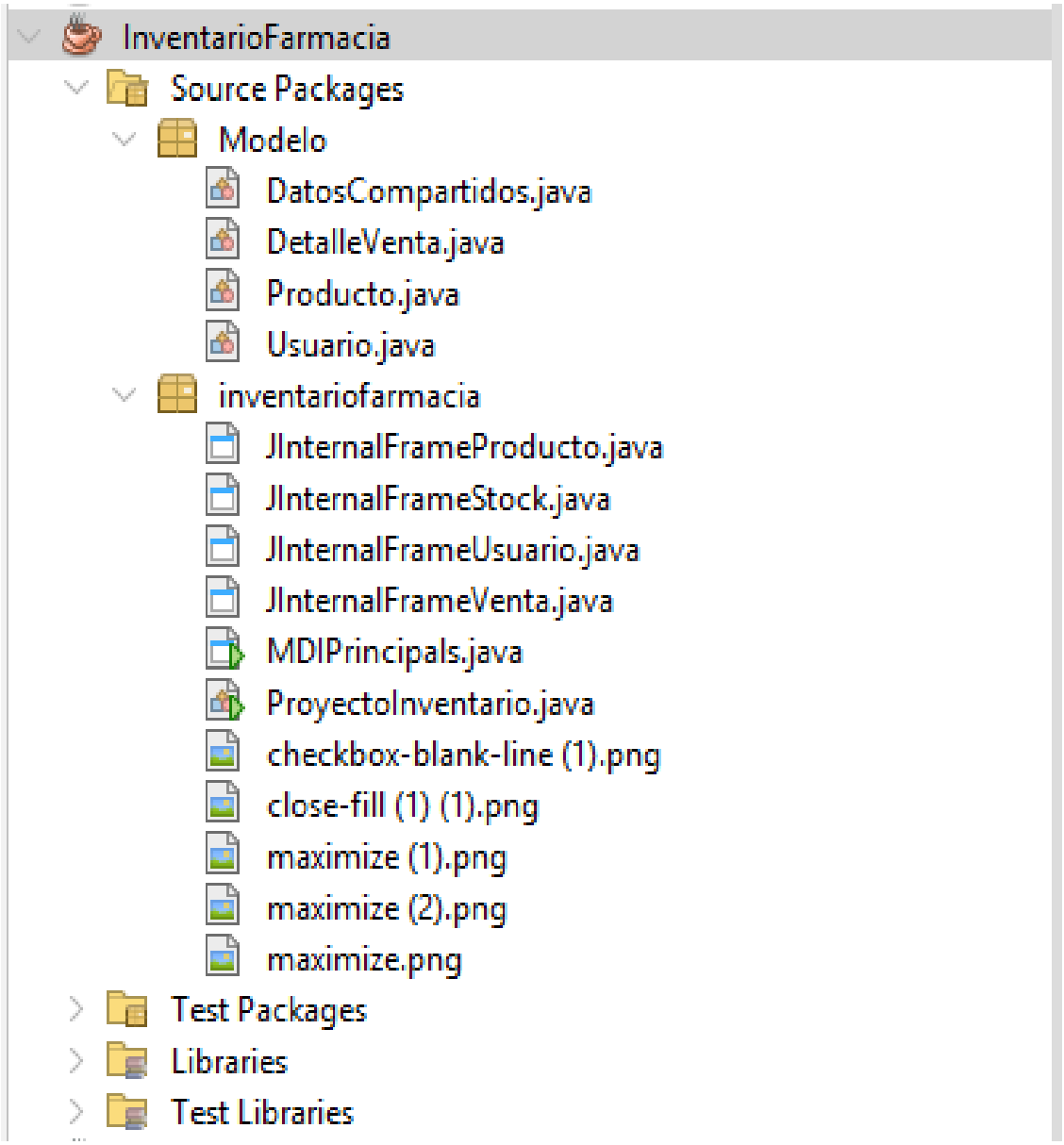

Inicio de sesión
Inicio de sesión
Correo
Contraseña
Recordarme
Iniciar Sesión
Continuar con Google
¿Olvidó su contraseña ?
Crear una cuenta


Inicio
Blenvenido
FARMACIA
VIDA SANA
Inicio

7.1 Enlace figma:

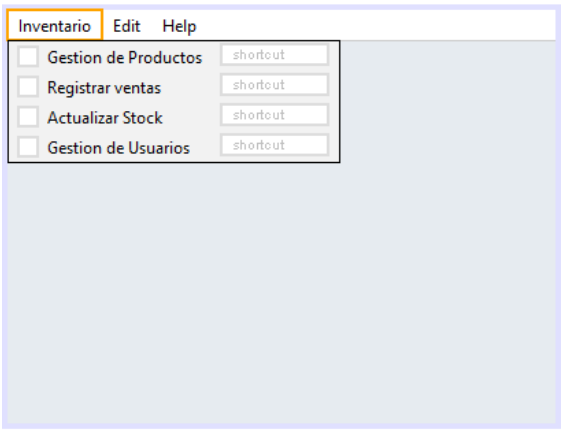
https://www.figma.com/design/Y9iRhPMGjEVMgRi2ViadmV/Inventario?node-id=0-1&t=ZrwaIQ7V1AvZzLzZ-1

8. Estructura del proyecto en NetBeans



9. Captura de interfaz y programación

9.1 Paquete Controlador: Formulario MDI (Interfaz y código).

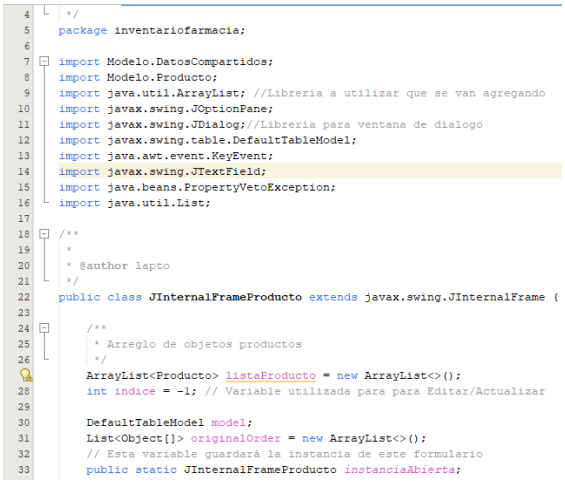



```java
package inventariofarmacia;

/**
 *
 * @author lapto
 */
public class MDIPrincipals extends javax.swing.JFrame {

    /**
     * Creates new form MDIPrincipals
     */
    public MDIPrincipals() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void GestionUsuariosMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        JInternalFrameUsuario jframeUsuario = new JInternalFrameUsuario();
        desktopPane.add(jframeUsuario);
        jframeUsuario.show();
    }
```

```java
    private void gestionProductoMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        JInternalFrameProducto jframeProducto = new JInternalFrameProducto();
        desktopPane.add(jframeProducto);
        jframeProducto.show();
    }

    private void RegistroVentaMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        JInternalFrameVenta jframeVenta = new JInternalFrameVenta();
        desktopPane.add(jframeVenta);
        jframeVenta.show();
    }

    private void actualizarStockMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        JInternalFrameStock jframeStock = new JInternalFrameStock();
        desktopPane.add(jframeStock);
        jframeStock.show();
    }
    //System.exit(0);

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MDIPrincipals().setVisible(true);
```

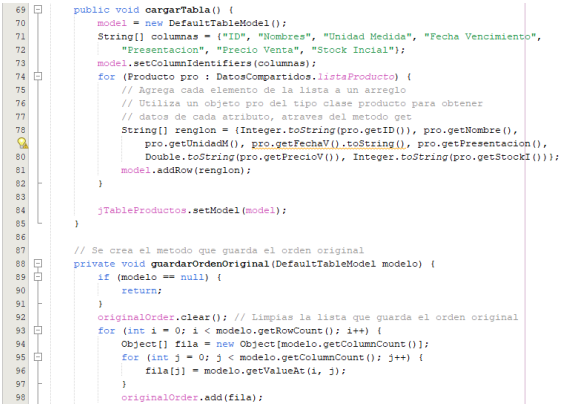
```java
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MDIPrincipals().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JMenuItem GestionUsuariosMenuItem;
    private javax.swing.JMenuItem RegistroVentaMenuItem;
    private javax.swing.JMenuItem aboutMenuItem;
    private javax.swing.JMenuItem actualizarStockMenuItem;
    private javax.swing.JMenuItem contentMenuItem;
    private javax.swing.JMenuItem copyMenuItem;
    private javax.swing.JMenuItem cutMenuItem;
    private javax.swing.JMenuItem deleteMenuItem;
    private javax.swing.JDesktopPane desktopPane;
    private javax.swing.JMenu editMenu;
    private javax.swing.JMenuItem gestionProductoMenuItem;
    private javax.swing.JMenu helpMenu;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenuItem pasteMenuItem;
    private javax.swing.JMenu productoMenu;
    // End of variables declaration

}
```

9.2 Paquete Vista: un formulario para cada entidad para realizar todas las transacciones (Interfaz y código).



```
 */
package inventariofarmacia;

import Modelo.DatosCompartidos;
import Modelo.Producto;
import java.util.ArrayList; //Libreria a utilizar que se van agregando
import javax.swing.JOptionPane;
import javax.swing.JDialog;//Libreria para ventana de dialogo
import javax.swing.table.DefaultTableModel;
import java.awt.event.KeyEvent;
import javax.swing.JTextField;
import java.beans.PropertyVetoException;
import java.util.List;

/**
 *
 * @author lapto
 */
public class JInternalFrameProducto extends javax.swing.JInternalFrame {

    /**
     * Arreglo de objetos productos
     */
    ArrayList<Producto> listaProducto = new ArrayList<>();
    int indice = -1; // Variable utilizada para para Editar/Actualizar

    DefaultTableModel model;
    List<Object[]> originalOrder = new ArrayList<>();
    // Esta variable guardará la instancia de este formulario
    public static JInternalFrameProducto instanciaAbierta;
```

```
    public JInternalFrameProducto() {  // Constructor del formulario
        initComponents();
        model = (DefaultTableModel) jTableProductos.getModel();

        // Le decimos a la variable estática
        instanciaAbierta = this;
        // Usamos el metodo para cargar la tabla
        cargarTabla();

    }

    private void limpiarCampos() { //Metodo para limpiar cajas de texto
        //se asigna un valor vacio a todas la cajas de texto
        //Se utiñiza el nombre de variable asignado
        this.jTextFieldID.setText(""); // Set para asignar un valor
        this.jTextFieldNombreP.setText("");
        this.jTextFieldCodigo.setText("");
        this.jTextFieldDescripcion.setText("");
        this.jTextFieldUnidadM.setText("");
        this.jTextFieldPrecioC.setText("");
        this.jTextFieldPorsentajeG.setText("");
        this.jTextFieldStockM.setText("");
        this.jTextFieldLote.setText("");
        this.FechaV.setText("");
        this.jTextFieldCategoria.setText("");
        this.jTextFieldPresentacion.setText("");
        this.jTextFieldPrecioV.setText("");
        this.jTextFieldStockI.setText("");

    }
```



```
    public void cargarTabla() {
        model = new DefaultTableModel();
        String[] columnas = {"ID", "Nombres", "Unidad Medida", "Fecha Vencimiento",
            "Presentacion", "Precio Venta", "Stock Incial"};
        model.setColumnIdentifiers(columnas);
        for (Producto pro : DatosCompartidos.listaProducto) {
            // Agrega cada elemento de la lista a un arreglo
            // Utiliza un objeto pro del tipo clase producto para obtener
            // datos de cada atributo, atraves del metodo get
            String[] renglon = {Integer.toString(pro.getID()), pro.getNombre(),
                pro.getUnidadM(), pro.getFechaV().toString(), pro.getPresentacion(),
                Double.toString(pro.getPrecioV()), Integer.toString(pro.getStockI())};
            model.addRow(renglon);
        }

        jTableProductos.setModel(model);
    }

    // Se crea el metodo que guarda el orden original
    private void guardarOrdenOriginal(DefaultTableModel modelo) {
        if (modelo == null) {
            return;
        }
        originalOrder.clear(); // Limpias la lista que guarda el orden original
        for (int i = 0; i < modelo.getRowCount(); i++) {
            Object[] fila = new Object[modelo.getColumnCount()];
            for (int j = 0; j < modelo.getColumnCount(); j++) {
                fila[j] = modelo.getValueAt(i, j);
            }
            originalOrder.add(fila);
```

```
    }
    // Crea el método para restaurar el orden original:
    public void restaurarOrden() {
        if (model == null || originalOrder.isEmpty()) {
            JOptionPane.showMessageDialog(null, "No hay un orden guardado para restaurar.", "Aviso", JOptionPane.WARNING_MESSAGE);
            return;
        }

        model.setRowCount(0); // Limpiar tabla
        for (Object[] row : originalOrder) {
            model.addRow(row);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButtonLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
        //Llama al metodo limpiar campos
        limpiarCampos();
    }

    private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {
        //validar que todos los campos de texto no esten vacios
```

```
private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    //validar que todos los campos de texto no esten vacios
    if ((jTextFieldID.getText().contentEquals(""))
            || (jTextFieldNombreP.getText().contentEquals(""))
            || (jTextFieldCodigo.getText().contentEquals(""))
            || (jTextFieldDescripcion.getText().contentEquals(""))
            || (jTextFieldUnidadM.getText().contentEquals(""))
            || (jTextFieldPrecioC.getText().contentEquals(""))
            || (jTextFieldPorsentajeG.getText().contentEquals(""))
            || (jTextFieldStockM.getText().contentEquals(""))
            || (jTextFieldLote.getText().contentEquals(""))
            || (FechaV.getText().contentEquals(""))
            || (jTextFieldCategoria.getText().contentEquals(""))
            || (jTextFieldPresentacion.getText().contentEquals(""))
            || (jTextFieldPrecioV.getText().contentEquals(""))
            || (jTextFieldStockI.getText().contentEquals(""))) {
        //Mensaje para informar si algun campo esta vacio // Agrego import
        JOptionPane.showMessageDialog(null,
                "Algunos campos estan vacios",
                "!Revise todos los campos!",
                JOptionPane.WARNING_MESSAGE);
    } else {
        //Get para obtener un valor
        //obtiene lo que esta en la caja de texto y lo convierte a tipo de dato
        int idProducto = Integer.parseInt(this.jTextFieldID.getText());
        String nombreProducto = this.jTextFieldNombreP.getText();
        String codigo = this.jTextFieldCodigo.getText();
        String descripcion = this.jTextFieldDescripcion.getText();
        String unidadMedida = this.jTextFieldUnidadM.getText().trim();
        Double precioCompra = Double.parseDouble(this.jTextFieldPrecioC.getText());
```

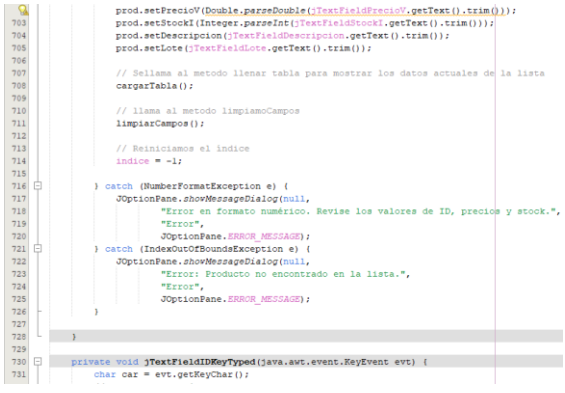
```
        String unidadMedida = this.jTextFieldUnidadM.getText().trim();
        Double precioCompra = Double.parseDouble(this.jTextFieldPrecioC.getText());
        Double porsentajeGanancia = Double.parseDouble(this.jTextFieldPorsentajeG.getText());
        int stockMinimo = Integer.parseInt(this.jTextFieldStockM.getText());
        String fechaVencimiento = this.FechaV.getText();
        String nombreCategoria = this.jTextFieldCategoria.getText();
        String presentacion = this.jTextFieldPresentacion.getText();
        Double precioVenta = Double.parseDouble(this.jTextFieldPrecioV.getText());
        int stockInicial = Integer.parseInt(this.jTextFieldStockI.getText());
        String loteProducto = this.jTextFieldLote.getText();

        try { //Bloque try (intentar) - catch (capturar) para capturar errores
            //Declarando un objeto de la clase producto,
            //cree un objeto de la clase producto
            //llama al metodo constructor de la clase producto
            Producto prod = new Producto(idProducto, nombreProducto, codigo, unidadMedida, precioCompra, porsentajeGanancia,
                    stockMinimo, fechaVencimiento, nombreCategoria, presentacion, precioVenta, stockInicial, descripcion, loteProducto);
            //Agrega al arreglo de objetos los datos del objeto prod
            // this.listaProducts.add(prod);
            DatosCompartidos.agregarProducto(prod);

            this.limpiarCampos(); //llama al metodo limpiar campos

            cargarTabla(); // Llama al metodo llenarTabla

            JOptionPane.showMessageDialog(null,
                    "Datos agregados correctamente");

        } catch (NumberFormatException nfe) {
            nfe.printStackTrace();
```

```
        } catch (NumberFormatException nfe) {
            nfe.printStackTrace();
            JOptionPane.showMessageDialog(null,
                    "Verifique que los camos númericos sean validos.",
                    "Error de formato",
                    JOptionPane.ERROR_MESSAGE);

        } catch (Exception e) { //capturar el error
            e.printStackTrace();
            JOptionPane.showMessageDialog(null,
                    "Datos no agregados:",
                    "Error",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jButtonEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        int fila = this.jTableProductos.getSelectedRow(); //Se obtiene #fila seleccionada
        // Se verifica que este seleccionada una fila de la tabla
        if (fila == -1) {
            JOptionPane.showMessageDialog(rootPane,
                    "Seleccione un registro de la tabla");
        } else {
            // Ventana de dialogo para preguntar si desea eliminar
            JDialog.setDefaultLookAndFeelDecorated(true);
            int resp = JOptionPane.showConfirmDialog(null,
                    "¿Esta seguro de eliminar?", "Eliminado...",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE);
```

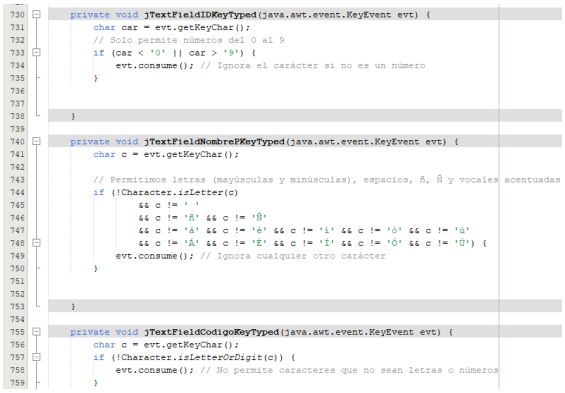
```
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE);
            // Si la respuesta es si se elimina
            if (resp == JOptionPane.YES_OPTION) {
                // Metodo remove para quitar de la lista de la fila seleccionada
                // Se le debe indicar la posicion del elemento de la lista
                // la numeracion de filas de la tabla es igual a las posiciones de la lista
                DatosCompartidos.listaProducto.remove(fila);
                // Se llama al metodo llenarTabla para mostrar los datos actuales de la lista
                cargarTabla();
            }
        }
    }

    private void jButtonEditarActionPerformed(java.awt.event.ActionEvent evt) {
        // Se verifica que este seleccionada una fila de la tabla
        int fila = this.jTableProductos.getSelectedRow(); //se obtiene #fila selecionada
        if (fila == -1) {
            JOptionPane.showMessageDialog(rootPane,
                    "Seleccione un registro de la tabla");
            return;
        }
        // Se asigna a la variable global
        indice = fila; // Se actualiza la variable gloval

        // Accedemos al objeto completo desde la lista
        Producto prod = DatosCompartidos.listaProducto.get(indice);
        //se rellena los camos directamente des de el objeto

        jTextFieldID.setText(String.valueOf(prod.getID()));
```

```
        jTextFieldID.setText(String.valueOf(prod.getID()));
        jTextFieldNombreP.setText(prod.getNombre());
        jTextFieldCodigo.setText(prod.getCodigo());
        jTextFieldUnidadM.setText(prod.getUnidadM());
        jTextFieldPrecioC.setText(String.valueOf(prod.getPrecioC()));
        jTextFieldPorsentajeG.setText(String.valueOf(prod.getPorsentajeG()));
        jTextFieldStockM.setText(String.valueOf(prod.getStockM()));
        FechaV.setText(prod.getFechaV().toString());
        jTextFieldCategoria.setText(prod.getCategoria());
        jTextFieldPresentacion.setText(prod.getPresentacion());
        jTextFieldPrecioV.setText(String.valueOf(prod.getPrecioV()));
        jTextFieldStockI.setText(String.valueOf(prod.getStockI()));
        jTextFieldDescripcion.setText(prod.getDescripcion());
        jTextFieldLote.setText(prod.getLote());


    }

    private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
        // Validamos si selecciono un producto para editar que los campos no esten vacios
        if (jTextFieldID.getText().trim().isEmpty()
                || jTextFieldNombreP.getText().trim().isEmpty()
                || jTextFieldCodigo.getText().trim().isEmpty()
                || jTextFieldUnidadM.getText().trim().isEmpty()
                || jTextFieldPrecioC.getText().trim().isEmpty()
                || jTextFieldPorsentajeG.getText().trim().isEmpty()
                || jTextFieldStockM.getText().trim().isEmpty()
                || FechaV.getText().trim().isEmpty()
                || jTextFieldCategoria.getText().trim().isEmpty()
                || jTextFieldPresentacion.getText().trim().isEmpty()
```

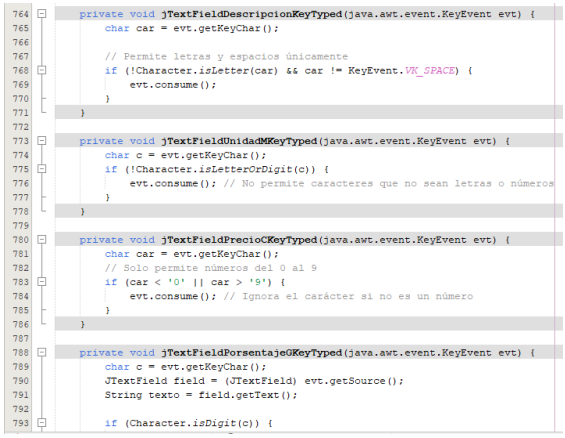
```
                || jTextFieldPresentacion.getText().trim().isEmpty()
                || jTextFieldPrecioV.getText().trim().isEmpty()
                || jTextFieldStockI.getText().trim().isEmpty()
                || jTextFieldDescripcion.getText().trim().isEmpty()
                || jTextFieldLote.getText().trim().isEmpty()
                || indice == -1) {
            // mensaje para informar si algunos campos estan vacios
            JOptionPane.showMessageDialog(null,
                    "Algunos campos están vacios o no ha seleccionado un producto",
                    "Actualizar...",
                    JOptionPane.WARNING_MESSAGE);
            limpiarCampos();
        }

        try {
            // Obtenemos el producto de la lista usando el indice
            Producto prod = DatosCompartidos.listaProducto.get(indice);

            // Actualizamos todos los campos
            prod.setID(Integer.parseInt(jTextFieldID.getText().trim()));
            prod.setNombre(jTextFieldNombreP.getText().trim());
            prod.setCodigo(jTextFieldCodigo.getText().trim());
            prod.setUnidadM(jTextFieldUnidadM.getText().trim());
            prod.setPrecioC(Double.parseDouble(jTextFieldPrecioC.getText().trim()));
            prod.setPorsentajeG(Double.parseDouble(jTextFieldPorsentajeG.getText().trim()));
            prod.SetStockM(Integer.parseInt(jTextFieldStockM.getText().trim()));
            prod.setFechaV(FechaV.getText().trim());
            prod.setCategoria(jTextFieldCategoria.getText().trim());
            prod.setPresentacion(jTextFieldPresentacion.getText().trim());
            prod.setPrecioV(Double.parseDouble(jTextFieldPrecioV.getText().trim()));
```



```
            prod.setPrecioV(Double.parseDouble(jTextFieldPrecioV.getText().trim()));
            prod.setStockI(Integer.parseInt(jTextFieldStockI.getText().trim()));
            prod.setDescripcion(jTextFieldDescripcion.getText().trim());
            prod.setLote(jTextFieldLote.getText().trim());

            // Sellama al metodo llenar tabla para mostrar los datos actuales de la lista
            cargarTabla();

            // llama al metodo limpiamoCampos
            limpiarCampos();

            // Reiniciamos el indice
            indice = -1;

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null,
                    "Error en formato numérico. Revise los valores de ID, precios y stock.",
                    "Error",
                    JOptionPane.ERROR_MESSAGE);
        } catch (IndexOutOfBoundsException e) {
            JOptionPane.showMessageDialog(null,
                    "Error: Producto no encontrado en la lista.",
                    "Error",
                    JOptionPane.ERROR_MESSAGE);
        }

    }

    private void jTextFieldIDKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
```

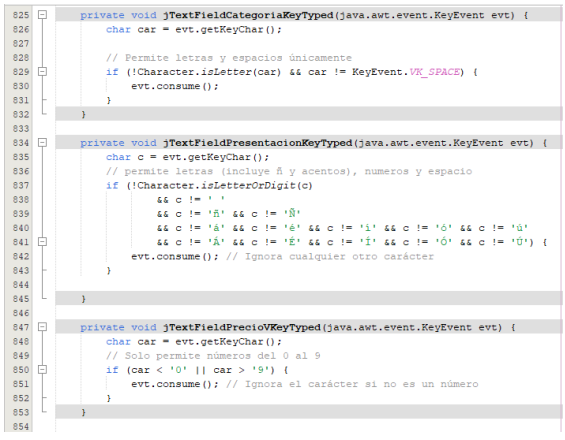


```
    private void jTextFieldIDKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }


    }

    private void jTextFieldNombrePKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();

        // Permitimos letras (mayúsculas y minúsculas), espacios, ñ, Ñ y vocales acentuadas
        if (!Character.isLetter(c)
                && c != ' '
                && c != 'ñ' && c != 'Ñ'
                && c != 'á' && c != 'é' && c != 'í' && c != 'ó' && c != 'ú'
                && c != 'Á' && c != 'É' && c != 'Í' && c != 'Ó' && c != 'Ú') {
            evt.consume(); // Ignora cualquier otro carácter
        }


    }

    private void jTextFieldCodigoKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }
```



```
    private void jTextFieldDescripcionKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();


        // Permite letras y espacios únicamente
        if (!Character.isLetter(car) && car != KeyEvent.VK_SPACE) {
            evt.consume();
        }
    }

    private void jTextFieldUnidadMKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }
    }

    private void jTextFieldPrecioCKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jTextFieldPorsentajeGKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        JTextField field = (JTextField) evt.getSource();
        String texto = field.getText();

        if (Character.isDigit(c)) {
```

```
        if (Character.isDigit(c)) {
            if (texto.contains("%")) {
                // No permitir números después del %
                evt.consume();
            }
        } else if (c == '%') {
            if (texto.contains("%") || texto.isEmpty()) {
                // No permitir más de un % o si se escribe como primer carácter
                evt.consume();
            }
        } else {
            // Bloquear cualquier otro carácter
            evt.consume();
        }

    }

    private void jTextFieldStockMKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jTextFieldLoteKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }
```



```
    private void jTextFieldCategoriaKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();


        // Permite letras y espacios únicamente
        if (!Character.isLetter(car) && car != KeyEvent.VK_SPACE) {
            evt.consume();
        }
    }

    private void jTextFieldPresentacionKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        // permite letras (incluye ñ y acentos), numeros y espacio
        if (!Character.isLetterOrDigit(c)
                && c != ' '
                && c != 'ñ' && c != 'Ñ'
                && c != 'á' && c != 'é' && c != 'í' && c != 'ó' && c != 'ú'
                && c != 'Á' && c != 'É' && c != 'Í' && c != 'Ó' && c != 'Ú') {
            evt.consume(); // Ignora cualquier otro carácter
        }


    }

    private void jTextFieldPrecioVKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }
```

```
    private void jTextFieldStockIKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jLabelMinimizarMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            this.setIcon(true);

        } catch (PropertyVetoException ex) {
            ex.printStackTrace();
        }


    }

    private void jLabelMaximizarMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            this.setMaximum(true);

        } catch (PropertyVetoException ex) {
            ex.printStackTrace();
        }
    }

    private void jLabelCerrarMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            this.setClosed(true);
```

```
    private void jLabelCerrarMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            this.setClosed(true);

        } catch (PropertyVetoException ex) {
            ex.printStackTrace();
        }
    }

    private void jButtonBuscarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            // Verifica que la tabla está inicializada
            if (jTableProductos == null) {
                JOptionPane.showMessageDialog(this, "La tabla no está inicializada.", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            // Obtiene el modelo de la tabla
            if (model == null) {
                model = (DefaultTableModel) jTableProductos.getModel();
            }

            // Verifica si la tabla tiene filas
            if (model.getRowCount() == 0) {
                JOptionPane.showMessageDialog(this, "No hay productos en la tabla.", "Aviso", JOptionPane.WARNING_MESSAGE);
                return;
            }

            // Guarda el orden original solo la primera vez
            if (originalOrder.isEmpty()) {
```


```java
package inventariofarmacia;

import Modelo.DatosCompartidos;
import Modelo.DetalleVenta;
import Modelo.Producto;
import java.beans.PropertyVetoException;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author lapto
 */
public class JInternalFrameVenta extends javax.swing.JInternalFrame {

    ArrayList<DetalleVenta> listaVenta = new ArrayList<>();
    DefaultTableModel modeloTabla;
    double total = 0;

    /**
     * Creates new form JInternalFrameVenta
     */
    public JInternalFrameVenta() {
        initComponents();

        inicializarTabla();

    }
```

```java
    // Configurar tabla
    private void inicializarTabla() {
        modeloTabla = new DefaultTableModel(
                new Object[]{"Producto", "Cantidad", "Precio", "Descuento", "Subtotal"}, 0
        );
        jTableProductos.setModel(modeloTabla);

    }

    // CALCULAR CAMBIO AUTOMÁTICAMENTE
    private void txtipodepagoActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            double pago = Double.parseDouble(txtipodepago.getText());
            double cambio = pago - total;
            jLabel23.setText("C$ " + String.format("%.2f", cambio));
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Ingrese una cantidad de pago válida.");
        }

    }
```

```java
    private void buscarActionPerformed(java.awt.event.ActionEvent evt) {
        String nombre = nombrebusca.getText().trim();

        Producto encontrado = DatosCompartidos.buscarProductoPorNombre(nombre);

        if (encontrado != null) {
            cerostock.setText(String.valueOf(encontrado.getStockI()));
            ceroprecio.setText(String.valueOf(encontrado.getPrecioV()));
        } else {
            JOptionPane.showMessageDialog(this, "Producto no encontrado");
            cerostock.setText("");
            ceroprecio.setText("");
        }

    }

    private void txtcantidadActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void agregarActionPerformed(java.awt.event.ActionEvent evt) {
        String nombreBuscado = nombrebusca.getText().trim();
        if (nombreBuscado.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Ingrese un nombre de producto.");
            return;
        }

        Producto encontrado = null;
        for (Producto p : DatosCompartidos.listaProducto) {
            if (p.getNombre().equalsIgnoreCase(nombreBuscado)) {
```

```java
        Producto encontrado = null;
        for (Producto p : DatosCompartidos.listaProducto) {
            if (p.getNombre().equalsIgnoreCase(nombreBuscado)) {
                encontrado = p;
                break;
            }
        }

        if (encontrado == null) {
            JOptionPane.showMessageDialog(this, "Producto no encontrado.");
            return;
        }

        int cantidad;
        try {
            cantidad = Integer.parseInt(txtcantidad.getText());
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Ingrese una cantidad válida.");
            return;
        }

        //Validar que la cantidad sea positiva
        if (cantidad <= 0) {
            JOptionPane.showMessageDialog(null, "La cantidad debe ser mayor a cero.");
            return;
        }

        //  Validacion de stock (se detiene la venta sino hay exixtencia)
        if (cantidad > encontrado.getStockI()) {
```

```java
    if (cantidad > encontrado.getStockI()) {
        JOptionPane.showMessageDialog(null, "Stock insuficiente. Solo quedan " + encontrado.getStockI() + " unidades.");
        return;
    }

    // Se resta la cantidad del stock
    // Esto modifica el objeto 'encontrado' que está en 'DatosCompartidos.listaProducto'
    int nuevoStock = encontrado.getStockI() - cantidad;
    encontrado.setStockI(nuevoStock);

    // Se actualiza la tabla automaticamente
    // Verificamos si el formulario de productos está abierto
    if (JInternalFrameProducto.instanciaAbierta != null) {
        // Si está abierto, llamamos a su método público para recargar la tabla
        JInternalFrameProducto.instanciaAbierta.cargarTabla();
    }

    // CÓDIGO DE AVISO DE STOCK BAJO
    // Comparamos el 'nuevo stock' que acabamos de calcular con el
    // 'stock mínimo' que tiene guardado el producto.
    // Esto NO detiene la venta, es solo un aviso.
    if (nuevoStock <= encontrado.getStockM()) {
        JOptionPane.showMessageDialog(null, // 'this' o 'null'
                "¡Aviso! Stock bajo para: " + encontrado.getNombre() + "\n" +
                "Quedan solo " + nuevoStock + " unidades (Mínimo: " + encontrado.getStockM() + ").",
                "Aviso de Stock Bajo",
                JOptionPane.WARNING_MESSAGE); // Icono de advertencia
    }
```

```java
            double precio = encontrado.getPrecioV();
            double descuento = 0;
            double subtotal = (precio * cantidad) - descuento;

            // Se actualiza el total general
            total += subtotal;
            lblTotal.setText("C$ " + String.format("%.2f", total));

            // Se crea el objeto DetalleVenta (de momento con pago y cambio = 0)
            DetalleVenta detalle = new DetalleVenta(
                    encontrado.getNombre(),
                    cantidad,
                    precio,
                    subtotal,
                    0, // pago aún no ingresado
                    0, // cambio aún no calculado
                    total // total acumulado hasta ahora
            );

            listaVenta.add(detalle);

            // Se muestra en la tabla
            modeloTabla.addRow(new Object[]{
                    encontrado.getNombre(), cantidad, precio, descuento, subtotal
            });

            // Limpia campos
            txtcantidad.setText("");
            nombrebusca.setText("");
            cerostock.setText("");
```

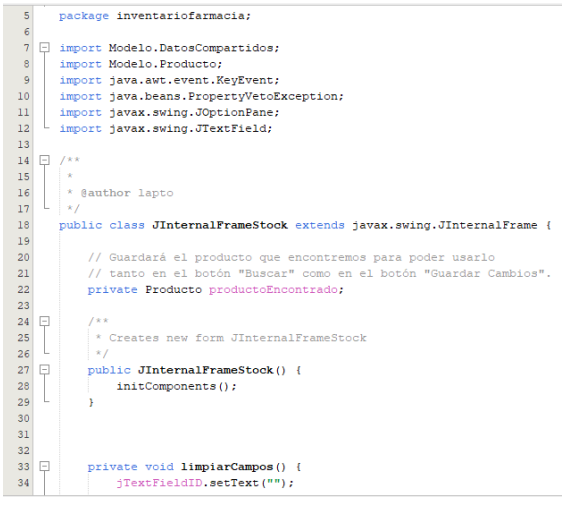
```java
        cerostock.setText("");
        ceroprecio.setText("");
    }

    private void Agregar2ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (listaVenta.isEmpty()) {
                JOptionPane.showMessageDialog(this, "No hay productos en la venta.");
                return;
            }

            double pago = Double.parseDouble(txtipodepago.getText());
            double cambio = pago - total;

            lblTotal.setText("C$ " + String.format("%.2f", total));
            jLabel23.setText("C$ " + String.format("%.2f", cambio));

            if (pago < total) {
                JOptionPane.showMessageDialog(this,
                        "El pago es menor al total. Faltan C$ " + String.format("%.2f", total - pago));
            } else {
                JOptionPane.showMessageDialog(this,
                        "Pago registrado correctamente.\nCambio: C$ " + String.format("%.2f", cambio));
            }

            // * Actualizamos todos los detalles con pago/cambio/total
            for (DetalleVenta detalle : listaVenta) {
                detalle.setPago(pago);
                detalle.setCambio(cambio);
                detalle.setTotal(total);
```

```java
            }

            // Limpiar campo de pago
            txtipodepago.setText("");

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Ingrese una cantidad de pago válida.");
        }
    }

    private void ventafinalizadaActionPerformed(java.awt.event.ActionEvent evt) {
        if (listaVenta.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No hay productos en la venta.", "Advertencia", JOptionPane.WARNING_MESSAGE);
            return;
        }

        int confirmacion = JOptionPane.showConfirmDialog(this,
                "¿Deseas finalizar la venta por un total de C$ " + String.format("%.2f", total) + "?",
                "Confirmar venta",
                JOptionPane.YES_NO_OPTION);

        if (confirmacion == JOptionPane.YES_OPTION) {
            JOptionPane.showMessageDialog(this, "✅ Venta finalizada con éxito. Total: C$ " + String.format("%.2f", total));

            // Se guarda la listaVenta  imprimir
            //  DatosCompartidos.listaVentas.addAll(listaVenta);
            // Limpiar tabla y campos
            modeloTabla.setRowCount(0);
            nombrebusca.setText("");
            txtcantidad.setText("");
```

```
            txtcantidad.setText("");
            txtipodepago.setText("");
            cexoprecio.setText("C$ 0");
            cerostock.setText("0");
            lblTotal.setText("C$ 0");
            jLabel23.setText("C$ 0");
            total = 0;
            listaVenta.clear();
        } else {
            JOptionPane.showMessageDialog(this, "Venta cancelada.");
        }
    }

    private void btnImprimirActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            //  Obtener el pago ingresado
            String textoPago = txtipodepago.getText().trim();
            if (textoPago.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Ingrese la cantidad de pago antes de imprimir la factura.");
                return;
            }

            double pago = Double.parseDouble(textoPago);

            //  Obtener el total de la etiqueta lblTotal
            String textoTotal = lblTotal.getText().replace("C$", "").trim();
            double total = Double.parseDouble(textoTotal);

            // Calcular el cambio
```

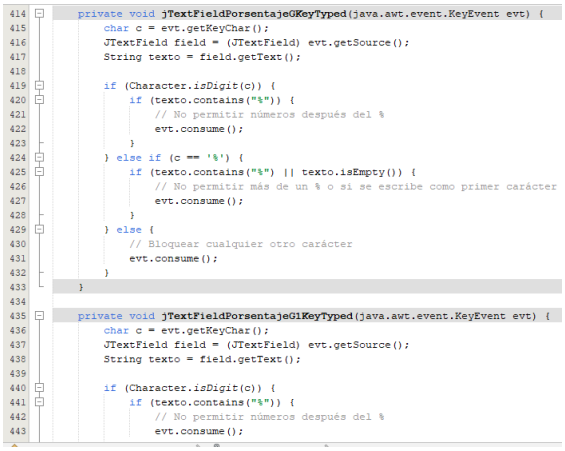
```
            // Calcular el cambio
            double cambio = pago - total;

            // Formatear los valores
            String pagoFormateado = String.format("%.2f", pago);
            String totalFormateado = String.format("%.2f", total);
            String cambioFormateado = String.format("%.2f", cambio);

            // Crear el texto de la factura
            StringBuilder factura = new StringBuilder("=== FACTURA ===\n\n");

            // Agregar productos desde la tabla
            for (int i = 0; i < modeloTabla.getRowCount(); i++) {
                factura.append(modeloTabla.getValueAt(i, 0)).append(" - ")
                       .append("Cant: ").append(modeloTabla.getValueAt(i, 1)).append(", ")
                       .append("Precio: C$").append(modeloTabla.getValueAt(i, 2)).append(", ")
                       .append("Subtotal: C$").append(modeloTabla.getValueAt(i, 4)).append("\n");
            }

            // Agregar totales, pago y cambio
            factura.append("\nTotal: C$ ").append(totalFormateado)
                   .append("\nPago: C$ ").append(pagoFormateado)
                   .append("\nCambio: C$ ").append(cambioFormateado);

            // Mostrar la factura
            JOptionPane.showMessageDialog(this, factura.toString(), "Factura", JOptionPane.INFORMATION_MESSAGE);

        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "El valor de pago no es válido. Ingrese un número.", "Error", JOptionPane.ERROR_MESSAGE);
        }
```

```
    private void cancelarActionPerformed(java.awt.event.ActionEvent evt) {
        // Limpiar el campo de texto
        txtipodepago.setText("");

        // Reiniciar etiquetas
        jLabel23.setText("C$ 0");
        lblTotal.setText("C$ 0");

        // resetear el total
        total = 0;

        // Mostrar mensaje
        JOptionPane.showMessageDialog(this, "Operación cancelada.");

    }

    private void jLabelCerrarMouseClicked(java.awt.event.MouseEvent evt) {
        try {
            this.setClosed(true);

        } catch (PropertyVetoException ex) {
            ex.printStackTrace();
        }
    }


    // Variables declaration - do not modify
    private javax.swing.JButton Agregar2;
    private javax.swing.JButton agregar;
    private javax.swing.JButton btnImprimir;
```

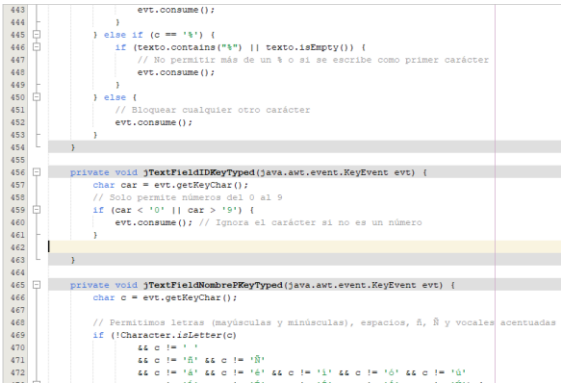


```
package inventariofarmacia;

import Modelo.DatosCompartidos;
import Modelo.Producto;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

/**
 *
 * @author lapto
 */
public class JInternalFrameStock extends javax.swing.JInternalFrame {

    // Guardará el producto que encontremos para poder usarlo
    // tanto en el botón "Buscar" como en el botón "Guardar Cambios".
    private Producto productoEncontrado;

    /**
     * Creates new form JInternalFrameStock
     */
    public JInternalFrameStock() {
        initComponents();
    }



    private void limpiarCampos() {
        jTextFieldID.setText("");
```

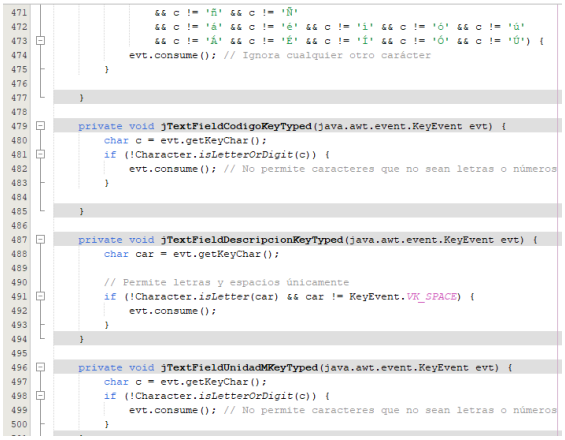
```

    private void limpiarCampos() {
        jTextFieldID.setText("");
        jTextFieldNombreP.setText("");
        jTextFieldCodigo.setText("");
        jTextFieldDescripcion.setText("");
        jTextFieldUnidadM.setText("");
        jTextFieldPrecioC.setText("");
        jTextFieldPorsentajeG1.setText("");
        jTextFieldStockM.setText("");
        jTextFieldLote.setText("");
        FechaV.setText("");
        jTextFieldCategoria.setText("");
        jTextFieldPresentacion.setText("");
        jTextFieldPrecioV.setText("");
        jTextFieldStockI.setText("");

        // También limpiamos la variable clave y el buscador
        this.productoEncontrado = null;
        jTextFieldBuscar.setText("");
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```



```
    private void jTextFieldPorsentajeGKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        JTextField field = (JTextField) evt.getSource();
        String texto = field.getText();

        if (Character.isDigit(c)) {
            if (texto.contains("%")) {
                // No permitir números después del %
                evt.consume();
            }
        } else if (c == '%') {
            if (texto.contains("%") || texto.isEmpty()) {
                // No permitir más de un % o si se escribe como primer carácter
                evt.consume();
            }
        } else {
            // Bloquear cualquier otro carácter
            evt.consume();
        }
    }

    private void jTextFieldPorsentajeG1KeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        JTextField field = (JTextField) evt.getSource();
        String texto = field.getText();

        if (Character.isDigit(c)) {
            if (texto.contains("%")) {
                // No permitir números después del %
                evt.consume();
```



```
                evt.consume();
            }
        } else if (c == '%') {
            if (texto.contains("%") || texto.isEmpty()) {
                // No permitir más de un % o si se escribe como primer carácter
                evt.consume();
            }
        } else {
            // Bloquear cualquier otro carácter
            evt.consume();
        }
    }

    private void jTextFieldIDKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }

    }

    private void jTextFieldNombrePKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();

        // Permitimos letras (mayúsculas y minúsculas), espacios, ñ, Ñ y vocales acentuadas
        if (!Character.isLetter(c)
                && c != ' '
                && c != 'ñ' && c != 'Ñ'
                && c != 'á' && c != 'é' && c != 'í' && c != 'ó' && c != 'ú'
```



```
                && c != 'Á' && c != 'É' && c != 'Í' && c != 'Ó' && c != 'Ú') {
            evt.consume(); // Ignora cualquier otro carácter
        }

    }

    private void jTextFieldCodigoKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }

    }

    private void jTextFieldDescripcionKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();

        // Permite letras y espacios únicamente
        if (!Character.isLetter(car) && car != KeyEvent.VK_SPACE) {
            evt.consume();
        }
    }

    private void jTextFieldUnidadMKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }
```

```
    }

    private void jTextFieldPrecioCKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jTextFieldStockMKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jTextFieldLoteKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isLetterOrDigit(c)) {
            evt.consume(); // No permite caracteres que no sean letras o números
        }
    }

    private void jTextFieldCategoriaKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();

        // Permite letras y espacios únicamente
        if (!Character.isLetter(car) && car != KeyEvent.VK_SPACE) {
```

```
        // Permite letras y espacios únicamente
        if (!Character.isLetter(car) && car != KeyEvent.VK_SPACE) {
            evt.consume();
        }
    }

    private void jTextFieldPresentacionKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        // permite letras (incluye Ñ y acentos), numeros y espacio
        if (!Character.isLetterOrDigit(c)
                && c != ' '
                && c != 'ñ' && c != 'Ñ'
                && c != 'á' && c != 'é' && c != 'í' && c != 'ó' && c != 'ú'
                && c != 'Á' && c != 'É' && c != 'Í' && c != 'Ó' && c != 'Ú') {
            evt.consume(); // Ignora cualquier otro carácter
        }

    }

    private void jTextFieldPrecioVKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jTextFieldStockIKeyTyped(java.awt.event.KeyEvent evt) {
        char car = evt.getKeyChar();
        // Solo permite números del 0 al 9
```

```
        // Solo permite números del 0 al 9
        if (car < '0' || car > '9') {
            evt.consume(); // Ignora el carácter si no es un número
        }
    }

    private void jButtonBuscarActionPerformed(java.awt.event.ActionEvent evt) {
        String nombreBuscado = jTextFieldBuscar.getText().trim();

        if (nombreBuscado.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Por favor, ingrese un nombre para buscar.");
            return;
        }

        // Reiniciamos la variable clave
        this.productoEncontrado = null;

        // Buscamos en la lista GLOBAL
        for (Producto p : DatosCompartidos.listaProducto) {
            if (p.getNombre().equalsIgnoreCase(nombreBuscado)) {
                this.productoEncontrado = p; // ¡Lo encontramos! Lo guardamos en la variable clave
                break; // Salimos del bucle
            }
        }

        //  llenamos las 14 cajas de texto
        if (this.productoEncontrado != null) {
            // Si lo encontramos, llenamos todos los campos

            // Usamos los "getters" del objeto 'productoEncontrado'
```

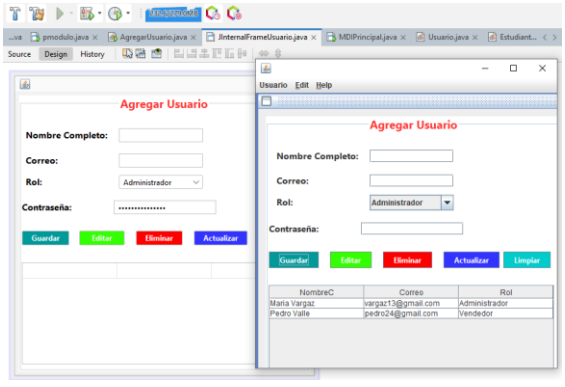
```
            // Usamos los "getters" del objeto 'productoEncontrado'
            // Los números (int, double) deben convertirse a String
            jTextFieldID.setText(String.valueOf(productoEncontrado.getID()));
            jTextFieldNombreP.setText(productoEncontrado.getNombre());
            jTextFieldCodigo.setText(productoEncontrado.getCodigo());
            jTextFieldDescripcion.setText(productoEncontrado.getDescripcion());
            jTextFieldUnidadM.setText(productoEncontrado.getUnidadM());
            jTextFieldPrecioC.setText(String.valueOf(productoEncontrado.getPrecioC()));
            jTextFieldPorsentajeG1.setText(String.valueOf(productoEncontrado.getPorsentajeG()));
            jTextFieldStockM.setText(String.valueOf(productoEncontrado.getStockM()));
            jTextFieldLote.setText(productoEncontrado.getLote());
            FechaV.setText(productoEncontrado.getFechaV());
            jTextFieldCategoria.setText(productoEncontrado.getCategoria());
            jTextFieldPresentacion.setText(productoEncontrado.getPresentacion());
            jTextFieldPrecioV.setText(String.valueOf(productoEncontrado.getPrecioV()));

            // El campo de stock/existencia
            jTextFieldStockI.setText(String.valueOf(productoEncontrado.getStockI()));

        } else {
            // Si no lo encontramos, mostramos un mensaje y limpiamos las cajas
            JOptionPane.showMessageDialog(this, "Producto no encontrado.");
            limpiarCampos(); // llama al metodo limpiarCampos
        }
    }

    private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {

        // 1. Verificamos que SÍ tengamos un producto cargado
        if (this.productoEncontrado == null) {
```

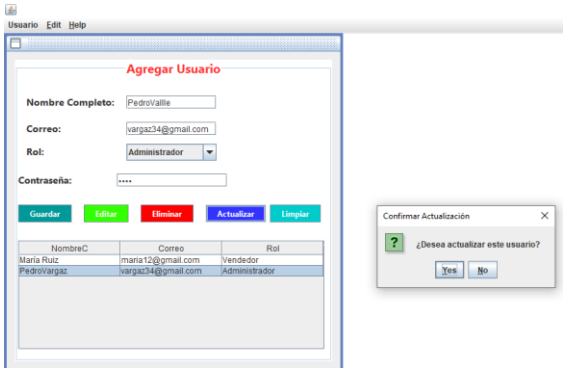
```
        // 1. Verificamos que SÍ tengamos un producto cargado
        if (this.productoEncontrado == null) {
            JOptionPane.showMessageDialog(this, "Primero debe buscar un producto antes de guardar.");
            return;
        }

        //validar que todos los campos de texto no esten vacios
        if ((jTextFieldID.getText().contentEquals(""))
                || (jTextFieldNombreP.getText().contentEquals(""))
                || (jTextFieldCodigo.getText().contentEquals(""))
                || (jTextFieldDescripcion.getText().contentEquals(""))
                || (jTextFieldUnidadM.getText().contentEquals(""))
                || (jTextFieldPrecioC.getText().contentEquals(""))
                || (jTextFieldPorsentajeG1.getText().contentEquals(""))
                || (jTextFieldStockM.getText().contentEquals(""))
                || (jTextFieldLote.getText().contentEquals(""))
                || (FechaV.getText().contentEquals(""))
                || (jTextFieldCategoria.getText().contentEquals(""))
                || (jTextFieldPresentacion.getText().contentEquals(""))
                || (jTextFieldPrecioV.getText().contentEquals(""))
                || (jTextFieldStockI.getText().contentEquals(""))) {
            //Mensaje para informar si algun campo esta vacio // Agrego import
            JOptionPane.showMessageDialog(null,
                    "Algunos campos estan vacios",
                    "!Revise todos los campos!",
                    JOptionPane.WARNING_MESSAGE);
            return;

        }
        try {
            // Recuperamos y convertimos todos los nuevos valores
```

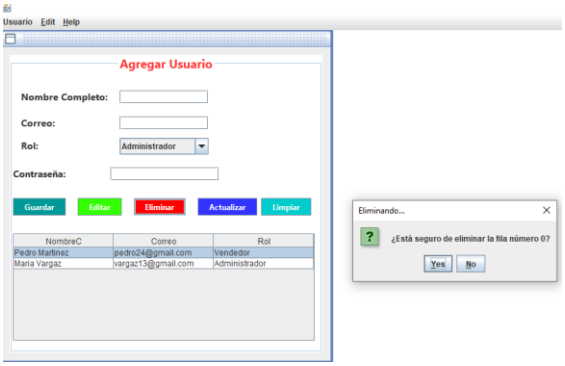
```
        try {
            // Recuperamos y convertimos todos los nuevos valores
            String nombreProducto = jTextFieldNombreP.getText().trim();
            String codigo = jTextFieldCodigo.getText().trim();
            String descripcion = jTextFieldDescripcion.getText().trim();
            String unidadMedida = jTextFieldUnidadM.getText().trim();
            String lote = jTextFieldLote.getText().trim();
            String fechaVencimiento = FechaV.getText().trim();
            String categoria = jTextFieldCategoria.getText().trim();
            String presentacion = jTextFieldPresentacion.getText().trim();
            // Convertimos los datos numericos
            double precioCompra = Double.parseDouble(jTextFieldPrecioC.getText().trim());
            double porsentage = Double.parseDouble(jTextFieldPorsentajeG1.getText().trim())
            int stockM = Integer.parseInt(jTextFieldStockM.getText().trim());
            double precioV = Double.parseDouble(jTextFieldPrecioV.getText().trim());
            int stockI = Integer.parseInt(jTextFieldStockI.getText().trim());
            int id = Integer.parseInt(jTextFieldID.getText().trim());

            this.productoEncontrado.setNombre(nombreProducto);
            this.productoEncontrado.setCodigo(codigo);
            this.productoEncontrado.setDescripcion(descripcion);
            this.productoEncontrado.setUnidadM(unidadMedida);
            this.productoEncontrado.setLote(lote);
            this.productoEncontrado.setFechaV(fechaVencimiento);
            this.productoEncontrado.setCategoria(categoria);
            this.productoEncontrado.setPresentacion(presentacion);
            this.productoEncontrado.setPrecioC(precioCompra);
            this.productoEncontrado.setPorsentajeG(porsentage);
            this.productoEncontrado.SetStockM(stockM);
            this.productoEncontrado.setPrecioV(precioV);
```

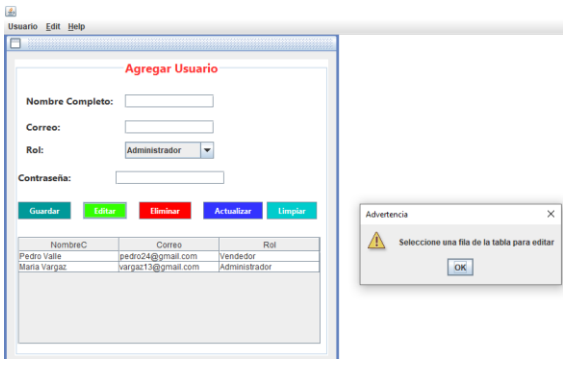
```
            this.productoEncontrado.setPrecioV(precioV);
            this.productoEncontrado.setStockI(stockI);
            this.productoEncontrado.setID(id);

            // Se le avisa al formulario de productos para actualizar la tabla
            if (JInternalFrameProducto.instanciaAbierta != null) {
                JInternalFrameProducto.instanciaAbierta.cargarTabla();
            }
            // Mostramos un mensaje confirmando la actualizacion
            JOptionPane.showMessageDialog(this, "Producto " + nombreProducto + "Actualizado Correctamente", "Exito", JOptionPane.INFORMATION_MESSAGE);

        } catch (NumberFormatException e) {
            JOptionPane.showConfirmDialog(this,
                    "Error de formato en campos numericos, verifique que los campos numericos (precio, stock) contengan solo números validos.",
                    "Error de conversión",
                    JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showConfirmDialog(this, "Ocurrio un error inesperado al actualizar: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void jButtonCancelarActionPerformed(java.awt.event.ActionEvent evt) {
        // llama al metodo limpiarCampos
        limpiarCampos();
    }
```

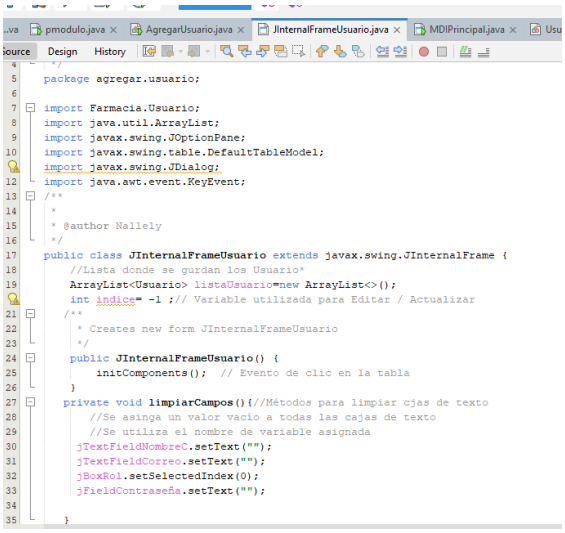
```
701     private void jLabelCerrarMouseClicked(java.awt.event.MouseEvent evt) {
702         try {
703             this.setClosed(true);
704 
705         } catch (PropertyVetoException ex) {
707             ex.printStackTrace();
708         }
709     }
710 
711     // Variables declaration - do not modify
712     private javax.swing.JFormattedTextField FechaV;
713     private javax.swing.JButton jButtonActualizar;
714     private javax.swing.JButton jButtonBuscar;
715     private javax.swing.JButton jButtonCancelar;
716     private javax.swing.JLabel jLabel1;
717     private javax.swing.JLabel jLabel10;
718     private javax.swing.JLabel jLabel11;
719     private javax.swing.JLabel jLabel12;
720     private javax.swing.JLabel jLabel13;
721     private javax.swing.JLabel jLabel14;
722     private javax.swing.JLabel jLabel2;
723     private javax.swing.JLabel jLabel3;
724     private javax.swing.JLabel jLabel4;
725     private javax.swing.JLabel jLabel5;
726     private javax.swing.JLabel jLabel6;
727     private javax.swing.JLabel jLabel7;
728     private javax.swing.JLabel jLabel8;
729     private javax.swing.JLabel jLabel9;
730     private javax.swing.JLabel jLabelCerrar;
```

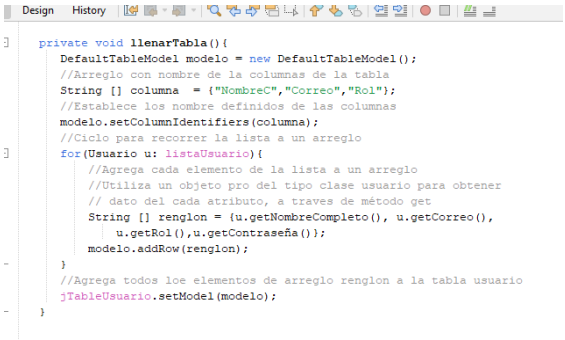


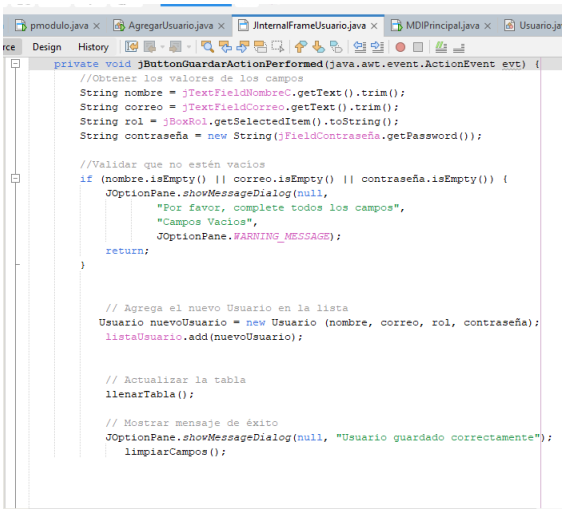


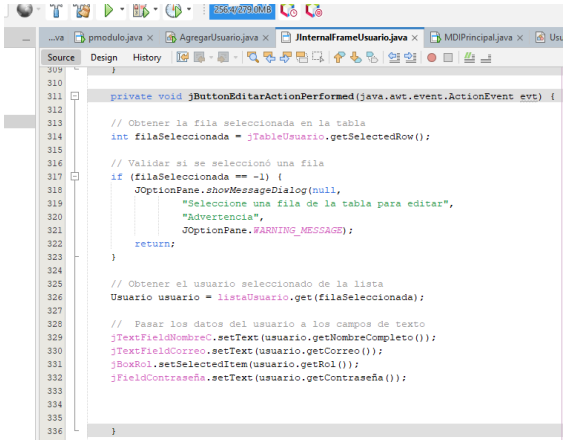

Usuario Edit Help
Agregar Usuario
Nombre Completo:
Correo:
Rol:
Administrador
Contraseña:
Guardar
Editar
Eliminar
Actualizar
Limpiar
NombreC
Correo
Rol
Pedro Martinez
pedro24@gmail.com
Vendedor
Maria Vargaz
vargaz13@gmail.com
Administrador
Eliminando...
¿Está seguro de eliminar la fila número 0?
Yes
No

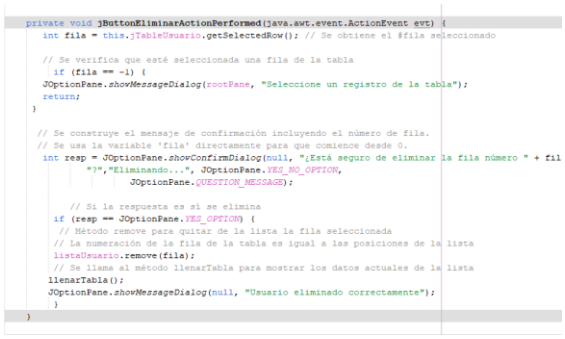

Usuario Edit Help
Agregar Usuario
Nombre Completo:
Correo:
Rol:
Administrador
Contraseña:
Guardar
Editar
Eliminar
Actualizar
Limpiar
NombreC
Correo
Rol
Pedro Valle
pedro24@gmail.com
Vendedor
Maria Vargaz
vargaz13@gmail.com
Administrador
Advertencia
Seleccione una fila de la tabla para editar
OK


pmodulo.java
AgregarUsuario.java
JInternalFrameUsuario.java
MDIPrincipal.java
Source
Design
History
package agregar.usuario;
import Farmacia.Usuario;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import javax.swing.JDialog;
import java.awt.event.KeyEvent;
/**
*
* @author Nallely
*/
public class JInternalFrameUsuario extends javax.swing.JInternalFrame {
//Lista donde se gurdan los Usuario*
ArrayList<Usuario> listaUsuario=new ArrayList<>();
int indice= -1 ;// Variable utilizada para Editar / Actualizar
/**
* Creates new form JInternalFrameUsuario
*/
public JInternalFrameUsuario() {
initComponents(); // Evento de clic en la tabla
}
private void limpiarCampos(){//Métodos para limpiar cjas de texto
//Se asinga un valor vacio a todas las cajas de texto
//Se utiliza el nombre de variable asignada
jTextFieldNombreC.setText("");
jTextFieldCorreo.setText("");
jBoxRol.setSelectedIndex(0);
jFieldContraseña.setText("");
}

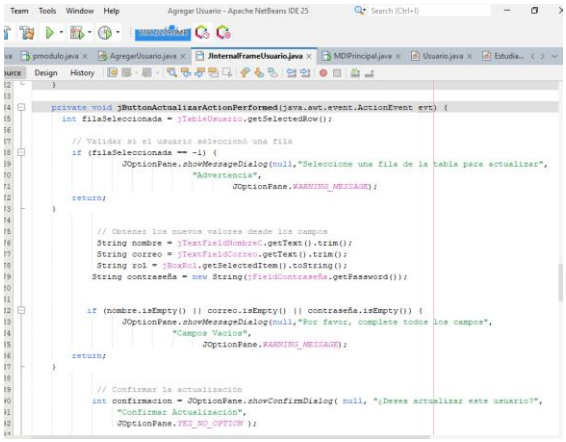

Design
History
private void llenarTabla(){
DefaultTableModel modelo = new DefaultTableModel();
//Arreglo con nombre de la columnas de la tabla
String [] columna = {"NombreC","Correo","Rol"};
//Establece los nombre definidos de las columnas
modelo.setColumnIdentifiers(columna);
//Ciclo para recorrer la lista a un arreglo
for(Usuario u: listaUsuario){
//Agrega cada elemento de la lista a un arreglo
//Utiliza un objeto pro del tipo clase usuario para obtener
// dato del cada atributo, a traves de método get
String [] renglon = {u.getNombreCompleto(), u.getCorreo(),
u.getRol(),u.getContraseña()};
modelo.addRow(renglon);
}
//Agrega todos loe elementos de arreglo renglon a la tabla usuario
jTableUsuario.setModel(modelo);
}


pmodulo.java
AgregarUsuario.java
JInternalFrameUsuario.java
MDIPrincipal.java
Design
History
private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {
//Obtener los valores de los campos
String nombre = jTextFieldNombreC.getText().trim();
String correo = jTextFieldCorreo.getText().trim();
String rol = jBoxRol.getSelectedItem().toString();
String contraseña = new String(jFieldContraseña.getPassword());
//Validar que no estén vacíos
if (nombre.isEmpty() || correo.isEmpty() || contraseña.isEmpty()) {
JOptionPane.showMessageDialog(null,
"For favor, complete todos los campos",
"Campos Vacios",
JOptionPane.WARNING_MESSAGE);
return;
}
// Agrega el nuevo Usuario en la lista
Usuario nuevoUsuario = new Usuario (nombre, correo, rol, contraseña);
listaUsuario.add(nuevoUsuario);
// Actualizar la tabla
llenarTabla();
// Mostrar mensaje de éxito
JOptionPane.showMessageDialog(null, "Usuario guardado correctamente");
limpiarCampos();

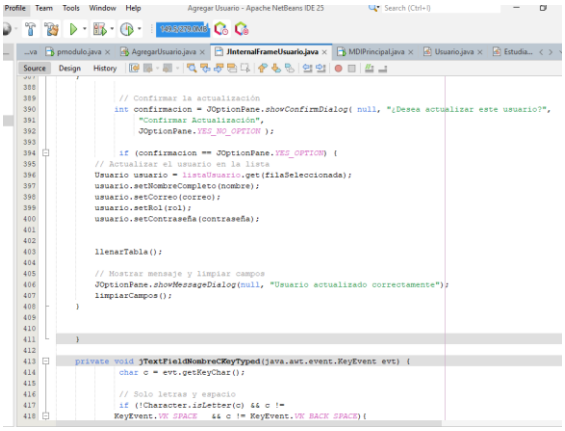

pmodulo.java
AgregarUsuario.java
JInternalFrameUsuario.java
MDIPrincipal.java
Source
Design
History
private void jButtonEditarActionPerformed(java.awt.event.ActionEvent evt) {
// Obtener la fila seleccionada en la tabla
int filaSeleccionada = jTableUsuario.getSelectedRow();
// Validar si se seleccionó una fila
if (filaSeleccionada == -1) {
JOptionPane.showMessageDialog(null,
"Seleccione una fila de la tabla para editar",
"Advertencia",
JOptionPane.WARNING_MESSAGE);
return;
}
// Obtener el usuario seleccionado de la lista
Usuario usuario = listaUsuario.get(filaSeleccionada);
// Pasar los datos del usuario a los campos de texto
jTextFieldNombreC.setText(usuario.getNombreCompleto());
jTextFieldCorreo.setText(usuario.getCorreo());
jBoxRol.setSelectedItem(usuario.getRol());
jFieldContraseña.setText(usuario.getContraseña());
}



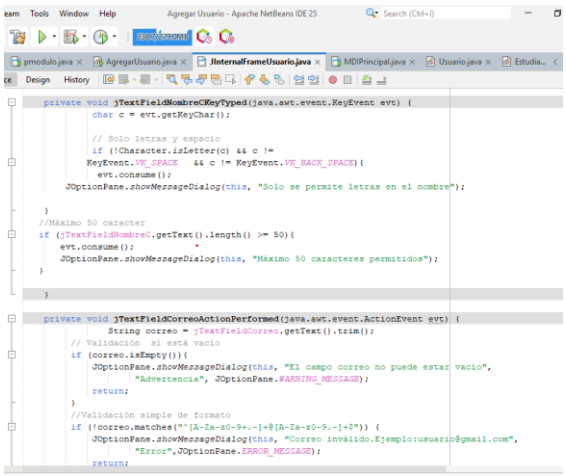
```
private void jButtonEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = this.jTableUsuario.getSelectedRow(); // Se obtiene el #fila seleccionado

    // Se verifica que esté seleccionada una fila de la tabla
      if (fila == -1) {
    JOptionPane.showMessageDialog(rootPane, "Seleccione un registro de la tabla");
    return;
   }

   // Se construye el mensaje de confirmación incluyendo el número de fila.
   // Se usa la variable 'fila' directamente para que comience desde 0.
    int resp = JOptionPane.showConfirmDialog(null, "¿Está seguro de eliminar la fila número " + fil
            "?","Eliminando...", JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE);

        // Si la respuesta es sí se elimina
      if (resp == JOptionPane.YES_OPTION) {
       // Método remove para quitar de la lista la fila seleccionada
       // La numeración de la fila de la tabla es igual a las posiciones de la lista
       listaUsuario.remove(fila);
       // Se llama al método llenarTabla para mostrar los datos actuales de la lista
      llenarTabla();
      JOptionPane.showMessageDialog(null, "Usuario eliminado correctamente");
       }
}
```



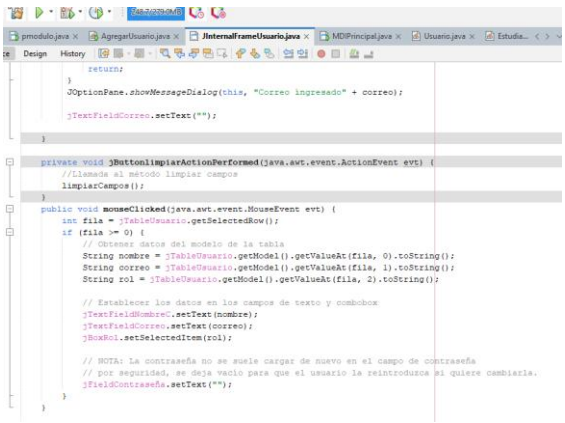
```
private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    int filaSeleccionada = jTableUsuario.getSelectedRow();

      // Validar si el usuario seleccionó una fila
      if (filaSeleccionada == -1) {
              JOptionPane.showMessageDialog(null,"Seleccione una fila de la tabla para actualizar",
                      "Advertencia",
                              JOptionPane.WARNING_MESSAGE);
      return;
   }

          // Obtener los nuevos valores desde los campos
          String nombre = jTextFieldNombreC.getText().trim();
          String correo = jTextFieldCorreo.getText().trim();
          String rol = jBoxRol.getSelectedItem().toString();
         String contraseña = new String(jFieldContraseña.getPassword());


       if (nombre.isEmpty() || correo.isEmpty() || contraseña.isEmpty()) {
              JOptionPane.showMessageDialog(null,"Por favor, complete todos los campos",
                      "Campos Vacios",
                              JOptionPane.WARNING_MESSAGE);
      return;
   }

          // Confirmar la actualización
          int confirmacion = JOptionPane.showConfirmDialog( null, "¿Desea actualizar este usuario?",
              "Confirmar Actualización",
              JOptionPane.YES_NO_OPTION );
```



```
            // Confirmar la actualización
            int confirmacion = JOptionPane.showConfirmDialog( null, "¿Desea actualizar este usuario?",
                "Confirmar Actualización",
                JOptionPane.YES_NO_OPTION );

            if (confirmacion == JOptionPane.YES_OPTION) {
          // Actualizar el usuario en la lista
          Usuario usuario = listaUsuario.get(filaSeleccionada);
          usuario.setNombreCompleto(nombre);
          usuario.setCorreo(correo);
          usuario.setRol(rol);
          usuario.setContraseña(contraseña);


          llenarTabla();

          // Mostrar mensaje y limpiar campos
          JOptionPane.showMessageDialog(null, "Usuario actualizado correctamente");
          limpiarCampos();
      }

  }

  private void jTextFieldNombreCKeyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();

            // Solo letras y espacio
            if (!Character.isLetter(c) && c !=
            KeyEvent.VK_SPACE    && c != KeyEvent.VK_BACK_SPACE){
```



```
  private void jTextFieldNombreCKeyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();

            // Solo letras y espacio
            if (!Character.isLetter(c) && c !=
          KeyEvent.VK_SPACE    && c != KeyEvent.VK_BACK_SPACE){
              evt.consume();
       JOptionPane.showMessageDialog(this, "Solo se permite letras en el nombre");

   }
  //Máximo 50 caracter
  if (jTextFieldNombreC.getText().length() >= 50){
      evt.consume();
      JOptionPane.showMessageDialog(this, "Máximo 50 caracteres permitidos");
  }

  }

  private void jTextFieldCorreoActionPerformed(java.awt.event.ActionEvent evt) {
          String correo = jTextFieldCorreo.getText().trim();
        // Validación  si está vacio
        if (correo.isEmpty()){
            JOptionPane.showMessageDialog(this, "El campo correo no puede estar vacio",
                    "Advertencia", JOptionPane.WARNING_MESSAGE);
            return;
        }
        //Validación simple de formato
        if (!correo.matches("^[A-Za-z0-9+._-]+@[A-Za-z0-9.-]+$")) {
            JOptionPane.showMessageDialog(this, "Correo inválido.Ejemplo:usuario@gmail.com",
                    "Error",JOptionPane.ERROR_MESSAGE);
            return;
```

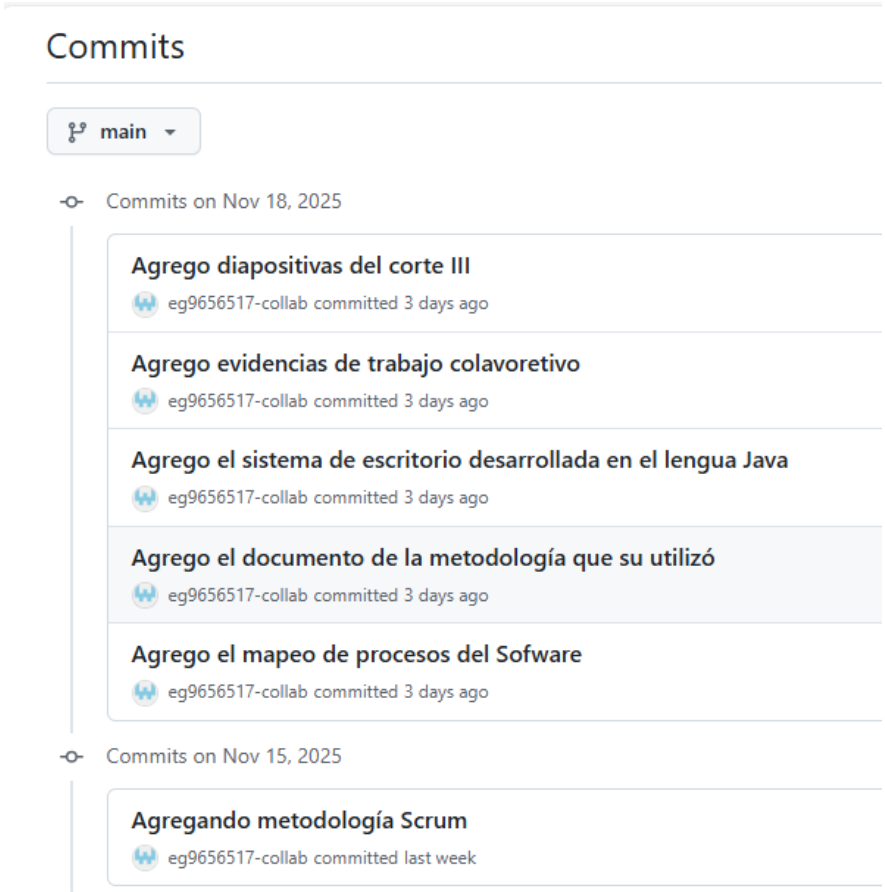


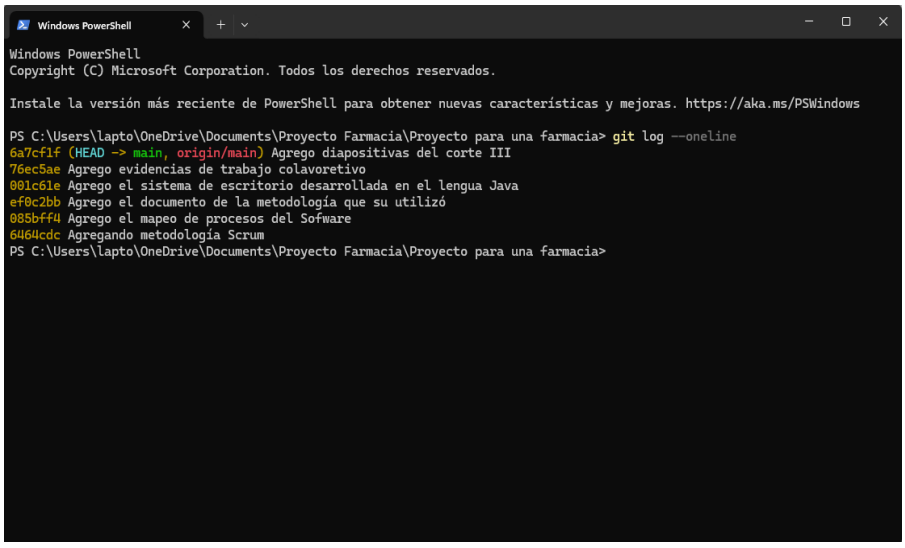
```
            return;
        }
        JOptionPane.showMessageDialog(this, "Correo ingresado" + correo);

        jTextFieldCorreo.setText("");

    }

  private void jButtonlimpiarActionPerformed(java.awt.event.ActionEvent evt) {
      //Llamada al método limpiar campos
      limpiarCampos();
  }
  public void mouseClicked(java.awt.event.MouseEvent evt) {
      int fila = jTableUsuario.getSelectedRow();
      if (fila >= 0) {
          // Obtener datos del modelo de la tabla
          String nombre = jTableUsuario.getModel().getValueAt(fila, 0).toString();
          String correo = jTableUsuario.getModel().getValueAt(fila, 1).toString();
          String rol = jTableUsuario.getModel().getValueAt(fila, 2).toString();

          // Establecer los datos en los campos de texto y combobox
          jTextFieldNombreC.setText(nombre);
          jTextFieldCorreo.setText(correo);
          jBoxRol.setSelectedItem(rol);

          // NOTA: La contraseña no se suele cargar de nuevo en el campo de contraseña
          // por seguridad, se deja vacío para que el usuario la reintroduzca si quiere cambiarla.
          jFieldContraseña.setText("");
      }
  }
```

10. Enlace del repositorio GitHub (Commit realizados Imagen)

[eg9656517-collab/Proyecto-Integrador-II]

10.1 Commit realizados





11. Referencias Bibliográficas

Ambulatoria del Hospital El Carmen de Maipú, para avanzar en la automatización de servicios farmacéuticos

https://repositorio.uchlile.cl/handle/2250/186964

Ardila Muñoz, S. A. (2024). Diseño, desarrollo e implementación de un componente para las plataformas web y escritorio de la empresa Sistemas y Computadores SYC SA.

https://repositorio.uts.edu.co:8080/xmlui/handle/123456789/14827

Agudelo Guisao, R. (2025). Diseño y desarrollo de un software que fomente el acceso a los servicios farmacéuticos asistenciales (dispensación, educación para la salud y farmacovigilancia) en pacientes que consultan Establecimientos Farmacéuticos Minoristas y Servicios Farmacéuticos ambulatorios.

https://bibliotecadigital.udea.edu.co/entities/publication/9e35bfda-3456-4e7e-835b-900d7a298**990**

García, L. (2021). Propiedades antioxidantes de extractos botánicos en el tratamiento de patologías dermatológicas. Farmacia Moderna, 15(3), 45-58.

https://doi.org/xx.xxxx/fm.2021.15.3.45.

González, M. J. F., Cabrera, C. a. Z., Prado, R. E. S., Japón, G. E. R., & Dávila, K. E. D. (2024). Farmacia Hospitalaria. In *Juan Cuevas eBooks*.
https://doi.org/10.56470/978-9942-627-95-7

Verdugo, R. M., Lamas, M. C., Latorre, A. D., Piqueres, R. F., Fernández-Llamazares, C., Vega, E. N., & Goitia, B. T. (2021). Desarrollo de la norma Q-PEX de certificación de calidad de la atención farmacéutica a pacientes externos de los servicios de Farmacia. *Journal of Healthcare Quality Research*, *36*(6), 324–332.
https://doi.org/10.1016/j.jhqr.2021.03.010

Manrique Rodríguez, D. S. (2025). Machine Lear Ning para la gestión de inventarios de medicamentos en el Hospital José Agurto Tello de Chosica.

https://repositorio.utp.edu.pe/handle/20.500.12867/11476

Rojas Raimundo, J. D. (2024). Implementación de un sistema de escritorio para el control de mermas en la empresa Perú Pharma.

https://repositorio.upla.edu.pe/handle/20.500.12848/7738

Quisaguano Casa, B. A. (2022). *Desarrollo de sistema para la gestión del inventario en FARMECC: Desarrollo de un sistema de escritorio* (Bachelor's thesis, Quito: EPN, 2022.).

https://bibdigital.epn.edu.ec/handle/15000/23106

Villagrán Gómez, C. E. (2021). Implementación de un sistema informático en la farmacia Ambulatoria del Hospital El Carmen de Maipú, para avanzar en la automatización de servicios farmacéuticos.

https://repositorio.uchlile.cl/handle/2250/186964

Román Ramos, F. J. (2024). Implementación de un sistema informático de control de compra, venta y stock de productos farmacéuticos para la farmacia "El Progreso"–Chimbote; 2024.Minoristas y Servicios Farmacéuticos ambulatorios.

https://repositorio.uladech.edu.pe/handle/20.500.13032/39882

Delgado, A. E. M., Rosero, G. A. B., Estrella, K. D. O., & Arias, E. B. (2025). Trayectoria profesional y laboral de los egresados del programa de Tecnología de Regencia de Farmacia. *Boletín Informativo CEI*, *12*(1), 33-35. 2024.Minoristas y Servicios Farmacéuticos ambulatorios.

https://revista.umariana.rdu.co/index.php/BoletinInformativoCEI/article/view/4512

12. Anexos

12.1 Informe de sesión de trabajo Corte I

No	**Nombres y apellidos**	**Aportes realizados por integrantes de equipos**	**Observaciones** (incumplimientos llegadas tardes, ausencia, no aporta al proyecto de equipo)
1	Jasser Josafaph Bermúdez Suazo	Tareas o funciones que el sistema automatizará, tomando en cuenta sus características	
2	Ana Gabriela Irías Murillo	Área de la empresa que hará uso del sistema de escritorio	
3	Eduardo Jesús Gudiel González	Objetivos a cumplir (con base en el análisis, diseño y desarrollo de la solución)	
4	Francis Nallely Ayala Calero	Empresa, organización o institución para la cual se desarrollará el sistema de escritorio	

12.2 Informe de sesión de trabajo Corte II

No	**Nombres y apellidos**	**Aportes realizados por integrantes de equipos**	**Observaciones** (incumplimientos llegadas tardes, ausencia, no aporta al proyecto de equipo)
1	Jasser Josafaph Bermúdez Suazo	Tabla de roles, historia de usuario de agregar productos y mapeo del software	
2	Ana Gabriela Irías Murillo	Tabla de historia de usuario (consultar productos)	
3	Eduardo Jesús Gudiel González	Tabla de historia de usuario (agregar usuarios, usuarios agregados), requerimientos funcionales, tabla del Sprint Backlog (2, 4), diseño de interfaces, mapeo del software e investigación.	
4	Francis Nallely Ayala Calero	Tabla de historias de usuario (reportes, inicio de sesión, actualizar stock), tabla de requerimientos funcionales, tabla del Sprint Backlog (1, 3 y 5), diseño de interfaces, mapeo del software e investigación.	

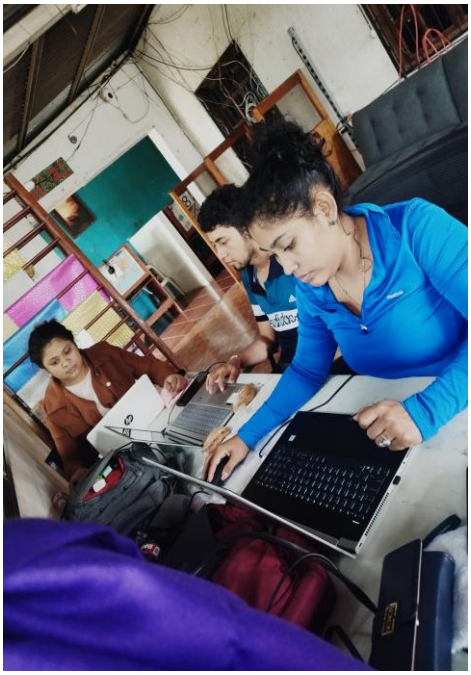
12.2.1 Evidencias de trabajo

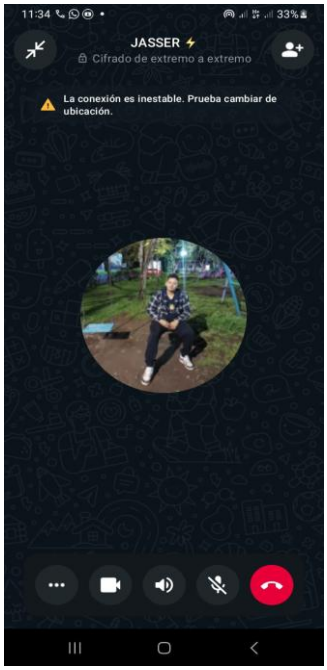
12. 3 Informe de sesión de trabajo Corte III

No	**Nombres y apellidos**	**Aportes realizados por integrantes de equipos**	**Observaciones** (incumplimientos llegadas tardes, ausencia, no aporta al proyecto de equipo)
1	Jasser Josafaph Bermúdez Suazo	Actualizó el mapeo del software y programó el formulario de registrar ventas.	
2	Ana Gabriela Irías Murillo	Diseñó el formulario de actualizar stock e hizo el índice del documento.	

3	Eduardo Jesús Gudiel González	Se encargó de terminar la tabla de requerimientos funcionales, organizó las tablas de historias de usuario, completó el sprint 1 y programó el formulario de Agregar Productos e hizo la conexión del formulario de ventas con el de agregar productos y programó la funcionalidad del formulario actualizar Stock	
4	Francis Nallely Ayala Calero	Se encargó de la organización de los Sprint Backlog 2 y 3 y programó el formulario de Agregar Usuario y Se encargó de la elaboración del BurnDown Chart del Sprint 2.	

12.3.1 Evidencias de trabajo





13. Cronograma de actividades.

Actividades /Tareas	Fechas inicio	Fechas fin	Hora de trabajo	Responsable de actividades
Selección del caso de estudio	09/08/2025	16/08/2025	2hrs	Equipo
Funcionalidades del sistema	17/08/2025	17/08/2025	3hrs	Equipo
objetivos y descripción de la problemática	18/08/2025	18/08/2025	1hrs	Equipo
Mapeo del Software	19/08/2025	19/08/2025	2hrs	Equipo

Planificación Scrum				
Tabla de roles del Equipo de desarrollo	6/09/2025	06/09/2025	2hrs	Equipo
Requerimientos funcionales	8/09/2025	8/09/2025	1hrs	Equipo
Historias de usuarios	11/09/2025	11/09/2025	3hrs	Equipo
Formato de product	15/09/2025	15/09/2025	2hrs	Equipo
Backlog priorizado	17/09/2025	17/09/202	3hrs	Equipo
Formato Sprint Backlog	18/09/2025	18/09/2025	1hrs	Equipo
Interfaz de usuario Figma	20/09/2025	20/09/2025	2hrs	Equipo
Creación de proyecto NetBeans	21/09/2025	21/09/2025	1hrs	Equipo
Paquete controlador	22/09/2025	22/09/2025	3hrs	Equipo
Paquete vista	23/09/2025	23/09/2025	1hrs	Equipo
Paquete modelo	25/09/2025	25/09/’25	3 hrs	Equipo
Referencias bibliográficas	24/09/2025	24/09/202	3hrs	Equipo
Enlace repositorio GitHub	26/09/2025	26/09/202	4hrs	Equipo
Anexos	26/09/2025	26/09/2025	2hrs	Equipo
Informes de sesión de trabajo corte I	27/09/2025	27/09/2025	3hrs	Equipo

Informe de sesión de trabajo corte II	27/09/2025	27/09/202	1hrs	Equipo
Informe de sesión de trabajo corte III	27/09/2025	27/09/202	2hrs	Equipo
Cronograma de actividades	27/09/2025	27/09/202	2hrs	Equipo