Revisión del proyecto.

Software administrativo para Panadería.

El proyecto mostrado a continuación permite administrar una panadería de manera “sencilla”, es decir va dirigido a panaderías que no tienen software asociado con la venta y administración del pan o aquellas que no se sientan conformes con el software que usan para su administración. Con el programa se espera poder controlar el inventario tanto de materia prima como de productos que venda la panadería con el fin de administrarlos para la venta del pan.

Además de la venta del pan se puede administrar la compra de materia prima necesaria para la elaboración de pan, cabe destacar que las compras son meramente informativas para la administración del dinero. El programa cuenta con una interfaz intuitiva para que el usuario sea capaz de manejarlo, además al realizar transacciones relacionadas con la venta del pan provee de un “ticket” con el cual el usuario puede ver las transacciones realizadas.

Al iniciar el programa permite ingresar al sistema de 3 formas, así como dar de alta 3 tipos de usuarios. Los usuarios empleados en este programa son:

Administrador.

Cliente.

Usuario.

El administrador podrá ser capaz de comprar materia prima para la administración del pan que se maneja en la misma panadería, una vez que exista materia prima suficiente para crear determinado pan se puede crear pan disminuyendo la materia prima necesaria, todos los movimientos anteriores son meramente informativos con el fin de tener una administración precisa del inventario.

Todas las consultas realizadas para manejar el inventario (compra de materia prima), así como las relacionadas con la venta del pan (creación y venta de pan), se harán a una base de datos.

Las operaciones básicas que proveerá el programa son:

Adquisición de inventario en cuanto a materia prima.

Creación de Pan por medio de la materia prima.

Venta de Pan con ticket.

CRUD de Usuarios, Administrador y de Cliente.

* + 1. Objetivos.
    2. Definición de actores:

Se describen en total 4 actores en el sistema. El Administrador es capaz de interactuar con todos los casos de usos.

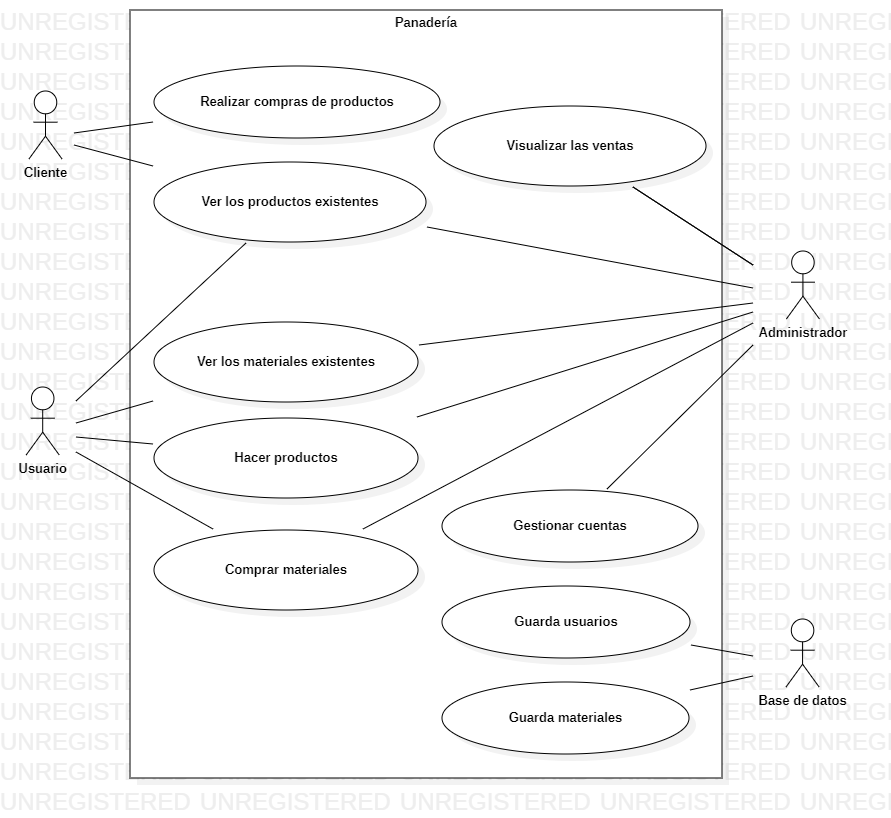
|  |  |
| --- | --- |
| ACTOR: | Administrador |
| CASOS DE USO: | Registrar usuarios, registrar clientes, registrar administradores, adquirir materia prima, elaborar pan con la materia prima, vender pan, visualizar ventas. |
| TIPO: | Primario |
| DESCRIPCIÓN | Actor principal y representa al dueño de la panadería capaz de realizar todas las funciones principales del programa. |

|  |  |
| --- | --- |
| ACTOR: | Usuario |
| CASOS DE USO: | Ver materia prima existente, comprar materia prima, ver productos existentes y crear pan con la materia prima. |
| TIPO: | Secundario |
| DESCRIPCIÓN | Es un actor secundario ya que solo crea pan, administra y compra materia prima. |

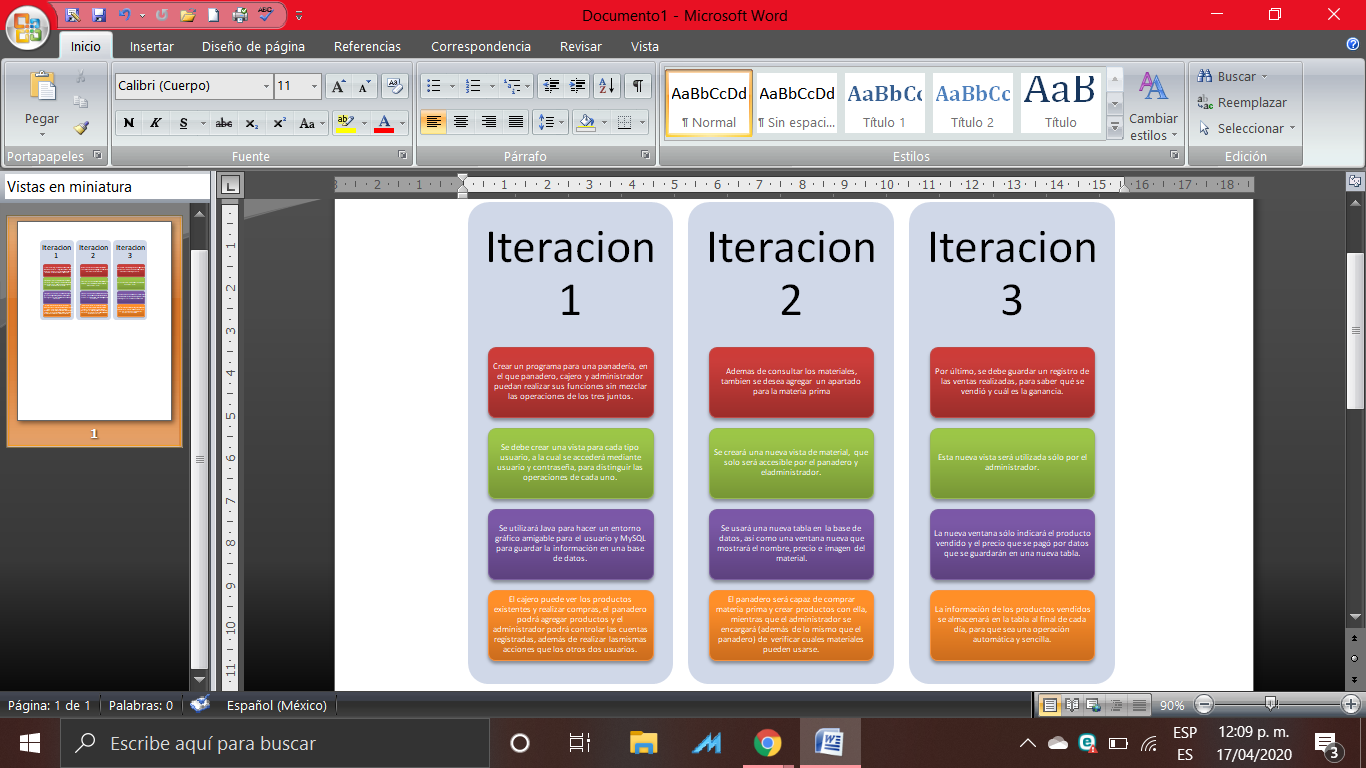
|  |  |
| --- | --- |
| ACTOR: | Cliente |
| CASOS DE USO: | Ver productos en existencia y realizar compra de productos. |
| TIPO: | Secundario |
| DESCRIPCIÓN | Es secundario ya que no es actor principal en el programa, es necesario para la panadería, pero no elabora como administrador o empleado en la misma, sino como cliente de la panadería. |

|  |  |
| --- | --- |
| ACTOR: | Base de Datos |
| CASOS DE USO: | Permite a los usuarios (cliente, usuario y administrador) realizar cambios o vistas con todo lo relacionado con el pan. |
| TIPO: | Secundario |
| DESCRIPCIÓN | Permite guardar los materiales de la panadería, así como a los usuarios dentro de la misma, para poder tener un control de los mismos. |

* + 1. Modelo de casos de Uso.



* + 1. Reglas de Negocio (RN)

1. VP0001.- El cliente al tratar de realizar un producto si existe disponibilidad de la materia prima involucrada se podrá llevar a cabo, caso contrario, se dará un aviso de que no fue posible realizar el producto por falta de material  señalando el material faltante.
2. VP0010.-El cliente al tratar comprar materia prima, deberá primero checarse el efectivo o equivalente en efectivo para comprobar si hay dinero suficiente para realizar la compra, caso de que no exista el efectivo suficiente se dará un aviso donde se menciona que no hay efectivo necesario para realizar la compra.
3. VP0011.-Si el cajero cuenta con el dinero suficiente y existe el producto puede llevarse a cabo la compra del mismo, de no ser así se mandará un aviso de fallo de compra por saldo insuficiente o falta del producto.
   * 1. Documento de Especificación de Requerimientos.
4. Requisitos Funcionales
5. RF1.- El administrador o el Panadero podrá crear un producto para ponerlo en venta en el sistema.
6. RF2.- El cajero podrá comprar los productos disponibles.
7. RF3.- Se visualizarán los productos en disponibilidad.
8. RF4.- Se visualizarán los materiales en disponibilidad.
9. RF5.- El Administrador podrá modificar los registros de usuarios, materiales y productos.
10. Requisitos No Funcionales
11. RNF1.- Medir tiempo de respuesta: Para llevar a cabo mediciones al programa se usará el programa JMH en el programa llevado a cabo en JAVA, en cuanto a la ejecución de querys en la base de datos se recabará el tiempo, debido a que la base de datos a usar es MYSQL se pueden recabar los tiempos en el mismo gestor.
12. RNF2.- Seguridad: Las claves para ingresar al sistema, tanto como administrador, usuario o cajero serán cifradas con el cifrado XOR y con cifrado CESAR.
13. RNF3.- Disponibilidad: El sistema sólo estará disponible en horario de trabajo, es decir cuando se necesite el uso, un tiempo aproximado serán 12 horas activo.
14. RNF4.- El dinero disponible en la caja disminuirá conforme el administrador o Panadero compre materia prima y aumentará cuando el usuario compre los beneficios.
16. Restricciones
    * 1. RX1.- Se empleará el lenguaje de programación Java y el gestor de base de datos MySQL
      2. Modelo de Procesos de Desarrollo.
17. Se optó por utilizar una Metodología Iterativa o Incremental, ya que el programa en un principio ya era completamente funcional, pero pudimos ver que se podían agregar nuevas funciones.
18. Usando esta metodología se puede tener un prototipo en una iteración, esto es importante ya que no se cuenta con mucho tiempo para el desarrollo de este proyecto, en la primera iteración se trabaja en los requisitos primordiales haciendo que el programa sea funcional en esta etapa, esto nos ayuda a ir sacando más requerimientos y evolucionando el proyecto cada que se avance a las demás iteraciones.
19. 

\*Cronograma que exprese.

- Número de iteraciones, duración de las iteraciones.

- Hitos y artefactos del proyecto.

* + 1. Herramientas de Software empleadas.
* Eclipse - El lenguaje de programación que mejor se adapta a nuestras necesidades fue Java, debido a que es un lenguaje orientado a objetos y es un lenguaje que todos conocemos. El editor que usamos fue Eclipse, debido a que tiene ciertas ventajas sobre otros como indicación de errores y facilidad para la conexión con bases de datos.
* NetBeans - Como alternativa a Eclipse, hubo algunas ocasiones en las que se usó este editor para hacer pruebas del proyecto y para observar detalles del mismo.
* MySQL - Uno de los sistemas gestores de bases de datos más conocidos tanto en el ámbito laboral como en el escolar, que posee varias utilidades, entre las cuales encontramos tanto el workbench para hacer la base de datos por medio de un diagrama ER como la misma terminal para trabajar por medio de sentencias.
* XAMPP - Como alternativa al instalador de MySQL, se utilizó este software que contiene MySQL, la diferencia con el instalador es que puede iniciar o detener MySQL, además de hacer uso de phpMyAdmin para gestionar la base de datos.
  + 1. Avance en el producto de SW.

# Código.

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

public class Panaderia

{

public static void main(String[] args)

{

String carpetaTrabajo = System.getProperty("use.dir");

System.out.println("La carpeta de trabajo es " + carpetaTrabajo);

new vista();

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

import java.sql.\*;

import java.util.\*;

public class Caja

{

private String ID;

private String Dinero;

public Caja()

{

}

public Caja(String id, String dinero)

{

this.ID = id;

this.Dinero = dinero;

}

// Metodos get

public String getID()

{

return ID;

}

public String getDinero()

{

return Dinero;

}

//Metodos set

public void setID(String id)

{

this.ID = id;

}

public void setDinero(String dinero)

{

this.Dinero = dinero;

}

public static Caja getCajaFromDB(String Caja, Properties prop)

{

Caja usuario = new Caja();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM CAJA WHERE ID = ?");

ps.setString(1,Caja);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String id = rs.getString("ID");

String dinero = rs.getString("Dinero");

usuario.setID(id);

usuario.setDinero(dinero);

con.close();

return usuario;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public static Caja Login (String Caja, Properties prop)

{

Caja usuario = new Caja();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM CAJA WHERE ID = ?");

ps.setString(1,Caja);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String dinero = rs.getString("Dinero");

String id = rs.getString("ID");

usuario.setDinero(dinero);

usuario.setID(id);

con.close();

return usuario;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public boolean Registro(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("INSERT INTO CAJA (ID, DINERO) VALUES (?,?)");

ps.setString(1, this.ID);

ps.setString(2, this.Dinero);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean cambiar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("UPDATE CAJA SET DINERO = ? WHERE ID = ? ");

// El titulo que llega de la Vista

ps.setString(1, this.Dinero);

ps.setString(2, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean borrar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM CAJA WHERE ID = ?");

ps.setString(1, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

import java.io.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class Materiales

{

private String ID;

private String nombre;

private String precio;

private String cantidad;

private String imagen;

public Materiales()

{

}

public Materiales(String ID,String nombre, String precio, String cantidad, String imagen)

{

this.ID = ID;

this.nombre= nombre;

this.cantidad = cantidad;

this.precio = precio ;

this.imagen = imagen;

}

// Metodos get

public String getID()

{

return ID;

}

public String getNombre()

{

return nombre;

}

public String getPrecio()

{

return precio;

}

public String getCantidad()

{

return cantidad;

}

public String getImagen()

{

return imagen;

}

//metodos Set

public void setID(String ID)

{

this.ID = ID;

}

public void setNombre(String nombre)

{

this.nombre = nombre;

}

public void setPrecio(String precio)

{

this.precio = precio;

}

public void setCantidad(String cantidad)

{

this.cantidad = cantidad;

}

public void setImagen(String imagen)

{

this.imagen = imagen;

}

public static Materiales getMaterialesFromDB(String Catalogo, Properties prop)

{

Materiales material = new Materiales();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM MATERIALES WHERE ID = ?");

ps.setString(1,Catalogo);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String ID=rs.getString("ID");

String nombre = rs.getString("Nombre");

String precio = rs.getString("Precio");

String cantidad = rs.getString("Cantidad");

String imagen = rs.getString("Imagen");

material.setID(ID);

material.setNombre(nombre);

material.setPrecio(precio);

material.setCantidad(cantidad);

material.setImagen(imagen);

con.close();

return material;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public boolean Registro(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("INSERT INTO MATERIALES (ID, NOMBRE,PRECIO,CANTIDAD,IMAGEN) VALUES (?,?,?,?,?)");

ps.setString(1, this.ID);

ps.setString(2, this.nombre);

ps.setString(3, this.precio);

ps.setString(4, this.cantidad);

ps.setString(5, this.imagen);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean cambiar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("UPDATE MATERIALES SET NOMBRE= ?, PRECIO = ?, CANTIDAD = ?, IMAGEN = ? WHERE ID = ? ");

// El titulo que llega de la Vista

ps.setString(1, this.nombre);

ps.setString(2, this.precio);

ps.setString(3, this.cantidad);

ps.setString(4, this.imagen);

ps.setString(5, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean borrar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM MATERIALES WHERE ID = ?");

ps.setString(1, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

import java.io.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class Productos

{

private String ID;

private String nombre;

private String precio;

private String cantidad;

private String imagen;

public Productos()

{

}

public Productos(String ID,String nombre, String precio, String cantidad, String imagen)

{

this.ID = ID;

this.nombre= nombre;

this.cantidad = cantidad;

this.precio = precio ;

this.imagen = imagen;

}

// Metodos get

public String getID()

{

return ID;

}

public String getNombre()

{

return nombre;

}

public String getPrecio()

{

return precio;

}

public String getCantidad()

{

return cantidad;

}

public String getImagen()

{

return imagen;

}

//metodos Set

public void setID(String ID)

{

this.ID = ID;

}

public void setNombre(String nombre)

{

this.nombre = nombre;

}

public void setPrecio(String precio)

{

this.precio = precio;

}

public void setCantidad(String cantidad)

{

this.cantidad = cantidad;

}

public void setImagen(String imagen)

{

this.imagen = imagen;

}

public static Productos getProductosFromDB(String Catalogo, Properties prop)

{

Productos material = new Productos();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM PRODUCTOS WHERE ID = ?");

ps.setString(1,Catalogo);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String ID=rs.getString("ID");

String nombre = rs.getString("Nombre");

String precio = rs.getString("Precio");

String cantidad = rs.getString("Cantidad");

String imagen = rs.getString("Imagen");

material.setID(ID);

material.setNombre(nombre);

material.setPrecio(precio);

material.setCantidad(cantidad);

material.setImagen(imagen);

con.close();

return material;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public static int getNumeroProductos( Properties prop)

{

Productos material = new Productos();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT count(\*) FROM materiales;");

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String Filas=rs.getString("count(\*)");

int filas=Integer.parseInt(Filas);

con.close();

return filas;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return 0;

}

public boolean Registro(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("INSERT INTO PRODUCTOS (ID, NOMBRE,PRECIO,CANTIDAD,IMAGEN) VALUES (?,?,?,?,?)");

ps.setString(1, this.ID);

ps.setString(2, this.nombre);

ps.setString(3, this.precio);

ps.setString(4, this.cantidad);

ps.setString(5, this.imagen);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean cambiar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("UPDATE PRODUCTOS SET NOMBRE= ?, PRECIO = ?, CANTIDAD = ?, IMAGEN = ? WHERE ID = ? ");

// El titulo que llega de la Vista

ps.setString(1, this.nombre);

ps.setString(2, this.precio);

ps.setString(3, this.cantidad);

ps.setString(4, this.imagen);

ps.setString(5, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean borrar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM PRODUCTOS WHERE ID = ?");

ps.setString(1, this.ID);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

import java.sql.\*;

import java.util.\*;

public class Usuarios

{

private String nombre;

private String contraseña;

private String tipo;

private String dinero;

public Usuarios()

{

}

public Usuarios(String nombre,String contraseña, String tipo)

{

this.nombre = nombre;

this.contraseña= contraseña;

this.tipo = tipo;

}

// Metodos get

public String getNombre()

{

return nombre;

}

public String getContraseña()

{

return contraseña;

}

public String getTipo()

{

return tipo;

}

//Metodos set

public void setNombre(String nombre)

{

this.nombre = nombre;

}

public void setContraseña(String contraseña)

{

this.contraseña = contraseña;

}

public void setTipo(String tipo)

{

this.tipo = tipo;

}

public static Usuarios getUsuarioFromDB(String Usuario, Properties prop)

{

Usuarios usuario = new Usuarios();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM USUARIOS WHERE NOMBRE = ?");

ps.setString(1,Usuario);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String nombre = rs.getString("Nombre");

String contraseña = rs.getString("Contraseña");

String tipo = rs.getString("Tipo");

usuario.setNombre(nombre);

usuario.setContraseña(contraseña);

usuario.setTipo(tipo);

con.close();

return usuario;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public static Usuarios Login (String Usuario, Properties prop)

{

Usuarios usuario = new Usuarios();

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM USUARIOS WHERE NOMBRE = ?");

ps.setString(1,Usuario);

boolean ok = ps.execute();

ResultSet rs = ps.getResultSet();

if(rs!=null && rs.next())

{

String tipo=rs.getString("Tipo");

String usuario1 = rs.getString("Nombre");

String contraseña = rs.getString("Contraseña");

usuario.setNombre(usuario1);

usuario.setContraseña(contraseña);

usuario.setTipo(tipo);

con.close();

return usuario;

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

return null;

}

public boolean Registro(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("INSERT INTO USUARIOS (NOMBRE, CONTRASEÑA, TIPO) VALUES (?,?,?)");

ps.setString(1, this.nombre);

ps.setString(2, this.contraseña);

ps.setString(3, this.tipo);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean cambiar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("UPDATE USUARIOS SET CONTRASEÑA = ?, TIPO = ? WHERE NOMBRE = ? ");

// El titulo que llega de la Vista

ps.setString(3, this.nombre);

ps.setString(1, this.contraseña);

ps.setString(2, this.tipo);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

public boolean borrar(Properties prop)

{

boolean exito = false;

try

{

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM USUARIOS WHERE NOMBRE = ?");

ps.setString(1, this.nombre);

exito = ps.executeUpdate() > 0;

con.close();

}

catch (Exception ex)

{

ex.printStackTrace();

}

return exito;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Panaderia;

/\*\*

\*

\* @author dvhda

\*/

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

import java.io.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.logging.Level;

import java.util.logging.Logger;

//Tabla

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.DefaultTableCellRenderer;

public class vista extends JFrame implements ActionListener

{

private boolean nuevo=true;

private JTextArea textarea1 =new JTextArea();

Productos newProd[] = new Productos[1];//idea multiproduco

int ComProd[]=new int[10];

//Tablas

JTable tablaProductos= new JTable();

JTable tablaMateriales=new JTable();

JScrollPane panelmat = new JScrollPane(tablaMateriales);

JScrollPane panelprod = new JScrollPane(tablaProductos);

//combo box

String Tipos[]={"Administrador","Cliente","Usuario"};

private JComboBox tipos = new JComboBox(Tipos);

String Materia[]={"Materiales","Huevo","Harina","Levadura","Leche","Azucar","Mantequilla","sal"};

private JComboBox materiales = new JComboBox(Materia);

String Producto[]={"Productos","Dona","Concha"};

private JComboBox productos = new JComboBox(Producto);

//Imagen

private JLabel jimagen = new JLabel();//Intento de imagen

private ImageIcon icon = new ImageIcon();//imagen

private JLabel imagen = new JLabel();

//Menu

private JMenuBar barraMenus = new JMenuBar();

private JMenu archivo = new JMenu("Archivo");

private JMenuItem crear = new JMenuItem("Crear");

private JMenuItem inicio = new JMenuItem("Login");

private JMenuItem Cmat = new JMenuItem("C.Materiales");

private JMenuItem Cprod = new JMenuItem("C.Productos");

private JMenuItem salir = new JMenuItem("Salir");

private JMenuItem Tprod = new JMenuItem("T.Productos");

private JMenuItem Tmat = new JMenuItem("T.Materiales");

private JMenuItem CRUDU = new JMenuItem("Usuarios");

private JMenuItem CRUDM = new JMenuItem("Materiales");

private JMenuItem CRUDP = new JMenuItem("Productos");

//Login

private JLabel et2 = new JLabel("Nombre:");

private JLabel et3 = new JLabel("Contraseña:");

private JTextField tNombre= new JTextField();

private JTextField tContraseña= new JTextField();

private JButton login = new JButton("Igresar");

private JButton registrar = new JButton("Registrar");

//CRUD

private JLabel et1 = new JLabel("Dinero:");

private JLabel et6 = new JLabel("Precio:");

private JLabel et4 = new JLabel("Cantidad:");

private JLabel et5 = new JLabel("ID:");

private JLabel et7 = new JLabel("Imagen:");

private JTextField tID = new JTextField();

private JTextField tDinero= new JTextField();

private JTextField tPrecio= new JTextField();

private JTextField tCantidad = new JTextField();

private JTextField tImagen= new JTextField();

private JButton buscarP = new JButton("Buscar");

private JButton buscarU = new JButton("Buscar");

private JButton buscarM = new JButton("Buscar");

private JButton registroP = new JButton("Regristrar");

private JButton registroU = new JButton("Regristrar");

private JButton registroM = new JButton("Regristrar");

private JButton cambiarP = new JButton("Cambiar");

private JButton cambiarU = new JButton("Cambiar");

private JButton cambiarM = new JButton("Cambiar");

private JButton borrarP = new JButton("Borrar");

private JButton borrarU = new JButton("Borrar");

private JButton borrarM = new JButton("Borrar");

//Operaciones

private JButton Cmaterial = new JButton("Comprar");

private JButton Cproducto = new JButton("Comprar");

private JButton producir = new JButton("Crear");

private JButton agregar = new JButton("Agregar");

//Properties

private Properties prop = new Properties(); // Para guardar la conf de la base de datos

// LA VISTA

public vista()

{

initComponents();

this.setTitle("Panaderia (CRUD)");

this.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

this.setLayout(null);

this.setBounds(100,100,600,450);

this.setJMenuBar(barraMenus);

// Carga las propiedades desde el archivo

try

{

prop.load(new FileInputStream("config.properties"));

}

catch (Exception ex)

{

ex.printStackTrace();

}

this.setVisible(true);

}

public void initComponents()

{

// Diseño del menu

barraMenus.add(archivo);

archivo.add(salir);

archivo.add(inicio);

archivo.add(Tprod);

archivo.add(Tmat);

archivo.add(crear);

archivo.add(Cprod);

archivo.add(Cmat);

archivo.add(CRUDM);

archivo.add(CRUDP);

archivo.add(CRUDU);

//TablaProductos

panelprod.setBounds(10,50,600,300);

this.add(panelprod);

//TablaMAteriales

panelmat.setBounds(10,50,600,300);

this.add(panelmat);

// Componentes Usuarios

jimagen.setBounds(350,30,250,200);

icon =new ImageIcon("Panadero.jpg");//imagen

//C:\Users\dvhda\Documents\NetBeansProjects\CRUD-Panaderia\imagenes

jimagen.setIcon(icon);//imagen

//Labels

et1.setBounds(10,30,100,30);

et2.setBounds(10,70,100,30);

et3.setBounds(10,110,100,30);

et4.setBounds(10,150,100,30);

et5.setBounds(10,190,100,30);

et6.setBounds(10,230,100,30);

et7.setBounds(10,110,100,30);

//Text

tID.setBounds(85,190,150,30);

tNombre.setBounds(85,70,150,30);

tContraseña.setBounds(85,110,150,30);

tDinero.setBounds(85,190,150,30);

tPrecio.setBounds(85,230,150,30);

tCantidad.setBounds(85,150,150,30);

tImagen.setBounds(85,110,150,30);

//Usuarios

buscarP.setBounds(250,70,80,30);

registroP.setBounds(50,265,100,30);

cambiarP.setBounds(160,265,100,30);

borrarP.setBounds(270,265,100,30);

//Materiales

buscarU.setBounds(250,70,80,30);

registroU.setBounds(50,265,100,30);

cambiarU.setBounds(160,265,100,30);

borrarU.setBounds(270,265,100,30);

//Productos

buscarM.setBounds(250,70,80,30);

registroM.setBounds(50,265,100,30);

cambiarM.setBounds(160,265,100,30);

borrarM.setBounds(270,265,100,30);

//Inicio

tipos.setBounds(85,150,150,30);

materiales.setBounds(85,110,150,30);

productos.setBounds(85,110,150,30);

registrar.setBounds(85,190, 120, 30);

login.setBounds(210,190,120,30);

//Operaciones

Cmaterial.setBounds(210,190,120,30);

Cproducto.setBounds(110,200,120,30);

producir.setBounds(210,190,120,30);

agregar.setBounds(110, 240, 120,30);

//TextArea

textarea1.setBounds(250,50,300,300);

add(textarea1);

textarea1.append("Pandaeria" +'\n'+"Ticket de compra"+ '\n'+"productos:");

textarea1.setEditable(false);

//ComboBox

add(tipos);

add(materiales);

add(productos);

//Labels

add(et1);

add(et2);

add(et3);

add(et4);

add(et5);

add(et6);

add(et7);

//Text

add(tID);

add(tNombre);

add(tContraseña);

add(tPrecio);

add(tImagen);

add(tCantidad);

add(tDinero);

//Usuario

add(buscarP);

add(registroP);

add(cambiarP);

add(borrarP);

//Productos

add(buscarU);

add(registroU);

add(cambiarU);

add(borrarU);

//Materiales

add(buscarM);

add(registroM);

add(cambiarM);

add(borrarM);

//Login

add(login);

add(registrar);

add(jimagen);//imagen

//Operaciones

add(Cmaterial);

add(Cproducto);

add(producir);

add(agregar);

//Visibilidad

//Menu

Tprod.setVisible(false);

Tmat.setVisible(false);

crear.setVisible(false);

Cmat.setVisible(false);

Cprod.setVisible(false);

CRUDP.setVisible(false);

CRUDM.setVisible(false);

CRUDU.setVisible(false);

//text area

textarea1.setVisible(false);

//Imagenes

jimagen.setVisible(true);

//ComboBox

tipos.setVisible(true);

materiales.setVisible(false);

productos.setVisible(false);

//Tabla

tablaProductos.setVisible(false);

tablaMateriales.setVisible(false);

panelprod.setVisible(false);

panelmat.setVisible(false);

//Text

tImagen.setVisible(false);

tDinero.setVisible(false);

tPrecio.setVisible(false);

tID.setVisible(false);

tCantidad.setVisible(false);

tNombre.setVisible(true);

tContraseña.setVisible(true);

//Button

cambiarU.setVisible(false);

cambiarP.setVisible(false);

cambiarM.setVisible(false);

borrarU.setVisible(false);

borrarM.setVisible(false);

borrarP.setVisible(false);

buscarP.setVisible(false);

buscarU.setVisible(false);

buscarM.setVisible(false);

registroP.setVisible(false);

registroU.setVisible(false);

registroM.setVisible(false);

registrar.setVisible(true);

login.setVisible(true);

Cmaterial.setVisible(false);

Cproducto.setVisible(false);

producir.setVisible(false);

agregar.setVisible(false);

//Label

et1.setVisible(false);

et2.setVisible(true);

et3.setVisible(true);

et4.setVisible(false);

et5.setVisible(false);

et6.setVisible(false);

et7.setVisible(false);

//Atcion Lisntener

salir.addActionListener(this);

inicio.addActionListener(this);

buscarP.addActionListener(this);

registroP.addActionListener(this);

cambiarP.addActionListener(this);

borrarP.addActionListener(this);

buscarM.addActionListener(this);

registroM.addActionListener(this);

cambiarM.addActionListener(this);

borrarM.addActionListener(this);

buscarU.addActionListener(this);

registroU.addActionListener(this);

cambiarU.addActionListener(this);

borrarU.addActionListener(this);

login.addActionListener(this);

Tprod.addActionListener(this);

Tmat.addActionListener(this);

registrar.addActionListener(this);

CRUDM.addActionListener(this);

CRUDP.addActionListener(this);

CRUDU.addActionListener(this);

Cmat.addActionListener(this);

Cprod.addActionListener(this);

Cmaterial.addActionListener(this);

Cproducto.addActionListener(this);

producir.addActionListener(this);

crear.addActionListener(this);

agregar.addActionListener(this);

class MyWindowAdapter extends WindowAdapter

{

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

}

addWindowListener(new MyWindowAdapter());

}

// EL CONTROLADOR (GESTIONA LOS EVENTOS)

public void actionPerformed(ActionEvent objetoEvento)

{

Object fuenteDelEvento = objetoEvento.getSource();

// Identifica sobre qué objeto generó el evento

if(fuenteDelEvento == salir) System.exit(0); // Cierra el Programa

if(fuenteDelEvento == Cprod)

{

quitar();

et4.setVisible(true);

tCantidad.setVisible(true);

productos.setVisible(true);

Cproducto.setVisible(true);

textarea1.setVisible(true);

agregar.setVisible(true);

}

if(fuenteDelEvento == crear)

{

quitar();

et4.setVisible(true);

tCantidad.setVisible(true);

productos.setVisible(true);

producir.setVisible(true);

}

if(fuenteDelEvento == Cmat)

{

quitar();

et4.setVisible(true);

tCantidad.setVisible(true);

materiales.setVisible(true);

Cmaterial.setVisible(true);

}

if(fuenteDelEvento == Tprod)

{

try {

llenarTablaProductos();

} catch (SQLException ex) {

Logger.getLogger(vista.class.getName()).log(Level.SEVERE, null, ex);

}

quitar();

tablaProductos.setVisible(true);

panelprod.setVisible(true);

}

if(fuenteDelEvento == inicio)

{

limpiarCampos();

quitar();

//login

et2.setVisible(true);

et3.setVisible(true);

tNombre.setVisible(true);

tContraseña.setVisible(true);

login.setVisible(true);

registrar.setVisible(true);

tipos.setVisible(true);

//Menu

Tprod.setVisible(false);

Tmat.setVisible(false);

crear.setVisible(false);

Cprod.setVisible(false);

CRUDP.setVisible(false);

CRUDM.setVisible(false);

CRUDU.setVisible(false);

}

if(fuenteDelEvento == CRUDM)

{

limpiarCampos();

quitar();

//CRUD

et2.setVisible(true);

et4.setVisible(true);

et5.setVisible(true);

et6.setVisible(true);

et7.setVisible(true);

tID.setVisible(true);

tNombre.setVisible(true);

tPrecio.setVisible(true);

tCantidad.setVisible(true);

tImagen.setVisible(true);

//Mteriales

buscarM.setVisible(true);

borrarM.setVisible(true);

cambiarM.setVisible(true);

registroM.setVisible(true);

//imagen

jimagen.setVisible(true);

}

if(fuenteDelEvento == CRUDP)

{

limpiarCampos();

quitar();

//CRUD

et2.setVisible(true);

et4.setVisible(true);

et5.setVisible(true);

et6.setVisible(true);

et7.setVisible(true);

tID.setVisible(true);

tNombre.setVisible(true);

tPrecio.setVisible(true);

tCantidad.setVisible(true);

tImagen.setVisible(true);

//Productos

buscarP.setVisible(true);

borrarP.setVisible(true);

cambiarP.setVisible(true);

registroP.setVisible(true);

//Imagen

jimagen.setVisible(true);

}

if(fuenteDelEvento == CRUDU)

{

limpiarCampos();

quitar();

//CRUD

et2.setVisible(true);

et3.setVisible(true);

tNombre.setVisible(true);

tContraseña.setVisible(true);

tipos.setVisible(true);

//Usuarios

buscarU.setVisible(true);

borrarU.setVisible(true);

cambiarU.setVisible(true);

registroU.setVisible(true);

}

if(fuenteDelEvento == Tmat)

{

try {

llenarTablaMateriales();

} catch (SQLException ex) {

Logger.getLogger(vista.class.getName()).log(Level.SEVERE, null, ex);

}

quitar();

tablaMateriales.setVisible(true);

panelmat.setVisible(true);

}

if(fuenteDelEvento == agregar)

{

int cantidad=0;

String producto = "";

switch(productos.getSelectedIndex())

{

case 1:

producto = "Dona";

cantidad = Integer.parseInt(tCantidad.getText());

ComProd[1]= ComProd[1]+cantidad;

System.out.println(ComProd[1]);//Arregas problema de compra de multiproductos

break;

case 2:

producto = "Concha";

cantidad = Integer.parseInt(tCantidad.getText());

ComProd[2]= ComProd[2]+cantidad;

System.out.println(ComProd[2]);

break;

}

textarea1.append("\n"+producto+" Cantidad: "+tCantidad.getText());

}

if(fuenteDelEvento == login)

{

// Hacer algo

if(tNombre.getText().length() == 0 || tContraseña.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura Usuario o Contraseña, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Usuarios newUser = Usuarios.Login(tNombre.getText(),prop); // Invoca al Modelo

if(newUser != null)

{

tNombre.equals(newUser.getNombre());

tContraseña.equals(newUser.getContraseña());

String Tipo = newUser.getTipo();

if(Tipo.equals("Administrador"))

{

quitar();

Tprod.setVisible(true);

Tmat.setVisible(true);

crear.setVisible(true);

Cmat.setVisible(true);

Cprod.setVisible(true);

CRUDP.setVisible(true);

CRUDM.setVisible(true);

CRUDU.setVisible(true);

}else if(Tipo.equals("Cliente"))

{

quitar();

Tprod.setVisible(true);

Tmat.setVisible(true);

Cmat.setVisible(true);

crear.setVisible(true);

try {

llenarTablaMateriales();

} catch (SQLException ex)

{

Logger.getLogger(vista.class.getName()).log(Level.SEVERE, null, ex);

}

tablaMateriales.setVisible(true);

panelmat.setVisible(true);

}else if(Tipo.equals("Usuario"))

{

quitar();

Cprod.setVisible(true);

tablaProductos.setVisible(true);

panelprod.setVisible(true);

try {

llenarTablaProductos();

} catch (SQLException ex)

{

Logger.getLogger(vista.class.getName()).log(Level.SEVERE, null, ex);

}

}

JOptionPane.showMessageDialog(this, "Bienvenido", "Aviso!",

JOptionPane.INFORMATION\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el usuario", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == producir)

{

// Hacer algo

if(tCantidad.getText().length() == 0 || productos.getSelectedIndex() == 0)

{

JOptionPane.showMessageDialog(this, "Captura cantidad y selecciona un producto, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

//CAmbiar materiales a productos

Productos newProd = Productos.getProductosFromDB(Integer.toString(productos.getSelectedIndex()),prop);

if(newProd!=null)

{

switch(productos.getSelectedIndex())

{

case 1:

int cantidad = Integer.parseInt(tCantidad.getText());

int cantidadP = Integer.parseInt(newProd.getCantidad());

int i=0, cantidadM=0, a=1;

Materiales newMat[] = new Materiales[2];

for(i=0;i<2;i++)

{

newMat[i] = Materiales.getMaterialesFromDB(Integer.toString(i+1),prop);//Primeros 3 materiales

}

for(i=0;i<2;i++)

{

if(Integer.parseInt(newMat[i].getCantidad())<cantidad)

{

JOptionPane.showMessageDialog(this, "No hay"+newMat[i].getNombre()+" suficiente!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

a = 0;

}

}

if (a==1)

{

for(i=0;i<2;i++)

{

cantidadM=Integer.parseInt(newMat[i].getCantidad())-cantidad;

newMat[i].setCantidad(Integer.toString(cantidadM));

}

cantidadP =cantidadP+cantidad;

newProd.setCantidad(Integer.toString(cantidadP));

for(i=0;i<2;i++)

{

newMat[i].cambiar(prop);

}

//Confirmacion

if(newProd.cambiar(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Creacion realizada: " + newProd.getNombre(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Creacion no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

break;

}

}

else

JOptionPane.showMessageDialog(this, "No se pudo crear el producto", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == Cproducto)

{

// Hacer algo

if(tCantidad.getText().length() == 0 || productos.getSelectedIndex() == 0)

{

JOptionPane.showMessageDialog(this, "Captura cantidad y selecciona un producto, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

//CAmbiar materiales a productos

int filas =Productos.getNumeroProductos(prop);

System.out.println(filas);

Productos newProd = Productos.getProductosFromDB(Integer.toString(productos.getSelectedIndex()),prop);

Caja newCash = Caja.getCajaFromDB("1", prop);

int A=0;

if(newProd != null && newCash!=null)

{

int cantidad = Integer.parseInt(tCantidad.getText());

int dinero = Integer.parseInt(newCash.getDinero());

int precio = Integer.parseInt(newProd.getPrecio());

int cantidadA = Integer.parseInt(newProd.getCantidad());

int total=0;

if(cantidadA>cantidad)

{

total =dinero+(precio\*cantidad);

cantidadA =cantidadA-cantidad;

newProd.setCantidad(Integer.toString(cantidadA));

newCash.setDinero(Integer.toString(total));

A=1;

}else

{

JOptionPane.showMessageDialog(this, "No hay produtos suficientes!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

//Confirmacion

if(A==1 && newCash.cambiar(prop) && newProd.cambiar(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Compra realizada: " + newProd.getNombre(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Compra no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Material o la Caja", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == Cmaterial)

{

// Hacer algo

if(tCantidad.getText().length() == 0 || materiales.getSelectedIndex() == 0)

{

JOptionPane.showMessageDialog(this, "Captura cantidad y selecciona un material, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Materiales newMat = Materiales.getMaterialesFromDB(Integer.toString(materiales.getSelectedIndex()),prop);

Caja newCash = Caja.getCajaFromDB("1", prop);

int A=0;

if(newMat != null && newCash!=null)

{

int cantidad = Integer.parseInt(tCantidad.getText());

int dinero = Integer.parseInt(newCash.getDinero());

int precio = Integer.parseInt(newMat.getPrecio());

int cantidadA = Integer.parseInt(newMat.getCantidad());

int total=0;

if(dinero>(precio\*cantidad))

{

total =dinero-(precio\*cantidad);

cantidadA =cantidadA+cantidad;

newMat.setCantidad(Integer.toString(cantidadA));

newCash.setDinero(Integer.toString(total));

A=1;

}else

{

JOptionPane.showMessageDialog(this, "No hay dinero suficiente!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

//Confirmacion

if(A==1 && newCash.cambiar(prop) && newMat.cambiar(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Compra realizada: " + newMat.getNombre(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Compra no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Material o la Caja", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == buscarP)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Productos newProduct = Productos.getProductosFromDB(tID.getText(),prop); // Invoca al Modelo

if(newProduct != null)

{

tID.setText(newProduct.getID());

tNombre.setText(newProduct.getNombre());

tPrecio.setText(newProduct.getPrecio());

tCantidad.setText(newProduct.getCantidad());

tImagen.setText(newProduct.getImagen());

icon =new ImageIcon(newProduct.getImagen());//imagen

jimagen.setIcon(icon);//imagen

}

else

JOptionPane.showMessageDialog(this, "No existe el producto", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == buscarU)

{

// Hacer algo

if(tNombre.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura Nombre, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Usuarios newUser = Usuarios.getUsuarioFromDB(tNombre.getText(),prop); // Invoca al Modelo

if(newUser != null)

{

tContraseña.setText(newUser.getContraseña());

tNombre.setText(newUser.getNombre());

String Tipo = newUser.getTipo();

System.out.println(Tipo);

if(Tipo.equals("Administrador"))

{

tipos.setSelectedIndex(0);

}else if(Tipo.equals("Cliente"))

{

tipos.setSelectedIndex(1);

}else if(Tipo.equals("Usuario"))

{

tipos.setSelectedIndex(2);

}

}

else

JOptionPane.showMessageDialog(this, "No existe el Usuario", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == registroP)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

// Creamos un nuevo libro con los datos de la vista

Productos newProduct = new Productos();

newProduct.setID(tID.getText());

newProduct.setNombre(tNombre.getText());

newProduct.setPrecio(tPrecio.getText());

newProduct.setCantidad(tCantidad.getText());

newProduct.setImagen(tImagen.getText());

// Y ejecutamos la alta

if(newProduct.Registro(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Registro agregado: " + tNombre.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == registroU || fuenteDelEvento == registrar)

{

// Hacer algo

if(tNombre.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura Nombre, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

// Creamos un nuevo libro con los datos de la vista

Usuarios newUser = new Usuarios();

newUser.setContraseña(tContraseña.getText());

newUser.setNombre(tNombre.getText());

int Tipo = tipos.getSelectedIndex();

if (Tipo == 0)

{

newUser.setTipo("Administrador");

}else if (Tipo == 1)

{

newUser.setTipo("Cliente");

}else if (Tipo == 2)

{

newUser.setTipo("Usuario");

}

// Y ejecutamos la alta

if(newUser.Registro(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Registro agregado: " + tNombre.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == cambiarP)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Productos newProductos = Productos.getProductosFromDB(tID.getText(),prop); // Invoca al Modelo

if(newProductos != null)

{

// Actualiza el atributos del objeto

newProductos.setID(tID.getText()); // Actualiza el titulo del objeto libro

newProductos.setNombre(tNombre.getText());

newProductos.setPrecio(tPrecio.getText());

newProductos.setCantidad(tCantidad.getText());

newProductos.setImagen(tImagen.getText());

if(newProductos.cambiar(prop)) // Si hubo éxito

JOptionPane.showMessageDialog(this, "Registro actualizado: " + tID.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Prducto", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == borrarP)

{

int respuesta = JOptionPane.showConfirmDialog(this, "Desea borrar este registro?", "Atención!!!", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if(respuesta==JOptionPane.YES\_OPTION) // Si el usuario está seguro

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Productos newProduct = Productos.getProductosFromDB(tID.getText(),prop); // Invoca al Modelo

if(newProduct != null)

{

if(newProduct.borrar(prop)) // Si hubo éxito

{

JOptionPane.showMessageDialog(this, "Registro eliminado: " + tID.getText(), "Aviso!",JOptionPane.WARNING\_MESSAGE);

limpiarCampos();

}

else JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Producto", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

}

if(fuenteDelEvento == borrarU)

{

int respuesta = JOptionPane.showConfirmDialog(this, "Desea borrar este registro?", "Atención!!!", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if(respuesta==JOptionPane.YES\_OPTION) // Si el usuario está seguro

{

// Hacer algo

if(tNombre.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura Nombre, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Usuarios newUser = Usuarios.getUsuarioFromDB(tNombre.getText(),prop); // Invoca al Modelo

if(newUser != null)

{

if(newUser.borrar(prop)) // Si hubo éxito

{

JOptionPane.showMessageDialog(this, "Registro eliminado: " + tNombre.getText(), "Aviso!",JOptionPane.WARNING\_MESSAGE);

limpiarCampos();

}

else JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Producto", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

}

if(fuenteDelEvento == buscarM)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Materiales newMaterial = Materiales.getMaterialesFromDB(tID.getText(),prop); // Invoca al Modelo

if(newMaterial != null)

{

tID.setText(newMaterial.getID());

tNombre.setText(newMaterial.getNombre());

tPrecio.setText(newMaterial.getPrecio());

tCantidad.setText(newMaterial.getCantidad());

tImagen.setText(newMaterial.getImagen());

icon =new ImageIcon(newMaterial.getImagen());//imagen

jimagen.setIcon(icon);//imagen

}

else

JOptionPane.showMessageDialog(this, "No existe el ID", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == registroM)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

// Creamos un nuevo libro con los datos de la vista

Materiales newMaterial = new Materiales();

newMaterial.setID(tID.getText());

newMaterial.setNombre(tNombre.getText());

newMaterial.setPrecio(tPrecio.getText());

newMaterial.setCantidad(tCantidad.getText());

newMaterial.setImagen(tImagen.getText());

// Y ejecutamos la alta

if(newMaterial.Registro(prop)) // Si la alta fue exitosa

JOptionPane.showMessageDialog(this, "Registro agregado: " + tID.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == cambiarM)

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Materiales newMaterial = Materiales.getMaterialesFromDB(tID.getText(),prop); // Invoca al Modelo

if(newMaterial != null)

{

newMaterial.setID(tID.getText()); // Actualiza el titulo del objeto libro

newMaterial.setNombre(tNombre.getText());

newMaterial.setPrecio(tPrecio.getText());

newMaterial.setCantidad(tCantidad.getText());

newMaterial.setImagen(tImagen.getText());

if(newMaterial.cambiar(prop)) // Si hubo éxito

JOptionPane.showMessageDialog(this, "Registro actualizado: " + tID.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Material", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == cambiarU)

{

// Hacer algo

if(tNombre.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura Nombre, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Usuarios newUser = Usuarios.getUsuarioFromDB(tNombre.getText(),prop); // Invoca al Modelo

if(newUser != null)

{

// Actualiza el titulo del objeto libro

newUser.setNombre(tNombre.getText());

newUser.setContraseña(tContraseña.getText());

int Tipo = tipos.getSelectedIndex();

if (Tipo == 0)

{

newUser.setTipo("Administrador");

}else if (Tipo == 1)

{

newUser.setTipo("Cliente");

}else if (Tipo == 2)

{

newUser.setTipo("Usuario");

}

if(newUser.cambiar(prop)) // Si hubo éxito

JOptionPane.showMessageDialog(this, "Registro actualizado: " + tNombre.getText(), "Aviso!",JOptionPane.INFORMATION\_MESSAGE);

else

JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Material", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

if(fuenteDelEvento == borrarM)

{

int respuesta = JOptionPane.showConfirmDialog(this, "Desea borrar este registro?", "Atención!!!", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if(respuesta==JOptionPane.YES\_OPTION) // Si el usuario está seguro

{

// Hacer algo

if(tID.getText().length() == 0)

{

JOptionPane.showMessageDialog(this, "Captura ID, no lo puedes dejar en blanco", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

else

{

Materiales newMaterial = Materiales.getMaterialesFromDB(tID.getText(),prop); // Invoca al Modelo

if(newMaterial != null)

{

if(newMaterial.borrar(prop)) // Si hubo éxito

{

JOptionPane.showMessageDialog(this, "Registro eliminado: " + tID.getText(), "Aviso!",JOptionPane.WARNING\_MESSAGE);

limpiarCampos();

}

else JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR\_MESSAGE);

}

else

JOptionPane.showMessageDialog(this, "No existe el Material", "Aviso!",

JOptionPane.ERROR\_MESSAGE);

}

}

}

}

private void limpiarCampos()

{

tID.setText("");

tNombre.setText("");

tContraseña.setText("");

tCantidad.setText("");

tPrecio.setText("");

tImagen.setText("");

}

private void quitar()

{

//login

et2.setVisible(false);

et3.setVisible(false);

tNombre.setVisible(false);

tContraseña.setVisible(false);

login.setVisible(false);

registrar.setVisible(false);

//ComboBox

tipos.setVisible(false);

materiales.setVisible(false);

productos.setVisible(false);

//Materiales

buscarM.setVisible(false);

borrarM.setVisible(false);

cambiarM.setVisible(false);

registroM.setVisible(false);

//Productos

buscarP.setVisible(false);

borrarP.setVisible(false);

cambiarP.setVisible(false);

registroP.setVisible(false);

//Materiales

buscarU.setVisible(false);

borrarU.setVisible(false);

cambiarU.setVisible(false);

registroU.setVisible(false);

//CRUD

et4.setVisible(false);

et5.setVisible(false);

et6.setVisible(false);

et7.setVisible(false);

tID.setVisible(false);

tPrecio.setVisible(false);

tCantidad.setVisible(false);

tImagen.setVisible(false);

//Tabla Productos

panelprod.setVisible(false);

tablaProductos.setVisible(false);

//Tabla Materiales

tablaMateriales.setVisible(false);

panelmat.setVisible(false);

//Imagen

jimagen.setVisible(false);

//Operaciones

Cmaterial.setVisible(false);

Cproducto.setVisible(false);

producir.setVisible(false);

agregar.setVisible(false);

//TextArea

textarea1.setVisible(false);

}

public void llenarTablaProductos() throws SQLException

{

try

{

DefaultTableModel modelo = new DefaultTableModel();

tablaProductos.setModel(modelo);

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM PRODUCTOS");

ResultSet rs = ps.executeQuery();

ResultSetMetaData rsMd = (ResultSetMetaData) rs.getMetaData();

int cantidadColumnas = rsMd.getColumnCount();

modelo.addColumn("ID");

modelo.addColumn("Nombre");

modelo.addColumn("Cantidad");

modelo.addColumn("Precio");

int[] anchos = {50,200,50,50};

for (int i = 0; i < tablaProductos.getColumnCount(); i++) {

tablaProductos.getColumnModel().getColumn(i).setPreferredWidth(anchos[i]);

}

while (rs.next()) {

Object[] filas = new Object[cantidadColumnas];

for (int i = 0; i < cantidadColumnas; i++)

{

filas[i] = rs.getObject(i + 1);

}

modelo.addRow(filas);

}

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public void llenarTablaMateriales() throws SQLException

{

try

{

DefaultTableModel modelo = new DefaultTableModel();

tablaMateriales.setModel(modelo);

String driver = prop.getProperty("dbdriver");

String host = prop.getProperty("dbhost");

String user = prop.getProperty("dbuser");

String password = prop.getProperty("dbpassword");

String name = prop.getProperty("dbname");

String url = host + name + "?user=" + user + "&password=" + password+"&useSSL=false";

System.out.println("Conexion a la BD: " + url);

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT \* FROM MATERIALES");

ResultSet rs = ps.executeQuery();

ResultSetMetaData rsMd = (ResultSetMetaData) rs.getMetaData();

int cantidadColumnas = rsMd.getColumnCount();

modelo.addColumn("ID");

modelo.addColumn("Cantidad");

modelo.addColumn("Precio");

modelo.addColumn("Nombre");

int[] anchos = {50,200,50,50};

for (int i = 0; i < tablaMateriales.getColumnCount(); i++) {

tablaMateriales.getColumnModel().getColumn(i).setPreferredWidth(anchos[i]);

}

while (rs.next()) {

Object[] filas = new Object[cantidadColumnas];

for (int i = 0; i < cantidadColumnas; i++)

{

filas[i] = rs.getObject(i + 1);

}

modelo.addRow(filas);

}

}

catch(Exception ex)

{

ex.printStackTrace();

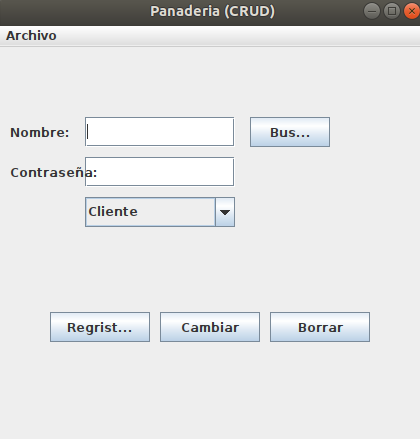
}

}

}

# Prototipos

Para la cuestión del CRUD de administrador, usuario y cliente se espera una ventana como la siguiente:



Donde se espera que se puedan cambiar datos como “nombre” y “contraseña”, hacer registros o borrar registros (CRUD), para elegir el usuario se espera poder cambiarlo por medio de un “combo Box” con las opciones “usuario, cliente y administrador”

* + 1. Pruebas