

Proyecto Final

Carlos Javier Mahecha Gómez
Jonatan Camilo Igua Contreras

Docente

Juan Carlos Gómez Cabanzo

Asignatura

Sistemas Transaccionales

NRC

4477

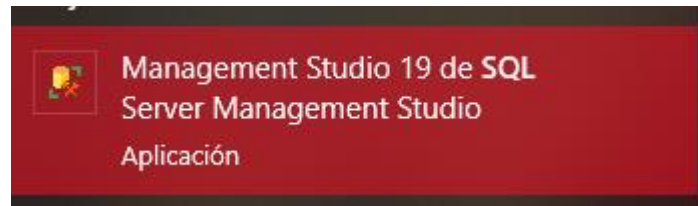
Corporación Universitaria Minuto de Dios

Bogotá D.C.

30/11/2023

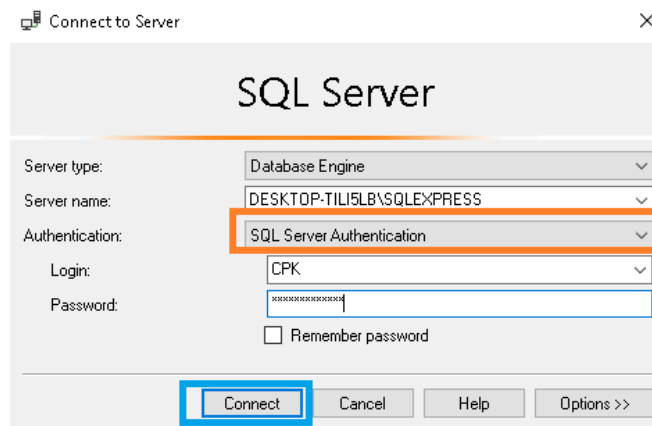
Base de datos

El motor de base de datos usado para crear el proyecto es SQL Server, por su versatilidad y grandes funciones a la hora de crear y gestionar las bases de datos.

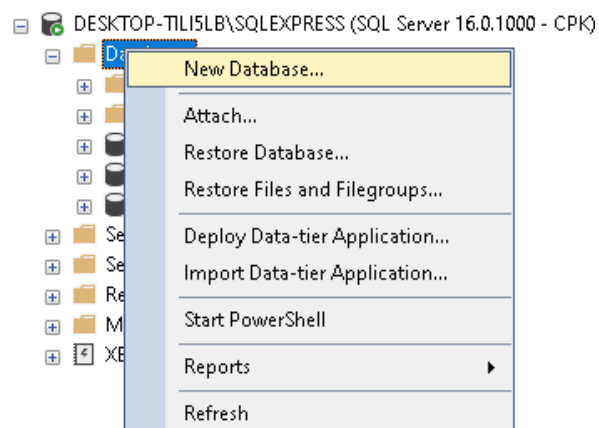


Crear Base de datos:

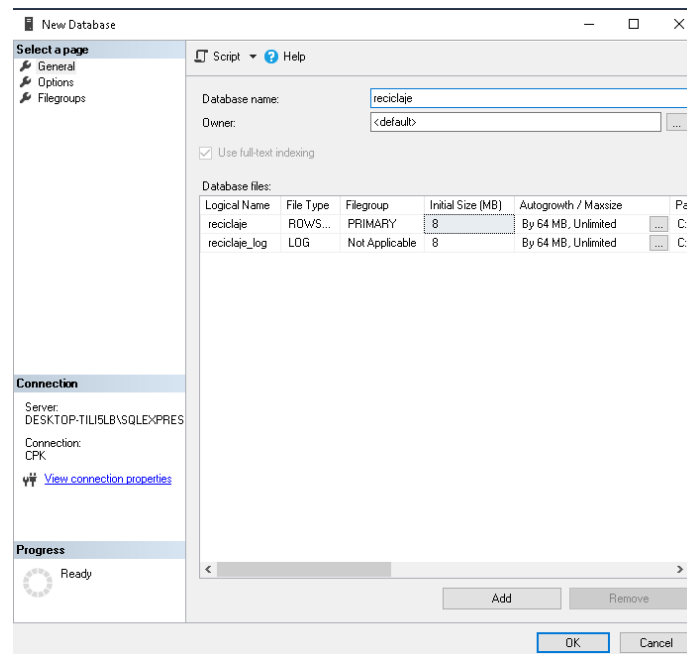
Para crear la base de datos, primero es necesario ingresar al motor de base de datos, se debe escoger en la opción “**Authentication**”, y colocar SQL Server Authentication, porque esta opción permite ingresar con un usuario y contraseña necesarios para crear la conexión con NetBeans.



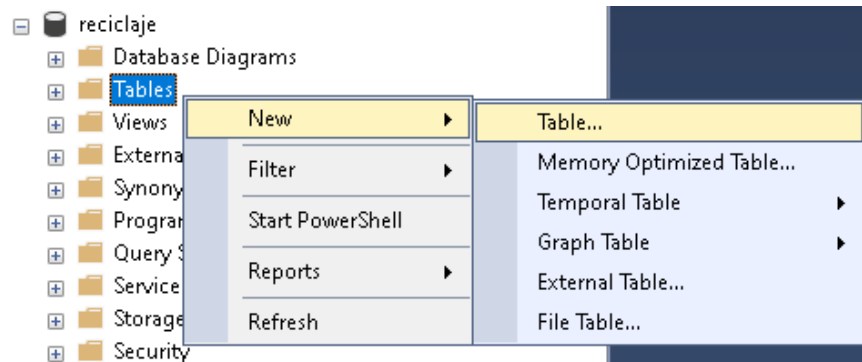
En la opción Databases, se da clic derecho y se selecciona New Database



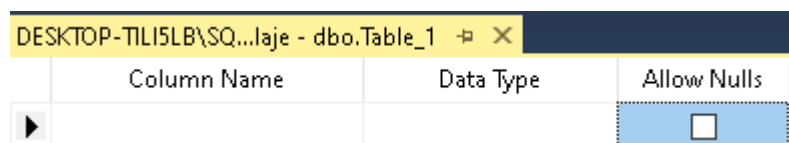
Se abrirá una nueva ventana en la cual se podrá nombrar la nueva base de datos a crear, en este caso se llama a la base de datos como “reciclaje”



Una vez creada la base de datos en la barra lateral aparecerá la base de datos con algunas carpetas creadas por defecto, para crear tablas en la base de datos se da clic derecho a la carpeta “Tables”, luego en New y luego en Table



Una vez creada la tabla aparecerán una serie de columnas y filas para crear los registros con su tipo de datos.



Tablas para la base de datos:

Tabla usuario:

Para crear la tabla usuario se crea una nueva tabla con los registros, id_usuario, Nombre, Apellidos, Telefono, Direccion, Nombre_Material y Cantidad_Material

	Column Name	Data Type	Allow Nulls
▶	id_usuario	int	<input type="checkbox"/>
	Nombre	varchar(10)	<input checked="" type="checkbox"/>
	Apellidos	varchar(20)	<input checked="" type="checkbox"/>
	Telefono	varchar(50)	<input checked="" type="checkbox"/>
	Direccion	varchar(30)	<input checked="" type="checkbox"/>
	Nombre_Material	varchar(50)	<input checked="" type="checkbox"/>
	Cantidad_Material	int	<input checked="" type="checkbox"/>

Se crean 4 registros con la información de los usuarios.

	id_usuario	Nombre	Apellidos	Telefono	Direccion	Nombre_Mate...	Cantidad_Mat...
▶	1	Wilson	Gonzales Carvajal	3115874869	Cll 22 # 22-1	Botellas	200
	2	Carlos	Mahecha Gomez	3165852808	Cll 52 # 3-2	Chatarra	25
	3	Samanta	Perez Paez	3204587621	Cll 45 # 4-6	Cajas	78
	4	Jonatan	Igua Contreras	3165475878	Cll 25 # 9-6	Alambre	150

Tabla almacen:

Para crear la tabla almacen se crea una nueva tabla con los registros, id_almacen, Tipo_Material, Nombre_Material, Cantidad_Material, Peso y Valor_kg

	Column Name	Data Type	Allow Nulls
▶	id_almacen	int	<input type="checkbox"/>
	Tipo_Material	varchar(20)	<input checked="" type="checkbox"/>
	Nombre_Material	varchar(25)	<input checked="" type="checkbox"/>
	Cantidad_Material	int	<input checked="" type="checkbox"/>
	Peso	float	<input checked="" type="checkbox"/>
	Valor_kg	money	<input checked="" type="checkbox"/>

Se crean 8 registros con la información de los materiales del almacén.

	id_almacén	Tipo_Material	Nombre_Mate...	Cantidad_Mat...	Peso	Valor_kg
▶	1	Plastico	Botellas	465	4	0,4700
	2	Carton	Cajas	70	72	20500,0000
	3	Cobre	Alambre	1	0,5	34,7020
	4	Aluminio	Latas	467	10,5	5,2000
	7	Vidrio	Ventanas	50	90	22000,0000
	8	Chatarra	Laminas	60	22000	200,0000

Tabla empresa:

Para crear la tabla empresa se crea una nueva tabla con los registros, id_empresa, Nombre_Empresa, Direccion, Telefono, Correo, Cantidad_Material, Nombre_Material y Pagos.

	Column Name	Data Type	Allow Nulls
▶	id_empresa	int	<input type="checkbox"/>
	Nombre_Empresa	varchar(30)	<input checked="" type="checkbox"/>
	Direccion	varchar(20)	<input checked="" type="checkbox"/>
	Telefono	varchar(50)	<input checked="" type="checkbox"/>
	Correo	varchar(40)	<input checked="" type="checkbox"/>
	Cantidad_Material	int	<input checked="" type="checkbox"/>
	Nombre_Material	varchar(50)	<input checked="" type="checkbox"/>
	Pagos	varchar(50)	<input checked="" type="checkbox"/>

Se crean 5 registros con la información de las empresas compradoras del material reciclable.

	id_empresa	Nombre_Empr...	Direccion	Telefono	Correo	Cantidad_Mat...	Nombre_Mate...	Pagos
▶	1	Clis S.A	CII 128 # 2-8	3125489012	Clis@gmail.com	301	Papel	exito
	2	ReciclaBogota	CII 1 # 32-1	3142596742	reciclabogota@...	54	Plastico	pendiente
	3	Reutilis	CII 12 # 20-1	3152459874	reutilis@gmail....	400	Carton	exito
	4	Si Al Reciclaje	CII 32 # 12-6	3205486974	sireciclaje@gm...	467	Aluminio	exito
	5	Limes	CII 20 # 20 -7	3154569874	limes@gmail.c...	214	Papel	pendiente

Tabla registro_Compra:

Para crear la tabla registro_Compra se crea una nueva tabla con los registros, id_Compra, id_usuario y id_almacén

Esta tabla es una tabla intermedia que permite la comunicación entre la tabla usuario y la tabla almacén.

	Column Name	Data Type	Allow Nulls
▶	id_Compra	int	<input type="checkbox"/>
	id_usuario	int	<input checked="" type="checkbox"/>
	id_almacen	int	<input checked="" type="checkbox"/>

Se crean 5 registros con los id de las tablas relacionadas.

	id_Compra	id_usuario	id_almacen
▶	1	2	5
	2	5	1
	3	4	3
	4	1	2
	5	3	4

Tabla devoluciones:

Para crear la tabla devoluciones se crea una nueva tabla con los registros, id_devolucion, nombre_Empresa y estado.

	Column Name	Data Type	Allow Nulls
▶	id_devolucion	int	<input type="checkbox"/>
	nombre_Empresa	varchar(50)	<input checked="" type="checkbox"/>
	estado	varchar(50)	<input checked="" type="checkbox"/>

Se crea 1 registro con la información de las empresas que solicitan una devolucion despues de la compra de un material reciclable.

	id_devolucion	nombre_Empr...	estado
▶	1	SaCI	Aprobado
*	NULL	NULL	NULL

Trigger y procesos almacenados aplicados en el proyecto

Trigger Devolución:

```
/*
Procedimiento Almacenado: Nuevo_Materiales

Descripción:
Este procedimiento maneja la inserción o actualización de registros en la tabla 'almacen' basándose en el nombre del material.
Si ya existe un registro con el mismo nombre de material, se realiza una actualización de la cantidad, peso y valor por kilogramo.
Si no existe, se realiza una inserción de un nuevo registro.

Parámetros:
- @tipo_Material: Tipo de material (varchar, longitud máxima: 20).
- @nombre_Material: Nombre del material (varchar, longitud máxima: 25).
- @cantidad_Material: Cantidad del material (entero).
- @peso: Peso del material (flotante).
- @valor_kg: Valor por kilogramo del material (moneda).
*/

-- Crear el procedimiento almacenado
Create procedure Nuevo_Materiales
@tipo_Material varchar(20),
@nombre_Material varchar(25),
@cantidad_Material int,
@peso float,
@valor_kg money
as
Begin
    -- Verificar si ya existe un registro con el mismo nombre de material
    if EXISTS(select * from almacen where Nombre_Material = @nombre_Material)
    Begin
        -- Actualizar el registro existente
        UPDATE almacen
        set
            Cantidad_Material = Cantidad_Material + @cantidad_Material,
            Peso = Peso + @peso,
            Valor_kg = Valor_kg + @valor_kg
        where
            Tipo_Material = @tipo_Material and
            Nombre_Material = @nombre_Material
    end
    else
    Begin
        -- Insertar un nuevo registro
        INSERT into almacen
        values
            (@tipo_Material, @nombre_Material, @cantidad_Material, @peso, @valor_kg)
    end
End
go
```

Trigger venta_resto_almacen:

Descripción:
Este trigger se activa después de realizar una inserción en la tabla 'empresa'. Su función es restar la cantidad de material vendido en la tabla 'almacen'. Se asume que la tabla 'empresa' contiene información sobre ventas, y la tabla 'almacen' contiene información sobre los materiales en stock.

Uso:
Este trigger se ejecuta automáticamente después de realizar una inserción en la tabla 'empresa'.

```
*/
-- Crear el trigger
CREATE TRIGGER venta_resto_almacen
ON empresa
AFTER INSERT
AS
BEGIN
    -- Declarar variables para almacenar el nombre del material y la nueva cantidad
    DECLARE @cantidadNueva AS INT
    DECLARE @nombreMaterial AS VARCHAR(50)

    -- Obtener el nombre del material y la nueva cantidad de la última inserción
    SELECT TOP 1
        @nombreMaterial = i.Nombre_Material,
        @cantidadNueva = i.Cantidad_Material
    FROM
        inserted i

    -- Actualizar la cantidad en la tabla 'almacen' restando la nueva cantidad vendida
    UPDATE almacen
    SET Cantidad_Material = Cantidad_Material - @cantidadNueva
    WHERE Nombre_Material = @nombreMaterial
END;
```

Proceso almacenado Nuevos_Materiales:

Descripción:
Este procedimiento maneja la inserción o actualización de registros en la tabla 'almacen' basándose en el nombre del material. Si ya existe un registro con el mismo nombre de material, se realiza una actualización de la cantidad, peso y valor por kilogramo. Si no existe, se realiza una inserción de un nuevo registro.

```
*/
```

```
-- Crear el procedimiento almacenado
CREATE PROCEDURE Nuevo_Materiales
    @tipo_Material varchar(20),
    @nombre_Material varchar(25),
    @cantidad_Material int,
    @peso float,
    @valor_kg money
AS
BEGIN
    -- Verificar si ya existe un registro con el mismo nombre de material
    IF EXISTS(SELECT * FROM almacen WHERE Nombre_Material = @nombre_Material)
    BEGIN
        -- Actualizar el registro existente
        UPDATE almacen
        SET
            Cantidad_Material = Cantidad_Material + @cantidad_Material,
            Peso = Peso + @peso,
            Valor_kg = Valor_kg + @valor_kg
        WHERE
            Tipo_Material = @tipo_Material AND
            Nombre_Material = @nombre_Material
    END
    ELSE
    BEGIN
        -- Insertar un nuevo registro
        INSERT INTO almacen
        VALUES
            (@tipo_Material, @nombre_Material, @cantidad_Material, @peso, @valor_kg)
    END
END
GO
```


Proceso almacenado Descuento_Compra:

```
/*
Procedimiento Almacenado: Descuento_Compra

Descripción:
Este procedimiento determina si una empresa ha realizado una compra de materiales igual o mayor a 300 unidades.
En caso afirmativo, se aplica un descuento del 15% en la próxima compra de cualquier material y se registra la aplicación del descuento en la minuta de la empresa.
En caso contrario, se establece el descuento en 0.
*/

-- Crear el procedimiento almacenado
CREATE PROCEDURE Descuento_Compra
    @id_empresa int
AS
BEGIN
    -- Verificar si la empresa ha realizado una compra mayor o igual a 300 unidades
    IF EXISTS(SELECT * FROM empresa WHERE Cantidad_Material >= 300 AND id_empresa = @id_empresa)
    BEGIN
        PRINT 'Esta empresa cuenta con una compra de materiales mayor o igual a 300, por lo tanto en la próxima compra de cualquier material que realice se le dará un descuento del 15%, de igual manera se registrará en la minuta de la empresa beneficiada la aplicación de un único descuento'

        -- Aplicar el descuento del 15%
        UPDATE empresa SET Descuento = 15 WHERE id_empresa = @id_empresa
    END
    ELSE
    BEGIN
        PRINT 'Esta empresa no tiene registrada una compra mayor o igual a 300, por lo tanto no aplica al descuento'

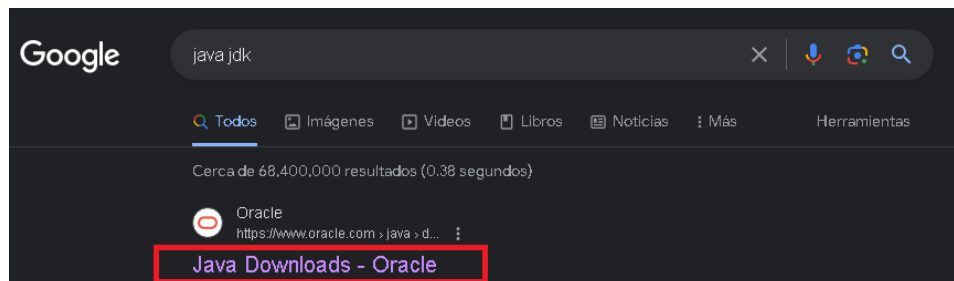
        -- Establecer el descuento en 0
        UPDATE empresa SET Descuento = 0 WHERE id_empresa = @id_empresa
    END
END
GO
```

Creación de la interfaz grafica

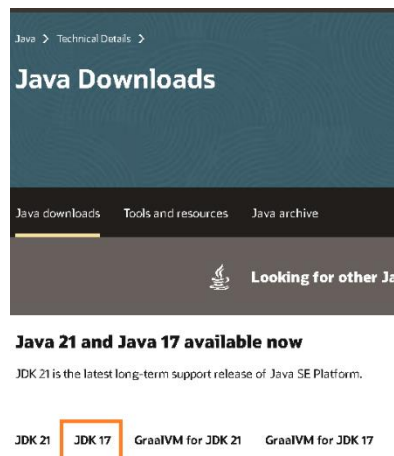
Para la creación del proyecto se usó el editor de código NetBeans, se implementó como sistema de gestión de base de datos SQL Server y se usó un Windows 10 para la creación del proyecto. Para la construcción del software se usó el lenguaje de programación de Java, por este motivo se tiene que descargar el JDK de Java para la construcción y ejecución del aplicativo de escritorio.

Instalación del JDK de Java:

Para instalar el JDK de Java, se tiene que buscar en la página oficial de Oracle y descargar la versión adecuada para el sistema operativo.



Cuando se ingresa a la página web, en el apartado de Downloads, se pueden descargar las diferentes versiones de Java, se recomienda no utilizar una versión reciente, por este motivo no se escoge la versión de JDK 21, por lo cual se optó por ocupar la versión JDK 17



Una vez descargado el JDK al ejecutarlo podemos instalarlo como cualquier otro programa de Windows.

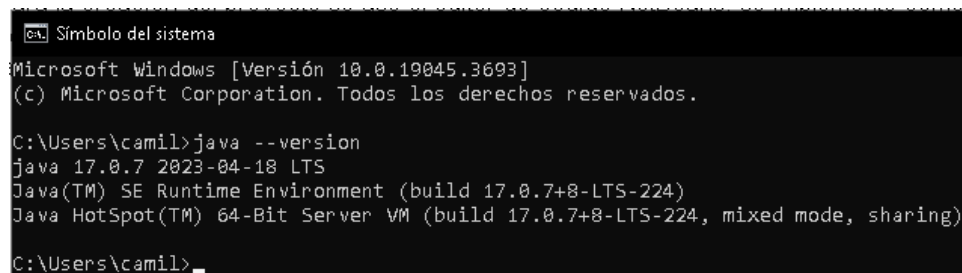
El archivo que se descarga es un ejecutable.exe



Si se ejecuta este archivo se abrirá una ventana para la instalación del JDK, simplemente se dan a siguiente para poder instalarlo en el equipo.



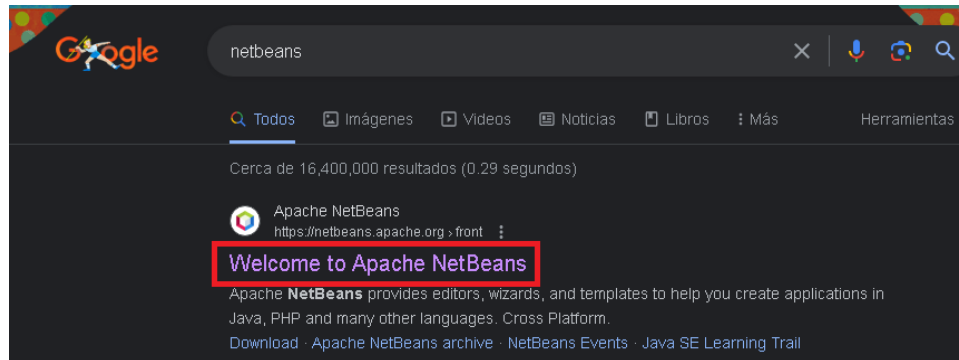
Para verificar si el JDK se instaló exitosamente, se abre el CMD de Windows y se ejecuta el comando **java -- version**



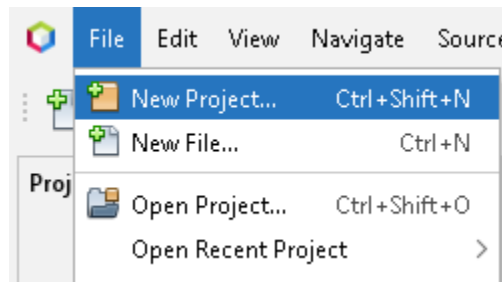
Creación del proyecto con NetBeans:

Para la creación del proyecto no basta con tener el JDK Java, si bien las herramientas del lenguaje ya se podrían ejecutar desde la consola de comandos, el entorno es muy limitado y complejo de usar, por este motivo se usa un IDE que permita hacer el desarrollo más rápido, fácil e intuitivo para el desarrollador.

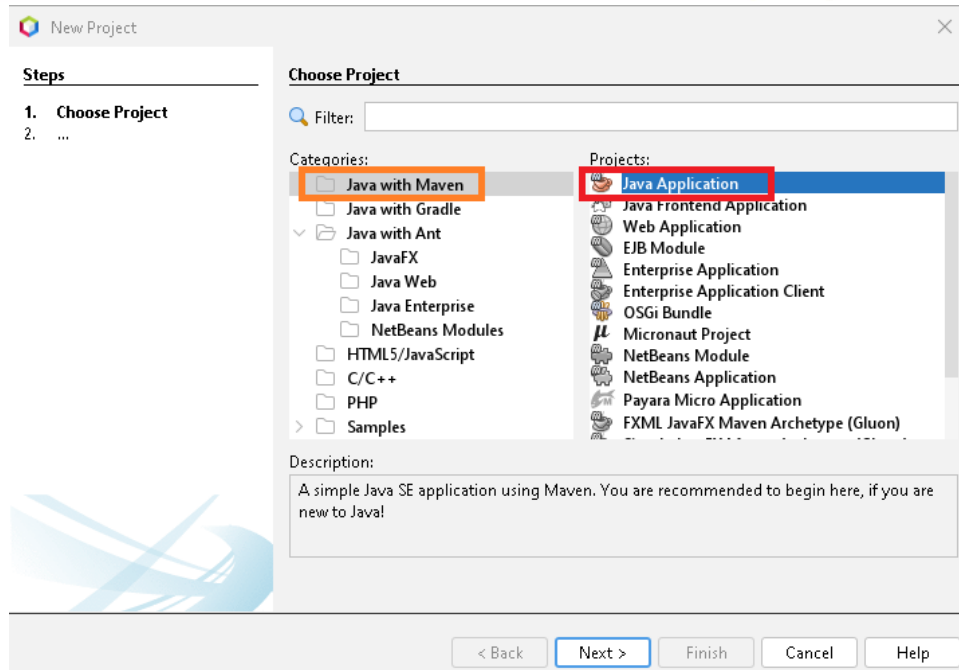
Para descargar esta herramienta se busca “Netbeans”, y se ingresa al primer resultado para descargar la herramienta.



Nuevo proyecto: Para crear un nuevo proyecto de java en netbeans es necesario dirigirse a la parte superior izquierda, en la opción File y seleccionar New Project

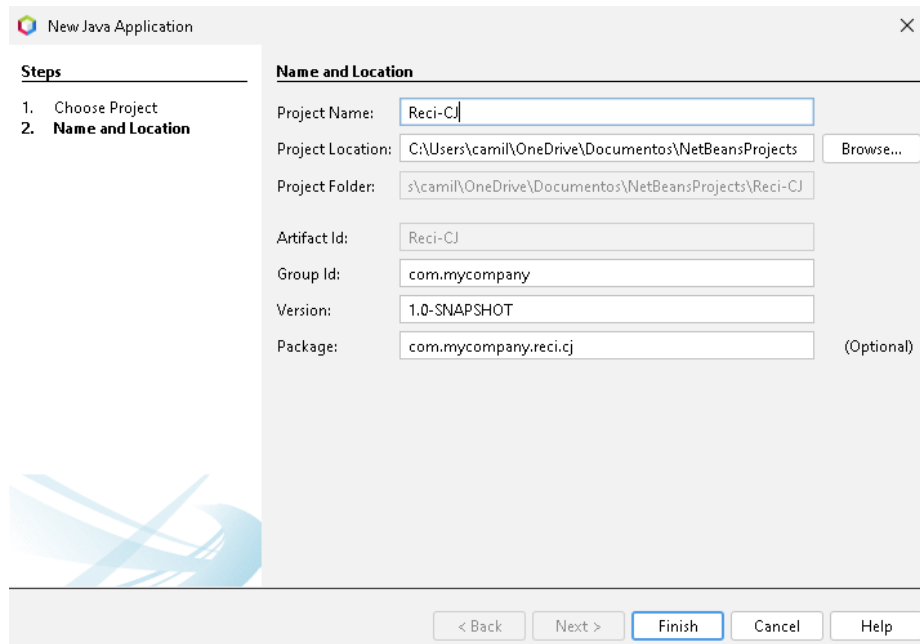


Al seleccionar esta opción se abre una nueva ventana en la cual seleccionar el tipo de proyecto de Java a crear, en este caso se escogió la opción de Java with Maven como categoría y como proyecto Java Application.



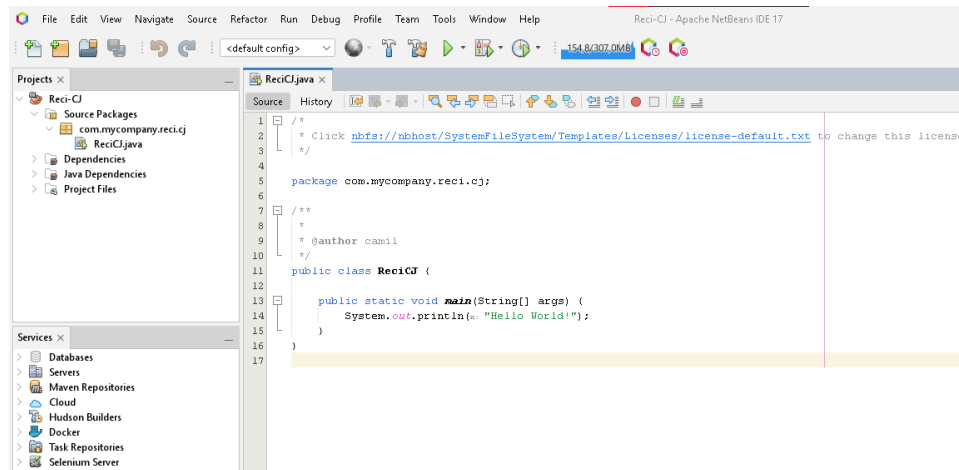
Luego de seleccionar la categoría y tipo del proyecto de Java, se procede a colocarle un nombre al nuevo proyecto y la ubicación en donde se va a aguardar, en este caso el proyecto se llamó

Reci-CJ y se guardara en la carpeta de NetBeansProjects



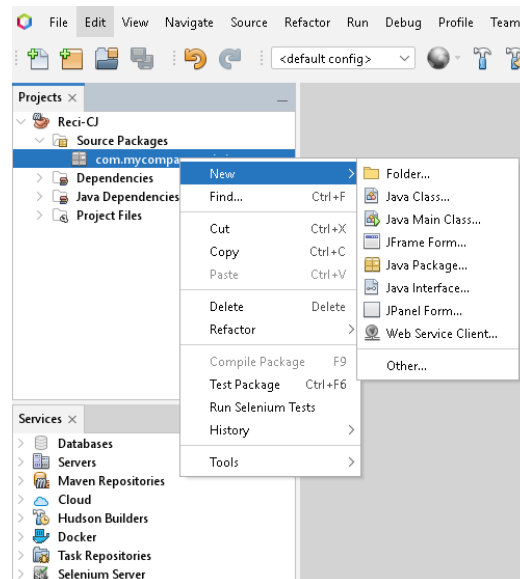
Cuando se crea el proyecto NetBeans crea una clase por defecto, pero en este caso esta clase no es necesaria porque se utilizarán JFrame From para la creación de la interfaz por este

motivo se puede eliminar, para eliminarla de selecciona el archivo RecicJ.java, luego se da clic derecho y se selecciona la opción **Delete** para eliminar la clase

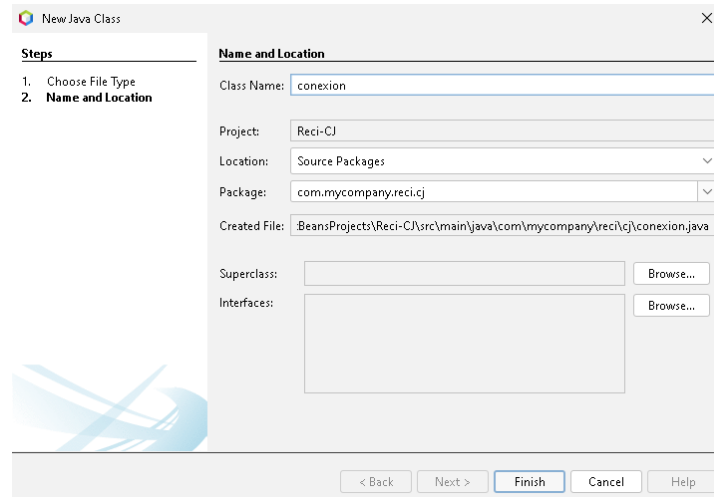


Creación de la Clase conexión:

Para crear una nueva clase en el proyecto, se selecciona el paquete, se da clic derecho, se escoge **New** y se selecciona **Java Class**



Luego se abre una ventana para colocar el nombre de la clase en este caso **conexión**



Dentro de la clase se crean variables para la conexión con la base de datos en SQL Server, como lo es la variable usuario, contra, bd, ip, puerto.

La variable **usuario** se usa para almacenar el nombre de usuario estipulado en SQL Server, la variable **contra** se usa para almacenar la contraseña de SQL Server, **bd** en ella se escribe el nombre de la base de datos a la cual hacer conexión, en **ip** la dirección del dispositivo para conectarse y en la variable **puerto**, se almacena el número de puerto configurado con anterioridad en SQL Server.

Se crea un método llamado “establecerConexion()”, en el cual se crea la lógica para poder establecer la conexión del proyecto de Netbeans, con el motor de base de datos SQLServer.

Código Clase Conexión:

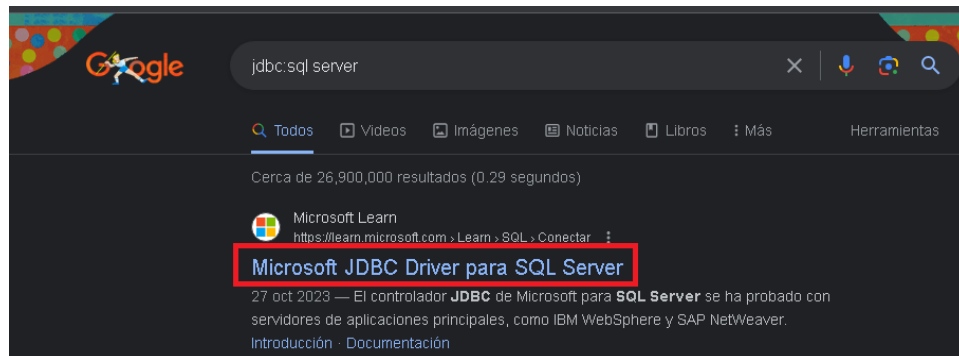
```

1
2 package com.mycompany.recic.cj;
3
4 import java.sql.Connection;
5 import java.sql.DriverManager;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.sql.SQLException;
9 import javax.swing.JOptionPane;
10
11 public class conexion {
12     Connection conectar = null;
13
14     String usuario = "admin";
15     String contra = "uniminuto2022";
16     String bd = "reciclaje";
17     String ip = "localhost";
18     String puerto = "1433";
19
20     String cadena = "jdbc:sqlserver://" + ip + ":" + puerto + "/" + bd;
21
22     public Connection establecerConexion() {
23         try {
24             String cadena = "jdbc:sqlserver://localhost:" + puerto + ";" + "databaseName=" + bd + ";trustServerCertificate = true;";
25             conectar = DriverManager.getConnection(uri: cadena, user: usuario, password: contra);
26         } catch (Exception e) {
27             JOptionPane.showMessageDialog(parentComponent: null, "Error conectar " + e.toString());
28         }
29         return conectar;
30     }
31 }

```

Driver JDBC:

Para poder conectarse a la base de datos de SQL Server es necesario descargar el controlador JDBC, para descargarlo se busca en internet **jbc:sql server**, se ingresa al primer resultado encontrado.



Al ingresar a la página web de Microsoft, se busca el apartado de descargas y se selecciona Microsoft JDBC Driver 12.4 en el formato de .zip, después de descargar el archivo e instalarlo, al ejecutar la línea 24 de la clase conexión, se conecta por medio del controlador a la base de datos en SQL Server

Descarga de Microsoft JDBC Driver para SQL Server

Artículo • 14/11/2023 • 15 colaboradores [Comentarios](#)

En este artículo

- Descargar
- Idiomas disponibles
- Uso del controlador JDBC con Maven Central
- Controladores no compatibles
- Pasos siguientes

Microsoft JDBC Driver para SQL Server es un controlador JDBC de tipo 4 que proporciona conectividad con bases de datos a través de las interfaces de programación de aplicaciones (API) estándar de JDBC disponibles en la plataforma Java. Las descargas del controlador están disponibles para todos los usuarios sin cargo adicional. Proporcionan acceso a SQL Server desde cualquier aplicación Java, servidor de aplicaciones o applet habilitado para Java.

Descargar

La versión 12.4 es la versión de disponibilidad general (GA) más reciente. Admite Java 8, 11, 17 y 20. Si tiene que usar un entorno de ejecución de Java anterior, vea la [matriz de compatibilidad de la especificación de Java y JDBC](#) para comprobar si existe una versión del controlador compatible que puede usar. Estamos trabajando continuamente para mejorar la compatibilidad con la conectividad de Java. Por tanto, se recomienda encarecidamente trabajar con la versión más reciente de Microsoft JDBC Driver.

[Descarga de Microsoft JDBC Driver 12.4 para SQL Server \(zip\) ¹²](#)
[Descarga de Microsoft JDBC Driver 12.4 para SQL Server \(tar.gz\) ¹³](#)

Problemas generados por la conexión:

El primer problema que surgió para conectar la base de datos fue causado por ingresar mal los valores en las variables de la clase, para solucionarlo simplemente se ingresaron los valores correspondientes a las variables dependiendo de los datos de SQL Server.

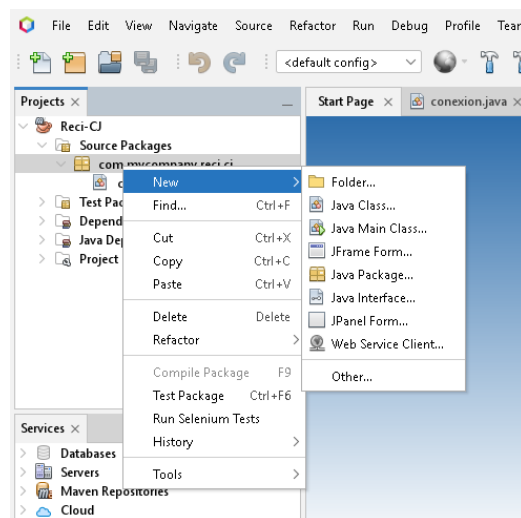
El segundo y último error generado fue más complejo de solucionar, el problema radicaba en que a la hora de intentar realizar la conexión SQL server impedía la conexión por motivos de seguridad de SSL, identificaba que la conexión no era segura porque el certificado de SQL server estaba deshabilitado, para solucionar este error en la línea 24 del código al final se agregó “**trustServerCertificate = true**”, habilitando el certificado de SQL permitiendo realizar la conexión de una manera exitosa.

```
trustServerCertificate = true;"
```

Creación de formularios para la aplicación:

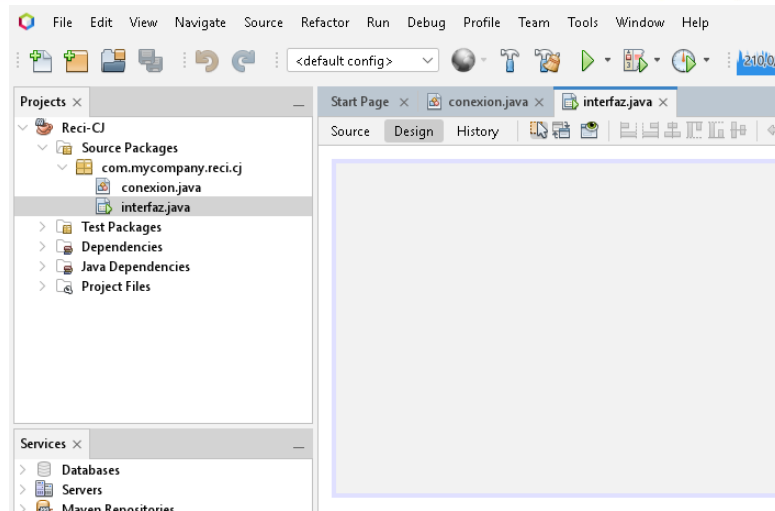
JFrame de la interfaz:

El formulario de interfaz será la ventana principal de la aplicación, para crear formularios en Netbeans, se selecciona el paquete, luego se da clic en la opción **New**, para seleccionar la opción **JFrame Form**.



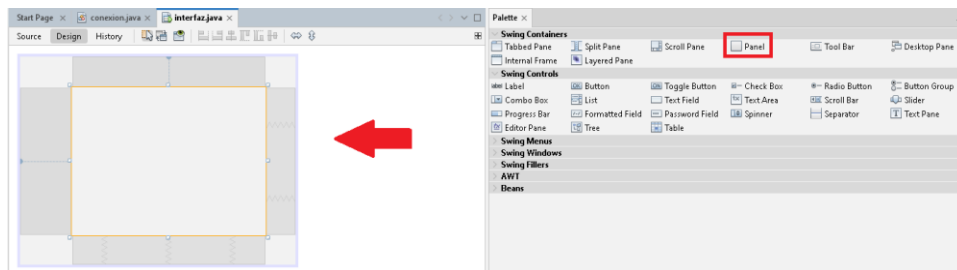
Después de seleccionar la opción se abre una ventana para poder darle nombre al proyecto en este caso se llamará “interfaz”, para la creación de los demás JFrame el procedimiento es el mismo solo cambia su nombre y el diseño de la ventana.

Una vez creado el JFrame, se abre un panel en blanco en el cual poder crear la vista de la ventana de la interfaz, en la cual se pueden crear botones, títulos, agregar color

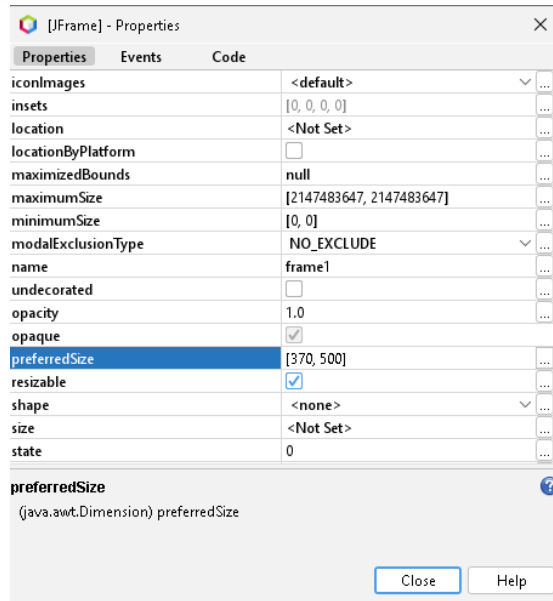


Como primer paso siempre se agregará un Jpanel, sobre el Frame que se crea automáticamente, para poder gestionar y posicionar los elementos como botones y titulo de una mejor manera.

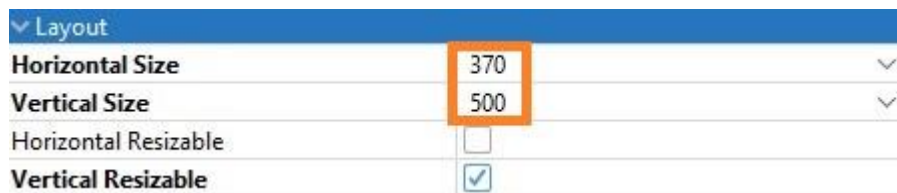
En la herramienta existe una opción llamada **Palette**, en este apartado se encuentran todos los elementos para la creación de la interfaz, para colocar un Panel en el Frame, lo seleccionamos y lo arrastramos al Frame.



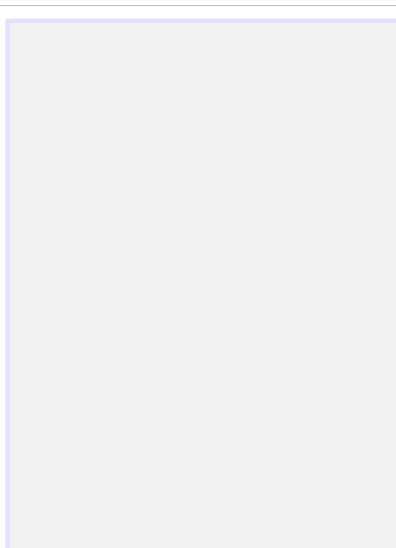
Para cambiar el tamaño del Frame se selecciona el Frame y va a la opción de propiedades, para cambiar el tamaño se modifican la opción de preferredSize



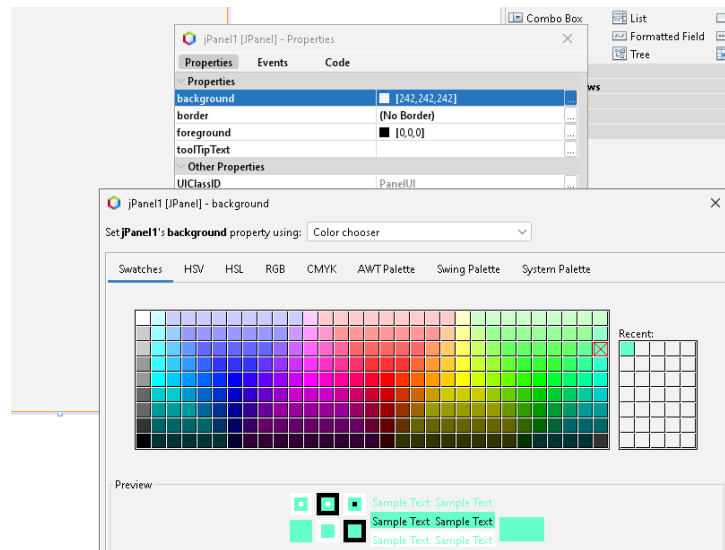
Como el Frame se modificó el JPANEL también se tiene que modificar con los siguientes valores.



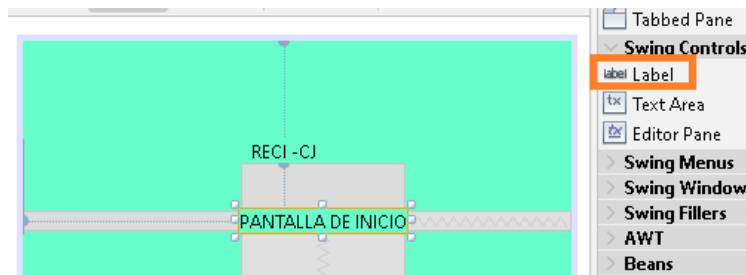
Si se configuro el Frame y el JPanel de la manera indicada se configuran las demás ventanas del aplicativo. El panel se vería de la siguiente manera:



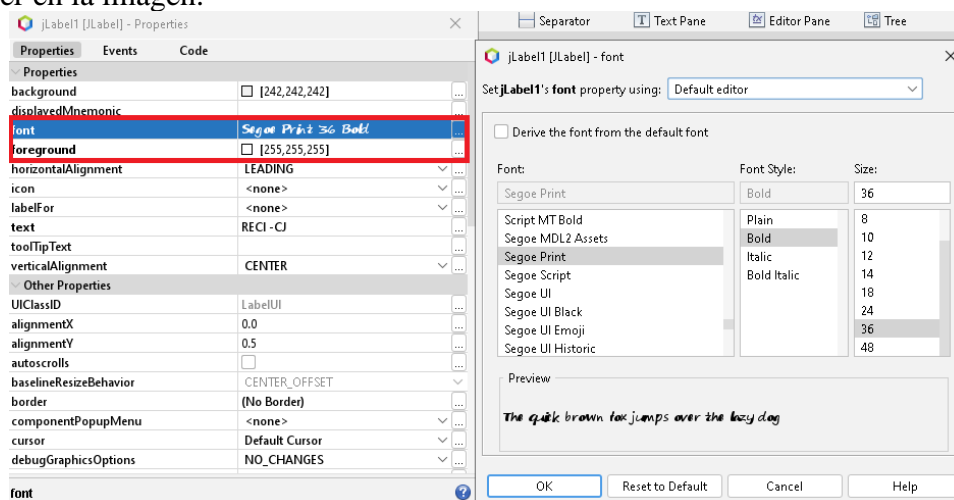
Para cambiar el color del JPanel en configuración en la opción background, se escoge el color para el fondo.



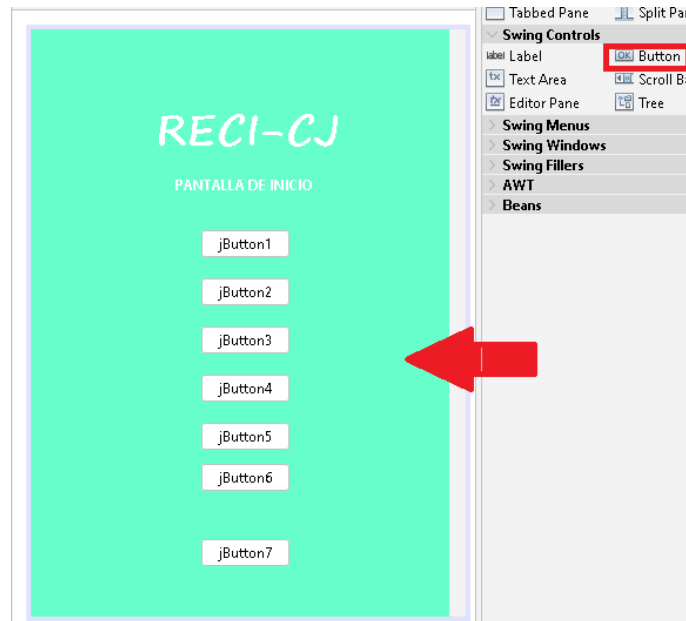
Para colocar Títulos en el panel se usa **Label**, se arrastra al panel y con doble clic se cambia el texto.



Para cambiar el tamaño, tipo de letra y color se selecciona el **Jpanel** y se cambia como se puede ver en la imagen.




Después de modificar los títulos se procede a agregar los botones de la interfaz principal.



Como se pudo apreciar se contarán con 7 botones, Realizar Compra, Realizar Venta, Realizar Devolución, Bases de Datos, Añadir Material, Generar Informe y Salir, estos botones tendrán un tipo de letra con su respectivo color y un tamaño de botón, estas configuraciones son iguales como las anteriores realizadas per enfocadas en los botones, la implementación de los botones quedaría de la siguiente manera.

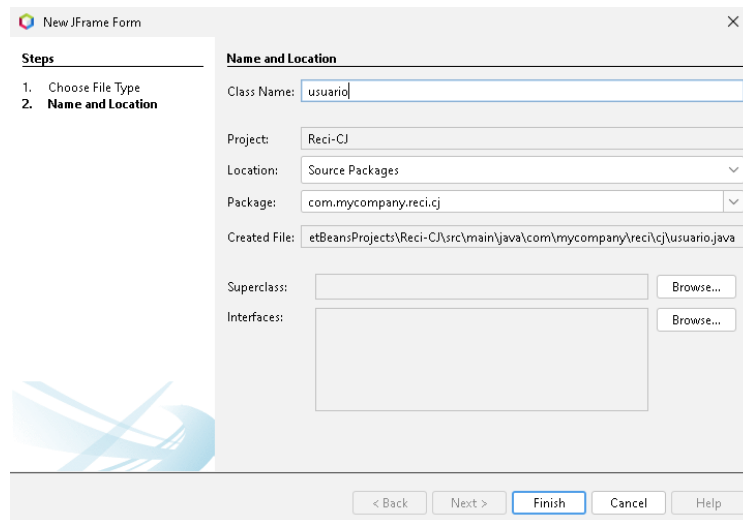


El tipo de letra y color usado para los botones fue el siguiente:

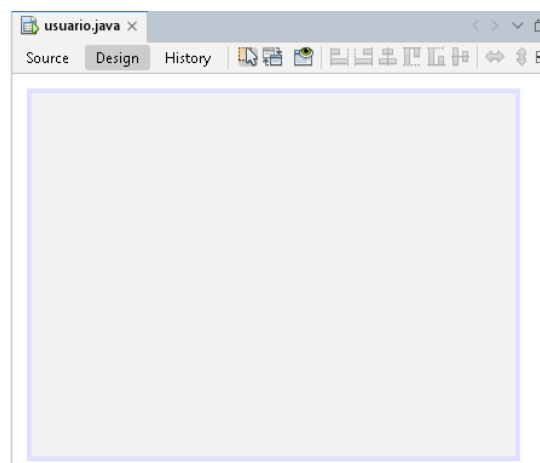
font	Segoe Script 12 Plain	...
foreground	 [0,0,255]	...

JFrame de usuario:

Para la creación del JFrame se crea desde el paquete del proyecto y se le asigna un nombre en este caso se nombró como “usuario”



Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón **“REALIZAR COMPRA”**



Para la creación de la interfaz de realizar compra se usaron 11 JLabel para los títulos, 9 Text Field para ingresar valores y 3 botones.

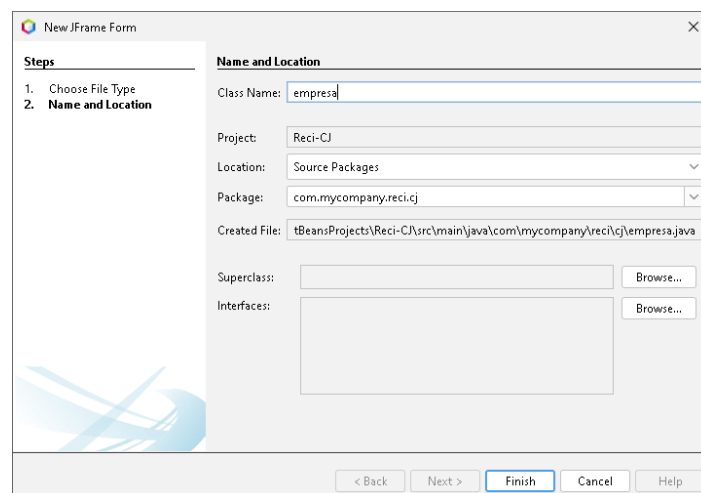
REALIZAR COMPRA
INGRESAR LOS SIGUIENTES DATOS

Nombre	<input type="text"/>
Apellidos	<input type="text"/>
Telefono	<input type="text"/>
Direccion	<input type="text"/>
Tipo Material	<input type="text"/>
Nombre Material	<input type="text"/>
Cantidad	<input type="text"/>
Precio	<input type="text"/>
Fecha	<input type="text"/>

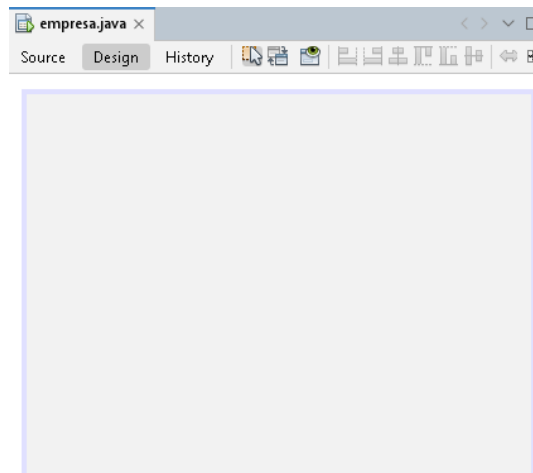
GUARDAR **LIMPIAR CAMPOS** **VOLVER**

JFrame de empresa:

Para la creación del JFrame se crea desde el paquete del proyecto y se le asigna un nombre en este caso se nombró como “empresa”



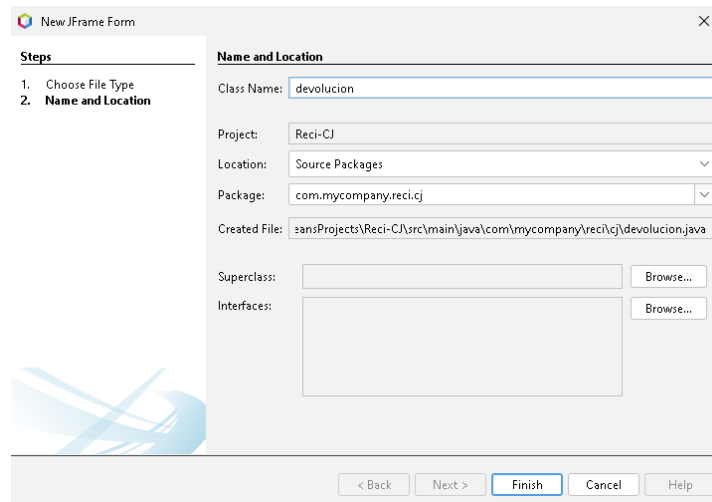
Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón **“REALIZAR VENTA”**



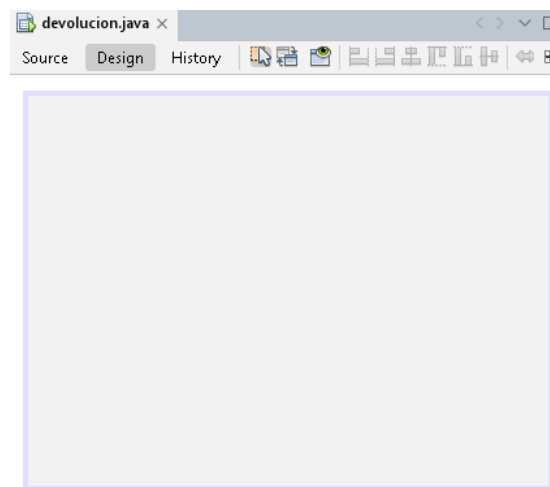
Para la creación de la interfaz de realizar venta se usaron 14 JLabel para los títulos y subtítulos, se crearon 9 Text Field para ingresar valores, se añadió un Radio button para la parte de Estado de pago y 3 botones.

JFrame de devolucion:

Para la creación del JFrame se crea desde el paquete del proyecto y se le asigna un nombre en este caso se nombró como “devolucion”



Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón **“REALIZAR DEVOLUCION”**



Para la creación de la interfaz solicitar devolucion compra se usaron 3 JLabel, el primero se usó para poder escribir las indicaciones de la sección de devoluciones, los otros dos se usaron para la parte de Nombre de Empresa y Resultado, por otro lado, se usaron 2 Text Field y dos botones

SOLICITAR DEVOLUCIÓN

Llenar el siguiente formulario para solicitar una devolución de un material. Además, si la devolución fue aprobada o no, se mostrará en el campo de resultado, hay que tener en cuenta que si la empresa esta pendiente, se podrá aceptar la devolución, de lo contrario será rechazada.

Nombre Empresa

Resultado

SOLICITAR DEVOLUCIÓN **VOLVER**

JFrame de baseDeDatos:

Para la creación del JFrame se crea desde el paquete del proyecto y se le asigna un nombre en este caso se nombró como “baseDeDatos”

New JFrame Form

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

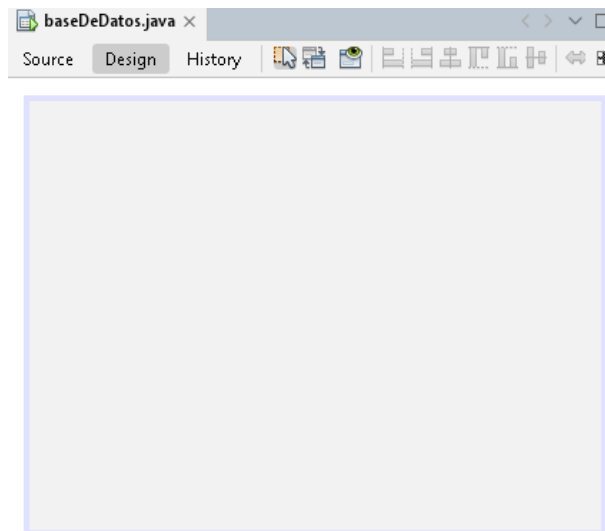
Created File:

Superclass:

Interfaces:

< Back Next > **Finish** Cancel Help

Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón “**BASES DE DATOS**”

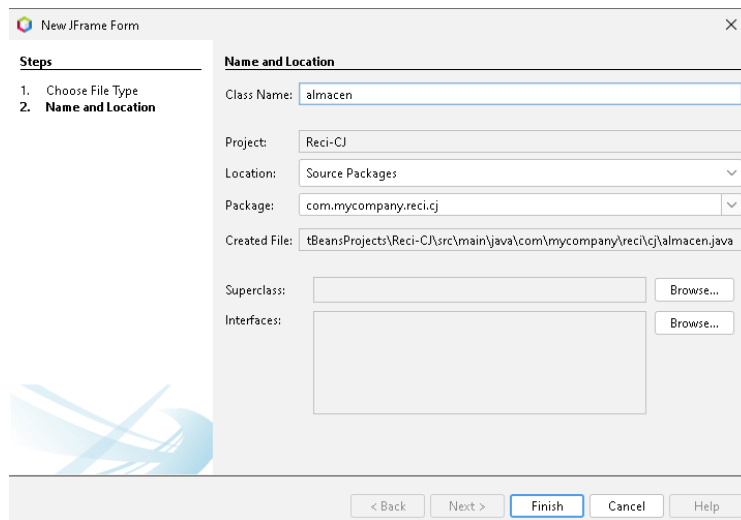


Para la creación de la interfaz de bases de datos se usaron 2 JLabel, el primero se usó para poder escribir las indicaciones de la sección y el segundo para el título, por otro lado, se crearon 4 botones utilizados para llamar a las tablas de la base de datos.

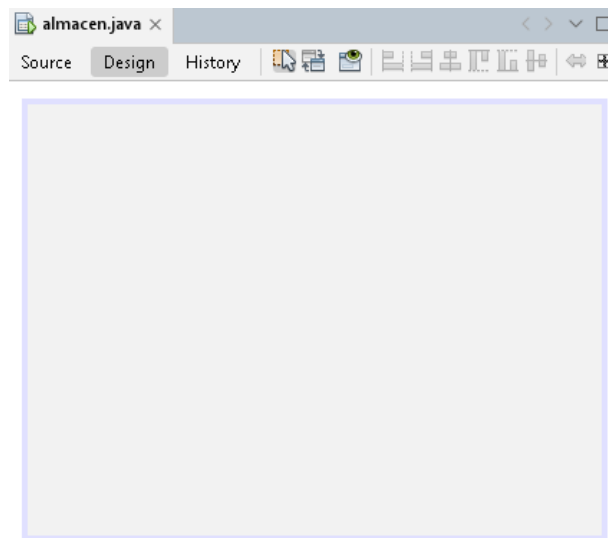


JFrame de almacen:

Para la creación del JFrame se crea desde el paquete del proyecto y se le asigna un nombre en este caso se nombró como “almacen”



Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón **“BASES DE DATOS”**



Para la creación de la interfaz de nuevo material se usaron 7 JLabel, se usaron 5 Text Field y tres botones.

NUEVO MATERIAL

En esta opción, se puede añadir un nuevo material a la tabla de almacen, el cual evaluara si el material existe se sumara la cantidad que se ingresa y si no existe se añade un nuevo registro.

Tipo Material

Nombre Material

Cantidad

Peso

Valor (kg)

GUARDAR **LIMPIAR CAMPOS** **VOLVER**

JFrame de informe:

New JFrame Form

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

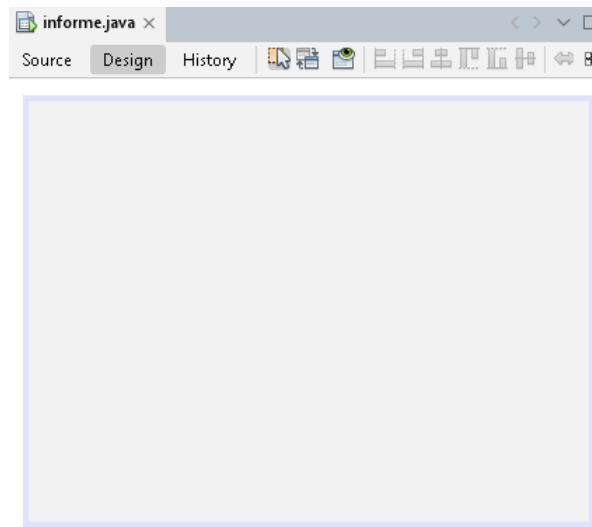
Created File:

Superclass:

Interfaces:

< Back Next > **Finish** Cancel Help

Una vez creado el archivo se abre una ventana con el JFrame principal en el cual comenzar a añadir los elementos de la sesión del botón **“GENERAR INFORME”**



Para la creación de la interfaz de Informe se usaron 7 JLabel, se usaron 5 Text Field y 1 botón para salir de la ventana.

A screenshot of the 'INFORME' application window. The window has a teal background. At the top, the word 'INFORME' is displayed in large, white, serif capital letters. Below it, the subtitle 'Generar un informe de las bases de datos' is written in a smaller, italicized, dark blue font. The main area contains five labels in italicized dark blue font, each followed by a white text input field: 'Tipo de material con mayor cantidad:', 'Promedio de los materiales de acuerdo al peso:', 'Suma de todos los material del almacén:', 'Usuario con mayor material vendido:', and 'Empresa con mayor material comprado:'. At the bottom, there is a blue rectangular button with the word 'SALIR' in white capital letters.

Explicación de la funcionalidad de aplicativo

Interfaz del panel principal del aplicativo RECI – CJ:

Para la creación de la interfaz principal se utilizó un Frame, en el cual contener todos los elementos necesarios, se creó un JPanel para colocar los JLabel con los textos y los botones que realizan alguna función dentro del aplicativo



Botón Realizar Compra

```
//Metodo de boton REALIZAR COMPRA
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase usuario, para la ventana de "Realizar Compra"
    usuario Uventa = new usuario();
    //Usar setVisible para hacer visible la ventana
    Uventa.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Realizar Venta:

```
//Metodo de boton REALIZAR VENTA
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase empresa, para la ventana de "Realizar Venta"
    empresa Cempresa = new empresa();
    //Usar setVisible para hacer visible la ventana
    Cempresa.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Realizar Devolución:

```
//Metodo de boton REALIZAR DEVOLUCION
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase devolucion, para la ventana de "Realizar Devolucion"
    devolucion d = new devolucion();
    //Usar setVisible para hacer visible la ventana
    d.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Bases de Datos:

```
//Metodo de boton BASES DE DATOS
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase baseDeDatos, para la ventana de "Bases de Datos"
    baseDeDatos bd = new baseDeDatos();
    //Usar setVisible para hacer visible la ventana
    bd.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Añadir Material:

```
//Metodo de boton AÑADIR MATERIAL
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase almacen, para la ventana de "Añadir Material"
    almacen a = new almacen();
    //Usar setVisible para hacer visible la ventana
    a.setVisible(b:true);
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Generar Informe:

```
//Metodo de boton GENERAR INFORME
private void btnInformeActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase informe, para la ventana de "Generar Informe"
    informe i = new informe();
    //Usar setVisible para hacer visible la ventana
    i.setVisible(b:true);
    //Se usa la instancia i para llamar al método "TipoMaterialMayor()"
    i.TipoMaterialMayor();
    //Se usa la instancia i para llamar al método "promedioMaterialesPeso()"
    i.promedioMaterialesPeso();
    //Se usa la instancia i para llamar al método "maximoUsuario()"
    i.maximoUsuario();
    //Se usa la instancia i para llamar al método "sumaMateriales()"
    i.sumaMateriales();
    //Se usa la instancia i para llamar al método "maximoEmpresa()"
    i.maximoEmpresa();
    //Se usa el método dispose para cerrar la ventana actual, la ventana de Interfaz
    this.dispose();
}
```

Botón Salir:

```
//Metodo de boton SALIR
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // Cerrar la aplicación al presionar el botón
    System.exit(status: 0);
}
```

Interfaz de realizar Compra:

En la interfaz de realizar compra se utilizaron un JPanel que será el soporte para colocar los elementos, 11 JLabel que se utilizaron para el título, descripción y nombre de los campos, 9 JTextfield donde el usuario ingresara los datos y 3 botones donde tiene las funciones de guardar datos limpiar los campos y volver al menú.

RECI-CJ (REALIZAR COMPRA)

REALIZAR COMPRA

INGRESAR LOS SIGUIENTES DATOS

Nombre

Apellidos

Telefono

Direccion

Tipo Material

Nombre Material

Cantidad

Precio

Fecha

Botón Guardar:

```
//Metodo de GUARDAR
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    //Se crea una instancia de la clase conexion, usada para establecer la
    //conexion con la base de datos
    conexion conn = new conexion();
    //Se crea una variable de tipo String llamada "Nombre", se usa para almacenar
    //los valores ingresados en los campos de texto de la interfaz
    String Nombre = txtNombre.getText();
    //Se crea una variable "Apellidos", para almacenar los valores ingresados
    String Apellidos = txtApellidos.getText();
    //Se crea una variable "Telefono", para almacenar los valores ingresados
    String Telefono = txtTelefono.getText();
    //Se crea una variable "Direccion", para almacenar los valores ingresados
    String Direccion = txtDireccion.getText();
    //Se crea una variable "txtNombre_Material", para almacenar los valores ingresados
    String Nombre_Material = txtNombre_Material.getText();
    //Se crea una variable de tipo int "Cantidad_Material", se usa Integer.parseInt
    //para capturar un dato de tipo String en una variable de tipo entero
    int Cantidad_Material = Integer.parseInt(txtCantidad_Material.getText());
    try{
        //Se establece conexion con la base de datos
        Connection con = conn.establecerConexion();
        //Se prepara la consulta SQL para insertar datos en la tabla usuario
        PreparedStatement ps = con.prepareStatement("INSERT INTO usuario (Nombre, Apellidos, Telefono, Direccion, Nombre_Material, Cantidad_Material)"
            + " VALUES (?, ?, ?, ?, ?, ?)");
    }
```

```
//Se establecen los valores de los parametros en la consula SQL
//los parametros son las variables anteriormente creadas
ps.setString( parameterIndex: 1, x: Nombre);
ps.setString( parameterIndex: 2, x: Apellidos);
ps.setString( parameterIndex: 3, x: Telefono);
ps.setString( parameterIndex: 4, x: Direccion);
ps.setString( parameterIndex: 5, x: Nombre_Material);
ps.setInt( parameterIndex: 6, x: Cantidad_Material);
//Se ejecuta la actualizacion en la base de datos, insertando un
// nuevo registro de una nueva compra
ps.executeUpdate();

//Se crea una instancia de la clase "facturacion", para mostrar una factura
facturacion f = new facturacion();

//Se establecen los valores en la instancia facturacion, para mostrar
//informacion sobre la compra, como el nombre, apellidos, telefono entre otros
f.setNombre( Nombre: txtNombre.getText());
f.setApellido( Apellido: txtApellidos.getText());
f.setTelefono( Telefono: txtTelefono.getText());
f.setTipoMaterial( Tipo_Material: txtTipo_Material.getText());
f.setNombreMaterial( NombreMaterial: txtNombre_Material.getText());
f.setCantidadMaterial( Cantidad_Material: txtCantidad_Material.getText());
f.setfecha( fecha: txtFecha.getText());
f.setprecio( precio: txtPrecio.getText());
f.settotal( total: txtPrecio.getText());
//Se hace visible la ventana de facturacion
f.setVisible( b: true);
} catch( SQLException e) {
    //Se envia un mensaje si ocurrio algun error
    JOptionPane.showMessageDialog( parentComponent: null, message: "Registro no guardado");
}
}
```

Botón Limpiar Campos:

```
//Metodo de LIMPIAR CAMPOS
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    //Limpiar los campos a su valor predeterminado
    // Limpiar el campo de texto del nombre
    txtNombre.setText( "");
    // Limpiar el campo de texto de los apellidos
    txtApellidos.setText( "");
    // Limpiar el campo de texto del teléfono
    txtTelefono.setText( "");
    // Limpiar el campo de texto de la dirección
    txtDireccion.setText( "");
    // Limpiar el campo de texto del tipo de material
    txtTipo_Material.setText( "");
    // Limpiar el campo de texto del nombre del material
    txtNombre_Material.setText( "");
    // Limpiar el campo de texto de la cantidad de material
    txtCantidad_Material.setText( "");
    // Limpiar el campo de texto del precio
    txtPrecio.setText( "");
    // Limpiar el campo de texto de la fecha
    txtFecha.setText( "");
}
```

Interfaz de realizar Venta:

En la interfaz de realizar venta se utilizaron un JPanel que será el soporte para colocar los elementos, 12 JLabel que se utilizaron para el título, descripción y nombre de los campos, 9 JTextfield donde el usuario ingresara los datos, 1 JRadius para especificar si la venta está en estado de pendiente o éxito y 3 botones donde tiene las funciones de guardar datos limpiar los campos y volver al menú.



The screenshot shows a Java Swing window titled "RECI-CJ (REALIZAR VENTA)". The window has a light blue header and a white body. The title "REALIZAR VENTA" is centered in bold black text, with the subtitle "INGRESAR LOS SIGUIENTES DATOS" below it. The form contains the following fields and controls:

- Nombre empresa:** A text input field.
- Dirección:** A text input field.
- Telefono:** A text input field.
- Correo:** A text input field.
- Tipo Material:** A text input field.
- Nombre Material:** A text input field.
- Cantidad Material:** A text input field.
- Precio:** A text input field.
- Estado de pago:** Two radio buttons labeled "exito" and "pendiente".
- Fecha:** A text input field.

At the bottom of the form are three blue buttons with white text: "GUARDAR", "LIMPIAR CAMPOS", and "VOLVER".

Botón Guardar:

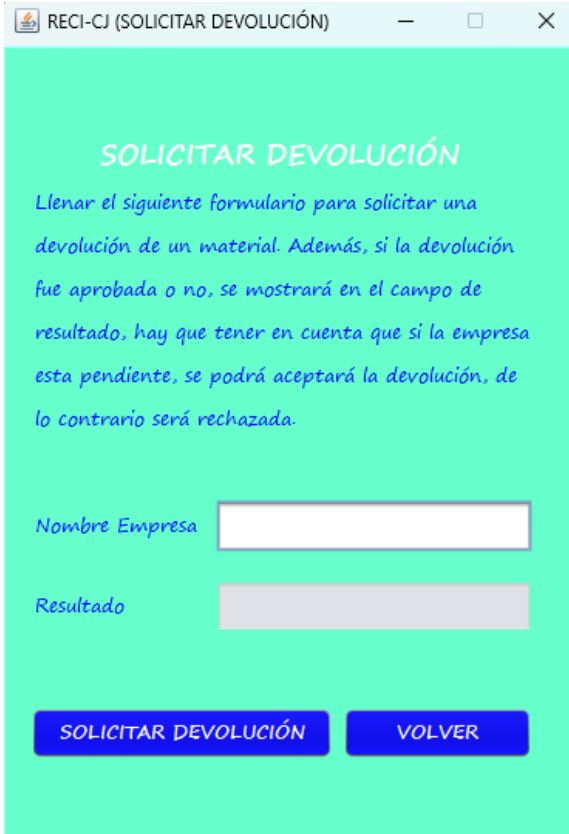
```
//Metodo del boton GUARDAR
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    //Se crea una instancia de la clase conexion, usada para establecer la
    //conexion con la base de datos
    conexion conn = new conexion();
    //Se crea una variable "Nombre_Empresa", para almacenar los valores ingresados
    String Nombre_Empresa = txtNombre_Empresa.getText();
    //Se crea una variable "Direccion", para almacenar los valores ingresados
    String Direccion = txtDireccion.getText();
    //Se crea una variable "Telefono", para almacenar los valores ingresados
    String Telefono = txtTelefono.getText();
    //Se crea una variable "Correo", para almacenar los valores ingresados
    String Correo = txtCorreo.getText();
    //Se crea una variable "Nombre_Material", para almacenar los valores ingresados
    String Nombre_Material = txtNombre_Material.getText();
    //Se crea una variable de tipo int "Cantidad_Material", se usa Integer.parseInt
    //para capturar un dato de tipo String en una variable de tipo entero
    int Cantidad_Material = Integer.parseInt(txtCantidad_Material.getText());
    //Se crea una variable para guardar la eleccion del Radio Button.
    String pagos;
    //Se crea una condicion IF, para verificar que opcion a seleccionado el usuario
    if(rbExito.isSelected() == true){
        //Si se selecciono el campo de exito, la variable de pago toma el valor de "exito"
        pagos = "exito";
    }
    else if(rbPendiente.isSelected() == true){
        //Si se selecciono el campo de pendiente, la variable de pago toma el valor de "pendiente"
        pagos = "pendiente";
    }
    else{
        //Se crea una condicion else, si el usuario no selecciona ningun valor la variable toma "no hay"
        pagos = "no hay";
    }
    try{
        //Se establece conexion con la base de datos
        Connection con = conn.establecerConexion();
        //Se prepara la consulta SQL para insertar datos en la tabla empresa
        PreparedStatement ps = con.prepareStatement("INSERT INTO empresa (Nombre_Empresa, Direccion, Telefono, Correo, "
            + "Nombre_Material, Cantidad_Material, Pagos) VALUES (?, ?, ?, ?, ?, ?, ?)");
        //Se establecen los valores de los parametros en la consulta SQL
        //los parametros son las variables anteriormente creadas
        ps.setString(parameterIndex: 1, x: Nombre_Empresa);
        ps.setString(parameterIndex: 2, x: Direccion);
        ps.setString(parameterIndex: 3, x: Telefono);
        ps.setString(parameterIndex: 4, x: Correo);
        ps.setString(parameterIndex: 5, x: Nombre_Material);
        ps.setInt(parameterIndex: 6, x: Cantidad_Material);
        ps.setString(parameterIndex: 7, x: pagos);
        //Se ejecuta la actualizacion en la base de datos, insertando un
        // nuevo registro de una nueva venta
        ps.executeUpdate();
        //Se crea una instancia de la clase "facturaVenta", para mostrar una factura
        facturaVenta f = new facturaVenta();
        //Se establecen los valores en la instancia facturaVenta, para mostrar
        //informacion sobre el NombreEmpresa,Direccion,Telefono,TipoMaterial entre otros
        f.setNombreEmpresa( Nombre: txtNombre_Empresa.getText());
        f.setDireccion( Direccion: txtDireccion.getText());
        f.setTelefono( Telefono: txtTelefono.getText());
        f.setTipoMaterial( TipoMaterial: txtTipo_Material.getText());
        f.setNombreMaterial( NombreMaterial: txtNombre_Material.getText());
        f.setCantidadMaterial( CantidadMaterial: txtCantidad_Material.getText());
        f.setPrecio( precio: txtPrecio.getText());
        f.setFecha( fecha: txtFecha.getText());
        f.setPrecioTotal( preciot: txtPrecio.getText());
        //Se hace visible la ventana de facturacion
        f.setVisible( b: true);
        //Se usa el metodo para cerrar la ventana de RealizarVenta
        this.dispose();
    } catch(SQLException e){
        //Se envia un mensaje si ocurrio algun error
        JOptionPane.showMessageDialog( parentComponent: null, message: "Registro no guardado");
    }
}
```

Botón Limpiar Campos:

```
//Metodo de LIMPIAR CAMPOS
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    //Limpiar los campos a su valor predeterminado
    // Limpiar el campo de texto de Nombre_Empresa
    txtNombre_Empresa.setText("");
    // Limpiar el campo de texto de la Direccion
    txtDireccion.setText("");
    // Limpiar el campo de texto del numero de Telefono
    txtTelefono.setText("");
    // Limpiar el campo de texto del Correo
    txtCorreo.setText("");
    // Limpiar el campo de texto de Tipo_Material
    txtTipo_Material.setText("");
    // Limpiar el campo de texto de Nombre_Material
    txtNombre_Material.setText("");
    // Limpiar el campo de texto de Cantidad_Material
    txtCantidad_Material.setText("");
    // Limpiar el campo de texto de Precio
    txtPrecio.setText("");
    // Limpiar el campo de texto de Fecha
    txtFecha.setText("");
}
```

Interfaz de Solicitar Devolución:

En la interfaz de solicitar devolución se utilizó un JPanel donde se colocarán los elementos, 9 JLabel para el título, descripción y nombre de los campos, 2 Jtextfield para ingresar el nombre de la empresa que se quiere hacer una devolución y otro para el resultado por último 2 botones para volver al menú y solicitar la devolución.



RECI-CJ (SOLICITAR DEVOLUCIÓN)

SOLICITAR DEVOLUCIÓN

Llenar el siguiente formulario para solicitar una devolución de un material. Además, si la devolución fue aprobada o no, se mostrará en el campo de resultado, hay que tener en cuenta que si la empresa esta pendiente, se podrá aceptar la devolución, de lo contrario será rechazada.

Nombre Empresa

Resultado

Botón Solicitar Devolución:

Este botón realiza la función de tomar el valor ingresado en el “txtNombreEmpresa” y llamar el método creado “obtenerResultadoYInsertar”.

```
//Metodo de boton Solicitar devolucion
private void btnSolicitarActionPerformed(java.awt.event.ActionEvent evt) {
    //Se crea una variable de tipo String para almacenar el valor del campo
    String nombreEmpresa = txtNombreEmpresa.getText();

    //Despues de obtener el valor del campo de la interfaz se llama al metodo
    //obtenerResultadoYInsertar para realizar la inserción y obtener el resultado
    obtenerResultadoYInsertar(nombreEmpresa);
}
```

Crear Método “obtenerResultadoYInsertar”:

Este método creado en la parte de la interfaz de devolución tiene como función insertar datos en la tabla de devoluciones que se encuentra en la base de datos de SQL server, de igual forma se trae la información que se encuentra en la tabla de devoluciones.

```
//Metodo para crear un nuevo registro en la tabla devolucion
private void obtenerResultadoYInsertar(String nombreEmpresa) {
    //Se crea una instancia de la clase conexion, usada para establecer la
    //conexion con la base de datos
    conexion conn = new conexion();
    try {
        //Se establece conexion con la base de datos
        Connection con = conn.establecerConexion();
        //Se prepara la consulta SQL para insertar un nuevo registro en la tabla devoluciones
        PreparedStatement ps = con.prepareStatement("INSERT INTO devoluciones (nombre_Empresa) VALUES (?)");
        //Se establecen los valores de los parametros en la consula SQL
        //los parametros en la primera posicion de la variable String nombreEmpresa{
        //variable brinda por el metodo obtenerResultado
        ps.setString(parameterIndex: 1, s: nombreEmpresa);
        //Se ejecuta la actualizacion en la base de datos, insertando un
        // nuevo registro de una nueva devolucion
        ps.executeUpdate();

        //Se crea una variable resultado, en la cual se almacena por medio de la instancia de conexion
        //el metodo obtenerResultado de la clase conexion, a este metodo se le pasa como parametro
        //la variable nombreEmpresa el cual contiene el nombre de la empresa ingresado por el usuario
        String resultado = conn.obtenerResultado(nombreEmpresa);
        //en el JTextField se agrega el parametro de resultado, obtenido del metodo obtenerResultado
        // de la clase conexion
        txtResultado.setText(s: resultado);
    } catch (SQLException e) {
        //Se envia un mensaje si ocurrio algun error
        JOptionPane.showMessageDialog(parentComponent: null, message: "Registro no guardado");
    }
}
```

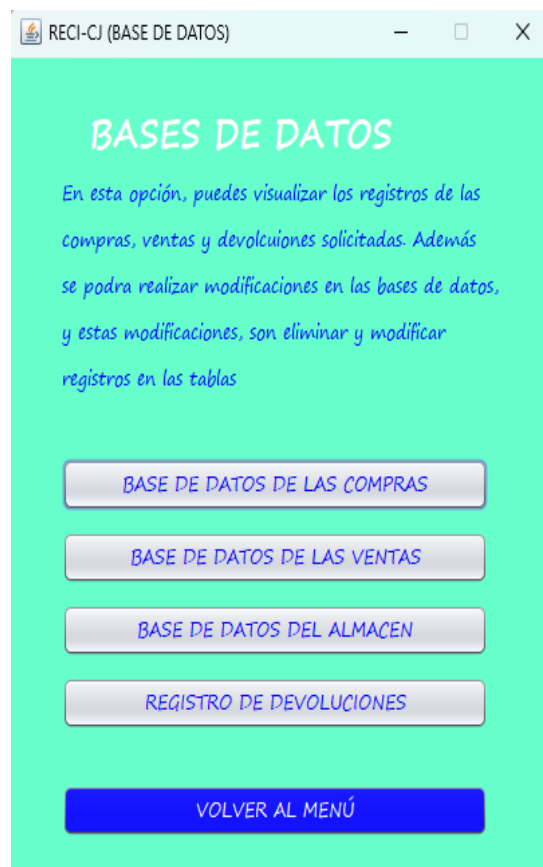
Método obtenerResultado de la Clase Conexión:

Este método tiene como función obtener el resultado en un campo de la tabla devoluciones de la base de datos de SQL server, además este método es llamado en “obtenerResultadoYInsertar” que se encuentra en la interfaz de devoluciones.

```
//Metodo obtenerResultado
//Se pasa como parametro del metodo una variable de tipo String nombreEmpresa
public String obtenerResultado(String nombreEmpresa) {
    //Se crea una variable de tipo String
    String resultado = "";
    try {
        //Se establece conexion con la base de datos
        Connection con = establecerConexion();
        // Se define la consulta SQL para obtener el estado de las devoluciones de la empresa especificada.
        String sql = "SELECT estado FROM devoluciones WHERE nombre_Empresa = ?";
        try (PreparedStatement ps = con.prepareStatement(sql)) {
            // Se establece el valor del parámetro en la consulta SQL.
            ps.setString(parameterIndex: 1, s: nombreEmpresa);
            // Se ejecuta la consulta y se obtiene el conjunto de resultados.
            try (ResultSet rs = ps.executeQuery()) {
                // Si hay resultados, se asigna el valor del estado a la variable resultado.
                if (rs.next()) {
                    resultado = rs.getString(columnLabel: "estado");
                } else {
                    // Si no hay resultados, se asigna un mensaje indicando que no se encontraron resultados.
                    resultado = "No se encontraron resultados";
                }
            }
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se imprime la traza de la excepción y se asigna un mensaje de error a la variable resultado.
        e.printStackTrace();
        resultado = "Error al obtener el resultado";
    }
}

// Se devuelve el resultado (estado de las devoluciones o mensaje de error).
return resultado;
}
```

Interfaz de Bases de Datos:



Botón Base de datos de las compras:

```
//Metodo de Base de datos de las Compras
private void btnUsuarioActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase tblUsuario, para la ventana de "Datos de las Compras"
    tblUsuario tUsuario = new tblUsuario();
    //Usar setVisible para hacer visible la ventana
    tUsuario.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual.
    this.dispose();
}
```

Boton Base de datos de las ventas:

```
//Metodo de Base de datos de las Ventas
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase tblEmpresa, para la ventana de "Datos de las Ventas"
    tblEmpresa tEmpresa = new tblEmpresa();
    //Usar setVisible para hacer visible la ventana
    tEmpresa.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual.
    this.dispose();
}
```

Botón Base de datos del almacén:

```
//Metodo de Base de datos del Almacen
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase tblAlmacen, para la ventana de "Datos del Almacen"
    tblAlmacen tAlmacen = new tblAlmacen();
    //Usar setVisible para hacer visible la ventana
    tAlmacen.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual.
    this.dispose();
}
```

Botón Registro de devoluciones:

```
//Metodo de Registro de Devoluciones
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    //Se instancia un objeto de la clase tblDevoluciones para la ventana de "Registro de Devoluciones"
    tblDevoluciones d = new tblDevoluciones();
    //Usar setVisible para hacer visible la ventana
    d.setVisible(b: true);
    //Se usa el método dispose para cerrar la ventana actual.
    this.dispose();
}
```


Interfaz de Tabla Usuario:



id_Usuarios	Nombre	Apellidos	Telefono	Direccion	Nombre_Material	Cantidad_Material
-------------	--------	-----------	----------	-----------	-----------------	-------------------

ACTUALIZAR TAB... ELIMINAR REGISTRO VOLVER

Botón Actualizar Tabla:



id_Usuarios	Nombre	Apellidos	Telefono	Direccion	Nombre_Material	Cantidad_Material
1	Wilson	Gonzales Carvajal	3115874869	Cll 22 # 22-1	Botellas	200
2	Carlos	Mahecha Gomez	3165852808	Cll 52 # 3-2	Chatarra	25
3	Samanta	Perez Paez	3204587621	Cll 45 # 4-6	Cajas	78
4	Jonatan	Igua Contreras	3165475878	Cll 25 # 9-6	Alambre	150
7	fdcf	hfgsdhfg	34324	DFSDG	FSDGS	24

ACTUALIZAR TAB... ELIMINAR REGISTRO VOLVER

Interfaz de Tabla Empresa:



id_empresa	Nombre_Emp...	Direccion	Telefono	Correo	Nombre_Mate...	Cantidad_Mat...	Pagos
------------	---------------	-----------	----------	--------	----------------	-----------------	-------

ACTUALIZAR TABLA ELIMINAR REGISTRO VOLVER

Botón Actualizar Tabla:

RECI-CJ (BD-VENTAS)

id_empresa	Nombre_Emp...	Direccion	Telefono	Correo	Nombre_Mate...	Cantidad_Mat...	Pagos
1	Clis S.A	Cll 128 # 2-8	3125489012	Clis@gmail.co...	Papel	301	exito
2	ReciclaBogota	Cll 1 # 32-1	3142596742	reciclabogota...	Plastico	54	pendiente
3	Reutilis	Cll 12 # 20-1	3152459874	reutilis@gmail...	Carton	400	exito
4	Si Al Reciclaje	Cll 32 # 12-6	3205486974	sireciclaje@g...	Aluminio	467	exito
5	Limes	Cll 20 # 20 -7	3154569874	limes@gmail....	Papel	214	pendiente

ACTUALIZAR TABLA ELIMINAR REGISTRO VOLVER

Interfaz de Tabla Almacén:

RECI-CJ (BD-ALMACEN)

id_almacen	Tipo_Material	Nombre_Material	Cantidad_Material	Peso	Valor_kg
------------	---------------	-----------------	-------------------	------	----------

ACTUALIZAR TABLA ELIMINAR REGISTRO VOLVER

Botón Actualizar Tabla:

RECI-CJ (BD-ALMACEN)

id_almacen	Tipo_Material	Nombre_Material	Cantidad_Material	Peso	Valor_kg
1	Plastico	Botellas	465	4	0,47
2	Carton	Cajas	70	72	20.500
3	Cobre	Alambre	1	0,5	34,702
4	Aluminio	Latas	467	10,5	5,2
7	Vidrio	Ventanas	50	90	22.000
8	Chatarra	Laminas	60	22.000	200

ACTUALIZAR TABLA ELIMINAR REGISTRO VOLVER

Interfaz de Tabla Devoluciones:

RECI-CJ (BD-DEVOLUCIONES)

id_devolucion	nombre_Empresa	estado
---------------	----------------	--------

ACTUALIZAR TAB... ELIMINAR REGISTRO VOLVER

Interfaz Añadir Material:



Botón Guardar:

```
//Metodo boton Guardar
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    // Se obtienen los valores de los campos de texto en la interfaz gráfica.
    String Tipo_Material = txtTipo_Material.getText();
    String Nombre_Material = txtNombre_Material.getText();
    int Cantidad_Material = Integer.parseInt(txtCantidad_Material.getText());
    float peso = Float.parseFloat(txtPeso.getText());
    float valorkg = Float.parseFloat(txtValorkg.getText());

    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Llamada al procedimiento almacenado de sql server llamado Nuevo_Materiales
        String storedProcedureCall = "{call Nuevo_Materiales(?, ?, ?, ?, ?)}";
        try (CallableStatement callableStatement = con.prepareCall(sql:storedProcedureCall)) {
            // Se establecen los valores de los parámetros del procedimiento almacenado.
            callableStatement.setString(parameterIndex: 1, x: Tipo_Material);
            callableStatement.setString(parameterIndex: 2, x: Nombre_Material);
            callableStatement.setInt(parameterIndex: 3, x: Cantidad_Material);
            callableStatement.setFloat(parameterIndex: 4, x: peso);
            callableStatement.setFloat(parameterIndex: 5, x: valorkg);

            // Ejecuta el procedimiento almacenado
            callableStatement.execute();
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

```
// Se muestra un mensaje de éxito al usuario.  
JOptionPane.showMessageDialog(parentComponent: null, message: "Procedimiento almacenado ejecutado exitosamente.");  
}  
} catch (SQLException e) {  
    // En caso de excepción SQL, se muestra un mensaje de error al usuario y se imprime la traza de la excepción.  
    JOptionPane.showMessageDialog(parentComponent: null, message: "Error al ejecutar el procedimiento almacenado");  
    e.printStackTrace();  
}  
}
```

Botón Limpiar Campos:

```
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {  
    // Se establece el texto de los campos de texto a cadena vacía para limpiar la interfaz.  
    txtTipo_Material.setText("");  
    txtNombre_Material.setText("");  
    txtCantidad_Material.setText("");  
    txtPeso.setText("");  
    txtValorKg.setText("");  
}
```

Interfaz Generar Informe:

RECICAJ (INFORME)

INFORME

Generar un informe de las bases de datos

Tipo de material con mayor cantidad:

Promedio de los materiales de acuerdo al peso:

Suma de todos los material del almacén:

Usuario con mayor material vendido:

Empresa con mayor material comprado:

SALIR

Tipo de material con mayor cantidad:

```
//Metodo de TipoMaterialMayor
public void TipoMaterialMayor() {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Se prepara la consulta SQL para obtener el valor máximo de la columna "cantidad_Material" en la tabla "almacen".
        PreparedStatement ps = con.prepareStatement("SELECT MAX(cantidad_Material) FROM almacen");
        // Se ejecuta la consulta y se obtiene el conjunto de resultados.
        ResultSet rs = ps.executeQuery();
        // Se verifica si hay resultados.
        if (rs.next()) {
            // Se obtiene el valor máximo de la columna "cantidad_Material".
            int maxCantidadMaterial = rs.getInt(columnIndex:1);
            // Se muestra el valor máximo en un componente de la interfaz gráfica (en este caso, txtTipoMAX).
            txtTipoMAX.setText(text: String.valueOf(i:maxCantidadMaterial));
        } else {
            // Si no hay resultados, se muestra un mensaje indicando que no hay datos en la tabla "almacen".
            JOptionPane.showMessageDialog(parentComponent: null, message: "No hay datos en la tabla almacen");
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se muestra un mensaje de error al usuario.
        JOptionPane.showMessageDialog(parentComponent: null, message: "No se pudo ejecutar");
    }
}
```

Promedio de los materiales de acuerdo al peso:

```
//Metodo de promedioMaterialesPeso
public void promedioMaterialesPeso() {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Se prepara la consulta SQL para obtener el promedio de la columna "peso" en la tabla "almacen".
        PreparedStatement ps = con.prepareStatement("SELECT AVG(peso) FROM almacen");
        // Se ejecuta la consulta y se obtiene el conjunto de resultados.
        ResultSet rs = ps.executeQuery();
        // Se verifica si hay resultados.
        if (rs.next()) {
            int promedio = rs.getInt(columnIndex:1);
            // Se muestra el valor promedio en un componente de la interfaz gráfica (en este caso, txtPromedio).
            txtPromedio.setText(text: String.valueOf(i:promedio));
        } else {
            // Si no hay resultados, se muestra un mensaje indicando que no hay datos en la tabla "almacen".
            JOptionPane.showMessageDialog(parentComponent: null, message: "No hay datos en la tabla almacen");
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se muestra un mensaje de error al usuario.
        JOptionPane.showMessageDialog(parentComponent: null, message: "No se pudo ejecutar");
    }
}
```

Suma de todos los materiales del almacén:

```
//Metodo sumaMateriales
public void sumaMateriales() {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Se prepara la consulta SQL para obtener la suma de la columna "cantidad_Material" en la tabla "almacen".
        PreparedStatement ps = con.prepareStatement("SELECT SUM(cantidad_Material) FROM almacen");
        // Se ejecuta la consulta y se obtiene el conjunto de resultados.
        ResultSet rs = ps.executeQuery();
        // Se verifica si hay resultados.
        if (rs.next()) {
            // Se obtiene el valor de la suma de la columna "cantidad_Material".
            int suma = rs.getInt(columnIndex:1);
            // Se muestra el valor de la suma en un componente de la interfaz gráfica (en este caso, txtSuma).
            txtSuma.setText(text:String.valueOf(i:suma));
        } else {
            // Si no hay resultados, se muestra un mensaje indicando que no hay datos en la tabla "almacen".
            JOptionPane.showMessageDialog(parentComponent:null, message:"No hay datos en la tabla almacen");
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se muestra un mensaje de error al usuario.
        JOptionPane.showMessageDialog(parentComponent:null, message:"No se pudo ejecutar");
    }
}
```

Usuario con mayor material vendido:

```
//Metodo maximoUsuario (Usuario con mayor material vendido)
public void maximoUsuario() {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Se prepara la consulta SQL para obtener el nombre del usuario con la máxima cantidad de materiales.
        PreparedStatement ps = con.prepareStatement("SELECT Nombre FROM usuario WHERE cantidad_Material = (SELECT MAX(cantidad_Material) FROM usuario)");
        // Se ejecuta la consulta y se obtiene el conjunto de resultados.
        ResultSet rs = ps.executeQuery();
        // Se verifica si hay resultados.
        if (rs.next()) {
            // Se obtiene el nombre del usuario con la máxima cantidad de materiales.
            String maxU = rs.getString(columnIndex:1);
            // Se muestra el nombre del usuario en un componente de la interfaz gráfica (en este caso, txtUsuarioMAX).
            txtUsuarioMAX.setText(text:maxU);
        } else {
            // Si no hay resultados, se muestra un mensaje indicando que no hay datos en la tabla "usuario".
            JOptionPane.showMessageDialog(parentComponent:null, message:"No hay datos en la tabla usuario");
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se muestra un mensaje de error al usuario.
        JOptionPane.showMessageDialog(parentComponent:null, message:"No se pudo ejecutar");
    }
}
```

Empresa con mayor material comprado:

```
//Metodo maximoEmpresa (Empresa con mayor material comprado)
public void maximoEmpresa() {
    // Se crea una instancia de la clase "conexion" para establecer la conexión con la base de datos.
    conexion conn = new conexion();
    try {
        // Se establece la conexión con la base de datos.
        Connection con = conn.establecerConexion();
        // Se prepara la consulta SQL para obtener el nombre de la empresa con la máxima cantidad de materiales.
        PreparedStatement ps = con.prepareStatement("SELECT Nombre_Empresa FROM empresa WHERE Cantidad_Material = (SELECT MAX(Cantidad_Material) FROM empresa)");
        // Se ejecuta la consulta y se obtiene el conjunto de resultados.
        ResultSet rs = ps.executeQuery();
        // Se verifica si hay resultados.
        if (rs.next()) {
            // Se obtiene el nombre de la empresa con la máxima cantidad de materiales.
            String maxE = rs.getString(columnIndex:1);
            // Se muestra el nombre de la empresa en un componente de la interfaz gráfica (en este caso, txtEmpresaMAX).
            txtEmpresaMAX.setText(text:maxE);
        } else {
            // Si no hay resultados, se muestra un mensaje indicando que no hay datos en la tabla "empresa".
            JOptionPane.showMessageDialog(parentComponent:null, message:"No hay datos en la tabla empresa");
        }
    } catch (SQLException e) {
        // En caso de excepción SQL, se muestra un mensaje de error al usuario.
        JOptionPane.showMessageDialog(parentComponent:null, message:"No se pudo ejecutar");
    }
}
```