Manual del Programador

Ver.1.0, 2020

Hecho por:

María Jose Ruiz Sánchez

Gisela Carolina Arevalo Gordillo

Dorian Alberto Ibáñez Nangüelú

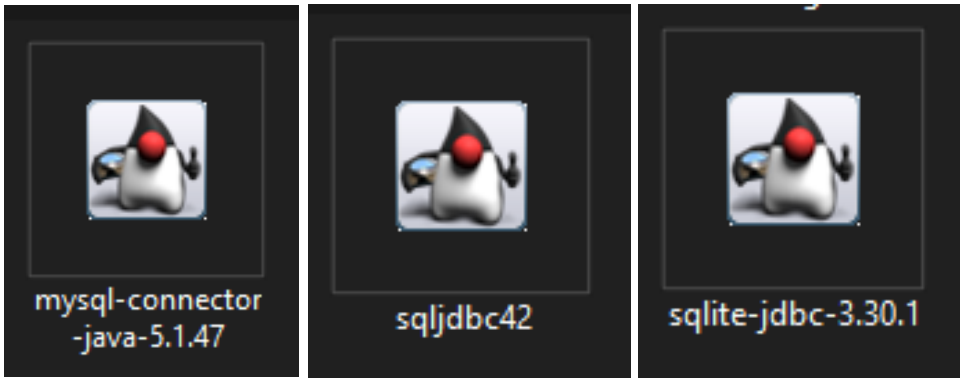
ÍNDICE

INTRODUCCIÓN

La finalidad que se tiene de este manual del programador es dar a conocer al lector todos los listados y códigos del programa Punto de Venta Papelería realizado. Por ello se tratará de forma amena y concisa todos los **Objetivos, Requerimientos técnico, Diagramas, Tablas de BD, etc.,** con el fin que el usuario pueda cambiar a gusto alguno de los valores y parámetros de los puntos expuestos para el programa del sistema Punto de Venta Papelería.

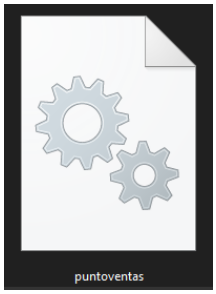
INSTALACIÓN

Para poder instalar el programa se tiene que tener 3 drives los cuales son:



El programa estará en su version ejecutable y tambien con un proyecto de eclipse para poder realizar modificaciones en el futuro

Y por ultimo se le proporcionara la base de datos conjunto al proyecto de eclipse



OBJETIVOS

FUNCIONAMIENTO DEL SISTEMA

* Iniciar Sesión (Login)
* Registro de nuevos usuarios
* Registro de producto
* Registro de Proveedores
* Inicio de sesión (tipo de usuario)
* Reporte de ventas
* Ticket de venta
* Base de datos
* Inventario (Filtrado)
* Inventario (Editar)
* Inventario (Eliminar)
* Lista de proveedores
* Lista de proveedores (Eliminar)
* Lista de proveedores (Editar)
* Lista de usuario
* Lista de usuario (Editar)
* Lista de usuario (Eliminar)

BASE DE DATOS

* Realizar consulta de datos
* Diagrama de Entidad-Relación
* Diagrama de Flujo de Datos

DOCUMENTACIÓN

* Poseer historias de usuario
* Tener los diagnósticos del sistema
* Poseer las distintas retrospectiva que se realizaban en el sistema

REQUERIMEINTOS TÉCNICOS DEL SISTEMA

ARQUITECTURA

La arquitectura decidida para este sisma es la Arquitectura MVC ya que el sistema ya está basado en esta arquitectura en donde se reutilizo para ahorrar tiempo.

HERRAMIENTAS

* Java 8.0
* Java FX
* MySQL WORKBENCH 8.0 CE
* Scene Builder 9-0-1
* mysql-connector-java 5.1.47
* Eclipse IDE for Java Developers 64 o 32 bits
* XAMPP 7.3.3
* VISUAL PARADIGM CE 15.1
* MICROSOFT OFFICE
*

SERVIDOR

* Memoria RAM de 4GB como mínimo, 8GB recomendado o superior.
* Procesador Intel PC Compatible, Xeon o similar a 2.5 GHz o superior (Itanium no recomendado).
* Conectividad a una red local vía TCP con protocolo http ó https.
* Sistema Operativo Microsoft Windows Server 2008, 2012 ó 2016, Estándar ó Enterprise, 32 ó 64 bits.
* Navegador Google Chrome 20 ó Mozilla Firefox 25 ó superiores (instalado localmente en el servidor sin ser navegador predeterminado).
* Navegador Microsoft Internet Explorer 9o superior.

USUARIO FINAL

* Windows 7, Windows 8, Windows 10.
* Instalador del programa

DIAGRAMA DFD

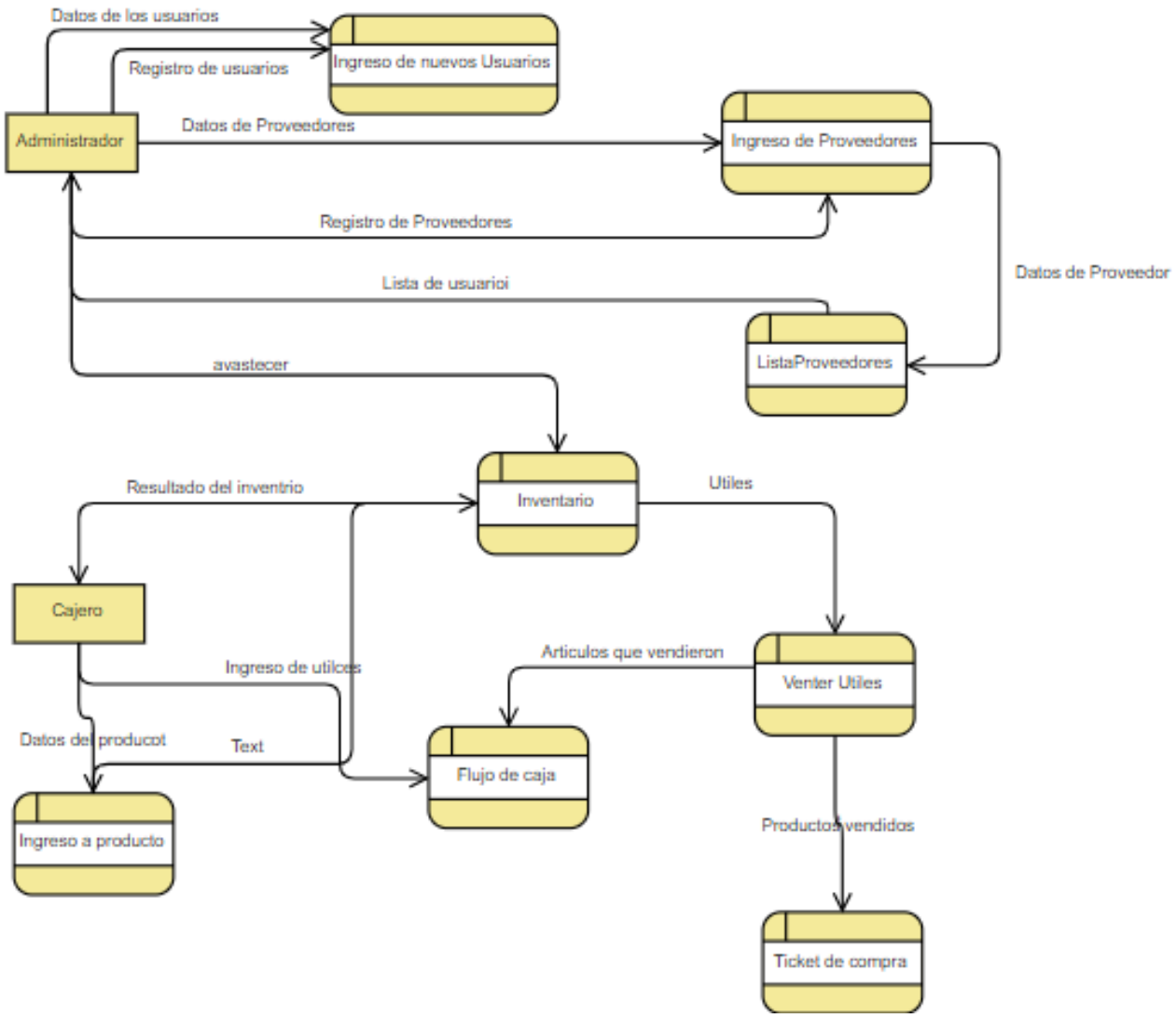
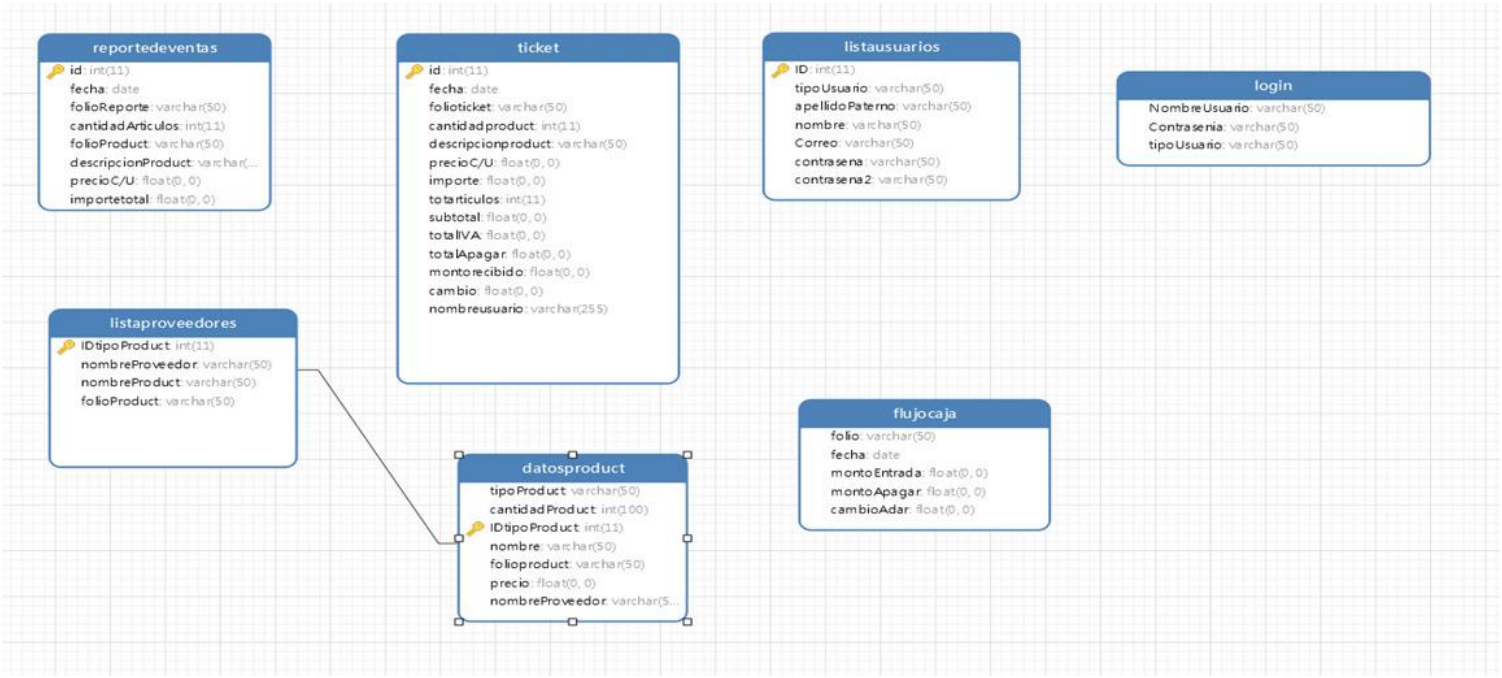


DIAGRAMA ENTIDAD-RELACIÓN Y NORMALIZACIÓN DE LA BASE DE DATOS

La normalización que se tomaron en cuenta para esta base de datos fue la primera normalización **Primera forma normal (1FN)** la cual consiste en que todos los atributos, valores almacenados en las columnas, deben ser indivisibles y no deben existir grupos de valores repetido.



DESCRIPCIÓN DE LAS TABLAS DE LA BD

Tabla Login

Campos:

NombreUsuario: varchar(50)

Constrasenia:varchart(50)

TipoUsuario:varchart(50)

Esta tabla se utiliza para el inicio de sesión, tiene solo los campos necesarios para el logeo del usuario.

Tabla listausuarios

Campos:

ID: int (11) llave primaria

tipoUsuario: varchar(50)

apellidoPaterno: varchar(50)

nombre: varchar(50)

Correo: varchar(50)

Contrasena: varchar(50)

contrasena2: varchar(50)

Tabla que contiene a los usuarios que se han registrado al sistema.

Tabla Flujo de caja

Campos

folio: varchar(50)

fecha: date

montoEntrada: float (0,0)

montoApagar: float(0,0)

cambioAdar: float(0,0)

Tabla que contiene el flujo de dinero que se tiene en la caja durante la venta

Tabla Ticket

Campos:

Id:int(11)

fecha: date

folioticket: varchar(50)

cantidadproduct: int(11)

descripcionproduct:varchart(50)

precioC/U:float(0,0)

importe:float(0,0)

totalarticulos: int(11)

subtotal: float (0,0)

totalIVA: float(0,0)

totalApagar: float(0,0)

montorecibido: float(0,0)

cambio: float(0,0)

nombreusuario: varchart(50)

Esta tabla contiene los campos necesarios que tendrá el ticket de compra

Tabla Reportedeventas

Campos:

Id: int(11)

fecha:

folioReporte: varchart(50)

cantidadArticulos:int(11)

folioProduct: varchart(50)

descripcionProduct: varchart(50)

precioC/U: float(0,0)

importetotal: float(0,0)

Esta tabla cuenta con los campos necesarios para generar un reporte de ventas. En este se desglosara la información necesaria que se requiera

Tabla Listaproveedores

Campo:

IDtipoProduct:int(11)

nombreProveedor:varchar(50)

nombreProduct:varchar(50)

folioProduct: varchart(50)

Esta tabla cuenta con los campos para saber el registro de los proveedores que se tienen registrados en el sistema, dicha tabla se relaciona con la tabla datosproduct ya que se utiliza quien es el proveedor de dicho producto que se encuentre registrado en el sistema

Seria una relación de uno a muchos donde un proveedor tendrá muchos productos.

Tabla Datosproduct

Campo:

tipoProduct:varchar(50)

cantidadProduct: int(100)

IDtipoProduct: int(11)

nombre: varchar(50)

folioprouct:varchar(50)

precio: float(0,0)

nombreProveedor:varchar(50)

Tabla que tiene los campos para el control del registro de los productos que se tiene registrados en el sistema, dicha tabla se relaciona con la tabla listaproveedores

DEFINICIÓN Y DESCRIPCIÓN DE VARIABLES USADAS EN EL SISTEMA

LOGIN

En esta unidad contiene el funcionamiento de poder ingresar al sistema

Variables

```
private Button entrar;
private Button Guardar;
private TextField CAJAusuario;
private PasswordField CAJAcontrasenia;
```

Implementación

```
public void guardar() throws SQLException{
      Connection cn = ConnectorMySQL.getConnection();
  try {
       System.out.println("HOLI cargar inicio secion");
       String sSQLL = "INSERT INTO login (NombreUsuario,Contrasenia)
       VALUES(?,?)";
      PreparedStatement stt = cn.prepareStatement(sSQLL);
      stt.setString(1,(CAJAusuario.getText()));
      stt.setString(2,CAJAcontrasenia.getText());

      stt.execute();
  }catch (SQLException e){
      e.printStackTrace();
  }
  CAJAusuario.clear();
  CAJAcontrasenia.clear();
}

public void cargarListado()   {
      try {


                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
                  primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
                  primaryLayout.show();
                  Stage nuevaEscena =(Stage)
this.entrar.getScene().getWindow();
                  nuevaEscena.close();

            } catch (Exception e) {
                  e.printStackTrace();
            }
}
```

INCIO

En esta unidad contiene el funcionamiento de mostrar las opciones de login y registro

Variables

```
	private Button Registrarse;

	private Button Regresar;
```

Implementación

```
public void cargarRegistro() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasREGISTRO.fxml
"));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasREGISTRO");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Registrarse.getScene().getWindow();
			nuevaEscena.close();


		} catch (Exception e) {
			e.printStackTrace();
		}

	}

public void cargarOPCIONBASEDATOS() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasOPCIONBasedat
os.fxml"));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasOPCIONBasedatos");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
			nuevaEscena.close();


		} catch (Exception e) {
			e.printStackTrace();
		}
```

```
	}

}
```

LISTADO

En esta unidad contiene el funcionamiento de poder visualizar a las opciones internas del sistema

Variables

```
	private Button Proveedores;
	private Button ventaYflujo;
	private Button Inventario;
	private Button Usuarios;
	private Button salirYregresar;
	private Button Reporte;
	private Button RegistrarProducto;
	private Button RegistroProvee;
```

Implementación

```
public void cargarProveedores() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasPROVEEDORES.f
xml"));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasPROVEEDORES");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Proveedores.getScene().getWindow();
			nuevaEscena.close();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}


public void cargarInventario() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasPRODUCTOS.fxm
l"));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasPRODUCTOS");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Inventario.getScene().getWindow();
			nuevaEscena.close();
```

```
			} catch (Exception e) {
				e.printStackTrace();
			}
		}


public void cargarUsuarios() {
			try {
				AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasUSUARIOS.fxml
"));
				Scene scene = new Scene (root2);
				Stage primaryLayout = new Stage();
				primaryLayout.setScene(scene);
				primaryLayout.setTitle("FXMLPuntoVentasUSUARIOS");
				primaryLayout.show();
				Stage nuevaEscena =(Stage)
this.Usuarios.getScene().getWindow();
				nuevaEscena.close();

			} catch (Exception e) {
				e.printStackTrace();
			}
		}


		try {
				AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLOGIN.fxml"))
;
				Scene scene = new Scene (root2);
				Stage primaryLayout = new Stage();
				primaryLayout.setScene(scene);
				primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
				primaryLayout.show();
				Stage nuevaEscena =(Stage)
this.salirYregresar.getScene().getWindow();
				nuevaEscena.close();

			} catch (Exception e) {
				e.printStackTrace();
			}
		}


public void cargarReporte() {
			try {
				AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasREPORTE.fxml"
));
				Scene scene = new Scene (root2);
				Stage primaryLayout = new Stage();
```

```
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasREPORTE");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.Reporte.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }


      public void cargarRegistroProvee() {
             try {
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasREGISTROProve
edor.fxml"));
                    Scene scene = new Scene (root2);
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasREGISTROProveedor");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.RegistroProvee.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }

      public void cargarVentaYFlujo() {
             try {
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasVENTAS.fxml")
);
                    Scene scene = new Scene (root2);
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasVENTAS");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.ventaYflujo.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }

public void cargarRegistroProducto() {
             try {
```

```
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasRegistroProdu
ctos.fxml"));
                    Scene scene = new Scene (root2);
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasRegistroProductos");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.RegistrarProducto.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }
```

VENTAHISTORIAL

En esta unidad contiene el funcionamiento de poder tener la venta

Variables

```
      private int conta=0;
      private float sumaCosto;
      private Button Salir;
      private TextField CAJAidproduct;
      private TableColumn<ClassProductos, String> tipoProduct;
      private TableColumn<ClassProductos, String> idTipoProduct;
      private TableColumn<ClassProductos, String> nombreProduct;
      private TableColumn<ClassProductos, String> folio;
      private TableColumn<ClassProductos, String> precio;
      private TableColumn<ClassProductos, String> nombreProvee;
      private TableView<ClassProductos> tablaProduct;
      private TextField CAJAmontopagar;
      private TextField CAJAmontoentrada;
      private TextField CAJAmonto;
      private TextField CAJAmontoIVA;
      private TextField CAJAdarcambio;
      ObservableList<ClassProductos> productList =
      FXCollections.observableArrayList();

      private Button Regresar;
```

Implementación

```
public void initialize() {
      tipoProduct.setCellValueFactory(new
PropertyValueFactory<>("tipoProducto"));
            idTipoProduct.setCellValueFactory(new
PropertyValueFactory<>("tipoProductoID"));
            nombreProduct.setCellValueFactory(new
PropertyValueFactory<>("nomProducto"));
            folio.setCellValueFactory(new
PropertyValueFactory<>("folioProducto"));
            precio.setCellValueFactory(new PropertyValueFactory<>("Precio"));
```

```
            nombreProvee.setCellValueFactory(new
PropertyValueFactory<>("nombreproveedor"));

    }


      public void getProductList(){
            int resp ;
            Connection connection = ConnectorMySQL.getConnection();
            String query = "SELECT * FROM datosproduct WHERE folioproduct = "+
CAJAidproduct.getText();
            Statement st;
            ResultSet rs;
            float cambio=0;
            try {
                  st = connection.createStatement();
                  rs = st.executeQuery(query);
                  ClassProductos Productos;
                  while(rs.next()) {
                        Productos = new
ClassProductos(rs.getString("tipoProduct"),rs.getInt("CantidadProducto"),rs.getI
nt("IDtipoProduct"),rs.getString("nombre"),rs.getString("folioproduct"),rs.getFl
oat("precio"),rs.getString("nombreProveedor"));
                        conta++;

                        productList.add(Productos);
                  }
                  float preciosSinIVA = calcularPrecio();
                  CAJAmonto.setText(String.valueOf(preciosSinIVA));
                  float iva=(float) (preciosSinIVA*(0.16));
                  CAJAmontoIVA.setText(String.valueOf(iva));
                  float preciosTotal = iva+preciosSinIVA;
                  CAJAmontopagar.setText(String.valueOf(preciosTotal));
                  tablaProduct.getItems().setAll(productList);
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }


      private void cambio() {
            float entrada = Float.valueOf(CAJAmontoentrada.getText());
            float total = Float.valueOf(CAJAmontopagar.getText());

            float cambio = entrada-total;

            CAJAdarcambio.setText(String.valueOf(cambio));
      }

      private float calcularPrecio() {
            float preciosSinIVA = 0;
            for (ClassProductos classProductos : productList) {
                  preciosSinIVA += classProductos.getPrecio();
            }
            return preciosSinIVA;
```

```
	}

    public void executeQuery(String query) {
            Connection conn = ConnectorMySQL.getConnection();
            Statement st;
            try {
                  st = conn.createStatement();
                  st.executeUpdate(query);
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }



      public void regresarLogin() {
            try {
                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLOGIN.fxml"))
;
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
                  primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
                  primaryLayout.show();
                  Stage nuevaEscena =(Stage) this.Salir.getScene().getWindow();
                  nuevaEscena.close();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }


public void cargarListado() {
            try {
                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
                  primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
                  primaryLayout.show();
                  Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
                  nuevaEscena.close();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
```

```
	public void archivoTXT() {
		for (int i = 0; i < productList.size(); i++) {


		 try {
		String ruta = "D://Eclipse
H.W//PuntoDeVentas//TXTtickets.txt";
		String contenido = "Contenido de ejemplo";
		File file = new File(ruta);
		// Si el archivo no existe es creado
		if (!file.exists()) {
		    file.createNewFile();
		}
		FileWriter fw = new FileWriter(file);
		BufferedWriter bw = new BufferedWriter(fw);
		bw.write(contenido);
		bw.close();
	  } catch (Exception e) {
		e.printStackTrace();
	  }

		}
	}


	private void generarTicket(int nRandom) throws FileNotFoundException,
UnsupportedEncodingException {
		PrintWriter writer = new PrintWriter("TXTtickets/" +
String.valueOf(nRandom) + ".txt", "UTF-8");
		for (ClassProductos classProductos : productList) {
			writer.println("Folio del ticket: " + nRandom);
			writer.println("-------------------------");
			writer.println("Tipo de Producto: " +
classProductos.getTipoProducto());
			writer.println("Nombre: " + classProductos.getNomProducto());
			writer.println("Folio: " +
classProductos.getFolioProducto());
			writer.println("Precio: $" + classProductos.getPrecio());
			writer.println("");
		}

		writer.println("-------------------------");
		writer.println("Subtotal: $" +  CAJAmonto.getText());
		writer.println("IVA: $" + CAJAmontoIVA.getText());
		writer.println("Total: $" + CAJAmontopagar.getText());
		writer.println("Pago: $" + CAJAmontoentrada.getText());
		writer.println("Cambio: $" + CAJAdarcambio.getText());

		writer.close();
		cargarListado();
	}


	private void finalizarCompra() {
```

```
            Random r = new Random();
          int nRandom = r.nextInt(999999999)+1;
          try {
                  generarTicket(nRandom);
            } catch (FileNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (UnsupportedEncodingException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }
```

PRODUCTOS

En esta unidad contiene el funcionamiento de poder visualizar una tabla que contiene los datos de los productos y tambien de poder editar algún registro dentro de la tabla.

Variables

```
      private Button Regresar;
      private Button buttoneliminar;
      private Button buttoneditar;
      private Button buttonAgregar;
      private TextField cajatipoProduct;
      private TextField cajaIDtipoProduct;
      private TextField cajanombreProduct;
      private TextField cajafolioProduct;
      private TextField cajaprecio;
      private TextField cajanombreProvee;
      private TextField CAJAnPRODUCT;
    private TableColumn<ClassProductos, String> tipoProduct;
    private TableColumn<ClassProductos, String> CantidadProducto;
    private TableColumn<ClassProductos, String> idTipoProduct;
    private TableColumn<ClassProductos, String> nombreProduct;
    private TableColumn<ClassProductos, String> folio;
    private TableColumn<ClassProductos, String> precio;
    private TableColumn<ClassProductos, String> nombreProvee;
    private TableView<ClassProductos> tablaProduct;
    ObservableList<ClassProductos> productList =
FXCollections.observableArrayList();
```

Implementación

```
public void initialize(URL url, ResourceBundle rb) {
      tipoProduct.setCellValueFactory(new
PropertyValueFactory<>("tipoProducto"));
      CantidadProducto.setCellValueFactory(new
PropertyValueFactory<>("cantidadProduct"));
            idTipoProduct.setCellValueFactory(new
PropertyValueFactory<>("tipoProductoID"));
            nombreProduct.setCellValueFactory(new
PropertyValueFactory<>("nomProducto"));
            folio.setCellValueFactory(new
PropertyValueFactory<>("folioProducto"));
```

```
            precio.setCellValueFactory(new PropertyValueFactory<>("Precio"));
            nombreProvee.setCellValueFactory(new
PropertyValueFactory<>("nombreproveedor"));
            ObservableList<ClassProductos> list = getPersonList();
      tablaProduct.getItems().setAll(list);

      final ObservableList<ClassProductos> tablaProducto =
tablaProduct.getSelectionModel().getSelectedItems();
          tablaProducto.addListener(selectorTablaProductos);
    }

      public ObservableList<ClassProductos> getPersonList(){
            ObservableList<ClassProductos> productList =
FXCollections.observableArrayList();
            Connection connection = ConnectorMySQL.getConnection();
            String query = "SELECT * FROM datosproduct";
            Statement st;
            ResultSet rs;

            try {
                  st = connection.createStatement();
                  rs = st.executeQuery(query);
                  ClassProductos Productos;
                  while(rs.next()) {
                        Productos = new
ClassProductos(rs.getString("tipoProduct"),rs.getInt("CantidadProduct"),rs.getIn
t("IDtipoProduct"),rs.getString("nombre"),rs.getString("folioproduct"),rs.getFlo
at("precio"),rs.getString("nombreProveedor"));
                        productList.add(Productos);
                  }
            }catch (Exception e) {
                  e.printStackTrace();
            }
            return productList;
      }



    public void executeQuery(String query) {
            Connection conn = ConnectorMySQL.getConnection();
            Statement st;
            try {
                  st = conn.createStatement();
                  st.executeUpdate(query);
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }



      private final ListChangeListener<ClassProductos> selectorTablaProductos =
            new ListChangeListener<ClassProductos>() {

                        @Override
```

```
                        public void onChanged(Change<? extends ClassProductos>
arg0) {
                                ponerProductoSeleccionada();

                        }
            };

      private void ponerProductoSeleccionada() {
        final ClassProductos producto =  productoSeleccionado();
        productList.indexOf(producto);
        if (productList != null) {
             cajatipoProduct.setText(producto.getTipoProducto());

      CAJAnPRODUCT.setText(Integer.toString(producto.getCantidadProduct()));

      cajaIDtipoProduct.setText(Integer.toString(producto.getTipoProductoID()))
;
             cajanombreProduct.setText(producto.getNomProducto());
             cajafolioProduct.setText(producto.getFolioProducto());
             cajanombreProvee.setText(producto.getNombreproveedor());
            cajaprecio.setText(Float.toString(producto.getPrecio()));
        }
    }

      public ClassProductos productoSeleccionado() {
      if (tablaProduct != null) {
          List<ClassProductos> tabla =
tablaProduct.getSelectionModel().getSelectedItems();
          if (tabla.size() == 1) {
              final ClassProductos seleccionada = tabla.get(0);
              return seleccionada;
          }
      }
      return null;
      }



      private void modificar(ActionEvent event) {
            String query = "UPDATE datosproduct SET
tipoProduct='"+cajatipoProduct.getText()+"',CantidadProducto='"+CAJAnPRODUCT.get
Text()+"',nombre='"+cajanombreProduct.getText()+"',folioproduct='"+cajafolioProd
uct.getText()+"',precio='"+cajaprecio.getText()+"',nombreProveedor='"+cajanombre
Provee.getText()+"' WHERE IDtipoProduct='"+cajaIDtipoProduct.getText()+"'";
             executeQuery(query);
             ObservableList<ClassProductos> list = getPersonList();
             tablaProduct.getItems().setAll(list);

              final ObservableList<ClassProductos> tablaPersonas =
tablaProduct.getSelectionModel().getSelectedItems();
            tablaPersonas.addListener(selectorTablaProductos);

    }

      private void eliminar(ActionEvent event) {
```

```
            String query = "DELETE FROM datosproduct WHERE
IDtipoProduct="+cajaIDtipoProduct.getText()+"";
            executeQuery(query);
            ObservableList<ClassProductos> list = getPersonList();
            tablaProduct.getItems().setAll(list);

             final ObservableList<ClassProductos> tablaPersonas =
tablaProduct.getSelectionModel().getSelectedItems();
              tablaPersonas.addListener(selectorTablaProductos);
    }
```

PROVEEDORES

En esta unidad contiene el funcionamiento de poder visualizar una tabla que contiene los datos personales de los proveedores y tambien de poder editar algún registro dentro de la tabla.

Variables

```
      private Button Regresar;
      private TextField CAJAnombrePovee;
      private TextField CAJAnombreProduct;
      private TextField CAJAfolioProduct;
      private TextField CAJAIDTipoProduct;
      private Button eliminar;
      private Button modificar;
    private TableView<ClassProveedores> tablaProvee;
    private TableColumn<ClassProveedores, String> ID;
    private TableColumn<ClassProveedores, String> nombre;
    private TableColumn<ClassProveedores, String> nomProduct;
    private TableColumn<ClassProveedores, String> folioProduct;
    ObservableList<ClassProveedores> proveedoresList =
FXCollections.observableArrayList();
```

Implementación

```
private void initialize() {
            ID.setCellValueFactory(new
PropertyValueFactory<>("tipoProductoID"));
            nombre.setCellValueFactory(new
PropertyValueFactory<>("nombreProveedor"));
            nomProduct.setCellValueFactory(new
PropertyValueFactory<>("nombreProducto"));
            folioProduct.setCellValueFactory(new
PropertyValueFactory<>("folioProducto"));
            ObservableList<ClassProveedores> list = getPersonList();
            tablaProvee.getItems().setAll(list);

      final ObservableList<ClassProveedores> tablaProvees =
tablaProvee.getSelectionModel().getSelectedItems();
          tablaProvees.addListener(selectortablaProveedor);
    }
```

```
	public ObservableList<ClassProveedores> getPersonList(){
		ObservableList<ClassProveedores> proveedoresList =
FXCollections.observableArrayList();
		Connection connection = ConnectorMySQL.getConnection();
		String query = "SELECT * FROM listaproveedores";
		Statement st;
		ResultSet rs;

		try {
			st = connection.createStatement();
			rs = st.executeQuery(query);
			ClassProveedores Proveedores;
			while(rs.next()) {
				Proveedores = new
ClassProveedores(rs.getInt("IDtipoProduct"),rs.getString("nombreProveedor"),rs.g
etString("nombreProduct"),rs.getString("folioProduct"));
				proveedoresList.add(Proveedores);
			}
		}catch (Exception e) {
			e.printStackTrace();
		}
		return proveedoresList;
	}



    public void executeQuery(String query) {
		Connection conn = ConnectorMySQL.getConnection();
		Statement st;
		try {
			st = conn.createStatement();
			st.executeUpdate(query);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}



	private final ListChangeListener<ClassProveedores> selectortablaProveedor
=
		new ListChangeListener<ClassProveedores>() {

			@Override
			public void onChanged(Change<? extends
ClassProveedores> arg0) {
				ponerProveedorSeleccionada();

			}
		};

	private void ponerProveedorSeleccionada() {
	  final ClassProveedores proveedor =  proveedorSeleccionado();
		proveedoresList.indexOf(proveedor);
	  if (proveedoresList != null) {
```

```
        CAJAIDTipoProduct.setText(Integer.toString(proveedor.getTipoProductoID())
);
              CAJAnombrePovee.setText(proveedor.getNombreProveedor());
              CAJAnombreProduct.setText(proveedor.getNombreProducto());
              CAJAfolioProduct.setText(proveedor.getFolioProducto());
//            cajanombreProvee.setText(proveedor.getNombreproveedor());
//             cajaprecio.setText(Float.toString(proveedor.getPrecio()));
          }
    }

        public ClassProveedores proveedorSeleccionado() {
        if (tablaProvee != null) {
            List<ClassProveedores> tabla =
tablaProvee.getSelectionModel().getSelectedItems();
            if (tabla.size() == 1) {
                final ClassProveedores seleccionada = tabla.get(0);
                return seleccionada;
            }
        }
        return null;
        }



        private void modificar(ActionEvent event) {
              String query = "UPDATE listaproveedores SET
IDtipoProduct="+CAJAIDTipoProduct.getText()+",nombreProveedor='"+CAJAnombrePovee
.getText()+"',nombreProduct='"+CAJAnombreProduct.getText()+"',folioProduct='"+CA
JAfolioProduct.getText()+"' WHERE IDtipoProduct = " +
CAJAIDTipoProduct.getText();
              executeQuery(query);
              ObservableList<ClassProveedores> list = getPersonList();
              tablaProvee.getItems().setAll(list);
//
//             final ObservableList<ClassProveedores> tablaPersonas =
tablaProvee.getSelectionModel().getSelectedItems();
//           tablaPersonas.addListener(selectortablaProveedor);

    }


        private void eliminar(ActionEvent event) {
              String query = "DELETE FROM listaproveedores WHERE
IDtipoProduct="+CAJAIDTipoProduct.getText()+"";
              executeQuery(query);
              ObservableList<ClassProveedores> list = getPersonList();
              tablaProvee.getItems().setAll(list);

               final ObservableList<ClassProveedores> tablaPersonas =
tablaProvee.getSelectionModel().getSelectedItems();
                tablaPersonas.addListener(selectortablaProveedor);
    }
```

```
	public void cargarListado() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
			nuevaEscena.close();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}
```

USUARIOS

En esta unidad contiene el funcionamiento de poder visualizar una tabla que contiene los datos personales de los usuarios al sistema y tambien de poder editar algún registro dentro de la tabla.

Variables

```
	private Button Salir;
	private Button editar;
	private Button eliminar;
	private TextField CAJAid;
	private TextField CAJAapellidoPater;
	private TextField CAJAnombres;
	private TextField CAJAtipoUsuario;
	private TextField CAJAcorreo;
    private TableColumn<ClassUsuarios, String> ID;
    private TableColumn<ClassUsuarios, String> apellidoPaterno;
    private TableColumn<ClassUsuarios, String> Nombres;
    private TableColumn<ClassUsuarios, String> tipoUsuario;
    private TableColumn<ClassUsuarios, String> correos;
    private TableView<ClassUsuarios> tablaUsuario;
    ObservableList<ClassUsuarios> UsuariosList =
FXCollections.observableArrayList();
```

Implementación

```
public void initialize(URL url, ResourceBundle rb) {
		ID.setCellValueFactory(new PropertyValueFactory<>("id"));
		apellidoPaterno.setCellValueFactory(new
PropertyValueFactory<>("apellidoP"));
		Nombres.setCellValueFactory(new
PropertyValueFactory<>("nombreUsuario"));
```

```
                    tipoUsuario.setCellValueFactory(new
PropertyValueFactory<>("tipoUsuario"));
                    correos.setCellValueFactory(new
PropertyValueFactory<>("coreeo"));

                    ObservableList<ClassUsuarios> list = getPersonList();
                    tablaUsuario.getItems().setAll(list);

              final ObservableList<ClassUsuarios> tablaUsuarioo =
tablaUsuario.getSelectionModel().getSelectedItems();
                  tablaUsuarioo.addListener(selectortablaUsuarioos);
           }

              public ObservableList<ClassUsuarios> getPersonList(){
                    ObservableList<ClassUsuarios> UsuariosList =
FXCollections.observableArrayList();
                    Connection connection = ConnectorMySQL.getConnection();
                    String query = "SELECT * FROM listausuarios";
                    Statement st;
                    ResultSet rs;

                    try {
                          st = connection.createStatement();
                          rs = st.executeQuery(query);
                          ClassUsuarios Usuarioss;
                          while(rs.next()) {
                                Usuarioss = new
ClassUsuarios(rs.getInt("ID"),rs.getString("tipoUsuario"),rs.getString("apellido
Paterno"),rs.getString("nombre"),rs.getString("Correo"));
                                UsuariosList.add(Usuarioss);
                          }
                    }catch (Exception e) {
                          e.printStackTrace();
                    }
                    return UsuariosList;
              }



           public void executeQuery(String query) {
                    Connection conn = ConnectorMySQL.getConnection();
                    Statement st;
                    try {
                          st = conn.createStatement();
                          st.executeUpdate(query);
                    } catch (Exception e) {
                          e.printStackTrace();
                    }
              }



              private final ListChangeListener<ClassUsuarios>
selectortablaUsuarioos =
                    new ListChangeListener<ClassUsuarios>() {
```

```
                                @Override
                                public void onChanged(Change<? extends
ClassUsuarios> arg0) {
                                      ponerUsuariosseleccionada();

                                }
                    };

              private void ponerUsuariosseleccionada() {
                final ClassUsuarios usuario =  Usuariosseleccionado();
                UsuariosList.indexOf(usuario);
                if (UsuariosList != null) {
                     CAJAid.setText(Integer.toString(usuario.getId()));
                     CAJAapellidoPater.setText(usuario.getApellidoP());
                     CAJAnombres.setText(usuario.getNombreUsuario());
                     CAJAtipoUsuario.setText(usuario.getTipoUsuario());
                     CAJAcorreo.setText(usuario.getCoreeo());

                }
           }

              public ClassUsuarios Usuariosseleccionado() {
              if (tablaUsuario != null) {
                  List<ClassUsuarios> tabla =
tablaUsuario.getSelectionModel().getSelectedItems();
                  if (tabla.size() == 1) {
                      final ClassUsuarios seleccionada = tabla.get(0);
                      return seleccionada;
                  }
              }
              return null;
              }



              private void modificar(ActionEvent event) {
                    String query = "UPDATE listausuarios SET
ID='"+CAJAid.getText()+"',tipoUsuario='"+CAJAtipoUsuario.getText()+"',apellidoPa
terno='"+CAJAapellidoPater.getText()+"',nombre='"+CAJAnombres.getText()+"',Corre
o='"+CAJAcorreo.getText()+"' WHERE ID='"+CAJAid.getText()+"'";
                    executeQuery(query);
                    ObservableList<ClassUsuarios> list = getPersonList();
                    tablaUsuario.getItems().setAll(list);

                     final ObservableList<ClassUsuarios> tablaPersonas =
tablaUsuario.getSelectionModel().getSelectedItems();
                   tablaPersonas.addListener(selectortablaUsuarioos);

           }


              private void eliminar(ActionEvent event) {
                    String query = "DELETE FROM listausuarios WHERE
ID="+CAJAid.getText()+"";
```

```
                    executeQuery(query);
                    ObservableList<ClassUsuarios> list = getPersonList();
                    tablaUsuario.getItems().setAll(list);

                     final ObservableList<ClassUsuarios> tablaPersonas =
tablaUsuario.getSelectionModel().getSelectedItems();
                      tablaPersonas.addListener(selectortablaUsuarioos);
          }


      public void regresarLogin() {
            try {
                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLOGIN.fxml"))
;
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
                  primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
                  primaryLayout.show();
                  Stage nuevaEscena =(Stage) this.Salir.getScene().getWindow();
                  nuevaEscena.close();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
```

REGISTRO

En esta unidad contiene el funcionamiento de poder registrar a los usuarios que podrán acceder al sistema

Variables

```
      private Button Cancelar;
      private TextField CAJAnombre;
      private TextField CAJAapellidoP;
      private TextField CAJAcorreo;
      private TextField CAJAcontra;
      private TextField CAJAcontra2;
```

Implementación

```
public void regresarLogin() {
            try {
                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLOGIN.fxml"))
;
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
                  primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
                  primaryLayout.show();
                  Stage nuevaEscena =(Stage)
this.Cancelar.getScene().getWindow();
                  nuevaEscena.close();
```

```
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }


      public void agregar()throws SQLException{
            Connection cn = ConnectorMySQL.getConnection();
       try {
            System.out.println("HOLI registro usuario");

           String sSQLL = "INSERT INTO listausuarios
(apellidoPaterno,nombre,Correo,contrasena,contrasena2) VALUES(?,?,?,?,?)";
           PreparedStatement stt = cn.prepareStatement(sSQLL);
           stt.setString(1,(CAJAnombre.getText()));
           stt.setString(2,(CAJAapellidoP.getText()));
           stt.setString(3,(CAJAcorreo.getText()));
           stt.setString(4,(CAJAcontra.getText()));
           stt.setString(5,(CAJAcontra2.getText()));
           stt.execute();
       }catch (SQLException e){
           e.printStackTrace();
       }
       CAJAnombre.clear();
       CAJAapellidoP.clear();
       CAJAcorreo.clear();
       CAJAcontra.clear();
       CAJAcontra2.clear();
   }
```

REGISTROPRODUCTO

En esta unidad contiene el funcionamiento de poder registrar los productos que contendrá el sistema

Variables

```
      private Button Regresar;
      private Button Guardar;
      private TextField cajatipoProduct;
      private TextField cajaNombre;
      private TextField cajaPrecio;
      private TextField cajaFolio;
      private TextField cajanombreProveedor;
      private TextField CAJAnproductos;
```

Implementación

```
public void cargarListado() {
            try {
                  AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
                  Scene scene = new Scene (root2);
                  Stage primaryLayout = new Stage();
                  primaryLayout.setScene(scene);
```

```
                    primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }

      public void agregar()throws SQLException{
             Connection cn = ConnectorMySQL.getConnection();
        try {
             System.out.println("HOLI");
            String sSQLL = "INSERT INTO datosproduct
(tipoProduct,cantidadProduct, nombre,folioProduct,precio,nombreProveedor)
VALUES(?,?,?,?,?,?)";
            PreparedStatement stt = cn.prepareStatement(sSQLL);
            stt.setString(1,(cajatipoProduct.getText()));
            stt.setInt(2,Integer.parseInt(CAJAnproductos.getText()));
            stt.setString(3,cajaNombre.getText());
            stt.setString(4,cajaFolio.getText());
            stt.setFloat(5,Float.parseFloat(cajaPrecio.getText()));
            stt.setString(6,cajanombreProveedor.getText());
            stt.execute();
        }catch (SQLException e){
            e.printStackTrace();
        }
        cajatipoProduct.clear();
        cajaNombre.clear();
        cajaFolio.clear();
        cajaPrecio.clear();
        cajanombreProveedor.clear();
        CAJAnproductos.clear();
        cargarListado();
    }
```

REGISTROPROVEEDOR

En esta unidad contiene el funcionamiento de poder registrar a los proveedores los cuales proporcionan productos para el sistema.

Variables

```
      private Button Salir;
      private Button Guardar;
      private TextField CAJAnombreProvee;
      private TextField CAJAnombreProduct;
      private TextField CAJAfolioproduct;
      private Spinner<String> CAJAfolioproduct1;

      private Button Regresar;
```

Implementación

```
	public void regresarLogin() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
			primaryLayout.show();
			Stage nuevaEscena =(Stage) this.Salir.getScene().getWindow();
			nuevaEscena.close();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public void cargarListado() {
		try {
			AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
			Scene scene = new Scene (root2);
			Stage primaryLayout = new Stage();
			primaryLayout.setScene(scene);
			primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
			primaryLayout.show();
			Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
			nuevaEscena.close();

		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public void spinner1()throws SQLException {
		ObservableList<String> lista = FXCollections.observableArrayList();
		Connection cn = ConnectorMySQL.getConnection();
		String uno="Select nombre from datosproduct";
		Statement st;
		ResultSet rs;
		try {
			st = cn.createStatement();
			rs = st.executeQuery(uno);
			while(rs.next()) {
				lista.add(rs.getString("nombre"));

			}
```

```
                    CAJAfolioproduct1.setValueFactory(new
SpinnerValueFactory.ListSpinnerValueFactory<String>((ObservableList<String>)
lista));

             }catch (Exception e) {
                    e.printStackTrace();
             }
      }

      public void agregar()throws SQLException{
             Connection cn = ConnectorMySQL.getConnection();
        try {
             System.out.println("HOLI agregar proveedores");
            String sSQLL = "INSERT INTO listaproveedores
(nombreProveedor,nombreProduct,folioProduct) VALUES(?,?,?)";
            PreparedStatement stt = cn.prepareStatement(sSQLL);
            stt.setString(1,(CAJAnombreProvee.getText()));
            stt.setString(2,CAJAnombreProduct.getText());
            stt.setString(3,CAJAfolioproduct.getText());
//            stt.setFloat(4,Float.parseFloat(cajaPrecio.getText()));
//            stt.setString(5,cajanombreProveedor.getText());
            stt.execute();
        }catch (SQLException e){
            e.printStackTrace();
        }
        CAJAnombreProvee.clear();
        CAJAnombreProduct.clear();
        CAJAfolioproduct.clear();
//        cajaPrecio.clear();
//        cajanombreProveedor.clear();
    }

    public ControllerREGISTROproveedores() throws SQLException {
        spinner1();
    }
```

REPORTES

En esta unidad contiene el funcionamiento de poder registrar las ganancias que se tuvieron durante el día de venta.

Variables

```
      private Button Salir;

      private Button Regresar;
```

Implementación

```
public void cargarListado() {
             try {
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
                    Scene scene = new Scene (root2);
```

```
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }
```

TICKET

En esta unidad contiene el funcionamiento de poder registrar las ganancias que se tuvieron durante el día de venta.

Variables

```
      private Button Salir;

      private Button Regresar;
```

Implementación

```
      public void regresarLogin() {
            try {
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLOGIN.fxml"))
;
                    Scene scene = new Scene (root2);
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasLOGIN");
                    primaryLayout.show();
                    Stage nuevaEscena =(Stage) this.Salir.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }

      public void cargarListado() {
            try {
                    AnchorPane root2 =
(AnchorPane)FXMLLoader.load(getClass().getResource("FXMLPuntoVentasLISTADO.fxml"
));
                    Scene scene = new Scene (root2);
                    Stage primaryLayout = new Stage();
                    primaryLayout.setScene(scene);
                    primaryLayout.setTitle("FXMLPuntoVentasLISTADO");
                    primaryLayout.show();
```

```
                    Stage nuevaEscena =(Stage)
this.Regresar.getScene().getWindow();
                    nuevaEscena.close();

             } catch (Exception e) {
                    e.printStackTrace();
             }
      }
```

DICCIONARIO DE DATOS

Se tienen siete entidades en la base de datos las cuales son:

- Reportedeventas
  Esta entidad conlleva de ID que es un número que lo identifica y tiene los atributos fecha, folioReporte, cantidadArticulos, folioProduct, descripcionProduct, precioC/U y importetotal.

- Login
  Esta entidad conlleva los atributos NombreUsuario, Constrasenia, TipoUsuario y no tiene ninguna relación con otras tablas.

- Listausuarios
  Esta entidad conlleva un ID que es un número que lo identifica y es la clave, y los atributos que tiene son tipoUsuario, apellidoPaterno, nombre, Correo, Contrasena, contrasena2.

- Flujodecaja
  Esta entidad conlleva un FOLIO el cual es un número que lo identifica y sus atributos son fecha, montoEntrada, montoApagar, cambioAdar.

- Ticket
  Esta entidad conlleva un ID que es un número que lo identifica y es la clave, y los atributos que tiene son fecha, folioticket, cantidadproduct, descripcionproduct, precioC/U, importe, totalarticulos, subtotal, totalIVA, totalApagar, montorecibido, cambio, nombreusuario.

- Listaproveedores
  Esta entidad conlleva un IDtipoProduct que es un número que lo identifica y es la clave y es la que está relacionada con la tabla DatosProducto, y sus atributos son nombreProveedor, nombreProduct, folioProduct.

- Datosproduct
  Esta entidad contiene un IDtipoProduct el cual está relacionado con la tabla Listaproveedores que es un número y es la clave, y sus atributos tipoProduct, cantidadProduct, nombre, folioprouct, precio, nombreProveedor.