



Asignatura: CONSTRUCCION DE SOFTWARE

Docente: WALDYR FREDY CERRON VALVERDE

NRC: 73866

Tema: IMPLEMENTACION DE UN PUNTO DE VENTA

Integrante:

- **BEDOYA ESPINOZA JORGE LUIS**

Link del video:

[https://drive.google.com/drive/folders/1sIFGrsSte7NPPAPBwGrG2wvuHEUUJfbU?](https://drive.google.com/drive/folders/1sIFGrsSte7NPPAPBwGrG2wvuHEUUJfbU?usp=sharing)

[usp=sharing](#)

INTRODUCCION

En el contexto actual de transformación digital, la automatización de procesos administrativos representa una necesidad creciente para pequeñas y medianas empresas. Este proyecto académico presenta el desarrollo progresivo de un sistema de facturación, inicialmente limitado a funciones de autenticación y registro de ventas, con el objetivo de evolucionar hacia una solución más robusta y funcional.

La versión inicial del sistema permite el acceso mediante credenciales y el registro básico de operaciones comerciales, almacenando los datos en una base de datos relacional. Sin embargo, se identifican dos carencias clave: la ausencia de generación de comprobantes digitales (factura en PDF) y la falta de validaciones que garanticen la integridad de los datos ingresados por el usuario.

A partir de este diagnóstico, se plantea una mejora estructurada bajo metodología ágil, simulando roles y sprints que permitan incorporar nuevas funcionalidades de forma ordenada y eficiente. Además, se integra el uso de inteligencia artificial generativa como herramienta de apoyo en el análisis, optimización y documentación del código, fortaleciendo el enfoque académico y técnico del proyecto.

Este informe documenta el proceso de evolución del sistema, las decisiones técnicas adoptadas, y las mejoras implementadas, con el propósito de demostrar una capacidad integral de desarrollo, análisis y presentación institucional.

1. Descripción del problema y la solución propuesta

En entornos comerciales pequeños, el proceso de facturación suele realizarse de forma manual o con herramientas limitadas, lo que genera errores frecuentes, pérdida de información y baja eficiencia operativa. La ausencia de comprobantes digitales y validaciones automáticas expone a los negocios a inconsistencias contables y dificultades en auditorías internas.

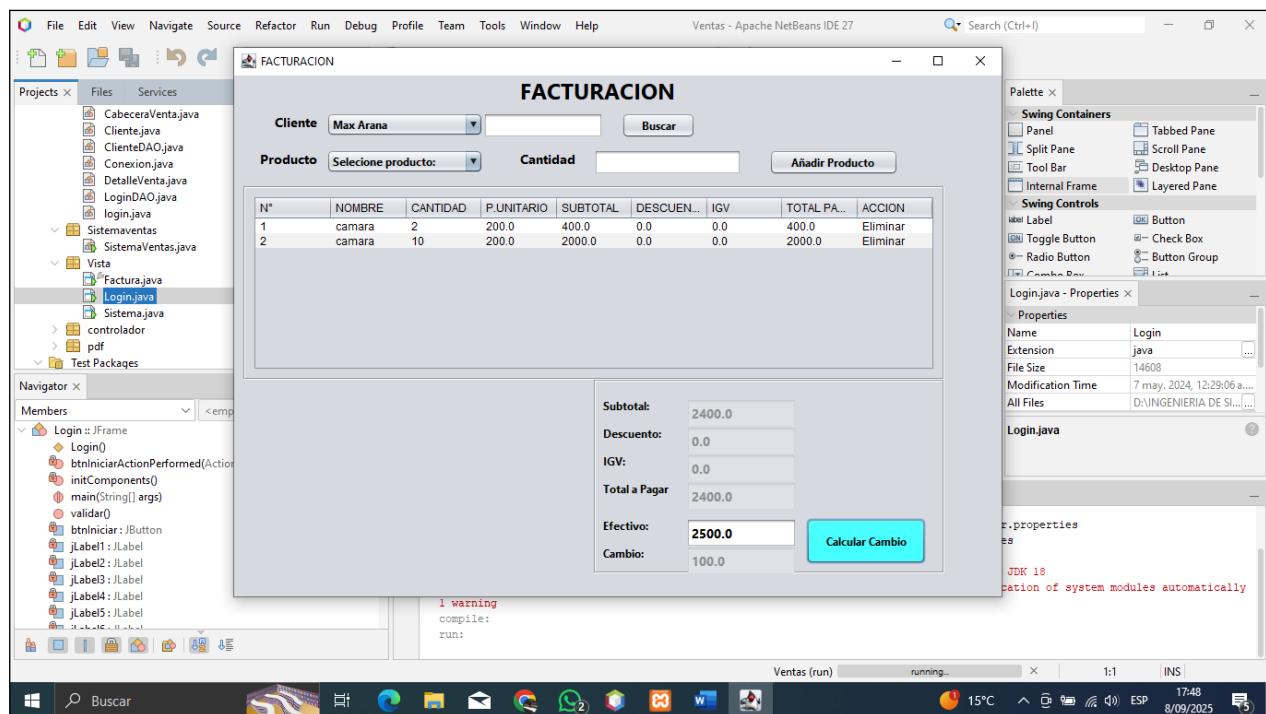
Problema identificado:

El sistema actual solo permite el acceso mediante login y el registro básico de ventas, sin generar comprobantes digitales ni validar correctamente los datos ingresados por el usuario.

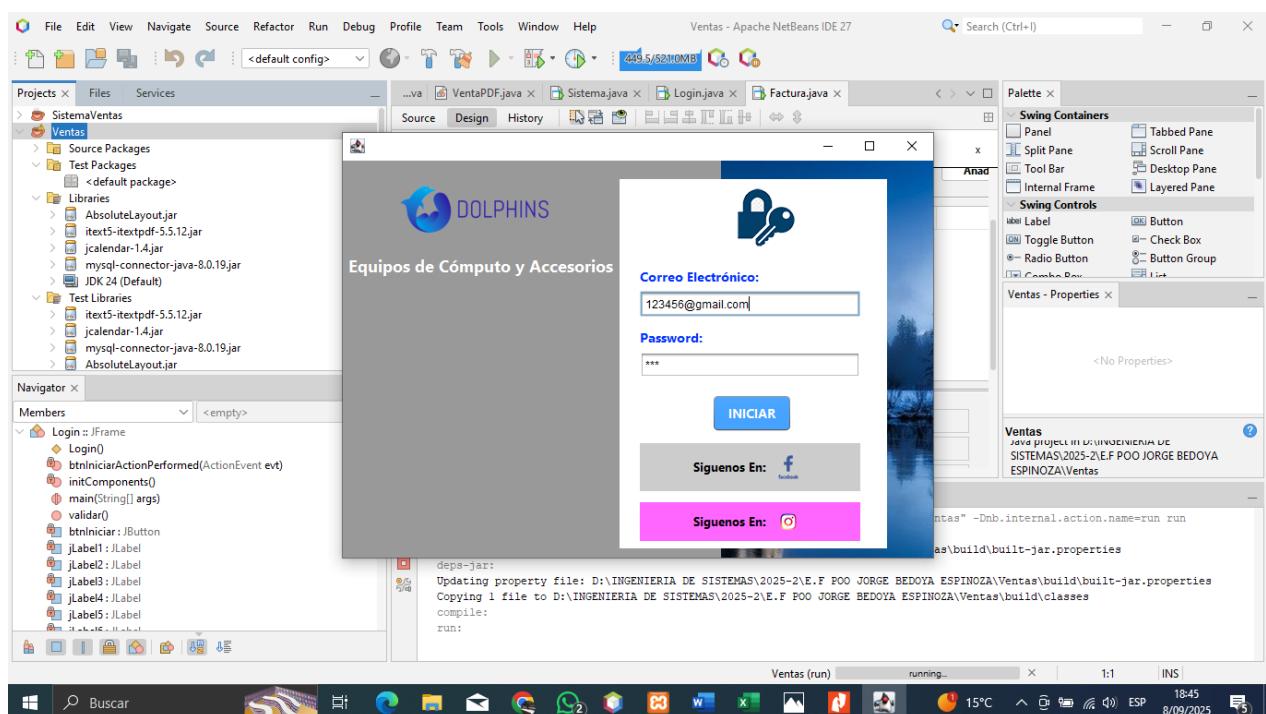
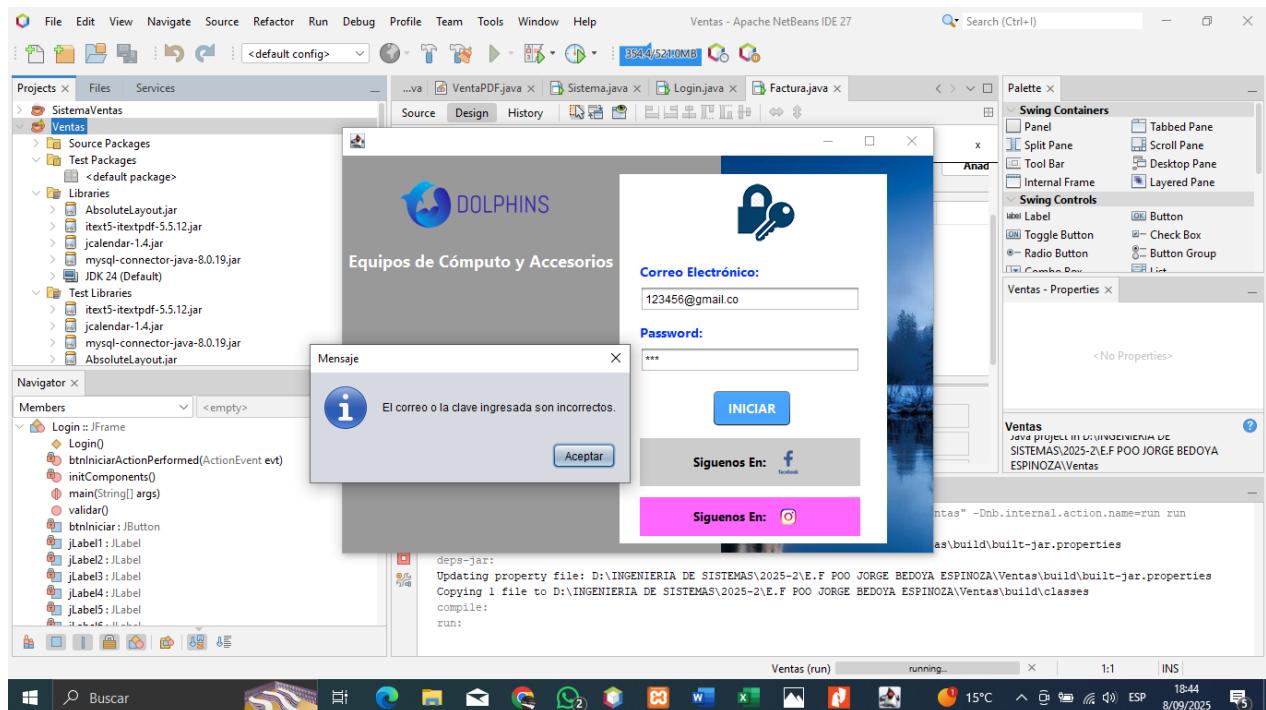
Solución propuesta: Se plantea la evolución del sistema hacia una versión más robusta que incluya:

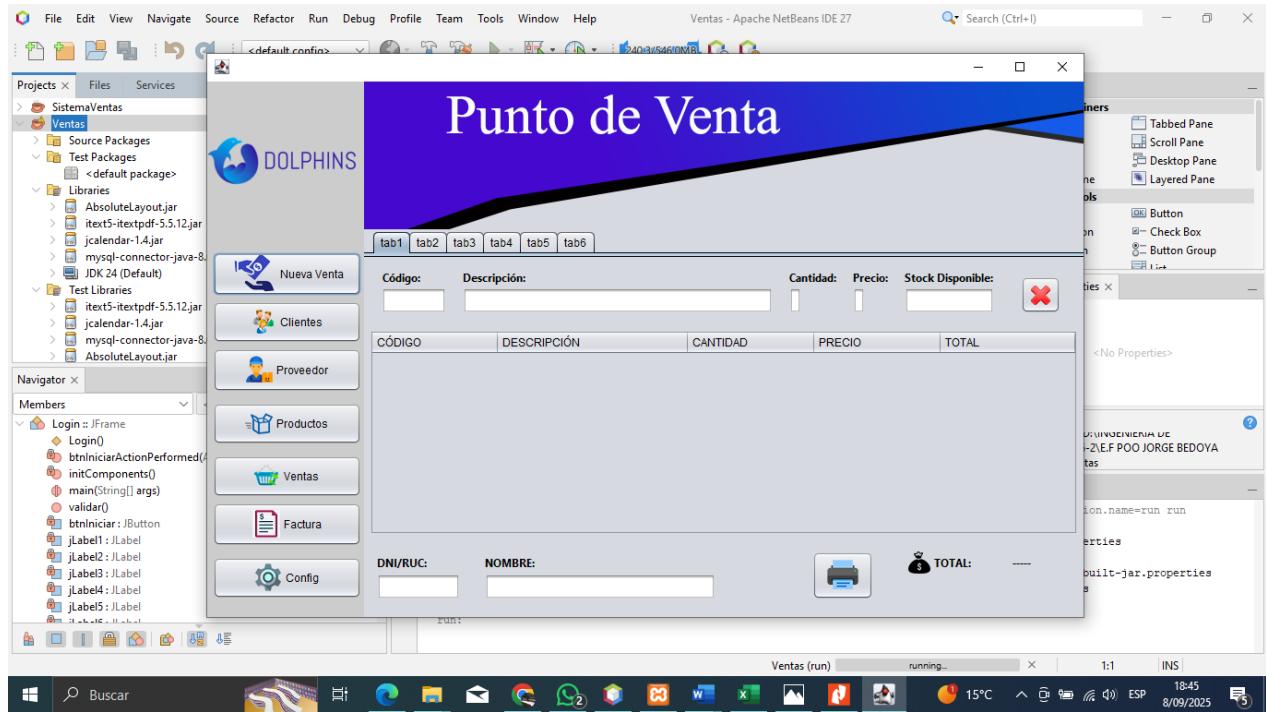
- **Generación automática de factura en PDF** con formato institucional.
- **Validaciones de entrada y manejo de excepciones** para garantizar la integridad de los datos.
- **Documentación técnica** que respalde cada mejora implementada.
- **Uso de IA generativa** para optimizar el desarrollo y enriquecer el proceso de documentación.

IMAGEN INICIAL DEL PROGRAMA

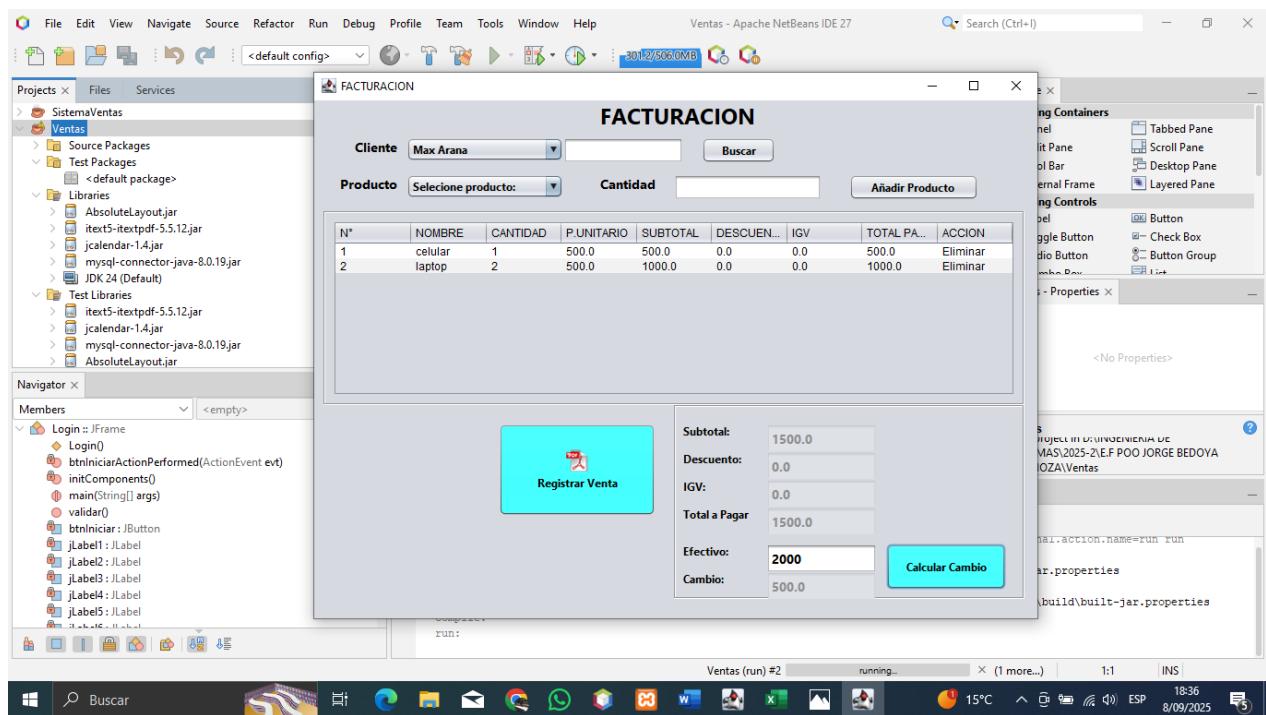


POSTERIOR DE HABER REALIZADO LAS MODIFICACIONES

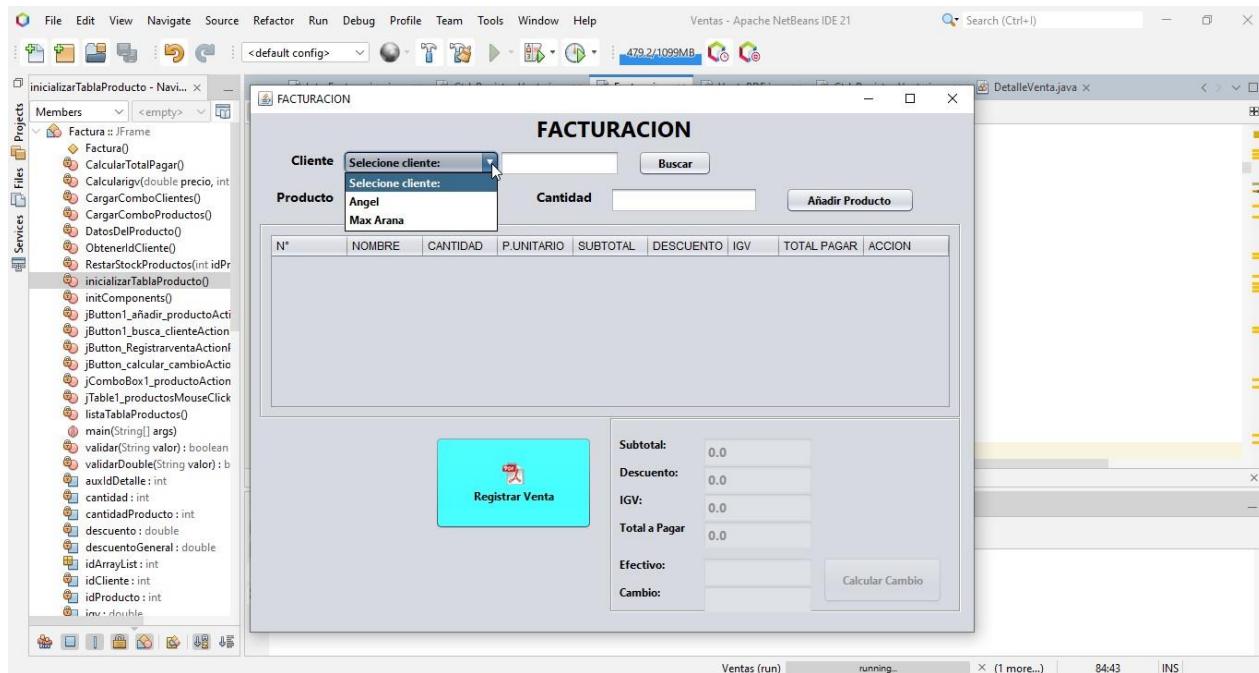




Seleccionamos la opción factura y se abrirá la siguiente imagen

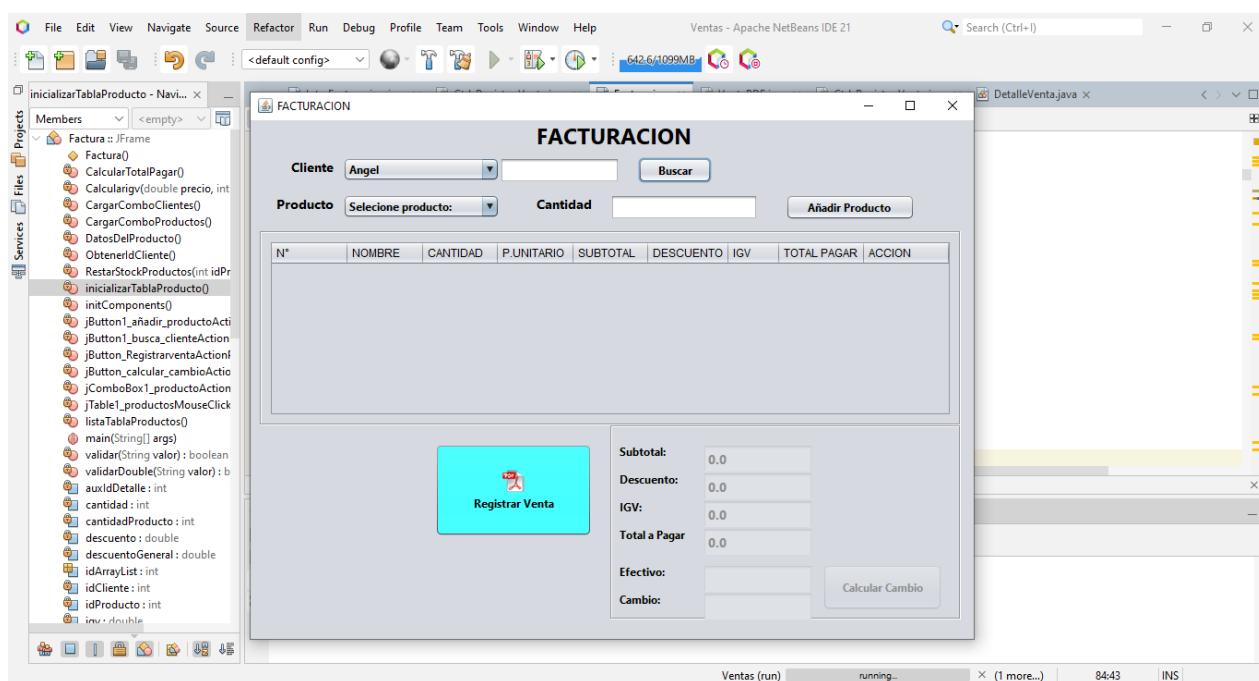
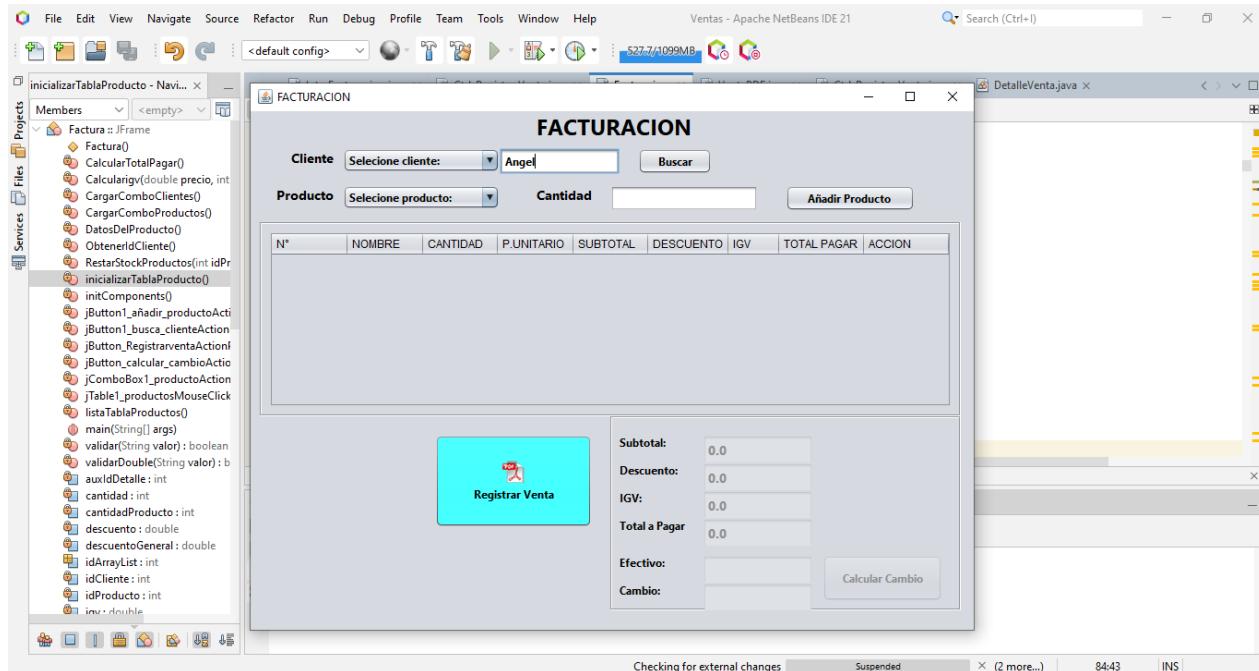


- La opción FACTURA consiste en la emisión de la factura al culminar la compra de los productos.
- **como primera funcionalidad es:**
En la opción SELECCIONA CLIENTE te da la opción de seleccionar los clientes que se encuentran en la base de datos.

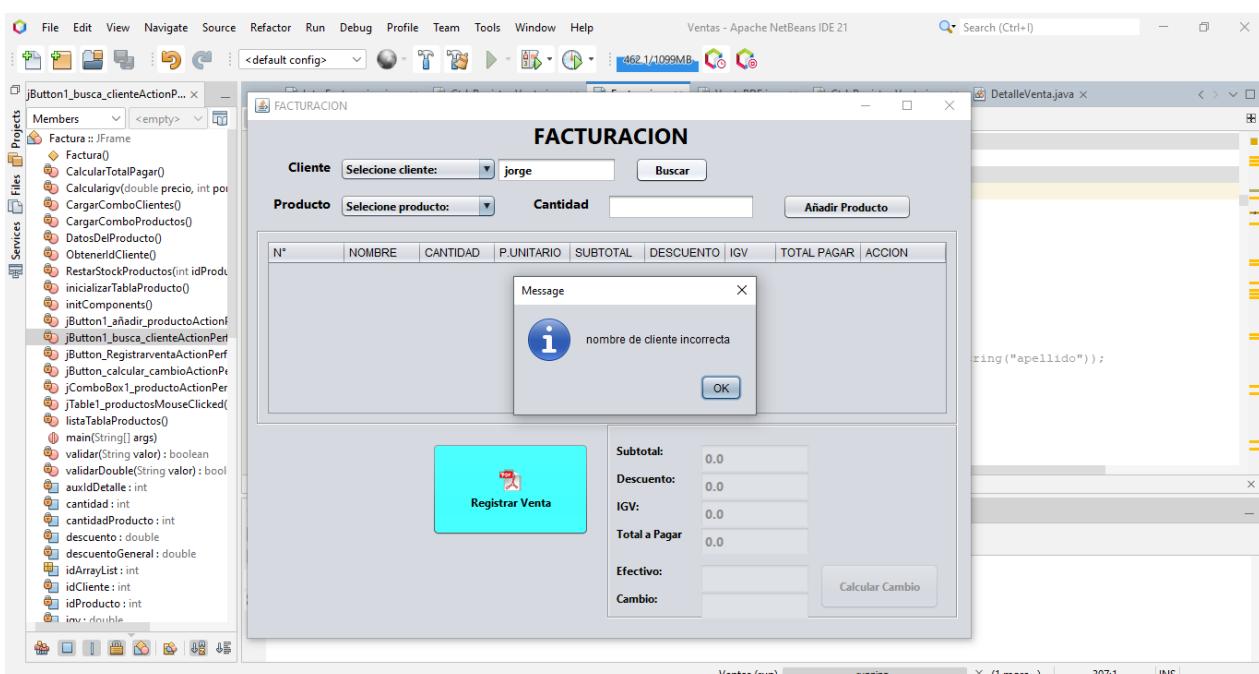
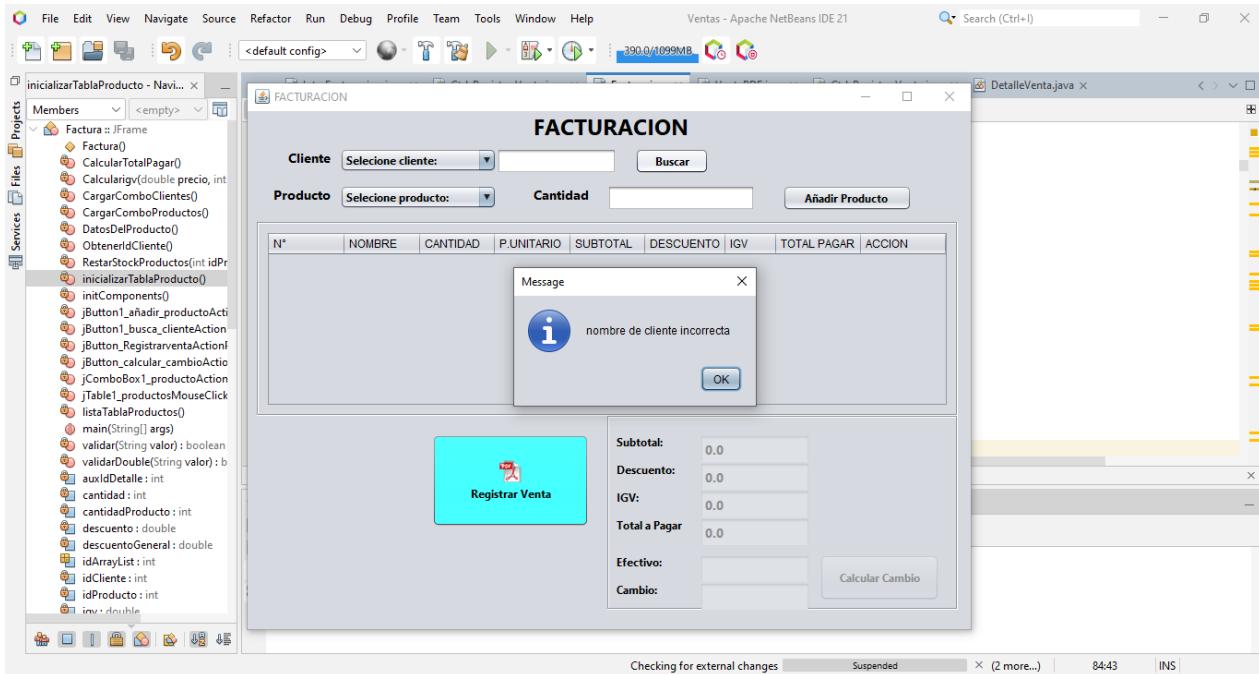


	id	dni	nombre	telefono	direccion
	1	1234598	Angel	924878	Lima - Perú
	2	77878878	Max Arana	999898999	Av. S/N

Así mismo la opción BUSCAR se encarga de buscar en la base de datos el usuario que escribes en el cuadro.

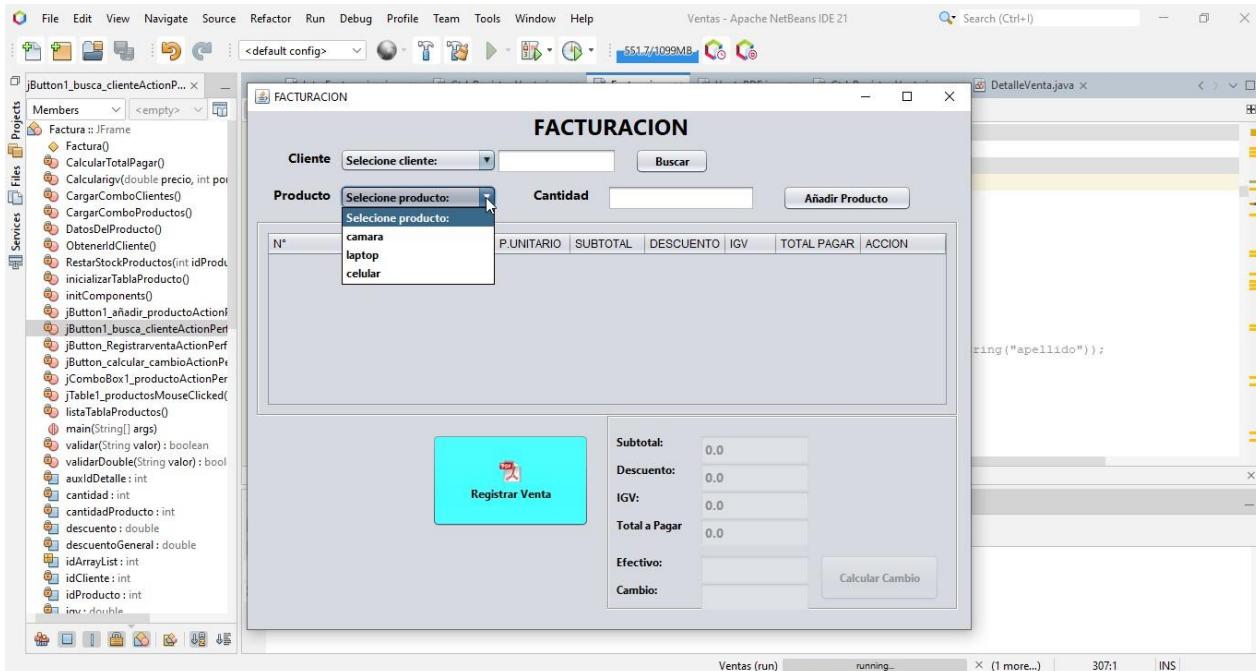
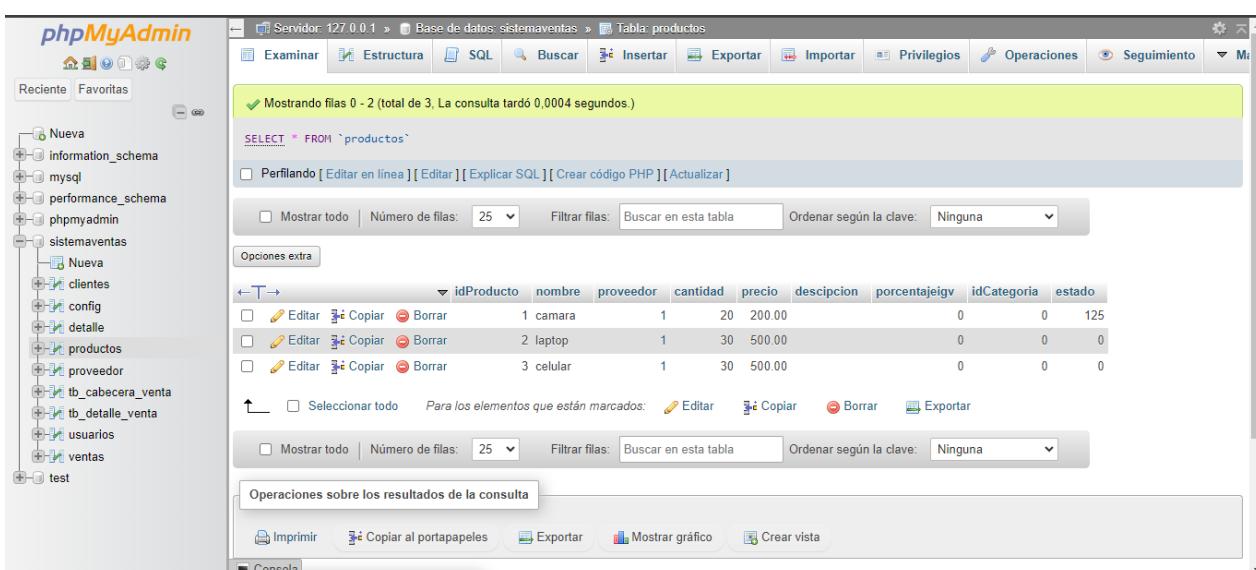


Por otro lado, si no ingresas nada o ingresas un nombre que no está en la base de datos te saldrá error.



- **Funcionalidad numero dos:**

la opción SELECCIONAR PRODUCTO te muestra todos los productos de la base de datos para que tu puedas seleccionar.

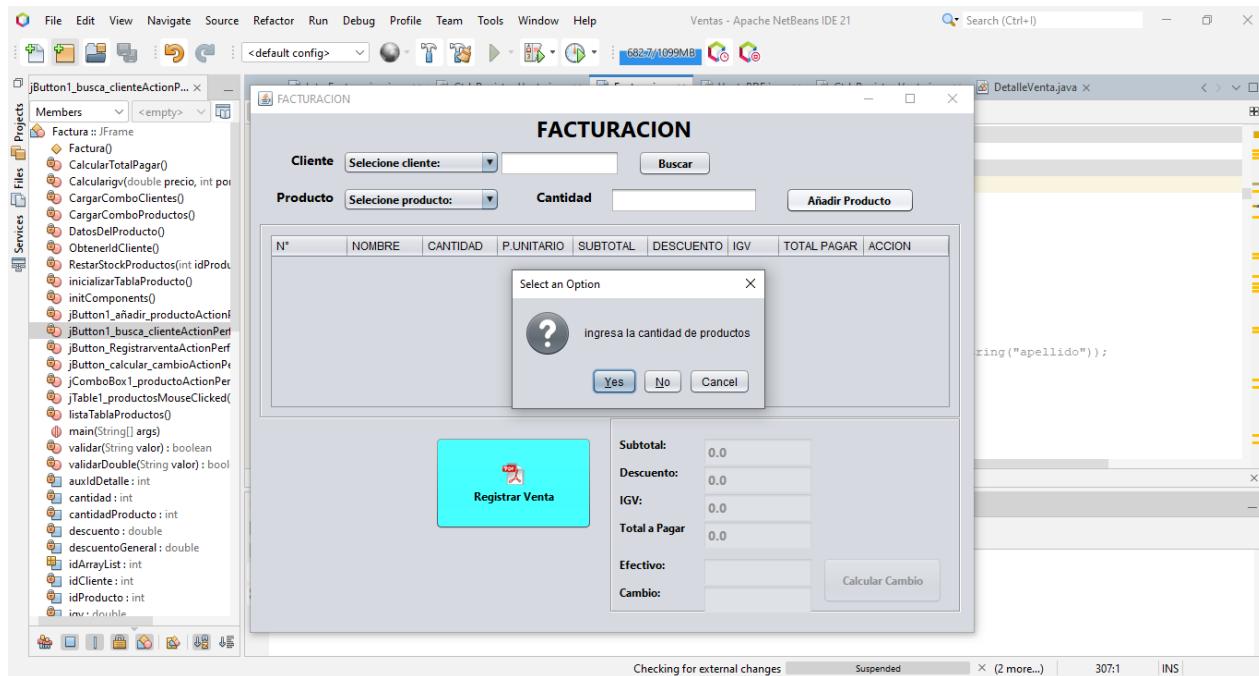



	idProducto	nombre	proveedor	cantidad	precio	descripcion	porcentajeigv	idCategoria	estado
<input type="checkbox"/>	1	camara		1	20	200.00	0	0	125
<input type="checkbox"/>	2	laptop		1	30	500.00	0	0	0
<input type="checkbox"/>	3	celular		1	30	500.00	0	0	0

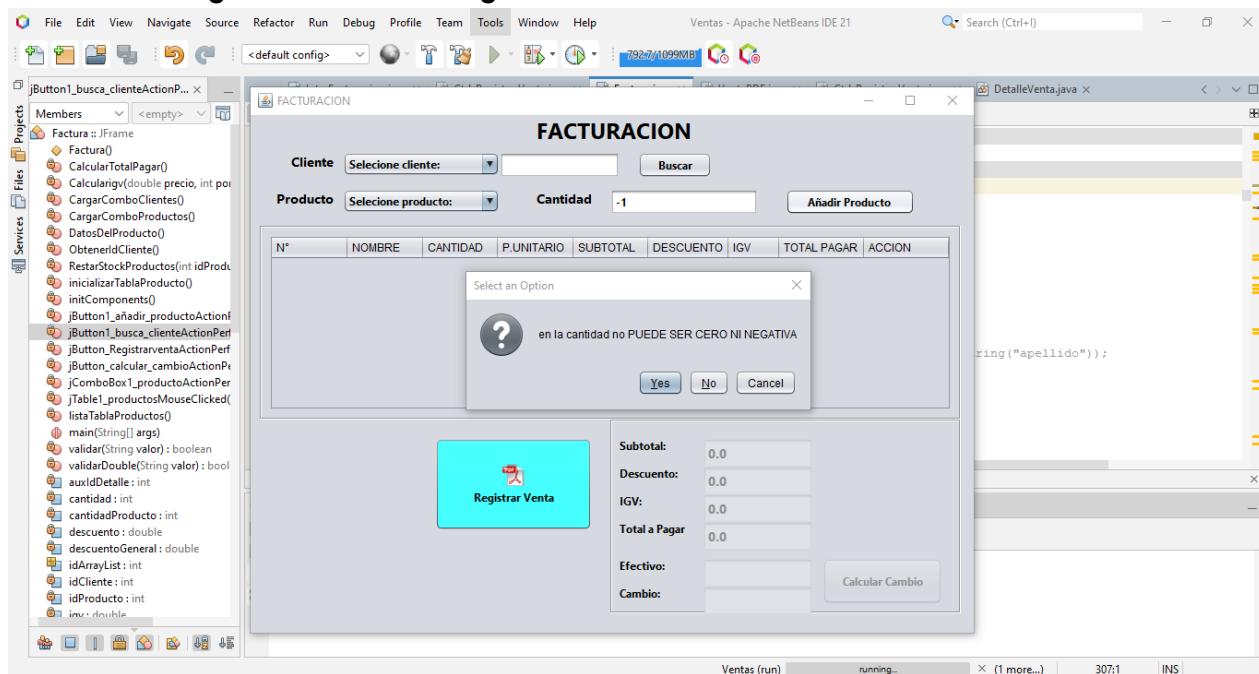
- **Funcionalidad número tres:**

la opción CANTIDAD se encarga de solicitarte un numero positivo mayor a cero y entero para saber que cantidad de productos estas comprando.

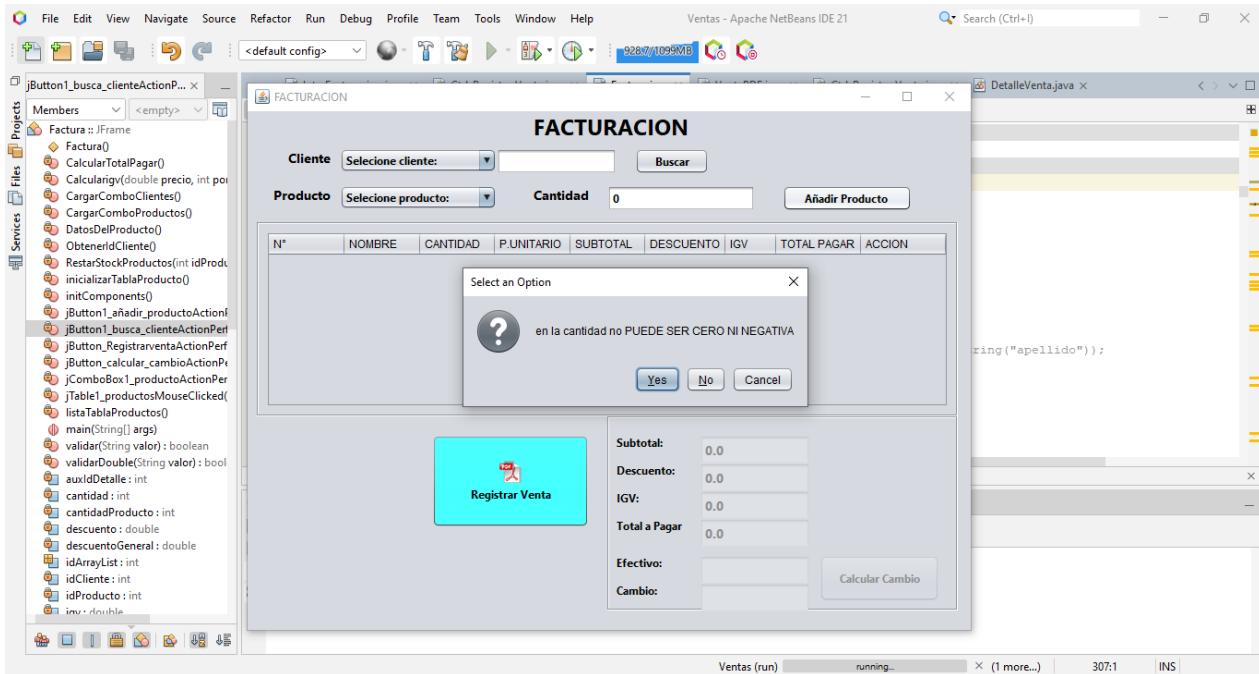
- **Sino ingresas nada**



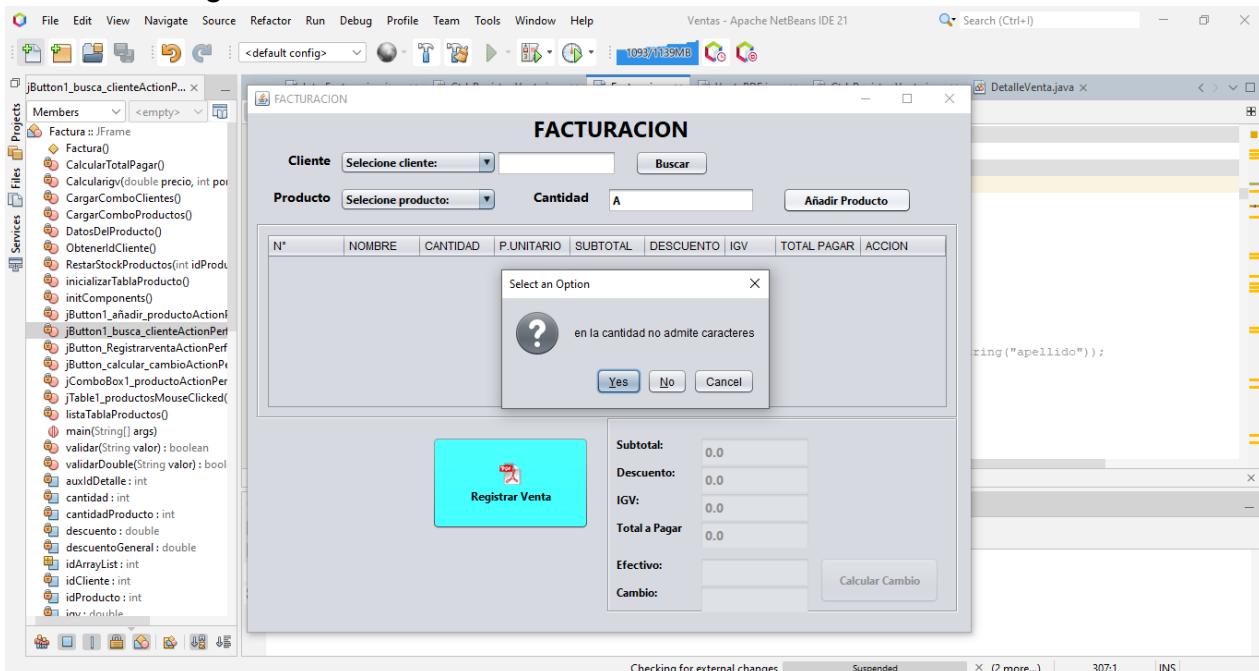
- Si ingresas un numero negativo



- Si ingresas el numero cero

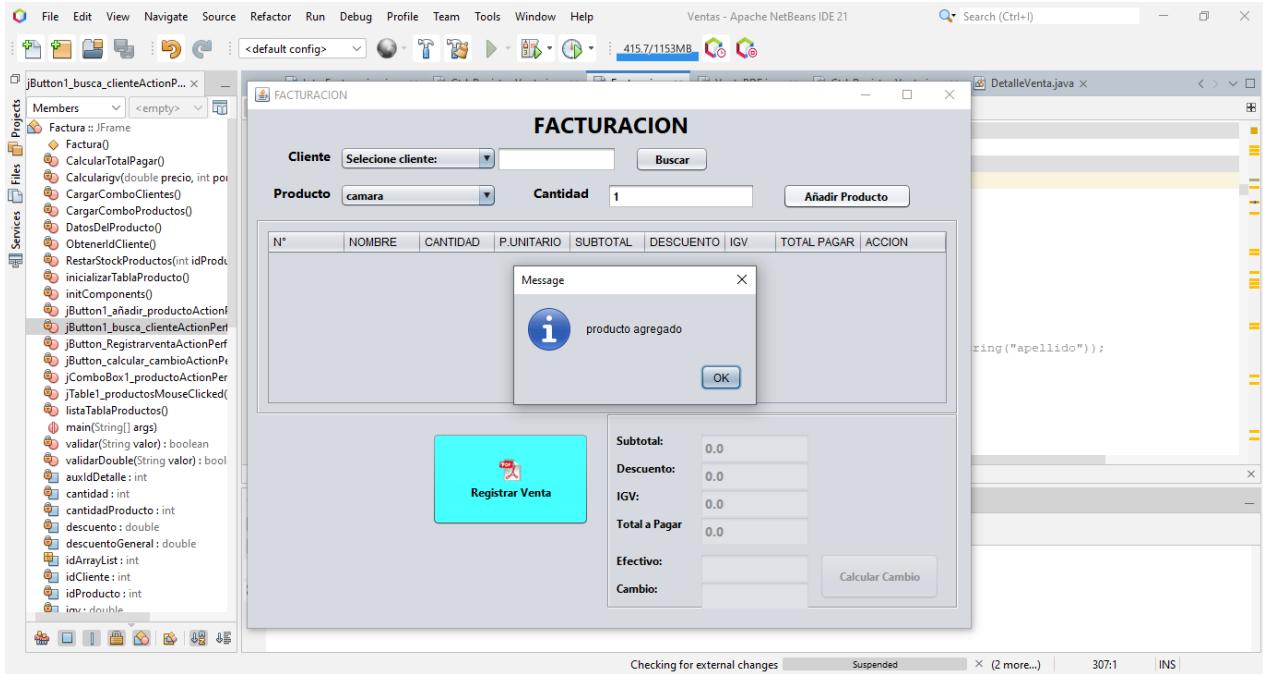


- Si ingresas una letra

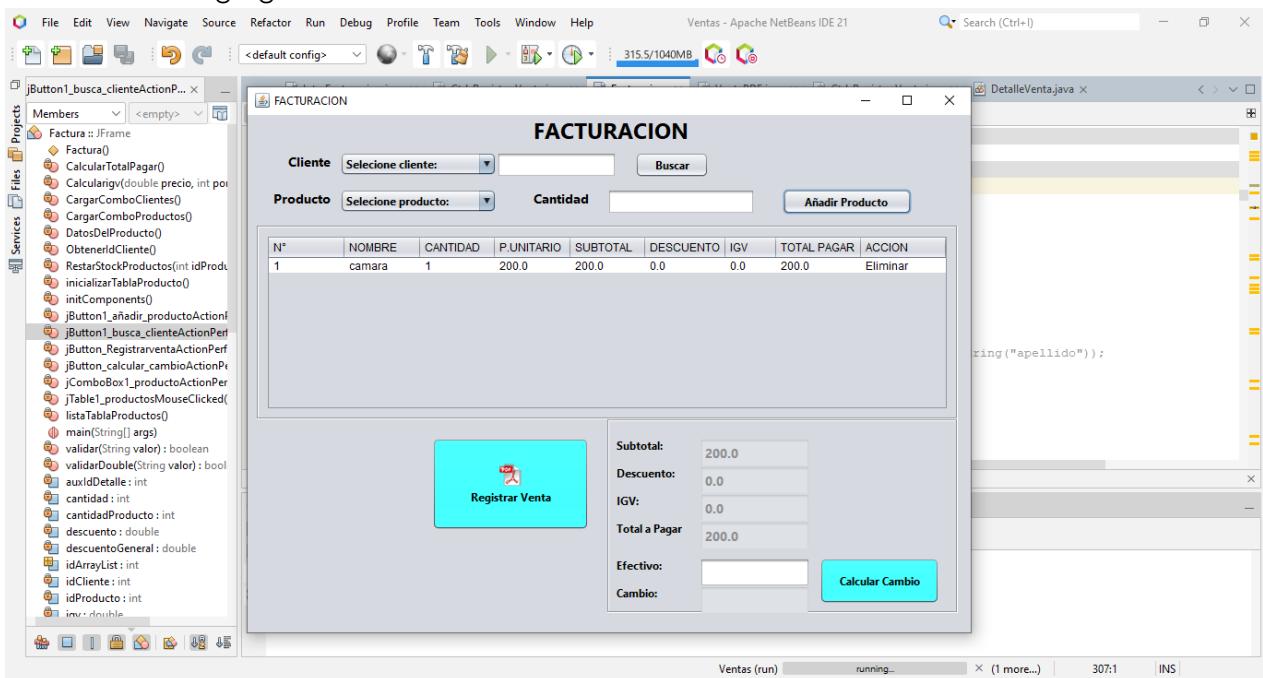


- **Funcionalidad número cuatro:**

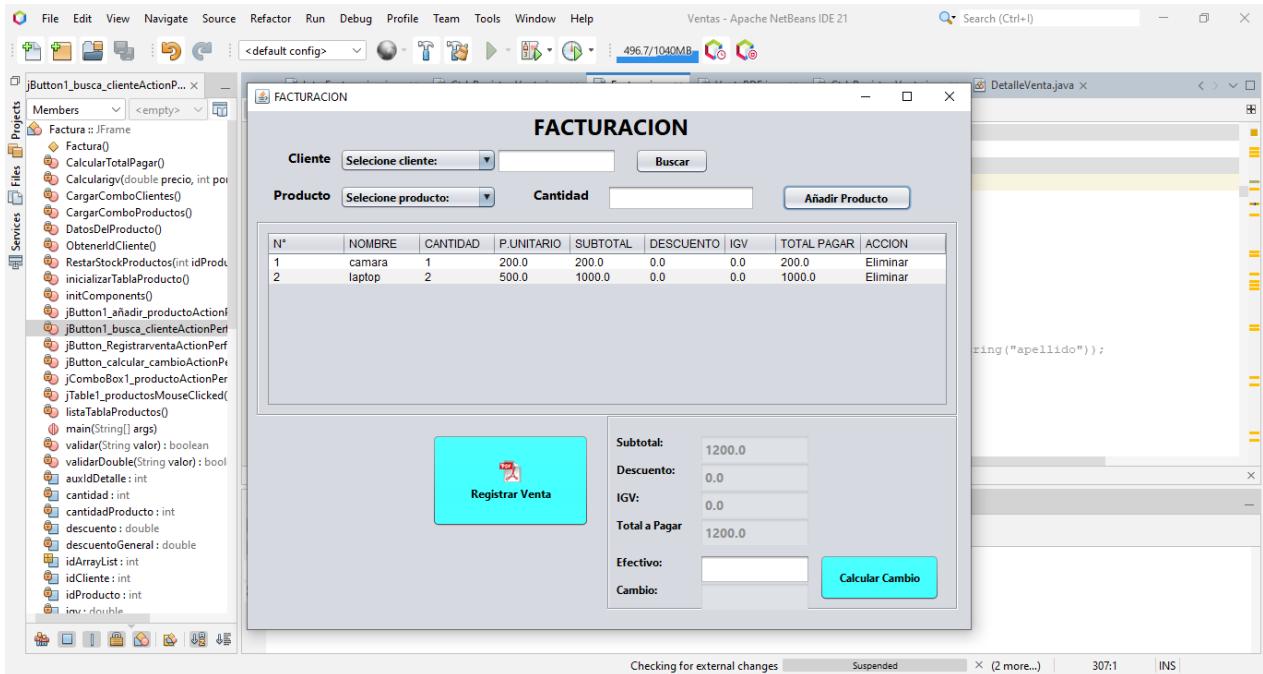
- la opción AÑADIR PRODUCTO, añade el producto seleccionado y la cantidad ingresada a la TABLA de la parte inferior.
- Seleccionamos el producto CAMARA y ingresamos la cantidad 1, si son opciones validas te sale el siguiente aviso.



La cual se agrega de manera correcta a la tabla inferior.



Si seleccionamos otro producto y registramos el numero 2

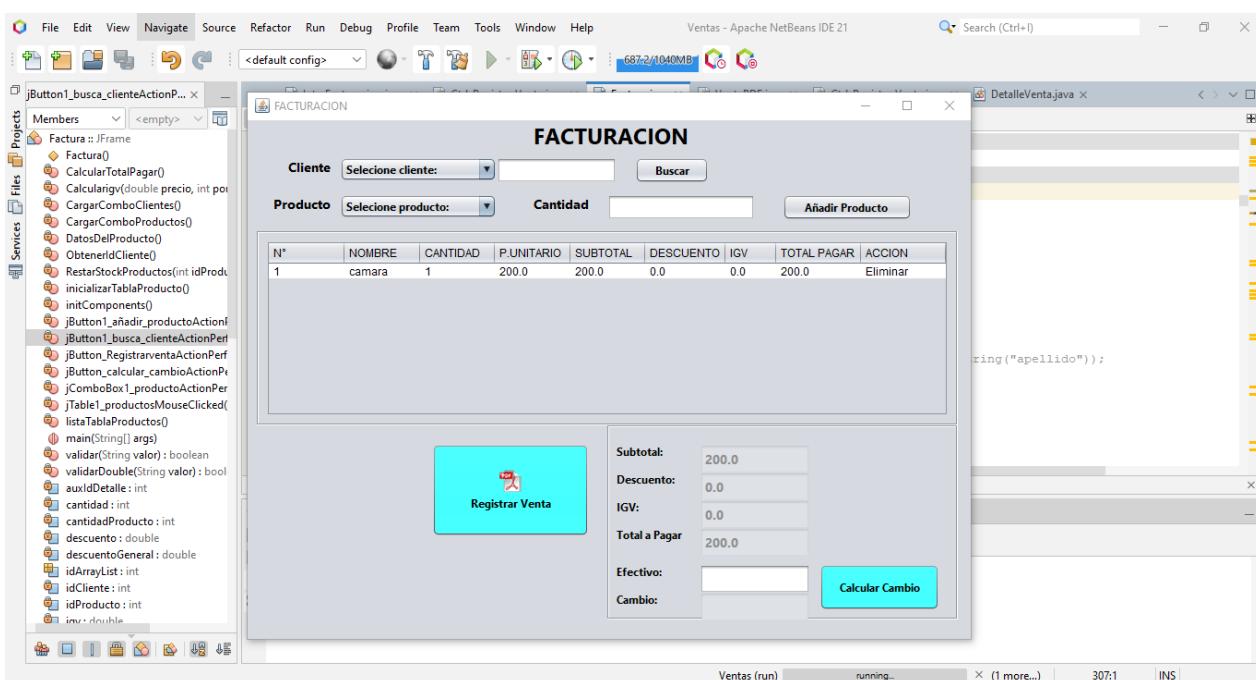
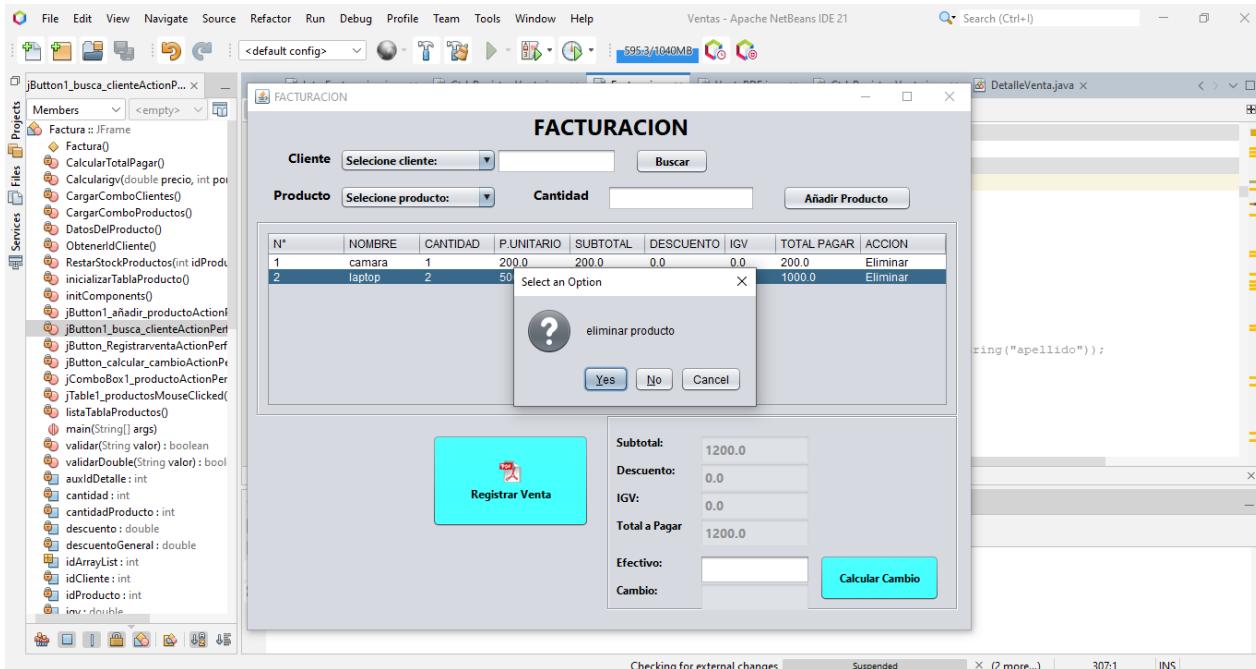


De igual manera se agrega a la tabla el producto, donde se puede evidenciar que se multiplica CANTIDAD POR SUBTOTAL donde nos da TOTAL PAGAR por producto.

	idProducto	nombre	proveedor	cantidad	precio	descripción	porcentajeIGV	idCategoria	estado
<input type="checkbox"/>	1	camara	1	20	200.00		0	0	125
<input type="checkbox"/>	2	laptop	1	30	500.00		0	0	0
<input type="checkbox"/>	3	celular	1	30	500.00		0	0	0

- **Funcionalidad número cinco:**

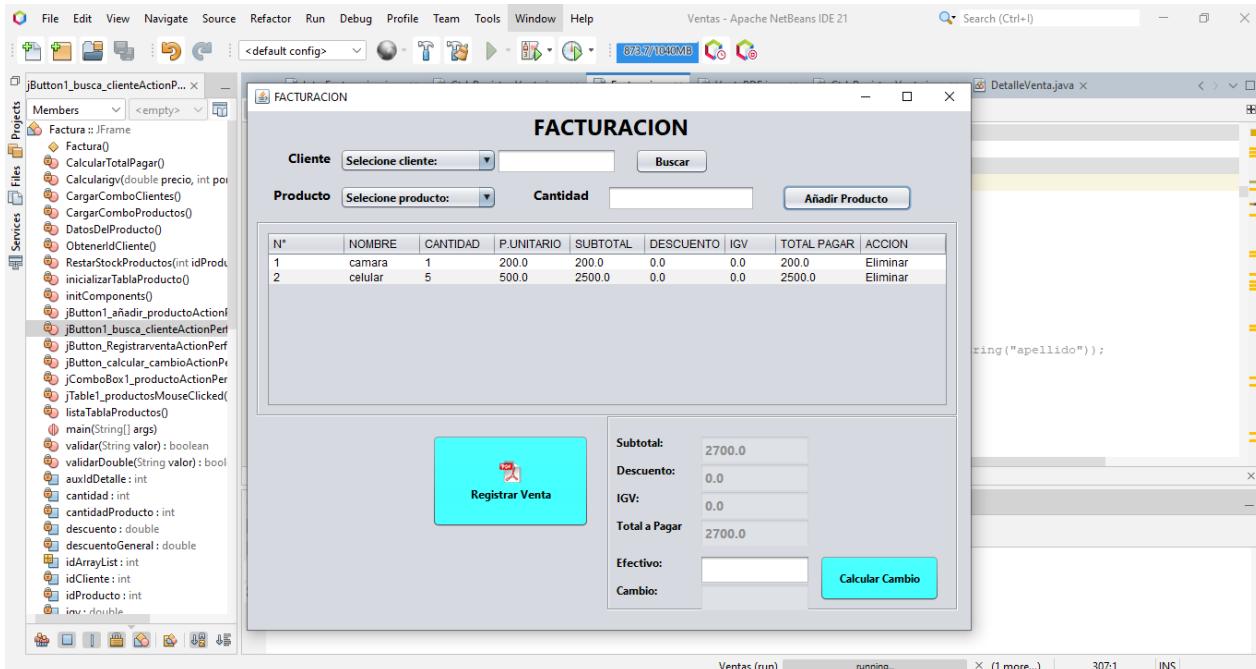
en la tabla en la columna ACCION se presenta una opción ELIMINAR, la cual cuando seleccionas dicha opción toda la fila del producto. Como se puede ver te consulta si estás seguro de eliminar dicho registro.



Se elimino el producto LAPTOP.

- Funcionalidad número seis:**

en la parte inferior se evidencia la siguiente información (Subtotal, Descuento, IGV, Total a Pagar), se calcula automáticamente cuando añades los productos con su respectiva cantidad.



Como se puede ver se suma el TOTAL PAGAR y se registra en Subtotal posterior realiza las operaciones aplicando tanto despuesto y IGV donde después de las operaciones te da el TOTAL A PAGAR.

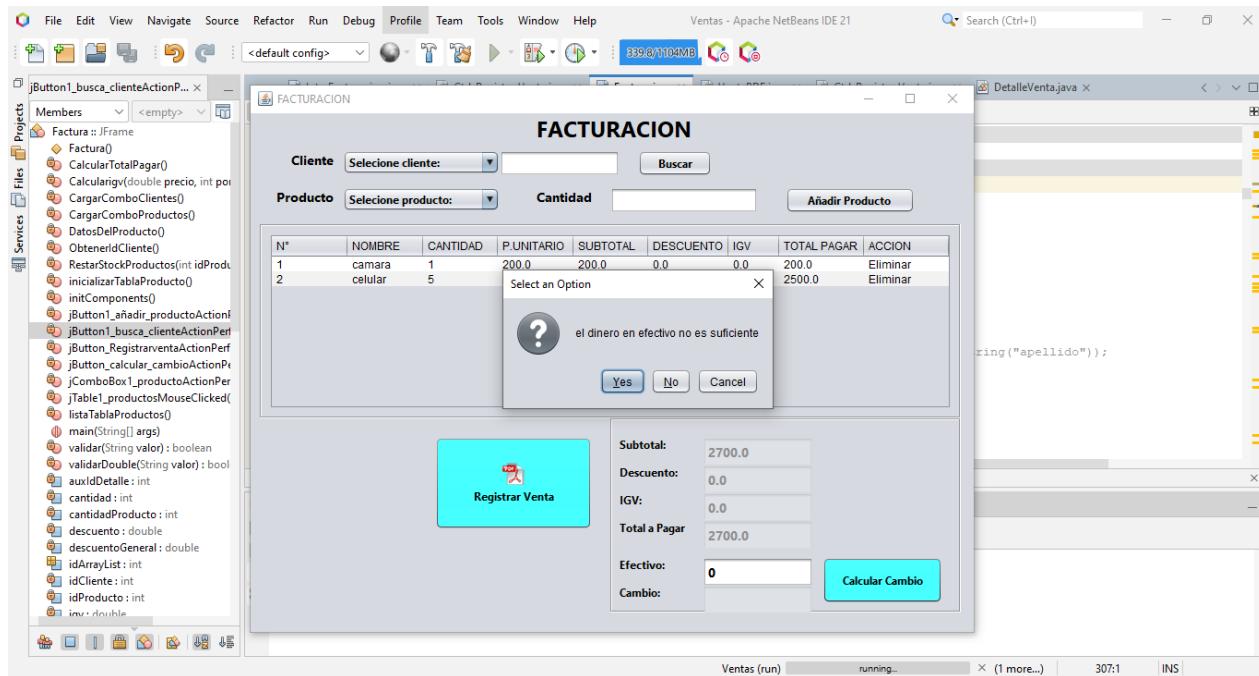
- Funcionalidad número siete:**

en la opción EFECTIVO se registra con que monto está haciendo el pago dicha opción solo permite un monto mayor a TOTAL A PAGAR, de lo contrario te dará error dado que no te pueden pagar con menos monto del que deben. Para poder probar dicha funcionalidad pasaremos a describir la siguiente funcionalidad.

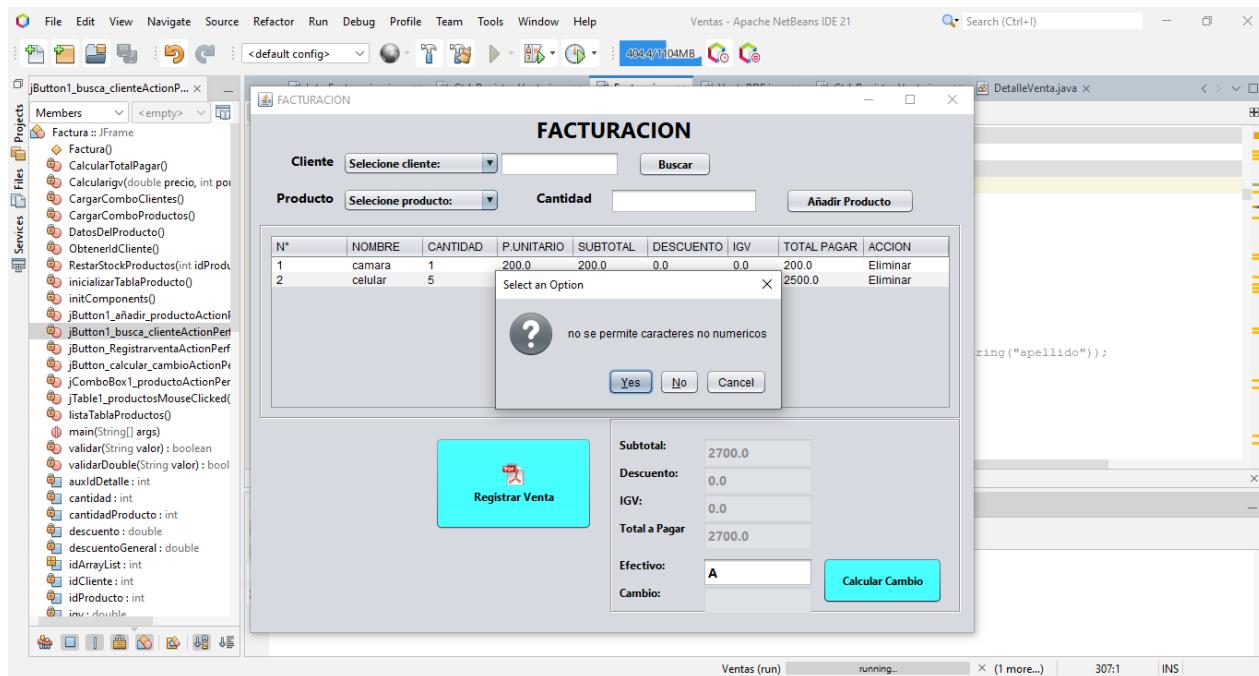
- Funcionalidad número ocho:**

la opción CALCULAR CAMBIO, calcula la diferencia entre el TOTAL A PAGAR y EFECTIVO.

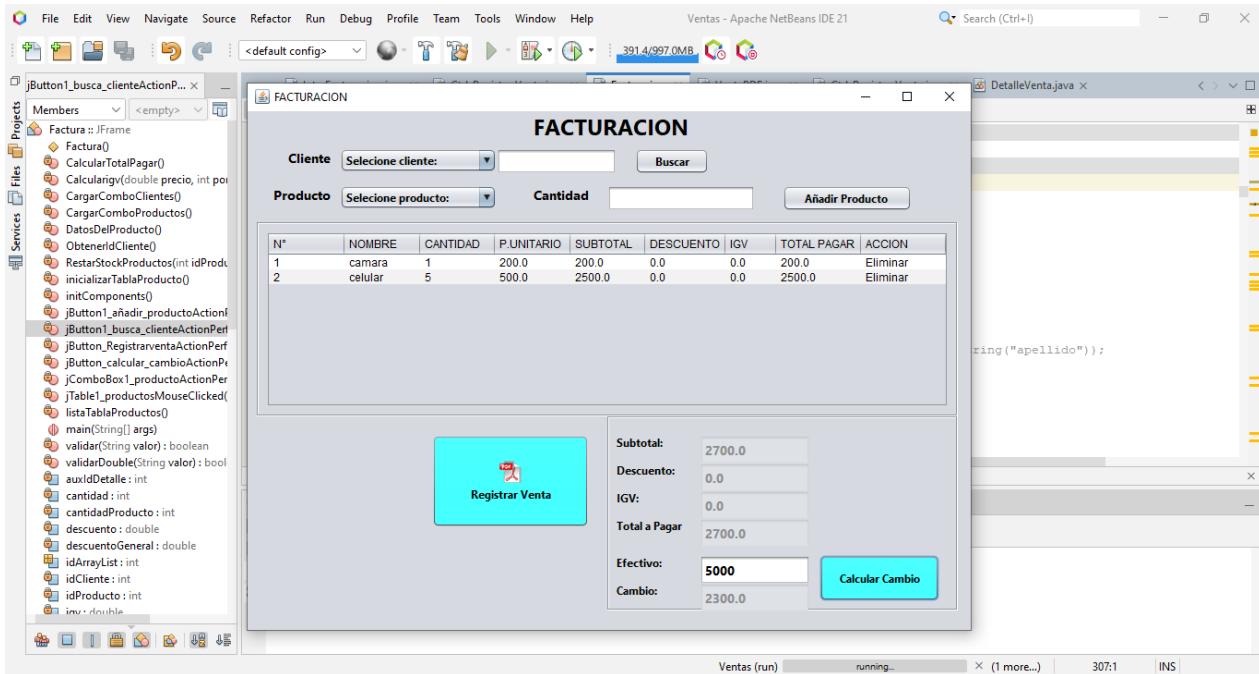
Si registras cero (0).



Si registras una letra



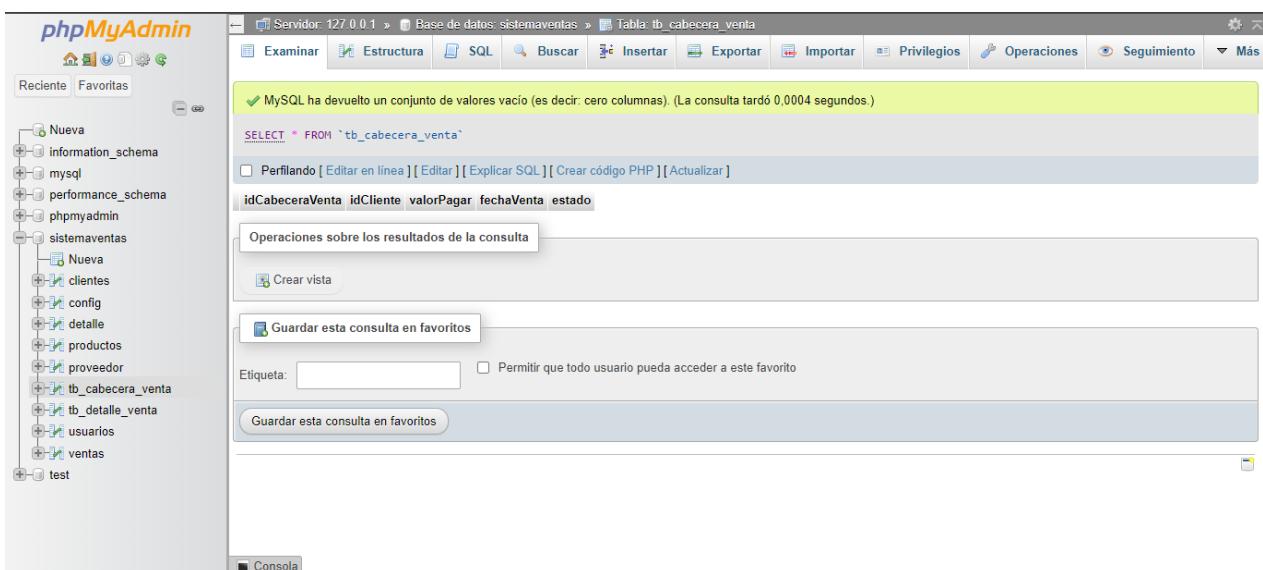
Si registras un numero valido

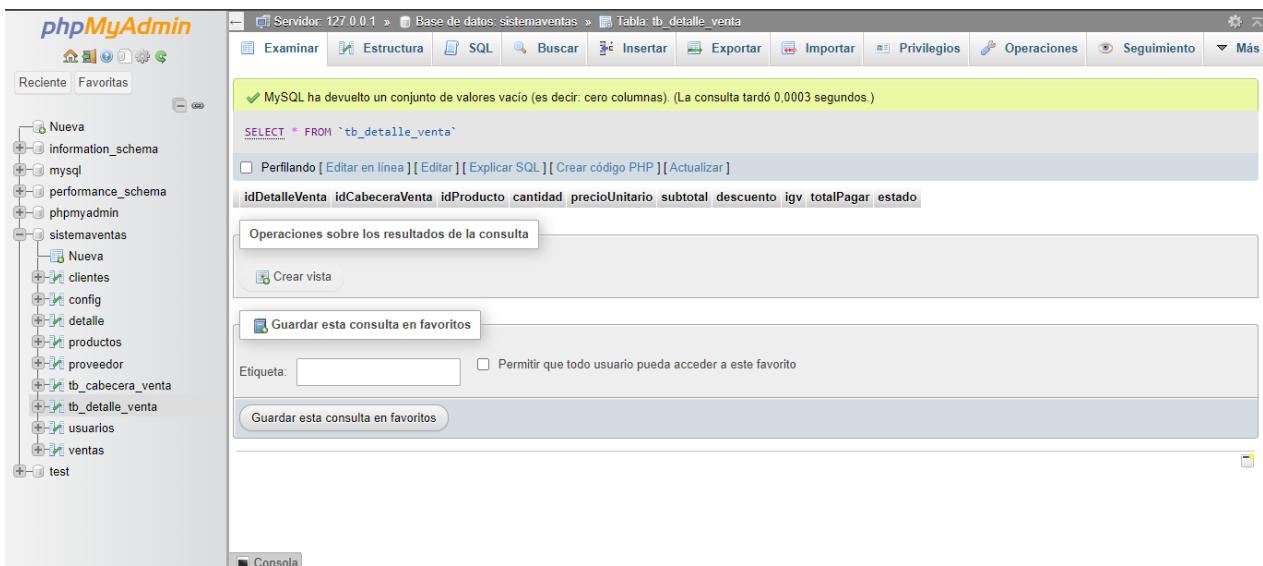


- Funcionalidad número nueve:**

la opción REGISTRAR VENTA, cuando eliges dicha opción pasara los siguientes casos:

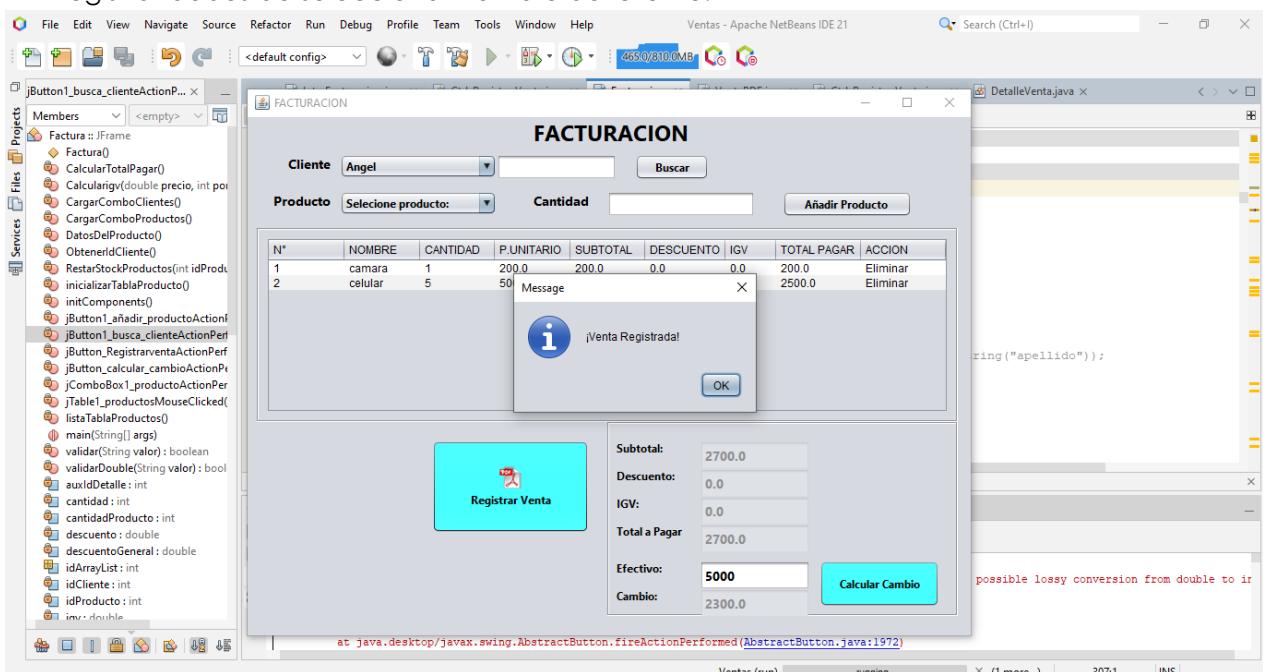
Caso número uno: los productos seleccionados se registrarán en la base de datos en la tabla TB_CABECERA_VENTA y TB_DETALLE_VENTA. Dado que son ya los productos vendidos.





The screenshot shows the phpMyAdmin interface connected to a MySQL server at 127.0.0.1. The current database is 'sistemataventas'. The 'tb_detalle_venta' table is selected. A query has been run: 'SELECT * FROM `tb_detalle_venta`', which returned an empty result set. The results page includes options to create a view or save the query as a favorite.

Cuando se registra de una manera correcta la venta, cabe precisar que debes de antes de registrar debes de seleccionar nombre del cliente.



The screenshot shows the Apache NetBeans IDE interface with a Java Swing application titled 'FACTURACION'. The application has a client selection dropdown ('Cliente: Angel'), a product selection dropdown ('Producto: Selecione producto:'), and a quantity input field ('Cantidad'). Below these is a table showing two items: 'camara' and 'celular'. A message dialog box is displayed in the center, stating '¡Venta Registrada!' (Sale registered!). On the right, the code editor shows the Java code for the application, and at the bottom, the terminal window shows some log messages.

Una vez realizada la venta esto pasara en la base de datos en la tabla PRODUCTO. Se disminuye los productos vendidos.

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: sistemaventas » Tabla: productos

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Ma

Mostrando filas 0 - 2 (total de 3. La consulta tardó 0,0014 segundos.)

SELECT * FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idProducto	nombre	proveedor	cantidad	precio	descripcion	porcentajeig	idCategoria	estado
<input type="checkbox"/>	1	camara	1	20	200.00		0	0	125
<input type="checkbox"/>	2	laptop	1	30	500.00		0	0	0
<input type="checkbox"/>	3	celular	1	30	500.00		0	0	0

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

Consola

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: sistemaventas » Tabla: productos

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Ma

Mostrando filas 0 - 2 (total de 3. La consulta tardó 0,0003 segundos.)

SELECT * FROM `productos`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idProducto	nombre	proveedor	cantidad	precio	descripcion	porcentajeig	idCategoria	estado
<input type="checkbox"/>	1	camara	1	19	200.00		0	0	125
<input type="checkbox"/>	2	laptop	1	30	500.00		0	0	0
<input type="checkbox"/>	3	celular	1	25	500.00		0	0	0

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

Consola

Así mismo en la tabla TB_CABECERA_VENTA y en TB_DETALLE_VENTA, se registran la información de los productos vendidos.

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: sistemaventas » Tabla: tb_cabecera_venta

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Ma

Mostrando filas 0 - 0 (total de 1. La consulta tardó 0,0003 segundos.)

SELECT * FROM `tb_cabecera_venta`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla

Opciones extra

	idCabeceraVenta	idCliente	valorPagar	fechaVenta	estado
<input type="checkbox"/>	1	1	2700.00	2024-05-12	1

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

Guardar esta consulta en favoritos

Consola

Permitir que todo usuario pueda acceder a este favorito

phpMyAdmin

Servidor: 127.0.0.1 » Base de datos: sistemaventas » Tabla: tb_detalle_venta

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Ma

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0,0005 segundos.)

SELECT * FROM `tb_detalle_venta`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Opciones extra

	idDetalleVenta	idCabeceraVenta	idProducto	cantidad	precioUnitario	subtotal	descuento	igv	totalPagar	estado
<input type="checkbox"/>	1	1	1	1	200.00	200.00	0.00	0.00	200.00	1
<input type="checkbox"/>	2	2	3	5	500.00	1500.00	0.00	0.00	1500.00	2

Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar Exportar

Mostrar todo Número de filas: 25 Filtrar filas: Buscar en esta tabla Ordenar según la clave: Ninguna

Operaciones sobre los resultados de la consulta

Imprimir Copiar al portapapeles Exportar Mostrar gráfico Crear vista

Guardar esta consulta en favoritos Consola

Por otro lado, crea la factura y lo exporta en formato PDF.

Venta_Max Arana_2025_09_08.pdf - Nitro Pro 10

ARCHIVO INICIO REVISAR FORMULARIOS PROTEGER AYUDA

Mano Zoom Ajustar Anchura Ajustar Página Rotar vista Ver

Selección Escribir QuickSign texto Herramientas PDF Combinar Portafolio Crear Word Excel PowerPoint otro PDF/A Convertir

RUC: 1058485710 NOMBRE: Jorge Bedoya Espinoza Factura: 001
 TELEFONO: 998485658 Fecha: 2025/09/08
 DIRECCION: Huancayo
 RAZON SOCIAL: Universidad Continental
 - Examen Final

Datos del cliente:

dni: 77878878	Nombre: Max Arana	Telefono: 999898999	Direccion: Av. S/N
---------------	-------------------	---------------------	--------------------

Cantidad:	Descripcion:	Precio Unitario:	Precio Total:
1	celular	500.0	500.0
2	laptop	500.0	1000.0

Total a pagar: 1500.0

Cancelación y firma

Gracias por su compra!

1 DE 1

Windows Search Bar Taskbar Icons Date: 8/09/2025 Time: 18:37 Temperature: 15°C

The screenshot shows the phpMyAdmin interface with the following details:

- Servidor:** 127.0.0.1
- Base de datos:** sistemaventas
- Tabla:** clientes
- Consulta ejecutada:** SELECT * FROM `clientes`
- Filas mostradas:** 0 - 1 (total de 2)
- Columnas:** id, dni, nombre, telefono, direccion
- Datos:**

	1	1234598	Angel	924878	Lima - Perú
	2	77878878	Max Arana	999898999	Av. S/N

Se crea un pdf de la factura con el nombre del cliente.

FIN DEL PROGRAMA.

2. Justificación del uso de metodologías ágiles

La metodología ágil permite desarrollar software de forma iterativa, adaptativa y centrada en la entrega de valor. En este proyecto, se simula el uso de **Scrum**, adaptado a un entorno individual, donde el desarrollador asume todos los roles: *Product Owner, Scrum Master y Developer*.

Rol asumido	Actividades realizadas
Product Owner	<ul style="list-style-type: none"> - Definición del problema: falta de comprobantes digitales y validaciones. - Priorización del backlog técnico. - Identificación de funcionalidades clave para el usuario final.
Scrum Master	<ul style="list-style-type: none"> - Planificación de sprints simulados. - Organización de tareas por entregables. - Evaluación continua del avance y ajustes en el enfoque. - Simulación de reuniones de seguimiento (diarias y retrospectiva).
Developer	<ul style="list-style-type: none"> - Implementación del login y registro de ventas. - Desarrollo de módulo de generación de factura en PDF. - Programación de validaciones y manejo de excepciones. - Refactorización del código para mejorar legibilidad y modularidad. - Integración de IA generativa para optimizar funciones y documentación.

Ventajas aplicadas:

- **Iteraciones cortas (sprints)** que permiten incorporar mejoras progresivas.
- **Backlog técnico** con tareas priorizadas: generación de PDF, validaciones, refactorización.
- **Flexibilidad** para ajustar funcionalidades según el avance y los hallazgos técnicos.

Sprint simulado:

- *Sprint 1:* Diagnóstico del sistema y planificación de mejoras.
- *Sprint 2:* Implementación de factura PDF.
- *Sprint 3:* Validaciones de entrada y manejo de errores.

ID	Tarea técnica	Prioridad	Sprint asignado	Estado	Entregable esperado
1	Implementar login con validación desde BD	Alta	Sprint 1	Completado	Pantalla de acceso funcional con validación real
2	Registrar ventas con cálculo básico	Alta	Sprint 1	Completado	Registro en BD con totales simples
3	Generar factura en PDF con formato institucional	Alta	Sprint 2	En desarrollo	Archivo PDF con datos de venta y cliente
4	Validar entradas numéricas y campos vacíos	Alta	Sprint 3	Pendiente	Mensajes de error y bloqueo de acciones inválidas
5	Manejar excepciones en tiempo de ejecución	Media	Sprint 3	Pendiente	Código robusto con try-catch y alertas
6	Refactorizar funciones repetitivas	Media	Sprint 3	Pendiente	Código modular y reutilizable
7	Documentar mejoras con apoyo de IA generativa	Media	Sprint 3	En desarrollo	Informe técnico y fragmentos de código optimizado



El siguiente diagrama representa la simulación de roles asumidos en el desarrollo individual del sistema, siguiendo un enfoque ágil. Cada bloque muestra las responsabilidades específicas que contribuyen a la evolución progresiva del proyecto.

3. Explicación del uso de inteligencia artificial generativa en el desarrollo

La inteligencia artificial generativa, en este caso representada por Copilot, ha sido utilizada como herramienta de apoyo en múltiples fases del desarrollo:

Aplicaciones concretas:

- **Análisis del código existente:** sugerencias para modularizar funciones y mejorar legibilidad.
- **Generación de código de validación:** expresiones regulares, estructuras condicionales, manejo de excepciones.
- **Diseño de estructura PDF:** formato, contenido, estilo institucional.

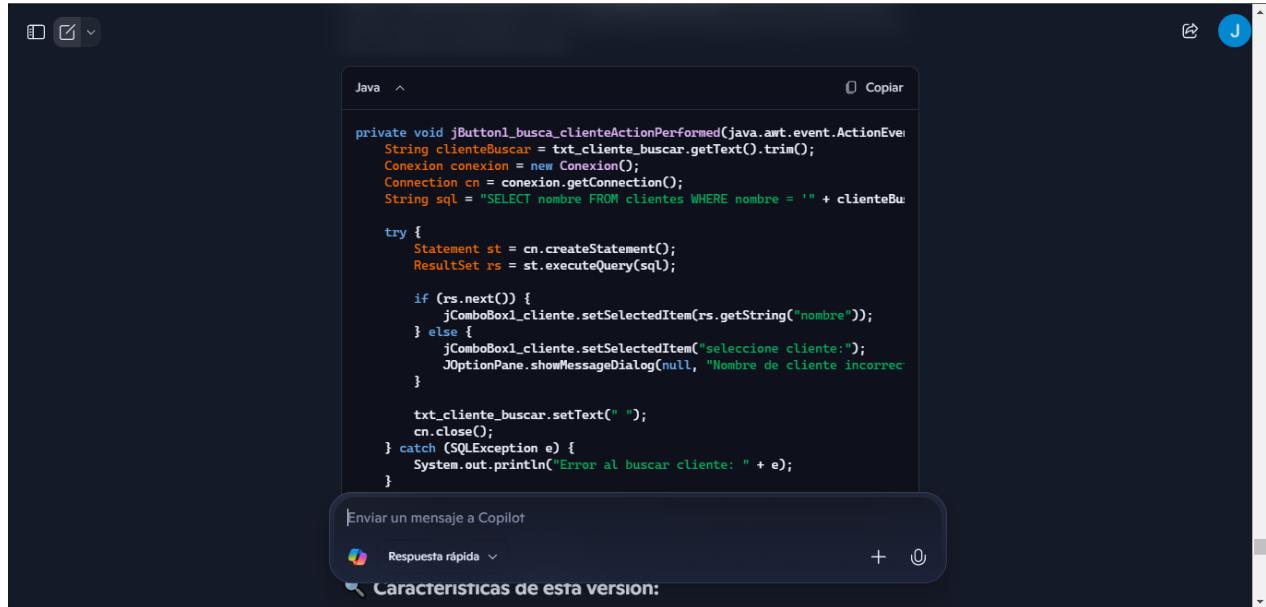
Impacto en el proyecto:

- Aceleración del desarrollo.
- Mejora en la calidad del código.

- Enriquecimiento de la presentación institucional del producto.

MEJORAMIENTO DEL CODIGO PARA LAS VALIDACIONES

Validacion 01 : validacion del cliente



```

private void jButton1_busca_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    String clienteBuscar = txt_cliente_buscar.getText().trim();
    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();
    String sql = "SELECT nombre FROM clientes WHERE nombre = '" + clienteBu
    try {
        Statement st = cn.createStatement();
        ResultSet rs = st.executeQuery(sql);

        if (rs.next()) {
            jComboBox1_cliente.setSelectedItem(rs.getString("nombre"));
        } else {
            jComboBox1_cliente.setSelectedItem("selecciona cliente:");
            JOptionPane.showMessageDialog(null, "Nombre de cliente incorrec
        }

        txt_cliente_buscar.setText("");
        cn.close();
    } catch (SQLException e) {
        System.out.println("Error al buscar cliente: " + e);
    }
}

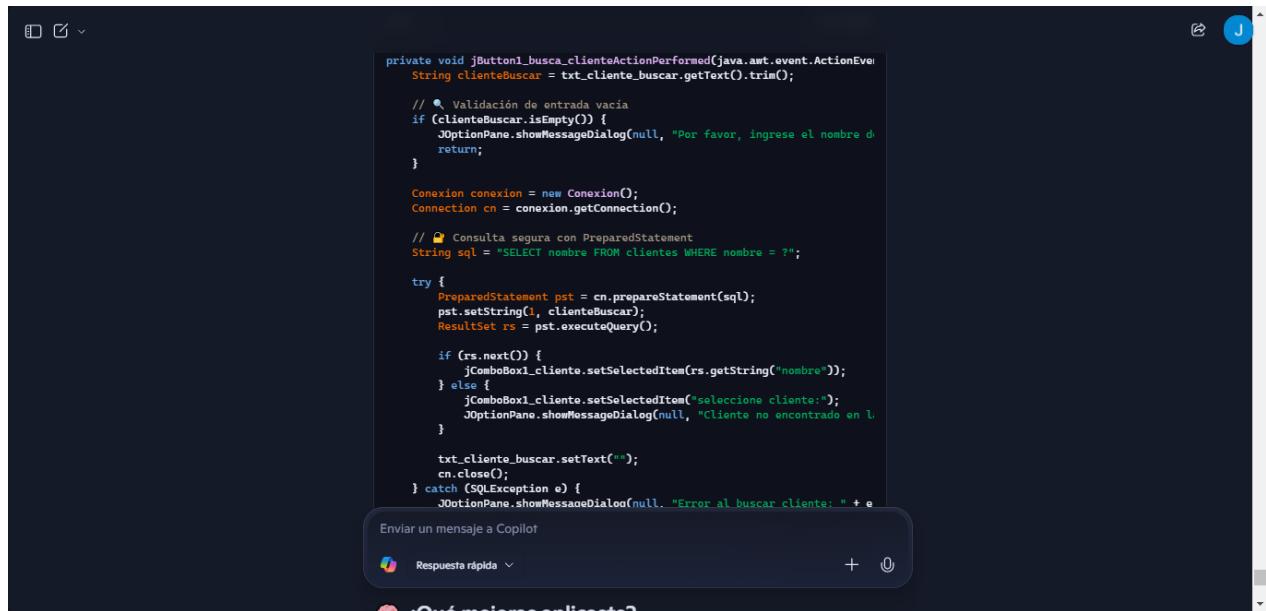
```

Enviar un mensaje a Copilot

Respuesta rápida ▾

Características de esta versión:

Sin validacion



```

private void jButton1_busca_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    String clienteBuscar = txt_cliente_buscar.getText().trim();

    // ✖ Validación de entrada vacía
    if (clienteBuscar.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Por favor, ingrese el nombre de cliente");
        return;
    }

    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();

    // ✖ Consulta segura con PreparedStatement
    String sql = "SELECT nombre FROM clientes WHERE nombre = ?";

    try {
        PreparedStatement pst = cn.prepareStatement(sql);
        pst.setString(1, clienteBuscar);
        ResultSet rs = pst.executeQuery();

        if (rs.next()) {
            jComboBox1_cliente.setSelectedItem(rs.getString("nombre"));
        } else {
            jComboBox1_cliente.setSelectedItem("selecciona cliente:");
            JOptionPane.showMessageDialog(null, "Cliente no encontrado en la base de datos");
        }

        txt_cliente_buscar.setText("");
        cn.close();
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error al buscar cliente: " + e);
    }
}

```

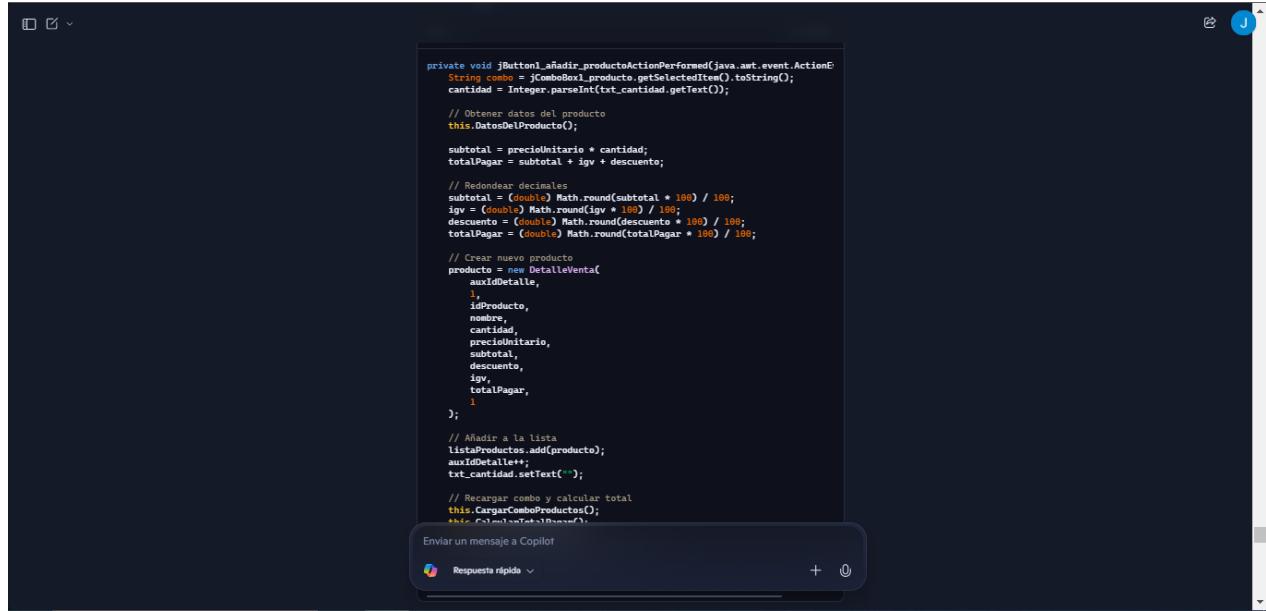
Enviar un mensaje a Copilot

Respuesta rápida ▾

Características de esta versión:

Con validacion del cliente

Validacion 02 : cantidad de producto y añadir producto



```
private void jButton1_agregar_productoActionPerformed(java.awt.event.ActionEvent evt) {
    String combo = JComboBox1_producto.getSelectedItem().toString();
    cantidad = Integer.parseInt(txt_cantidad.getText());

    // Obtener datos del producto
    this.DatosDelProducto();

    subtotal = precioUnitario * cantidad;
    totalPagar = subtotal + igt + descuento;

    // Redondear decimales
    subtotal = (double) Math.round(subtotal * 100) / 100;
    igt = (double) Math.round(igt * 100) / 100;
    descuento = (double) Math.round(descuento * 100) / 100;
    totalPagar = (double) Math.round(totalPagar * 100) / 100;

    // Crear nuevo producto
    producto = new DetalleVenta(
        auxIdDetalle,
        1,
        idProducto,
        nombre,
        cantidad,
        precioUnitario,
        subtotal,
        descuento,
        igt,
        totalPagar,
        1
    );
    // Añadir a la lista
    listaProductos.add(producto);
    auxIdDetalle++;
    txt_cantidad.setText("");
    // Recargar combo y calcular total
    this.CargaComboProductos();
    this.CalcularTotal();
}
```

Enviar un mensaje a Copilot

Respueta rápida ▾

Sin validacion

```
.añadir_productoActionPerformed(java.awt.event.ActionEvent evt) {
    _producto
    ls.jComboBox1_producto.getSelectedItem().toString();

    if (txt_cantidad.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Seleccione un producto");

        cantidad
        if (txt_cantidad.getText().matches("\\D+")) {
            JOptionPane.showMessageDialog(null, "Por favor ingrese solo números");
            return;
        }
        cantidad = Integer.parseInt(txt_cantidad.getText());
        if (cantidad <= 0) {
            JOptionPane.showMessageDialog(null, "La cantidad debe ser mayor que cero");
            return;
        }
        subtotal = precioUnitario * cantidad;
        totalPagar = subtotal + igv + descuento;
        // Redondear decimal
        subtotal = (double) Math.round(subtotal * 100) / 100;
        totalPagar = (double) Math.round(totalPagar * 100) / 100;
    }
}
```

Enviar un mensaje a Copilot



Respuesta rápida ▾

+



```
subtotal = precioUnitario * cantidad;
totalPagar = subtotal + igv + descuento;
// Redondear decimal
subtotal = (double) Math.round(subtotal * 100) / 100;
igv = (double) Math.round(igv * 100) / 100;
descuento = (double) Math.round(descuento * 100) / 100;
totalPagar = (double) Math.round(totalPagar * 100) / 100;
// Se crea un nuevo producto
producto = new DetalleVenta(
    auxIdDetalle,
    1,
    idProducto,
    nombre,
    cantidad,
    precioUnitario,
    subtotal,
    descuento,
    igv,
    totalPagar,
    1
);
// Añadir a la lista
ListaProductos.add(producto);
JOptionPane.showMessageDialog(null, "producto agregado");
auxIdDetalle++;
txt_cantidad.setText("");
```



Enviar un mensaje a Copilot



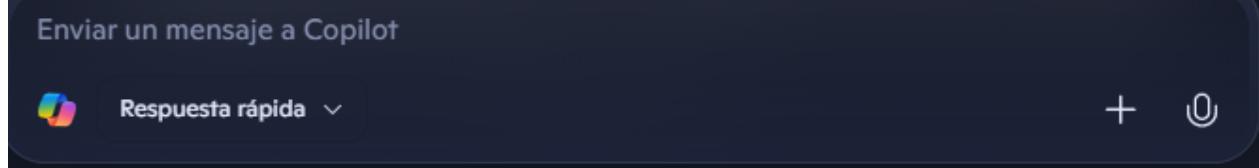
Respuesta rápida ▾

+ 0

```
listaProductos.add(producto);
JOptionPane.showMessageDialog(null, "producto agregado");
auxIdDetalle++;
txt_cantidad.setText("");
// Volver a cargar combo productos
this.CargarComboProductos();
this.CalcularTotalPagar();
txt_efectivo.setEnabled(true);
jButton_calcular_cambio.setEnabled(true);
else {
    JOptionPane.showMessageDialog(null, "cantidad supera al stock");
}
{
    JOptionPane.showConfirmDialog(null, "en la cantidad no PUEDE SER CERO NI NEGA
    ionPane.showConfirmDialog(null, "en la cantidad no admite caracteres");

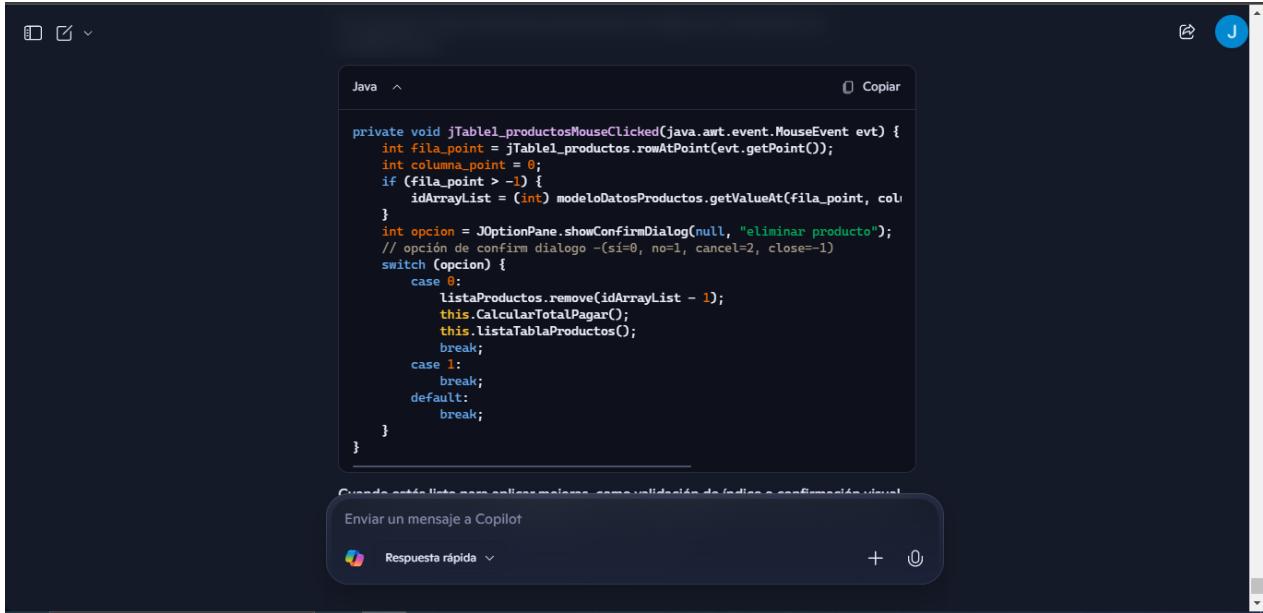
    ie.showConfirmDialog(null, "ingresa la cantidad de productos");

    iductos();
```



Con validacion de la cantidad y añadir producto

Validacion 03 : eliminacion de producto o compra



```

Java ^ Copiar

private void jTable1_productosMouseClicked(java.awt.event.MouseEvent evt) {
    int fila_point = jTable1_productos.rowAtPoint(evt.getPoint());
    int columna_point = 0;
    if (fila_point > -1) {
        idArrayList = (int) modeloDatosProductos.getValueAt(fila_point, columna_point);
    }
    int opcion = JOptionPane.showConfirmDialog(null, "eliminar producto");
    // opción de confirm dialogo -(si=0, no=1, cancel=2, close=-1)
    switch (opcion) {
        case 0:
            listaProductos.remove(idArrayList - 1);
            this.CalcularTotalPagar();
            this.listaTablaProductos();
            break;
        case 1:
            break;
        default:
            break;
    }
}

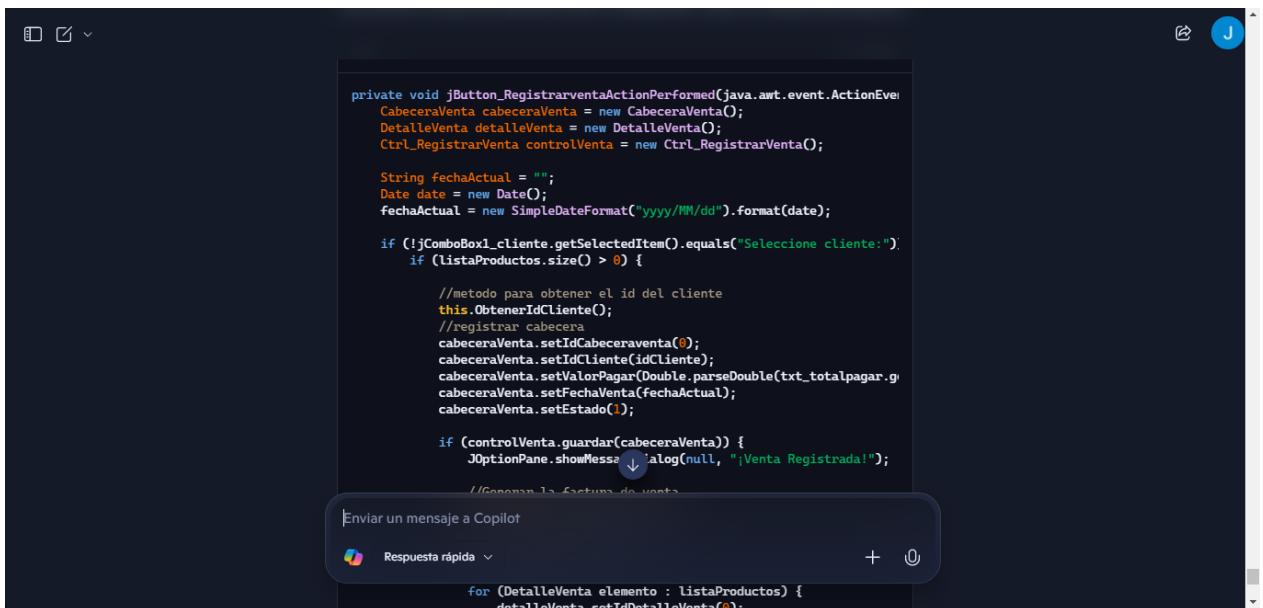
```

Cuando se da click en el icono de la fila de la tabla de productos

Enviar un mensaje a Copilot

Resposta rápida + 0

Validacion 04 : imprimir en pdf la factura



```

private void jButton_RegistrarventaActionPerformed(java.awt.event.ActionEvent evt) {
    CabeceraVenta cabeceraVenta = new CabeceraVenta();
    DetalleVenta detalleVenta = new DetalleVenta();
    Ctrl_RegistrarVenta controlVenta = new Ctrl_RegistrarVenta();

    String fechaActual = "";
    Date date = new Date();
    fechaActual = new SimpleDateFormat("yyyy/MM/dd").format(date);

    if (!comboBox1_cliente.getSelectedItem().equals("Seleccione cliente"))
        if (listaProductos.size() > 0) {
            //metodo para obtener el id del cliente
            this.ObtenerIdCliente();
            //registrar cabecera
            cabeceraVenta.setIdCabeceraVenta(0);
            cabeceraVenta.setIdCliente(idCliente);
            cabeceraVenta.setValorPagar(Double.parseDouble(txt_totalpagar.getText()));
            cabeceraVenta.setFechaVenta(fechaActual);
            cabeceraVenta.setEstado(1);

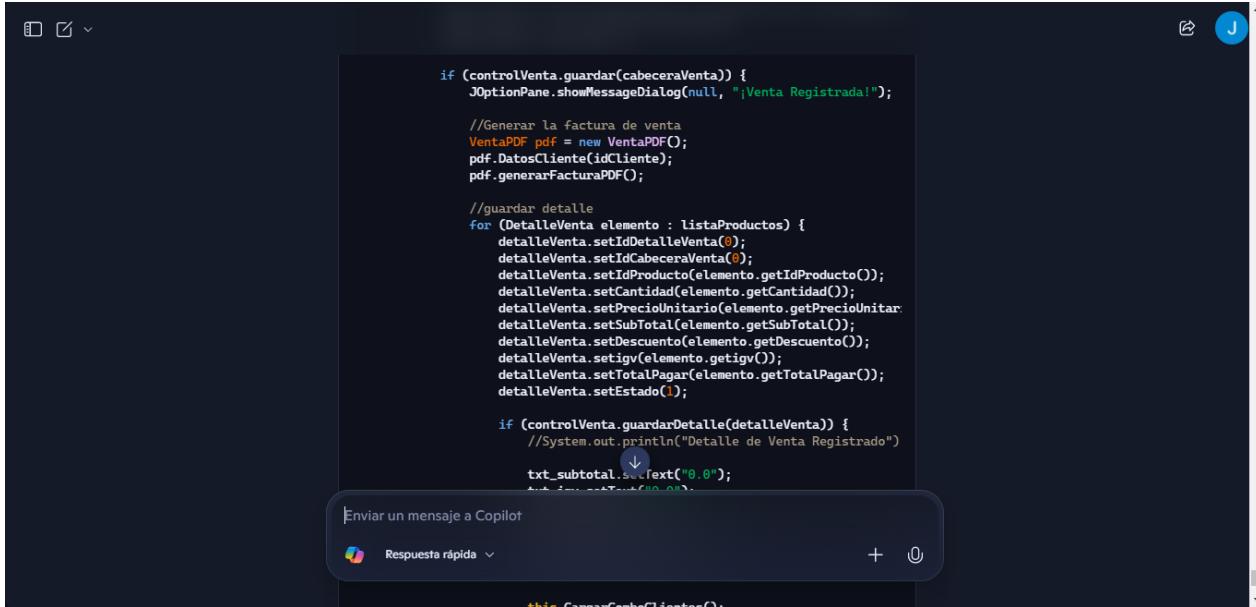
            if (controlVenta.guardar(cabeceraVenta)) {
                JOptionPane.showMessageDialog(null, "|Venta Registrada|");
                //Generar la factura de venta
            }
        }
}

for (DetalleVenta elemento : listaProductos) {
    detalleVenta.setIDDetalleVenta(0);
}

```

Enviar un mensaje a Copilot

Resposta rápida + 0



```

if (controlVenta.guardar(cabeceraVenta)) {
    JOptionPane.showMessageDialog(null, "|Venta Registrada!");

    //Generar la factura de venta
    VentaPDF pdf = new VentaPDF();
    pdf.DatosCliente(idCliente);
    pdf.generarFacturaPDF();

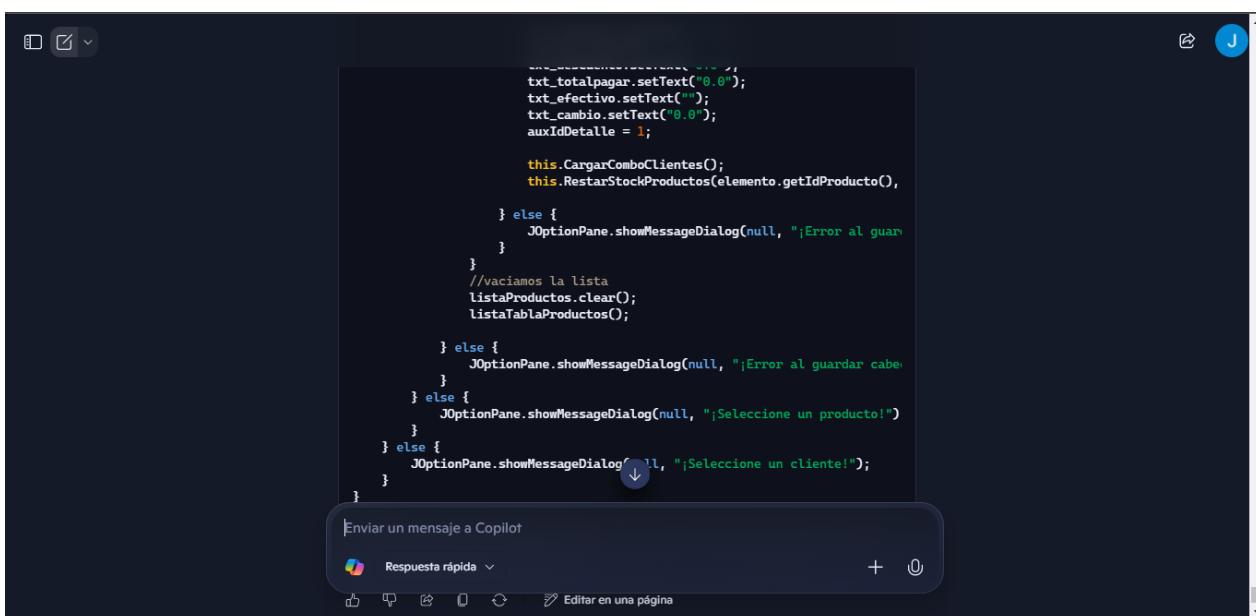
    //guardar detalle
    for (DetalleVenta elemento : listaProductos) {
        detalleVenta.setIdDetalleVenta(0);
        detalleVenta.setIdCabeceraVenta(0);
        detalleVenta.setIdProducto(elemento.getIdProducto());
        detalleVenta.setCantidad(elemento.getCantidad());
        detalleVenta.setPrecioUnitario(elemento.getPrecioUnitario());
        detalleVenta.setSubTotal(elemento.getSubTotal());
        detalleVenta.setDescuento(elemento.getDescuento());
        detalleVenta.setIgv(elemento.getIgv());
        detalleVenta.setTotalPagar(elemento.getTotalPagar());
        detalleVenta.setEstado(1);

        if (controlVenta.guardarDetalle(detalleVenta)) {
            //System.out.println("Detalle de Venta Registrado")
            txt_subtotal.setText("0.0");
        }
    }
}

```

Enviar un mensaje a Copilot

Resuesta rápida



```

txt_totalpagar.setText("0.0");
txt_efectivo.setText("");
txt_cambio.setText("0.0");
auxIdDetalle = 1;

this.CargarComboClientes();
this.RestarStockProductos(elemento.getIdProducto(),

} else {
    JOptionPane.showMessageDialog(null, "|Error al guardar cabecera");
}
//vaciamos la lista
listaProductos.clear();
listaTablaProductos();

} else {
    JOptionPane.showMessageDialog(null, "|Error al guardar cabecera");
} else {
    JOptionPane.showMessageDialog(null, "|Seleccione un producto!");
} else {
    JOptionPane.showMessageDialog(null, "|Seleccione un cliente!");
}
}

```

Enviar un mensaje a Copilot

Resuesta rápida

4. Detalles sobre la implementación del manejo de excepciones

El manejo de excepciones es fundamental para garantizar la estabilidad del sistema y evitar comportamientos inesperados. En esta fase del proyecto, se implementarán validaciones que respondan a los siguientes casos:

Casos contemplados:

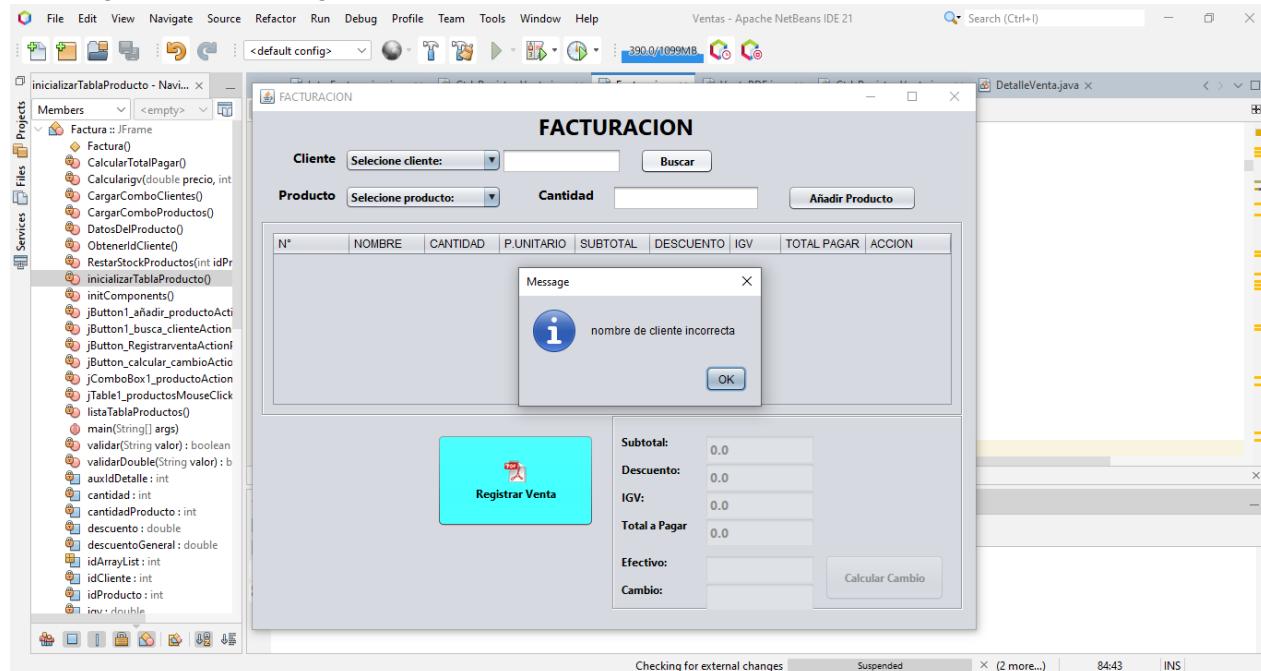
- Campos vacíos o nulos.
- Letras en campos numéricos (cantidad, efectivo).
- Cantidad negativa o cero.
- Pago insuficiente.
- Cliente no registrado.

Técnicas utilizadas:

- Condicionales (`if`, `try-catch`) para interceptar errores.
- Mensajes personalizados para el usuario.
- Bloqueo de acciones hasta corregir el error.

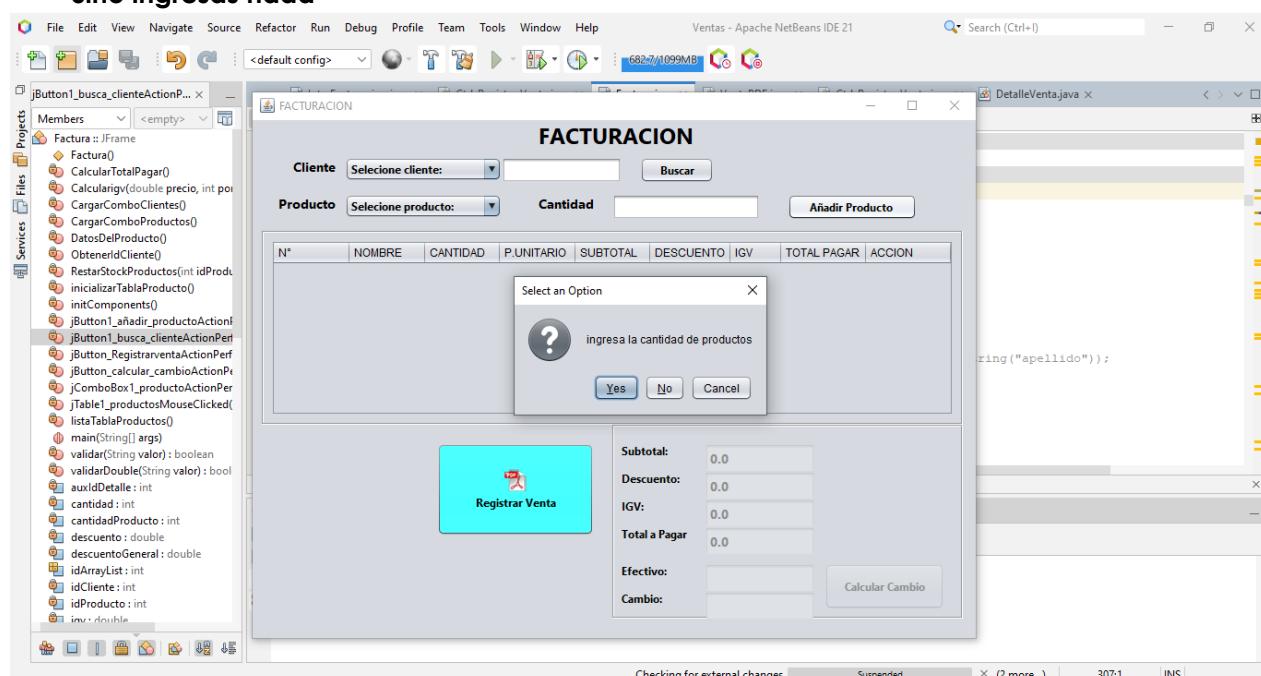
VERIFICACION DE EXCEPCIONES

Si no ingresas nada o ingresas un nombre que no está en la base de datos te saldrá error.

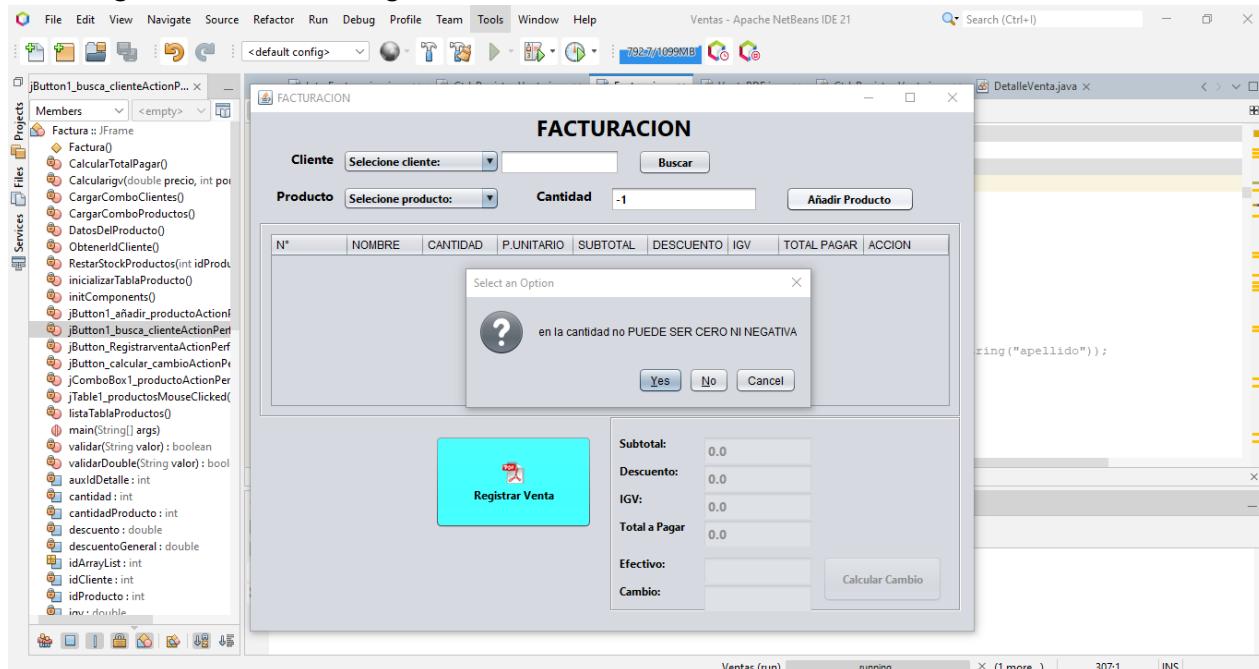


La opción CANTIDAD se encarga de solicitarte un numero positivo mayor a cero y entero para saber que cantidad de productos estas comprando.

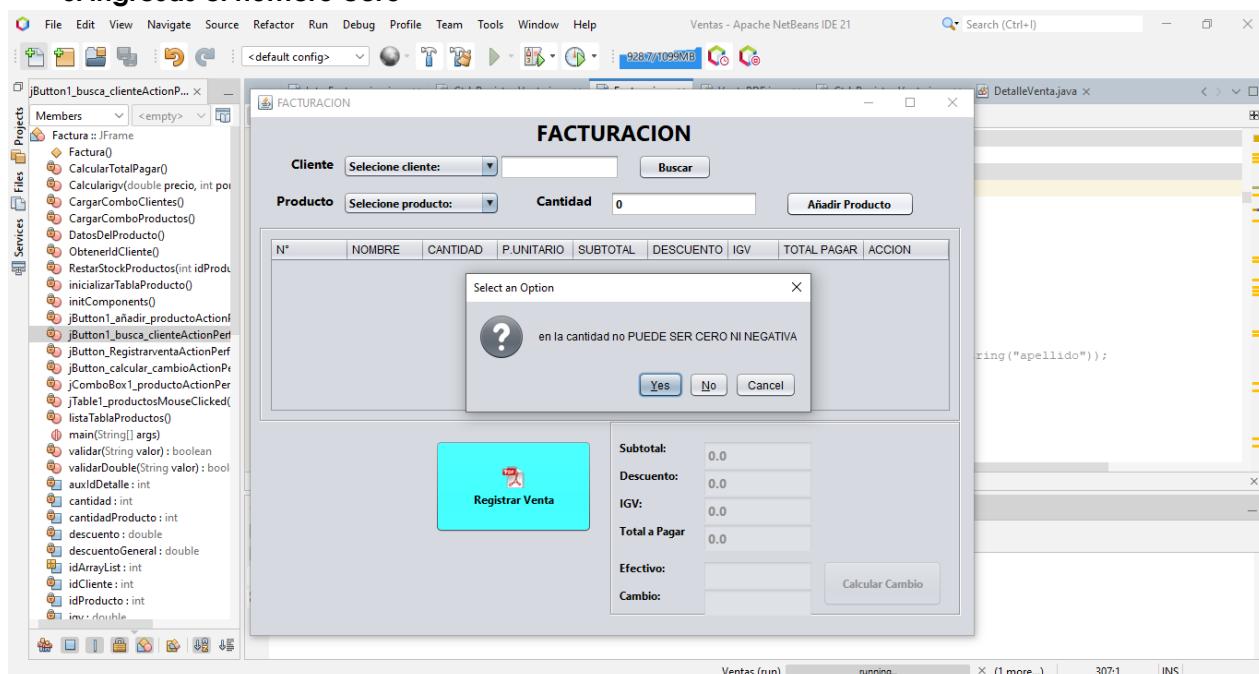
Sino ingresas nada



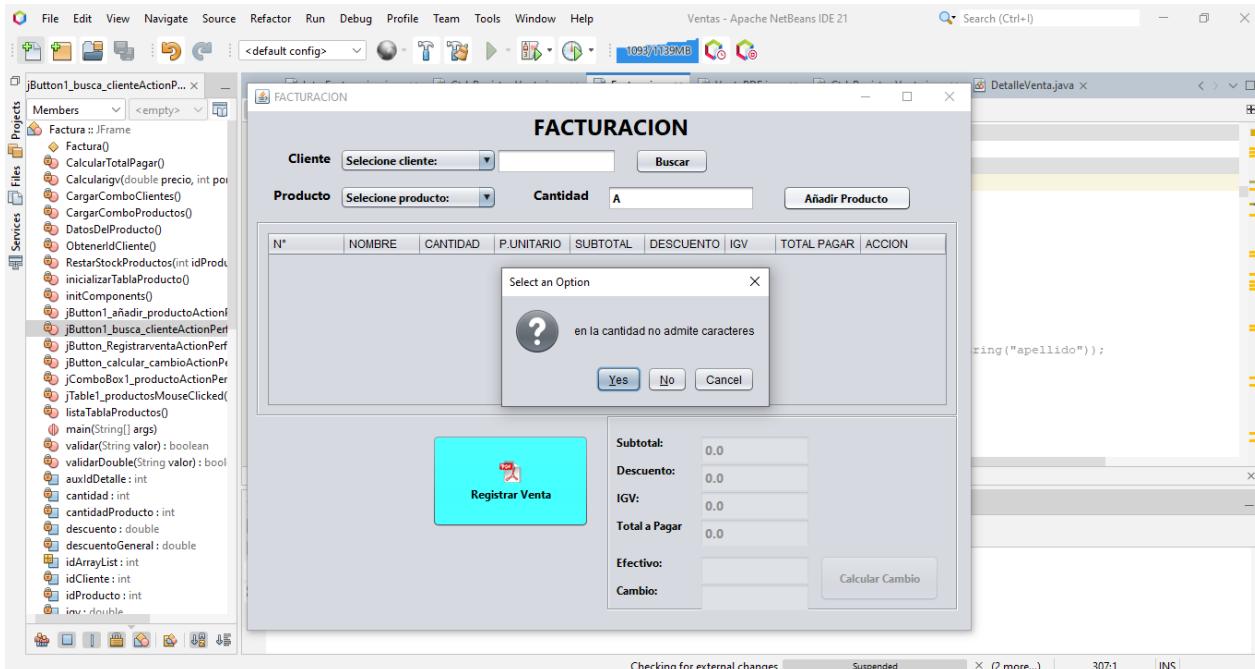
Si ingresas un numero negativo



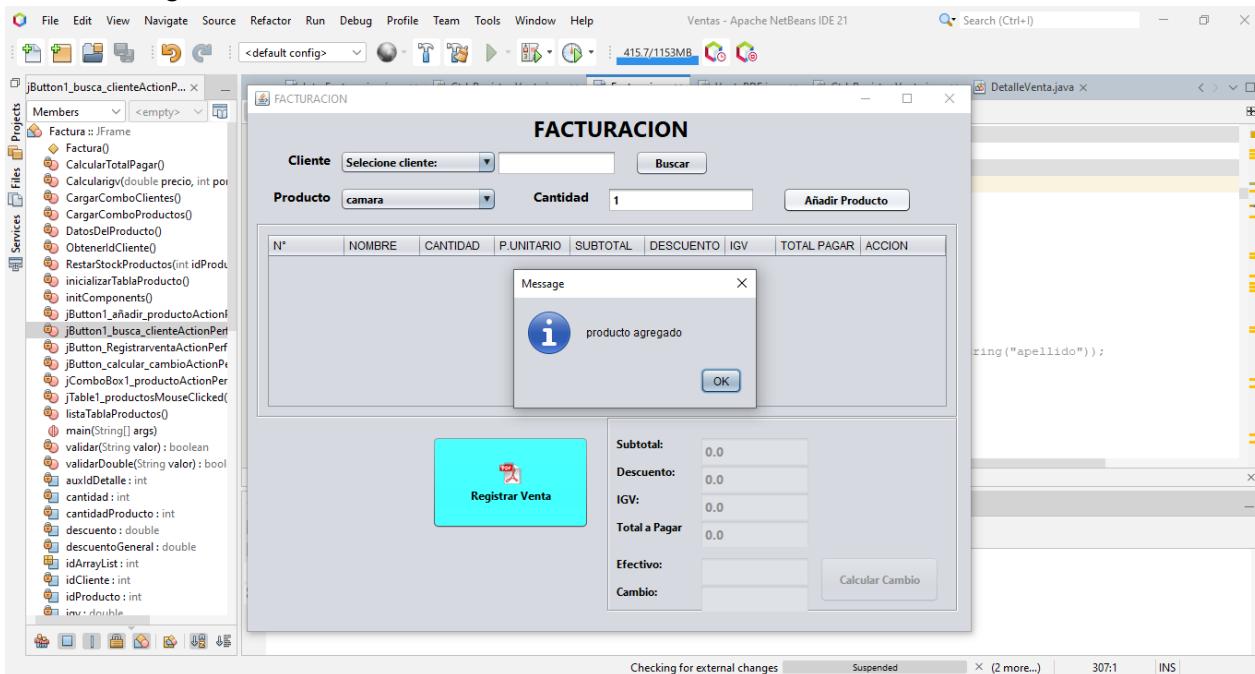
Si ingresas el numero cero



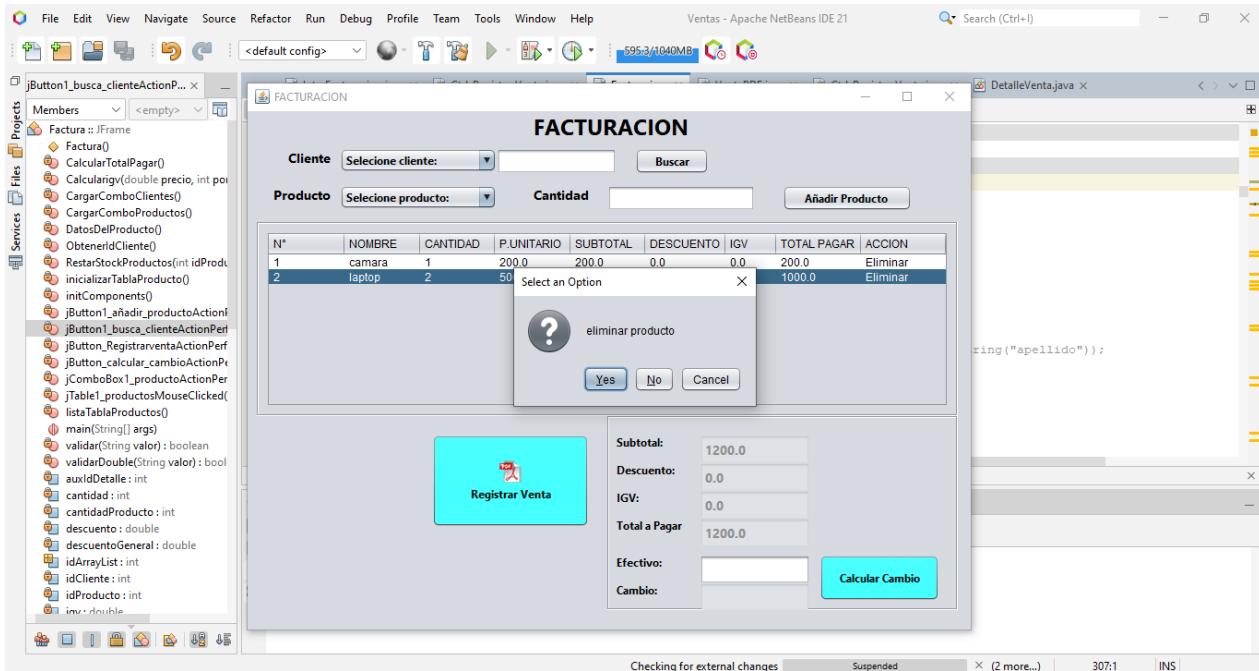
Si ingresas una letra



Seleccionamos el producto CAMARA y ingresamos la cantidad 1, si son opciones validas te sale el siguiente aviso.

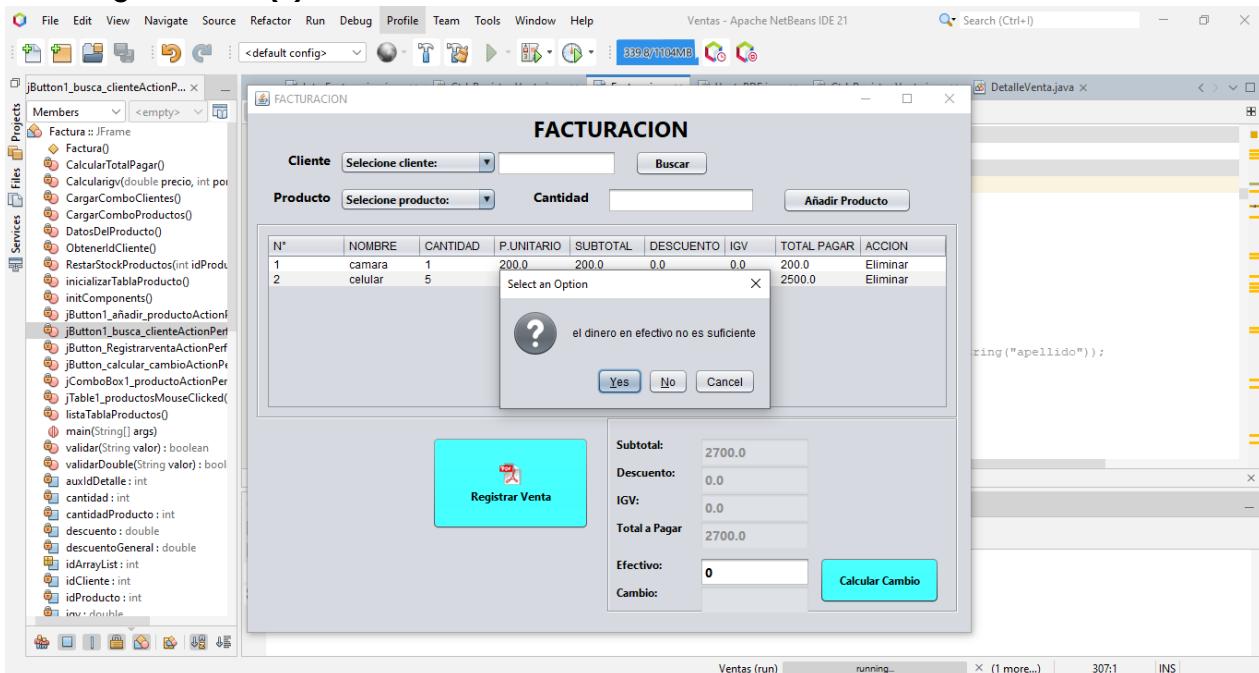


en la tabla en la columna ACCION se presenta una opción ELIMINAR, la cual cuando seleccionas dicha opción toda la fila del producto. Como se puede ver te consulta si estas seguro de eliminar dicho registro.

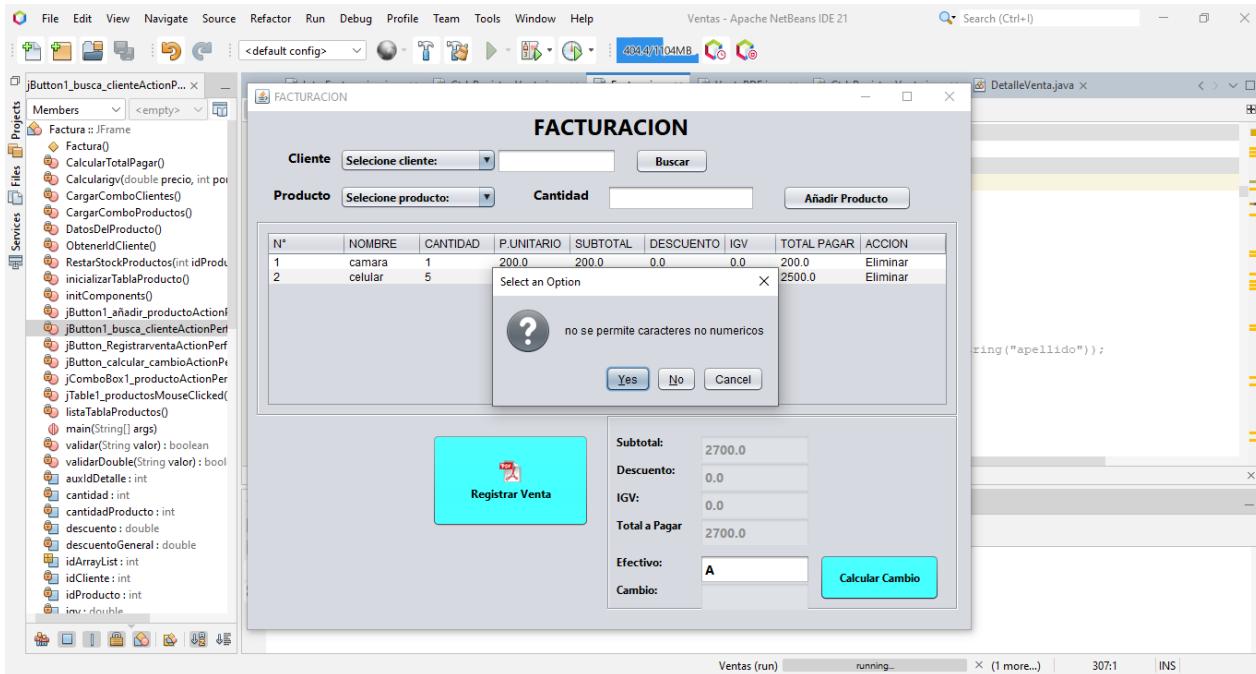


la opción CALCULAR CAMBIO, calcula la diferencia entre el TOTAL A PAGAR y EFECTIVO.

Si registras cero (0).



Si registras una letra



5. Análisis de código limpio, estilo y refactorización

El concepto de **código limpio** implica que el software sea legible, mantenible y funcional. En este proyecto se aplican los siguientes principios:

Buenas prácticas aplicadas:

- **Nombres descriptivos** para variables y métodos (`CalcularTotalPagar`, `DatosDelProducto`).
- **Separación de responsabilidades**: cada clase cumple una función específica (modelo, controlador, vista).
- **Modularidad**: funciones reutilizables y bien delimitadas.
- **Eliminación de redundancias**: refactorización de bloques repetitivos.

CODIGO REALIZADO

```
private void jButton1_busca_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    String clienteBuscar = txt_cliente_buscar.getText().trim();
    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();
    String sql = "select nombre from clientes where nombre ='" + clienteBuscar + "'";
    Statement st;
    try {
        st = cn.createStatement();
        ResultSet rs = st.executeQuery(sql);
        if (rs.next()) {
            txt_nombre.setText(rs.getString("nombre"));
            txt_apellido.setText(rs.getString("apellido"));
            txt_direccion.setText(rs.getString("direccion"));
            txt_ciudad.setText(rs.getString("ciudad"));
            txt_pais.setText(rs.getString("pais"));
            txt_telefono.setText(rs.getString("telefono"));
            txt_email.setText(rs.getString("email"));
        } else {
            JOptionPane.showMessageDialog(null, "No se encontró ningún cliente con ese nombre.");
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Ocurrió un error al conectar con la base de datos.");
    }
}
```

```
st = cn.createStatement();

ResultSet rs = st.executeQuery(sql);

if (rs.next()) {

    jComboBox1_cliente.setSelectedItem(rs.getString("nombre") + " " + rs.getString("apellido"));

} else {

    jComboBox1_cliente.setSelectedItem("seleccione cliente:");

    JOptionPane.showMessageDialog(null, "nombre de cliente incorrecta");

}

txt_cliente_buscar.setText(" ");

cn.close();

} catch (SQLException e) {

    System.out.println("error al buscar cliente" + e);

}

}
```

CODIGO LIMPIO

```
private void jButton1_busca_clienteActionPerformed(java.awt.event.ActionEvent evt) {
    // Obtener y limpiar el texto ingresado por el usuario
    String clienteBuscar = txt_cliente_buscar.getText().trim();

    // Validación básica: evitar consultas vacías
    if (clienteBuscar.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Ingrese un nombre de cliente.");
        return; // Se detiene el flujo si no hay texto
    }

    // Conexión a la base de datos usando try-with-resources para cerrar
    // automáticamente
    try (Connection cn = new Conexion().getConnection()) {

        // Consulta segura con PreparedStatement para evitar inyección SQL
        String sql = "SELECT nombre FROM clientes WHERE nombre = ?";
        PreparedStatement pst = cn.prepareStatement(sql);
```

```

pst.setString(1, clienteBuscar); // Se inserta el valor de forma
segura

// Ejecutar la consulta
ResultSet rs = pst.executeQuery();

// Si se encuentra el cliente, se selecciona en el combo
if (rs.next()) {
    jComboBox1_cliente.setSelectedItem(rs.getString("nombre"));
} else {
    // Si no se encuentra, se muestra mensaje y se reinicia el combo
    jComboBox1_cliente.setSelectedItem("Seleccione cliente:");
    JOptionPane.showMessageDialog(null, "Nombre de cliente
incorrecto.");
}

// Limpiar el campo de búsqueda
txt_cliente_buscar.setText("");

} catch (SQLException e) {
    // Manejo de errores con mensaje visible al usuario
    JOptionPane.showMessageDialog(null, "Error al buscar cliente: " +
e.getMessage());
}
}
}

```

Aspecto	Mejora aplicada
Seguridad	Se reemplazó Statement por PreparedStatement para evitar inyección SQL.
Limpieza de código	Se eliminó el espacio innecesario en txt_cliente_buscar.setText(" ").
Validación de entrada	Se agregó control para evitar búsquedas vacías.
Manejo de recursos	Se usó try-with-resources para cerrar automáticamente la conexión.
Mensajes claros	Se mejoró el mensaje de error para el usuario y se eliminó el System.out.println.
Estilo y legibilidad	Se estructuró el código con sangrías claras y nombres consistentes.

- **Refactorización del módulo de búsqueda de cliente**

Versión original	Problema detectado	Mejora aplicada
Uso de Statement	Vulnerabilidad a inyección SQL	Reemplazo por PreparedStatement
Campo sin validación	Possible consulta vacía	Validación con isEmpty()
Conexión manual	Riesgo de no cerrar conexión	Uso de try-with-resources
Mensaje en consola	No visible para el usuario final	Reemplazo por JOptionPane

CODIGO REALIZADO

```
private void jButton1_añadir_productoActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // jButton1_añadir_producto  
  
    String combo=this.jComboBox1_producto.getSelectedItem().toString();  
  
    //validar el producto  
  
    if (combo.equalsIgnoreCase("Seleccione producto")){  
  
        JOptionPane.showMessageDialog(null, "seleccione un producto");  
  
    } else {  
  
        //valida la cantidad  
  
        if(!txt_cantidad.getText().isEmpty()){  
  
            // validar que el usuario ingrese caracteres numericos  
  
            boolean validacion = validar(txt_cantidad.getText());  
  
            if (validacion == true){  
  
                // validar que la cantidad sea mayor a cero  
  
                if (Integer.parseInt(txt_cantidad.getText())>0){  
  
                    cantidad = Integer.parseInt(txt_cantidad.getText());  
  
                    //ejecutar metodo datos del producto  
  
                    this.DatosDelProducto();  
  
                    //validar la cantidad de producto seleccionado no sea mayor al stok de la base de datos  
  
                    if(cantidad <= cantidadProducto){  
  
                        subtotal=precioUnitario * cantidad;  
  
                        totalPagar=subtotal + igv +descuento;  
  
                        // redondear decimal  
  
                        subtotal=(double)Math.round(subtotal * 100)/100;  
  
                        igv=(double)Math.round(igv * 100)/100;  
                }  
            }  
        }  
    }  
}
```

```
descuento=(double)Math.round(descuento * 100)/100;

totalPagar=(double)Math.round(totalPagar * 100)/100;

// SE CREA UN NUEVO PRODUCTO

producto = new DetalleVenta(

    auxIdDetalle,
    1,
    idProducto,
    nombre,
    cantidad,
    precioUnitario,
    subtotal,
    descuento,
    igv,
    totalPagar,
    1
);

//añadir a la lista

listaProductos.add(producto);

 JOptionPane.showMessageDialog(null, "producto agregado");

auxIdDetalle++;

txt_cantidad.setText("");

// VOLVER A CARGAR COMBO PRODUCTOS

this.CargarComboProductos();

this.CalcularTotalPagar();

txt_efectivo.setEnabled(true);
```

```
    jButton_calcular_cambio.setEnabled(true);

} else{

    JOptionPane.showMessageDialog(null, "cantidad supera al stock");

}

} else {

    JOptionPane.showConfirmDialog(null, "en la cantidad no PUEDE SER CERO NI
NEGATIVA");

}

} else {

    JOptionPane.showConfirmDialog(null, "en la cantidad no admite caracteres");

}

} else {

    JOptionPane.showConfirmDialog(null, "ingresa la cantidad de productos");

}

}

this.listaTablaProductos();

}
```

CODIGO LIMPIO

```
private void
jButton1_añadir_productoActionPerformed(java.awt.event.ActionEvent evt) {
    // Obtener el producto seleccionado del combo
    String combo = jComboBox1_producto.getSelectedItem().toString();

    // Validación: evitar que se procese si no se ha seleccionado un producto
    if (combo.equalsIgnoreCase("Seleccione producto")) {
        JOptionPane.showMessageDialog(null, "Seleccione un producto");
        return; // Se detiene el flujo si no hay selección válida
    }

    // Validación: verificar que el campo de cantidad no esté vacío
    String cantidadTexto = txt_cantidad.getText().trim();
    if (cantidadTexto.isEmpty()) {
```

```
 JOptionPane.showConfirmDialog(null, "Ingrese la cantidad de
productos");
    return;
}

// Validación: verificar que el texto ingresado sea numérico
boolean validacion = validar(cantidadTexto); // Método personalizado
if (!validacion) {
    JOptionPane.showConfirmDialog(null, "La cantidad no admite
caracteres");
    return;
}

// Validación: verificar que la cantidad sea mayor a cero
cantidad = Integer.parseInt(cantidadTexto);
if (cantidad <= 0) {
    JOptionPane.showConfirmDialog(null, "La cantidad no puede ser cero ni
negativa");
    return;
}

// Obtener datos del producto desde la base de datos o estructura interna
this.DatosDelProducto();

// Validación: verificar que la cantidad no supere el stock disponible
if (cantidad > cantidadProducto) {
    JOptionPane.showMessageDialog(null, "La cantidad supera el stock
disponible");
    return;
}

//Cálculo de montos
subtotal = precioUnitario * cantidad;
totalPagar = subtotal + igv + descuento;

//Redondeo de valores a dos decimales
subtotal = Math.round(subtotal * 100) / 100.0;
igv = Math.round(igv * 100) / 100.0;
descuento = Math.round(descuento * 100) / 100.0;
totalPagar = Math.round(totalPagar * 100) / 100.0;

//Creación del objeto DetalleVenta con todos los datos
producto = new DetalleVenta(
    auxIdDetalle,
    1,
    idProducto,
    nombre,
    cantidad,
    precioUnitario,
    subtotal,
    descuento,
    igv,
    totalPagar,
    1
);

//Añadir el producto a la lista de ventas
listaProductos.add(producto);
JOptionPane.showMessageDialog(null, "Producto agregado");

// Actualización de estado y limpieza de campos
```

```

auxIdDetalle++;
txt_cantidad.setText("");
this.CargarComboProductos();
this.CalcularTotalPagar();
txt_efectivo.setEnabled(true);
jButton_calcular_cambio.setEnabled(true);

// Actualizar la tabla visual de productos
this.listaTablaProductos();
}

```

Aspecto	Mejora aplicada
Limpieza visual	Se eliminaron sangrías innecesarias y se organizó el código por bloques funcionales
Estilo uniforme	Se usaron nombres consistentes (cantidadTexto) y se evitó duplicación de lógica
Refactorización lógica	Se aplicaron return para evitar anidaciones profundas y mejorar la lectura
Comentarios técnicos	Se añadieron comentarios claros para cada validación y proceso
Redondeo profesional	Se usó Math.round(...)/100.0 para mayor precisión y estilo

- **Refactorización del módulo de búsqueda de cliente**

Versión original	Problema detectado	Mejora aplicada
Uso de Statement	Vulnerabilidad a inyección SQL	Reemplazo por PreparedStatement
Campo sin validación	Possible ejecución de consulta vacía	Validación con isEmpty()
Conexión manual	Riesgo de no cerrar conexión correctamente	Uso de try-with-resources
Mensaje en consola	No visible para el usuario final	Reemplazo por JOptionPane
Espacios innecesarios	txt_cliente_buscar.setText(" ") genera confusión	Se reemplaza por setText("")
Código anidado	Dificulta la lectura y mantenimiento	Uso de return para simplificar estructura

CODIGO REALIZADO

```

private void jTable1_productosMouseClicked(java.awt.event.MouseEvent evt) {

    int fila_point=jTable1_productos.rowAtPoint(evt.getPoint());
    int columna_point=0;
    if(fila_point>-1){

```

```
idArrayList = (int) modeloDatosProductos.getValueAt(fila_point, columna_point);

}

int opcion=JOptionPane.showConfirmDialog(null, "eliminar producto");

// opcion de confirm dialogo -(si=0, no=1,cancel =2, close=-1)

switch (opcion){

    case 0:

        listaProductos.remove(idArrayList - 1);

        this.CalcularTotalPagar();

        this.listaTablaProductos();

        break;

    case 1:

        break;

    default:

        break;

    }

}
```

CODIGO LIMPIO

```
private void jTable1_productosMouseClicked(java.awt.event.MouseEvent evt) {
    // Obtener la fila seleccionada en la tabla
    int filaSeleccionada = jTable1_productos.rowAtPoint(evt.getPoint());
    int columnaid = 0;

    // Validar que se haya hecho clic sobre una fila válida
    if (filaSeleccionada > -1) {
        // Obtener el ID del producto desde la primera columna
        idArrayList = (int) modeloDatosProductos.getValueAt(filaSeleccionada,
columnaid);

        // Mostrar cuadro de confirmación para eliminar
        int opcion = JOptionPane.showConfirmDialog(null, "¿Eliminar
producto?");

        // Procesar la respuesta del usuario
        switch (opcion) {
```

```
        case 0: // Sí
            listaProductos.remove(idArrayList - 1); // Elimina el producto
            de la lista
            this.CalcularTotalPagar(); // Recalcula el total
            this.listaTablaProductos(); // Actualiza la tabla
            visual
            break;

        case 1: // No
            // No se realiza ninguna acción
            break;

        default: // Cancelar o cerrar
            // No se realiza ninguna acción
            break;
    }
}
```

Aspecto	Mejora aplicada
Nombres descriptivos	filasSeleccionada, columnasID mejoran la comprensión del código
Comentarios claros	Se explica cada bloque funcional para facilitar mantenimiento
Separación lógica	Se agrupan operaciones por propósito: selección, confirmación, acción
Estilo uniforme	Sangrías consistentes y estructura visual clara

- Refactorización del módulo de eliminación de productos desde la tabla

Versión original	Problema detectado	Mejora aplicada
Nombres genéricos	Variables como fila_point y columna_point poco descriptivas	Renombradas como filaSeleccionada y columnalID para mayor claridad
Código sin estructura	Lógica agrupada sin separación funcional	División por bloques: selección, confirmación, acción
Comentarios ausentes	Dificultad para entender el propósito de cada línea	Comentarios técnicos explicando cada paso
Estilo visual inconsistente	Sangrías y saltos poco definidos	Estilo uniforme y legible
Validación implícita	No se indica claramente qué se valida	Validación explícita de fila seleccionada

CODIGO REALIZADO

```
private void jButton_RegistrarventaActionPerformed(java.awt.event.ActionEvent evt) {  
  
    CabeceraVenta cabeceraVenta = new CabeceraVenta();  
  
    DetalleVenta detalleVenta = new DetalleVenta();  
  
    Ctrl_RegistrarVenta controlVenta = new Ctrl_RegistrarVenta();
```

```
String fechaActual = "";  
  
Date date = new Date();  
  
fechaActual = new SimpleDateFormat("yyyy/MM/dd").format(date);  
  
  
  
if (!jComboBox1_cliente.getSelectedItem().equals("Seleccione cliente:")) {  
  
    if (listaProductos.size() > 0) {  
  
        //metodo para obtener el id del cliente  
  
        this.ObtenerIdCliente();  
  
        //registrar cabecera  
  
        cabeceraVenta.setIdCabeceraventa(0);  
  
        cabeceraVenta.setIdCliente(idCliente);  
  
        cabeceraVenta.setValorPagar(Double.parseDouble(txt_totalpagar.getText()));  
  
        cabeceraVenta.setFechaVenta(fechaActual);  
  
        cabeceraVenta.setEstado(1);  
  
  
  
        if (controlVenta.guardar(cabeceraVenta)) {  
  
            JOptionPane.showMessageDialog(null, "¡Venta Registrada!");  
  
  
  
            //Generar la factura de venta  
  
            VentaPDF pdf = new VentaPDF();  
  
            pdf.DatosCliente(idCliente);  
  
            pdf.generarFacturaPDF();  
  
  
  
            //guardar detalle
```

```
for (DetalleVenta elemento : listaProductos) {  
  
    detalleVenta.setIdDetalleVenta(0);  
  
    detalleVenta.setIdCabeceraVenta(0);  
  
    detalleVenta.setIdProducto(elemento.getIdProducto());  
  
    detalleVenta.setCantidad(elemento.getCantidad());  
  
    detalleVenta.setPrecioUnitario(elemento.getPrecioUnitario());  
  
    detalleVenta.setSubTotal(elemento.getSubTotal());  
  
    detalleVenta.setDescuento(elemento.getDescuento());  
  
    detalleVenta.setIgv(elemento.getIgv());  
  
    detalleVenta.setTotalPagar(elemento.getTotalPagar());  
  
    detalleVenta.setEstado(1);  
  
  
if (controlVenta.guardarDetalle(detalleVenta)) {  
  
    //System.out.println("Detalle de Venta Registrado");  
  
  
    txt_subtotal.setText("0.0");  
  
    txt_igv.setText("0.0");  
  
    txt_descuento.setText("0.0");  
  
    txt_totalpagar.setText("0.0");  
  
    txt_efectivo.setText("");  
  
    txt_cambio.setText("0.0");  
  
    auxIdDetalle = 1;  
  
  
this.CargarComboClientes();  
  
this.RestarStockProductos(elemento.getIdProducto(), elemento.getCantidad());
```

```
    } else {  
  
        JOptionPane.showMessageDialog(null, "¡Error al guardar detalle de venta!");  
  
    }  
  
}  
  
//vaciamos la lista  
  
listaProductos.clear();  
  
listaTablaProductos();  
  
}  
  
} else {  
  
    JOptionPane.showMessageDialog(null, "¡Error al guardar cabecera de venta!");  
  
}  
  
} else {  
  
    JOptionPane.showMessageDialog(null, "¡Seleccione un producto!");  
  
}  
  
} else {  
  
    JOptionPane.showMessageDialog(null, "¡Seleccione un cliente!");  
  
}  
  
}
```

CODIGO LIMPIO

```
private void jButton_RegistrarventaActionPerformed(java.awt.event.ActionEvent evt) {  
    // Instanciar objetos necesarios  
    CabeceraVenta cabeceraVenta = new CabeceraVenta();  
    DetalleVenta detalleVenta = new DetalleVenta();  
    Ctrl_RegistrarVenta controlVenta = new Ctrl_RegistrarVenta();  
  
    // Obtener fecha actual en formato yyyy/MM/dd
```

```
String fechaActual = new SimpleDateFormat("yyyy/MM/dd").format(new Date());  
  
    // Validar selección de cliente  
    if (jComboBox1_cliente.getSelectedItem().equals("Seleccione cliente:")) {  
        JOptionPane.showMessageDialog(null, ";Seleccione un cliente!");  
        return;  
    }  
  
    // Validar que haya productos en la lista  
    if (listaProductos.isEmpty()) {  
        JOptionPane.showMessageDialog(null, ";Seleccione un producto!");  
        return;  
    }  
  
    // Obtener ID del cliente  
    this.ObtenerIdCliente();  
  
    // Configurar cabecera de venta  
    cabeceraVenta.setIdCabeceraVenta(0);  
    cabeceraVenta.setIdCliente(idCliente);  
    cabeceraVenta.setValorPagar(Double.parseDouble(txt_totalpagar.getText()));  
    cabeceraVenta.setFechaVenta(fechaActual);  
    cabeceraVenta.setEstado(1);  
  
    // Registrar cabecera en la base de datos  
    if (!controlVenta.guardar(cabeceraVenta)) {  
        JOptionPane.showMessageDialog(null, ";Error al guardar cabecera de venta!");  
        return;  
    }  
  
    JOptionPane.showMessageDialog(null, ";Venta Registrada!");  
  
    // Generar factura PDF  
    VentaPDF pdf = new VentaPDF();  
    pdf.DatosCliente(idCliente);  
    pdf.generarFacturaPDF();  
  
    // Registrar cada detalle de producto  
    for (DetalleVenta elemento : listaProductos) {  
        detalleVenta.setIdDetalleVenta(0);  
        detalleVenta.setIdCabeceraVenta(0); // Se puede actualizar si se obtiene el ID real  
        detalleVenta.setIdProducto(elemento.getIdProducto());  
        detalleVenta.setCantidad(elemento.getCantidad());  
        detalleVenta.setPrecioUnitario(elemento.getPrecioUnitario());  
        detalleVenta.setSubTotal(elemento.getSubTotal());  
        detalleVenta.setDescuento(elemento.getDescuento());  
        detalleVenta.setIgv(elemento.getIgv());  
        detalleVenta.setTotalPagar(elemento.getTotalPagar());  
        detalleVenta.setEstado(1);  
  
        // Guardar detalle en la base de datos  
        if (!controlVenta.guardarDetalle(detalleVenta)) {  
            JOptionPane.showMessageDialog(null, ";Error al guardar detalle de venta!");  
            continue;  
        }  
  
        // Actualizar interfaz y stock
```

```

txt_subtotal.setText("0.0");
txt_igv.setText("0.0");
txt_descuento.setText("0.0");
txt_totalpagar.setText("0.0");
txt_efectivo.setText("");
txt_cambio.setText("0.0");
auxIdDetalle = 1;

this.CargarComboClientes();
this.RestarStockProductos(elemento.getIdProducto(),
elemento.getCantidad());
}

// Limpiar lista y actualizar tabla
listaProductos.clear();
listaTablaProductos();
}
}

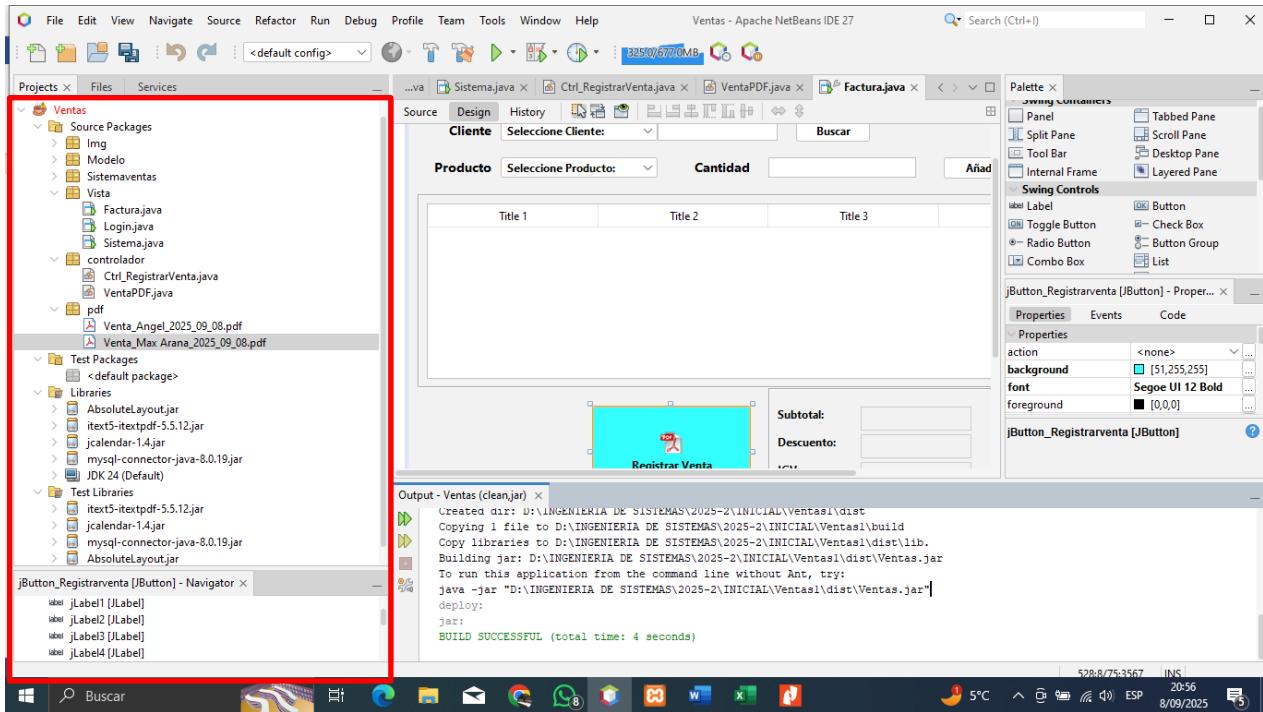
```

Aspecto	Mejora aplicada
Legibilidad	Se agrupan bloques funcionales con comentarios claros
Validaciones explícitas	Se usan return para evitar anidaciones profundas
Nombres consistentes	Variables y métodos con nombres descriptivos
Eliminación de redundancia	Se evita repetir validaciones y estructuras innecesarias
Separación lógica	Flujo dividido en: validación, cabecera, detalle, interfaz

- **Refactorización del módulo de registro de venta**

Versión original	Problema detectado	Mejora aplicada
Código anidado	Dificultad para leer y mantener	Uso de return para simplificar estructura
Comentarios ausentes	Poco claro el propósito de cada bloque	Comentarios técnicos por sección funcional
Repetición de lógica	Validaciones duplicadas	Consolidación de condiciones
Estilo visual inconsistente	Sangrías y saltos poco definidos	Estilo uniforme y legible
Flujo poco modular	Todo en un solo bloque	Separación por validación, cabecera, detalle

ESTRUCTURA MODULAR DEL PROYECTO



6. CONCLUSIONES

- El desarrollo progresivo del sistema de facturación presentado en este trabajo evidencia una sólida capacidad técnica y organizativa para abordar problemas reales mediante soluciones digitales. A partir de una versión inicial con funcionalidades básicas —login y registro de ventas— se identificaron carencias clave que permitieron justificar la evolución del sistema bajo un enfoque ágil y académico.
- La incorporación de nuevas funcionalidades como la generación de factura en PDF y el manejo de validaciones y excepciones no solo fortaleció la robustez del sistema, sino que permitió aplicar principios de código limpio, modularidad y reutilización. Estas mejoras fueron planificadas y ejecutadas mediante una simulación metodológica basada en Scrum, donde se asumieron roles estratégicos que facilitaron la organización del trabajo y la entrega de resultados concretos por sprint.
- El uso de inteligencia artificial generativa como herramienta de apoyo técnico y documental aportó valor al proceso, permitiendo optimizar funciones, enriquecer la presentación institucional y simular escenarios de mejora. Esta integración demuestra una actitud proactiva frente a la innovación y una capacidad para adaptarse a nuevas tecnologías en el desarrollo de software.

CODIGO FUENTE DEL LA OPCION FACTURA (JAVA)

package Vista;

```
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.awt.Dimension;
import java.sql.Connection;
import Modelo.Conexion;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.ArrayList;
import Modelo.DetalleVenta;
import controlador.VentaPDF;
import java.text.SimpleDateFormat;
import java.util.Date;
import Modelo.CabeceraVenta;
import Modelo.DetalleVenta;
import controlador.Ctrl_RegistrarVenta;
import static java.awt.image.ImageObserver.WIDTH;
import java.sql.PreparedStatement;
import javax.swing.Icon;
import javax.swing.ImageIcon;
```

```
public class Factura extends javax.swing.JFrame {
// modelo de datos
    private DefaultTableModel modeloDatosProductos;
    // LISTA PARA EL DETALLE DE VENTA DE PRODUCTOS
    ArrayList<DetalleVenta> listaProductos = new ArrayList<>();
    private DetalleVenta producto;
    private int idCliente=0;
    private int idProducto=0;
    private String nombre="";
    private int cantidadProducto=0;
    private double precioUnitario =0.0;
    private int porcentajeIgv=0;
    private int cantidad =0;
    private double subtotal =0.0;
    private double descuento =0.0;
    private double igv =0.0;
    private double totalPagar =0.0;
    // variables para calculos globales
    private double subtotalGeneral = 0.0;
    private double descuentoGeneral = 0.0;
    private double igvGeneral = 0.0;
    private double totalPagarGeneral = 0.0;
    // fin de variables de calculos globales
```

```
private int auxIdDetalle=1;
//Conexion cx;

public Factura() {
    initComponents();
    this.setSize(new Dimension(800, 600));
    this.setTitle("FACTURACION");

    //cargar los clientes en la lista
    this.CargarComboClientes();
    this.CargarComboProductos();
    this.inicializarTablaProducto();
    txt_efectivo.setEnabled(false);
    jButton_calcular_cambio.setEnabled(false);
    txt_subtotal.setText("0.0");
    txt_igv.setText("0.0");
    txt_descuento.setText("0.0");
    txt_totalpagar.setText("0.0");
    //cx=new Conexion();
    // cx.conectar();

}

//metodo para inicializar los productos
private void inicializarTablaProducto(){
    modeloDatosProductos = new DefaultTableModel();
    modeloDatosProductos.addColumn("Nº");
    modeloDatosProductos.addColumn("NOMBRE");
    modeloDatosProductos.addColumn("CANTIDAD");
    modeloDatosProductos.addColumn("P.UNITARIO");
    modeloDatosProductos.addColumn("SUBTOTAL");
    modeloDatosProductos.addColumn("DESCUENTO");
    modeloDatosProductos.addColumn("IGV");
    modeloDatosProductos.addColumn("TOTAL PAGAR");
    modeloDatosProductos.addColumn("ACCION");
    // agregar dato tabla
    this.jTable1_productos.setModel(modeloDatosProductos);

}

//metodo para presentar informacion de la tabla
private void listaTablaProductos(){
    this.modeloDatosProductos.setRowCount(listaProductos.size());
    for(int i=0; i< listaProductos.size(); i++){
        this.modeloDatosProductos.setValueAt(i+1, i, 0);
        this.modeloDatosProductos.setValueAt(listaProductos.get(i).getNombre(), i, 1);
        this.modeloDatosProductos.setValueAt(listaProductos.get(i).getCantidad(), i, 2);
        this.modeloDatosProductos.setValueAt(listaProductos.get(i).getPrecioUnitario(), i, 3);
        this.modeloDatosProductos.setValueAt(listaProductos.get(i).getSubTotal(), i, 4);
        this.modeloDatosProductos.setValueAt(listaProductos.get(i).getDescuento(), i, 5);
    }
}
```

```
this.modeloDatosProductos.setValueAt(listaProductos.get(i).getIgv(), i, 6);
this.modeloDatosProductos.setValueAt(listaProductos.get(i).getTotalPagar(), i, 7);
this.modeloDatosProductos.setValueAt("Eliminar", i, 8);

}

// añadir jTable
jTable1_productos.setModel(modeloDatosProductos);
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jComboBox1_cliente = new javax.swing.JComboBox();
    jComboBox1_producto = new javax.swing.JComboBox();
    txt_cliente_buscar = new javax.swing.JTextField();
    txt_cantidad = new javax.swing.JTextField();
    jButton1_busca_cliente = new javax.swing.JButton();
    jButton1_añadir_producto = new javax.swing.JButton();
    jPanel1 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTable1_productos = new javax.swing.JTable();
    jPanel2 = new javax.swing.JPanel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    txt_subtotal = new javax.swing.JTextField();
    txt_descuento = new javax.swing.JTextField();
    txt_igv = new javax.swing.JTextField();
    txt_totalpagar = new javax.swing.JTextField();
    txt_efectivo = new javax.swing.JTextField();
    txt_cambio = new javax.swing.JTextField();
    jButton_calcular_cambio = new javax.swing.JButton();
    jButton_Registrarventa = new javax.swing.JButton();
    jLabel1_walpaper = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

    jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
    jLabel1.setText("FACTURACION");
    getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(310, -1, -1));

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

    jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
    jLabel1.setText("FACTURACION");
    getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(310, -1, -1));
}
```

```
jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel2.setText("Cliente");
getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(10,
40, 80, -1));

jLabel3.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel3.setText("Producto");
getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(10,
80, 80, -1));

jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel4.setText("Cantidad");
getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(290,
80, 80, -1));

jComboBox1_cliente.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jComboBox1_cliente.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Seleccione Cliente:", "Item 2", "Item 3", "Item 4" }));
getContentPane().add(jComboBox1_cliente,
new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 40, 170, -1));

jComboBox1_producto.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jComboBox1_producto.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Seleccione Producto:", "Item 2", "Item 3", "Item 4" }));
jComboBox1_producto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jComboBox1_productoActionPerformed(evt);
    }
});
getContentPane().add(jComboBox1_producto,
new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 80, 170, -1));

txt_cliente_buscar.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
txt_cliente_buscar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txt_cliente_buscarActionPerformed(evt);
    }
});
getContentPane().add(txt_cliente_buscar,
new org.netbeans.lib.awtextra.AbsoluteConstraints(270, 40, 130, -1));

txt_cantidad.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
getContentPane().add(txt_cantidad,
new org.netbeans.lib.awtextra.AbsoluteConstraints(390, 80, 160, -1));
```

```
jButton1_busca_cliente.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton1_busca_cliente.setText("Buscar");
jButton1_busca_cliente.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1_busca_clienteActionPerformed(evt);
    }
});
getContentPane().add(jButton1_busca_cliente,
new org.netbeans.lib.awtextra.AbsoluteConstraints(420, 40, 80, -1));new

jButton1_añadir_producto.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton1_añadir_producto.setText("Añadir Producto ");
jButton1_añadir_producto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1_añadir_productoActionPerformed(evt);
    }
});
getContentPane().add(jButton1_añadir_producto,
new org.netbeans.lib.awtextra.AbsoluteConstraints(580, 80, 140, -1));new

jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());
jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jTable1_productos.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Title 1", "Title 2", "Title 3", "Title 4"
    }
));
jTable1_productos.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jTable1_productosMouseClicked(evt);
    }
});
 jScrollPane1.setViewportView(jTable1_productos);

jPanel1.add(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10,
740, 190));new

getContentPane().add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 120, 760, 210));

jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());
jPanel2.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
jLabel5.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel5.setText("Subtotal:");
jPanel2.add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 20, -1, -1));

jLabel6.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel6.setText("Descuento:");
jPanel2.add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 50, -1, -1));

jLabel7.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel7.setText("IGV:");
jPanel2.add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, -1, -1));

jLabel8.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel8.setText("Total a Pagar");
jPanel2.add(jLabel8, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 110, -1, -1));

jLabel9.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel9.setText("Efectivo:");
jPanel2.add(jLabel9, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 150, -1, -1));

jLabel10.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel10.setText("Cambio:");
jPanel2.add(jLabel10, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 180, -1, -1));

txt_subtotal.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
txt_subtotal.setEnabled(false);
jPanel2.add(txt_subtotal, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 20, 120, -1));

txt_descuento.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
txt_descuento.setEnabled(false);
jPanel2.add(txt_descuento, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 50, 120, -1));

txt_igv.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
txt_igv.setEnabled(false);
jPanel2.add(txt_igv, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 80, 120, -1));

txt_totalpagar.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
txt_totalpagar.setEnabled(false);
jPanel2.add(txt_totalpagar, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 110, 120, -1));

txt_efectivo.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jPanel2.add(txt_efectivo, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 150, 120, -1));
```

```

txt_cambio.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
txt_cambio.setEnabled(false);
jPanel2.add(txt_cambio, new org.netbeans.lib.awtextra.AbsoluteConstraints(100, 180,
120, -1));

jButton_calcular_cambio.setBackground(new java.awt.Color(51, 255, 255));
jButton_calcular_cambio.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton_calcular_cambio.setText("Calcular Cambio");
jButton_calcular_cambio.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton_calcular_cambioActionPerformed(evt);
    }
});
jPanel2.add(jButton_calcular_cambio,
new org.netbeans.lib.awtextra.AbsoluteConstraints(230, 150, 130, 50));

getContentPane().add(jPanel2,
new org.netbeans.lib.awtextra.AbsoluteConstraints(390, 330, 380, 210));

jButton_Registrarventa.setBackground(new java.awt.Color(51, 255, 255));
jButton_Registrarventa.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton_Registrarventa.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Img/pdf.png"))); // NOI18N
jButton_Registrarventa.setText("Registrar Venta");
jButton_Registrarventa.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
jButton_Registrarventa.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
jButton_Registrarventa.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton_RegistrarventaActionPerformed(evt);
    }
});
getContentPane().add(jButton_Registrarventa,
new org.netbeans.lib.awtextra.AbsoluteConstraints(200, 350, 170, 100));

jLabel1_walpaper.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
getContentPane().add(jLabel1_walpaper,
new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 150, 600));

pack();
}// </editor-fold>

private void jButton1_busca_clienteActionPerformed(java.awt.event.ActionEvent evt) {
// Obtener y limpiar el texto ingresado por el usuario
String clienteBuscar = txt_cliente_buscar.getText().trim();

// Validación básica: evitar consultas vacías
if (clienteBuscar.isEmpty()) {
    JOptionPane.showMessageDialog(null, "Ingrese un nombre de cliente.");
}

```

```
        return; // Se detiene el flujo si no hay texto
    }

// Conexión a la base de datos usando try-with-resources para cerrar automáticamente
try (Connection cn = new Conexion().getConnection()) {

    // Consulta segura con PreparedStatement para evitar inyección SQL
    String sql = "SELECT nombre FROM clientes WHERE nombre = ?";
    PreparedStatement pst = cn.prepareStatement(sql);
    pst.setString(1, clienteBuscar); // Se inserta el valor de forma segura

    // Ejecutar la consulta
    ResultSet rs = pst.executeQuery();

    // Si se encuentra el cliente, se selecciona en el combo
    if (rs.next()) {
        jComboBox1_cliente.setSelectedItem(rs.getString("nombre"));
    } else {
        // Si no se encuentra, se muestra mensaje y se reinicia el combo
        jComboBox1_cliente.setSelectedItem("Seleccione cliente:");
        JOptionPane.showMessageDialog(null, "Nombre de cliente incorrecto.");
    }

    // Limpiar el campo de búsqueda
    txt_cliente_buscar.setText("");

} catch (SQLException e) {
    // Manejo de errores con mensaje visible al usuario
    JOptionPane.showMessageDialog(null, "Error al buscar cliente: " + e.getMessage());
}
}

private void jButton1_añadir_productoActionPerformed(java.awt.event.ActionEvent evt) {
// Obtener el producto seleccionado del combo
String combo = jComboBox1_producto.getSelectedItem().toString();

// 🔎 Validación: evitar que se procese si no se ha seleccionado un producto
if (combo.equalsIgnoreCase("Seleccione producto")) {
    JOptionPane.showMessageDialog(null, "Seleccione un producto");
    return; // Se detiene el flujo si no hay selección válida
}

// 🔎 Validación: verificar que el campo de cantidad no esté vacío
String cantidadTexto = txt_cantidad.getText().trim();
if (cantidadTexto.isEmpty()) {
    JOptionPane.showConfirmDialog(null, "Ingrese la cantidad de productos");
    return;
}
```

```
// 🔎 Validación: verificar que el texto ingresado sea numérico
boolean validacion = validar(cantidadTexto); // Método personalizado
if (!validacion) {
    JOptionPane.showConfirmDialog(null, "La cantidad no admite caracteres");
    return;
}

// 🔎 Validación: verificar que la cantidad sea mayor a cero
cantidad = Integer.parseInt(cantidadTexto);
if (cantidad <= 0) {
    JOptionPane.showConfirmDialog(null, "La cantidad no puede ser cero ni negativa");
    return;
}

// 🔍 Obtener datos del producto desde la base de datos o estructura interna
this.DatosDelProducto();

// 🔎 Validación: verificar que la cantidad no supere el stock disponible
if (cantidad > cantidadProducto) {
    JOptionPane.showMessageDialog(null, "La cantidad supera el stock disponible");
    return;
}

// 📈 Cálculo de montos
subtotal = precioUnitario * cantidad;
totalPagar = subtotal + igv + descuento;

// 🔍 Redondeo de valores a dos decimales
subtotal = Math.round(subtotal * 100) / 100.0;
igv = Math.round(igv * 100) / 100.0;
descuento = Math.round(descuento * 100) / 100.0;
totalPagar = Math.round(totalPagar * 100) / 100.0;

// 📦 Creación del objeto DetalleVenta con todos los datos
producto = new DetalleVenta(
    auxIdDetalle,
    1,
    idProducto,
    nombre,
    cantidad,
    precioUnitario,
    subtotal,
    descuento,
    igv,
    totalPagar,
    1
);
```

```

// + Añadir el producto a la lista de ventas
listaProductos.add(producto);
JOptionPane.showMessageDialog(null, "Producto agregado");

// 🔍 Actualización de estado y limpieza de campos
auxIdDetalle++;
txt_cantidad.setText("");
this.CargarComboProductos();
this.CalcularTotalPagar();
txt_efectivo.setEnabled(true);
jButton_calcular_cambio.setEnabled(true);

// 📁 Actualizar la tabla visual de productos
this.listaTablaProductos();

}

private void jComboBox1_productoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton_calcular_cambioActionPerformed(java.awt.event.ActionEvent evt) {
if(!txt_efectivo.getText().isEmpty()){
    // validamos
    boolean validacion =validarDouble(txt_efectivo.getText());
    if(validacion==true){
        // validar el efectivo que ingresamos sea mayor a total a pagar
        double efc=Double.parseDouble(txt_efectivo.getText().trim());
        double top=Double.parseDouble(txt_totalpagar.getText().trim());
        if (efc < top){
            JOptionPane.showConfirmDialog(null, "el dinero en efectivo no es suficiente");
        }
        else {
            double cambio=(efc - top);
            double cambi = (double)Math.round(cambio * 100d)/100;
            String camb=String.valueOf(cambi);
            txt_cambio.setText(camb);
        }
    }
    else{
        JOptionPane.showConfirmDialog(null, "no se permite caracteres no numericos");
    }
}
else{
    JOptionPane.showConfirmDialog(null, "ingrese dinero en efectivo");
}
}

int idArrayList=0;
private void jTable1_productosMouseClicked(java.awt.event.MouseEvent evt) {

```

```

// 🔍 Obtener la fila seleccionada en la tabla
int filaSeleccionada = jTable1_productos.rowAtPoint(evt.getPoint());
int columnalD = 0;

// 🔎 Validar que se haya hecho clic sobre una fila válida
if (filaSeleccionada > -1) {
    //>ID Obtener el ID del producto desde la primera columna
    idArrayList = (int) modeloDatosProductos.getValueAt(filaSeleccionada, columnalD);

    // 🗑 Mostrar cuadro de confirmación para eliminar
    int opcion = JOptionPane.showConfirmDialog(null, "¿Eliminar producto?");

    //🎯 Procesar la respuesta del usuario
    switch (opcion) {
        case 0: // Sí
            listaProductos.remove(idArrayList - 1); // Elimina el producto de la lista
            this.CalcularTotalPagar(); // Recalcula el total
            this.listaTablaProductos(); // Actualiza la tabla visual
            break;

        case 1: // No
            // No se realiza ninguna acción
            break;

        default: // Cancelar o cerrar
            // No se realiza ninguna acción
            break;
    }
}
}

private void jButton_RegistrarventaActionPerformed(java.awt.event.ActionEvent evt) {
// 📦 Instanciar objetos necesarios
CabeceraVenta cabeceraVenta = new CabeceraVenta();
DetalleVenta detalleVenta = new DetalleVenta();
Ctrl_RegistrarVenta controlVenta = new Ctrl_RegistrarVenta();

// 📅 Obtener fecha actual en formato yyyy/MM/dd
String fechaActual = new SimpleDateFormat("yyyy/MM/dd").format(new Date());

// 🔎 Validar selección de cliente
if (jComboBox1_cliente.getSelectedItem().equals("Seleccione cliente:")) {
    JOptionPane.showMessageDialog(null, "¡Seleccione un cliente!");
    return;
}

// 🔎 Validar que haya productos en la lista
if (listaProductos.isEmpty()) {

```

```
JOptionPane.showMessageDialog(null, "¡Seleccione un producto!");
return;
}

// ID Obtener ID del cliente
this.ObtenerIdCliente();

// ☰ Configurar cabecera de venta
cabeceraVenta.setIdCabeceraVenta(0);
cabeceraVenta.setIdCliente(idCliente);
cabeceraVenta.setValorPagar(Double.parseDouble(txt_totalpagar.getText()));
cabeceraVenta.setFechaVenta(fechaActual);
cabeceraVenta.setEstado(1);

// ☐ Registrar cabecera en la base de datos
if (!controlVenta.guardar(cabeceraVenta)) {
    JOptionPane.showMessageDialog(null, "¡Error al guardar cabecera de venta!");
    return;
}

JOptionPane.showMessageDialog(null, "¡Venta Registrada!");

// ☰ Generar factura PDF
VentaPDF pdf = new VentaPDF();
pdf.DatosCliente(idCliente);
pdf.generarFacturaPDF();

// ✅ Registrar cada detalle de producto
for (DetalleVenta elemento : listaProductos) {
    detalleVenta.setIdDetalleVenta(0);
    detalleVenta.setIdCabeceraVenta(0); // Se puede actualizar si se obtiene el ID real
    detalleVenta.setIdProducto(elemento.getIdProducto());
    detalleVenta.setCantidad(elemento.getCantidad());
    detalleVenta.setPrecioUnitario(elemento.getPrecioUnitario());
    detalleVenta.setSubTotal(elemento.getSubTotal());
    detalleVenta.setDescuento(elemento.getDescuento());
    detalleVenta.setIgv(elemento.getIgv());
    detalleVenta.setTotalPagar(elemento.getTotalPagar());
    detalleVenta.setEstado(1);

    // ☐ Guardar detalle en la base de datos
    if (!controlVenta.guardarDetalle(detalleVenta)) {
        JOptionPane.showMessageDialog(null, "¡Error al guardar detalle de venta!");
        continue;
    }

    // 🔄 Actualizar interfaz y stock
    txt_subtotal.setText("0.0");
    txt_igv.setText("0.0");
```

```

txt_descuento.setText("0.0");
txt_totalpagar.setText("0.0");
txt_efectivo.setText("");
txt_cambio.setText("0.0");
auxIdDetalle = 1;

this.CargarComboClientes();
this.RestarStockProductos(elemento.getIdProducto(), elemento.getCantidad());
}

// ✎ Limpiar lista y actualizar tabla
listaProductos.clear();
listaTablaProductos();

}

private void txt_cliente_buscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Factura.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Factura.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Factura.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalArgumentException ex) {
}
}

```

```
java.util.logging.Logger.getLogger(Factura.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(Factura.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}
//</editor-fold>
//</editor-fold>
//</editor-fold>
//</editor-fold>

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

// Variables declaration - do not modify
private javax.swing.JButton jButton1_añadir_producto;
private javax.swing.JButton jButton1_busca_cliente;
private javax.swing.JButton jButton_Registrarventa;
private javax.swing.JButton jButton_calcular_cambio;
private javax.swing.JComboBox<String> jComboBox1_cliente;
private javax.swing.JComboBox<String> jComboBox1_producto;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel1_walpaper;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
public static javax.swing.JScrollPane jScrollPane1;
public static javax.swing.JTable jTable1_productos;
private javax.swing.JTextField txt_cambio;
private javax.swing.JTextField txt_cantidad;
private javax.swing.JTextField txt_cliente_buscar;
private javax.swing.JTextField txt_descuento;
private javax.swing.JTextField txt_efectivo;
private javax.swing.JTextField txt_igv;
```

```
private javax.swing.JTextField txt_subtotal;
public static javax.swing.JTextField txt_totalpagar;
// End of variables declaration

// metodo para cargar clientes
private void CargarComboClientes(){
    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();
    String sql="select * from clientes";
    Statement st;
    try{
        st = cn.createStatement();
        ResultSet rs = st.executeQuery(sql);
        jComboBox1_cliente.removeAllItems();
        jComboBox1_cliente.addItem("Seleccione cliente:");
        while (rs.next()){
            jComboBox1_cliente.addItem(rs.getString("nombre"));
            //jComboBox1_cliente.addItem(rs.getString("nombre")+" "+rs.getString("apellido"));
        }
    } catch (SQLException e){
        System.out.println("Error al cargar cliente, !"+e);
    }
}
// metodo para cargar productos
private void CargarComboProductos(){
    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();
    String sql="select * from productos";
    Statement st;
    try{
        st = cn.createStatement();
        ResultSet rs = st.executeQuery(sql);
        jComboBox1_producto.removeAllItems();
        jComboBox1_producto.addItem("Seleccione producto:");
        while (rs.next()){
            jComboBox1_producto.addItem(rs.getString("nombre"));
        }
    } catch (SQLException e){
        System.out.println("Error al cargar producto, !"+e);
    }
}

// metodo para validar que el usuario no ingrese caracteres no numericos
private boolean validar(String valor){
```

```
try {
    int num= Integer.parseInt(valor);
    return true;
}

}catch(NumberFormatException e){
    return false;
}
}

// metodo para validar que el usuario no ingrese caracteres no numericos
private boolean validarDouble(String valor){
try {
    double num= Double.parseDouble(valor);
    return true;
}

}catch(NumberFormatException e){
    return false;
}
}

private void DatosDelProducto(){
    try {
        String sql="select * from productos where nombre= "+this.jComboBox1_producto.getSelectedItem()+"";
        Conexion conexion = new Conexion();
        Connection cn = conexion.getConnection();
        Statement st;
        st=cn.createStatement();
        ResultSet rs=st.executeQuery(sql);
        while (rs.next()){
            idProducto=rs.getInt("idProducto");
            nombre=rs.getString("nombre");
            cantidadProducto=rs.getInt("cantidad");
            precioUnitario=rs.getDouble("precio");
            porcentajeigv=rs.getInt("porcentajeigv");
            this.Calcularigv(precioUnitario, porcentajeigv);
        }
    }catch (SQLException e){
        System.out.println("Error al obtener datos del producto");
    }
}

//Metodo para calcular igv
private double Calcularigv(double precio, int porcentajeigv){
    int p_igv=porcentajeigv;
    switch (p_igv){
        case 0:
            igv=0.0;
    }
}
```

```

        break;
    case 12:
        igv=(precio * cantidad)*0.12;
    case 14:
        igv=(precio*cantidad)*0.14;
        break;
    default:
        break;
    }
    return igv;
}
// metodo para calcular total a pagar los productos
private void CalcularTotalPagar(){
subtotalGeneral =0;
descuentoGeneral = 0;
igvGeneral = 0;
totalPagarGeneral = 0;
for(DetalleVenta elemento : listaProductos){
    subtotalGeneral +=elemento.getSubTotal();
    descuentoGeneral +=elemento.getDescuento();
    igvGeneral +=elemento.getIgv();
    totalPagarGeneral +=elemento.getTotalPagar();
}
//REDONDEAR DECIMALES
subtotalGeneral= (double) Math.round(subtotalGeneral *100)/100;
igvGeneral =(double) Math.round(igvGeneral *100)/100;
descuentoGeneral =(double) Math.round(descuentoGeneral *100)/100;
totalPagarGeneral =(double) Math.round(totalPagarGeneral *100)/100;
//enviar datos a la vista
txt_subtotal.setText(String.valueOf(subtotalGeneral));
txt_igv.setText(String.valueOf(igvGeneral));
txt_descuento.setText(String.valueOf(descuentoGeneral));
txt_totalpagar.setText(String.valueOf(totalPagarGeneral));
}
private void ObtenerIdCliente() {
try {
    String sql = "select * from clientes where concat(nombre) = '' +
this.jComboBox1_cliente.getSelectedItem() + """;
    Conexion conexion = new Conexion();
    Connection cn = conexion.getConnection();
    Statement st;
    st = cn.createStatement();
    ResultSet rs = st.executeQuery(sql);
    while (rs.next()) {
        idCliente = rs.getInt("id");
    }
} catch (SQLException e) {
    System.out.println("Error al obtener id del cliente, " + e);
}

```

```
        }
    }

    private void RestarStockProductos(int idProducto, int cantidad) {
        int cantidadProductosBaseDeDatos = 0;
        try {
            Conexion conexion = new Conexion();
            Connection cn = conexion.getConnection();
            String sql = "select idProducto, cantidad from productos where idProducto = '" +
idProducto + "'";
            Statement st;
            st = cn.createStatement();
            ResultSet rs = st.executeQuery(sql);
            while (rs.next()) {
                cantidadProductosBaseDeDatos = rs.getInt("cantidad");
            }
            cn.close();
        } catch (SQLException e) {
            System.out.println("Error al restar cantidad 1, " + e);
        }

        try {
            Conexion conexion = new Conexion();
            Connection cn = conexion.getConnection();
            PreparedStatement consulta = cn.prepareStatement("update productos set
cantidad=? where idProducto = '" + idProducto + "'");
            int cantidadNueva = cantidadProductosBaseDeDatos - cantidad;
            consulta.setInt(1, cantidadNueva);
            if(consulta.executeUpdate() > 0){
                //System.out.println("Todo bien");
            }
            cn.close();
        } catch (SQLException e) {
            System.out.println("Error al restar cantidad 2, " + e);
        }
    }
}
```