PROBLEMA A RESOLVER:

Auxiliara tiendas que ocupan un sistema de gestion que aborde varias funcionalidades, como ventas, compras, gestion de almace, generacion de reportes y mantenimiento de usuarios.

La problematica que busca resolver:

- Gestion de inventario: Garantizar que la tienda siempre tenga suficiente stock de productos para satisfacer la demanda
- Proceso de Ventas y Compras: Optimizar y hacer mas eficiente el proceso de ventas y compras registrando adecuadamente las transacciones y actualizando el inventario
- Generacion de Reportes: Proporciona informes detallados sobre las ventas, compras, y el estado actual del inventario para facilitar la toma de desiciones
- Mantenimiento de Usuarios: Gestionar los usuarios que interactuan con el sistema, lo que puede incluir empleados que realizan ventas y administradores que realizan funciones mas avanzadas.

Descripcion de la solucion:

La solucion a este problema puede estructurarse en varias clases, a continuacion la division en clases y conceptos:

Interfaces.

VentaOperaciones: Define las operaciones relacionadas con el proceso de ventas.
CompraOperaciones Define las operaciones relacionadas con el proceso de compras
AlmacenOperaciones: Define las operaciones relacionadas con el gestion del almacen
ReporteGenerador: Define las operaciones para generar diferentes tipos de infores
UsuarioOperaciones: Define las operaciones relacionadas con la gestion de usuarios

Clases Abstractas:
OperacionesBase: Puede contener metodos comunes a las operaciones de venta, compra, almacen, etc.

Clases Concretas:

VentanaVenta: Implementa la interfaz VentaOperaione sy gestiona las operaciones de venta
VentanaCompra: Implementa la interfaz CompraOperaciones y gestiona las operaciones de compra
Almacen: Implementa la interfaz AlmacenOperaciones y gestiona las operaciones del almacen
GeneradorDeReportes: Implementa la interfaz ReporteGenerador y se encarga de generar informes
GestorDeUsuarios: Implementa la interfaz UsuarioOperaciones y gestiona las operaciones relacionadas con usuarios

Clase Principal (Tienda):

Contiene el metodo main y se encarga de inicializaar y coordinar las diferenctes operaciones y ventanas de la tienda

Clases de Conexion y Utilidades:

Conexion: Gestiona la conexion a la base de datos.
EscuchadorBotones: Maneja los eventos de los botones en las Ventanas

Clase	Descripción	Métodos	Adicionales (Clases Internas, variables)
Tienda	La clase Tienda representa la interfaz gráfica de una aplicación de tienda. Fue creada para gestionar las operaciones de compra, venta, almacenamiento, generación de reportes y proporcionar ayuda al usuario. Además, gestiona la apertura de ventanas secundarias correspondientes a cada opción y permite cambiar la configuración de la base de datos	Constructor Tienda(): ● Descripción: Constructor de la clase Tienda que inicializa la interfaz gráfica y establece el tamaño de la ventana. También agrega un EscuchadorAccion a los botones de compra, venta, almacenamiento, ayuda y reportes, también inicializa un objeto Conexion para la gestión de la base de datos. ● Argumentos: Ninguno. ● Valor de retorno: Ninguno. Abrir(String ventana): ● Descripción: Abre la ventana correspondiente según el parámetro ventana proporcionado, que puede ser "Compra", "Venta", "Almacen", "Ayuda" o "Reportes". Realiza acciones específicas para cada caso, como la creación de nuevas ventanas y la visualización de paneles. ● Argumentos: ventana (String): El nombre de la ventana a abrir. ● Valor de retorno: Ninguno. mostrarPaneles(): ● Descripción: Hace visible los paneles de la interfaz gráfica. ● Argumentos: Ninguno. ● Valor de retorno: Ninguno main(String args[]): ● Descripción: Método principal que inicia la aplicación. Configura el aspecto visual y muestra la ventana principal de la tienda. ● Argumentos: args (String[]): Argumentos de la línea de comandos. ● Valor de retorno: Ninguno. abrirSubMenu(): ● Descripción: Abre una ventana secundaria para cambiar la configuración de la base de datos,	EscuchadorAccion (Método `actionPerformed(ActionEvent evento)'): ● Descripción: Clase que implementa la interfaz ActionListener y se utiliza para manejar los eventos de acción de los botones en la interfaz gráfica de la tienda. ● Argumentos: evento (ActionEvent): Evento de acción que desencadena la llamada al método. ● Valor de retorno: Ninguno. VerificadorProductos: ● Descripción: Clase que extiende Thread para verificar la cantidad de productos en el almacén a intervalos

		solicitando al usuario la URL, el nombre de usuario y la contraseña. ● Argumentos: Ninguno. ● Valor de retorno: Ninguno. cambiarBaseDeDatos(): ● Descripción: Cambia la configuración de la base de datos según la información ingresada por el usuario en la ventana de cambio de base de datos. ● Argumentos: Ninguno. ● Valor de retorno: Ninguno.	regulares. ● Métodos: ● run(): Método principal del hilo que verifica continuamente la cantidad de productos. ● verificarCantidadProductos(): Método privado que realiza la verificación de la cantidad de productos. ● Argumentos de entrada: Ninguno. ● Valor de retorno: Ninguno EscuchadorVentana: ● Descripción: Implementa la interfaz WindowListener para gestionar eventos de ventana. Se utiliza para cerrar la aplicación cuando se cierra la ventana principal. ● Métodos windowOpened(WindowEvent e), windowClosing(WindowEvent e), windowClosed(WindowEvent e), windowIconified(WindowEvent e), windowDeiconified(WindowEv

			ent e), windowActivated(WindowEvent e), windowDeactivated(WindowEvent e): ● Descripción: Métodos vacíos que implementan la interfaz WindowListener. Se utilizan para manejar diferentes eventos de ventana, pero en este caso, solo windowClosing se utiliza para cerrar la aplicación. ● Argumentos: ● e (WindowEvent): Evento de ventana que desencadenó el método. ● Valor de retorno: Ninguno. .
Almacen	La clase Almacen representa un panel de interfaz de usuario que muestra la información sobre los productos almacenados en una tienda. Permite visualizar la lista de productos disponibles, así	Constructor Almacen(Tienda tienda, Conexion Conexion): ● Descripción: Inicializa la instancia de la clase Almacen. ● Argumentos: tienda (Tienda): Instancia de la clase Tienda a la que está asociado el almacén, Conexion (Conexion): Instancia de la Clase Conexion a la que esta asociado el almacen. ● Valor de Retorno: Ninguno Regresa(ActionEvent evento)r: ● Descripción: Oculta el panel actual de Almacen y muestra los paneles principales de la tienda. ● Argumentos: evento (ActionEvent): Evento de acción que desencadena la llamada al método.	

	como agregar nuevos productos al panel.	● Valor de retorno: Ninguno. llenarTabla(): ● Descripción: Llena la tabla en el panel con la información de los productos almacenados en la base de datos. ● Argumentos: Ninguno. ● Valor de retorno: Ninguno. calcularDinero ● Descripción: Calcula la suma total del dinero generado por las ventas y muestra el resultado en un campo de texto. ● Argumentos de entrada: Ninguno. ● Valores de retorno: Ninguno.	
Conexion	La clase Conexion proporciona métodos para establecer y gestionar la conexión a una base de datos MySQL. Incluye un método para establecer la conexión y otro para obtener la conexión.	conectar(Connection connection): ● Descripción: Establece la conexión a la base de datos utilizando los parámetros de conexión especificados en la clase. ● Argumentos (Entrada): Connection connection - Objeto de conexión a la base de datos que se actualizará con la nueva conexión. ● Valor de retorno: Ninguno. ● Excepciones: SQLException: Excepción lanzada en caso de error durante la conexión. Connection conectar() ● Descripción: Establece una conexión a la base de datos y devuelve el objeto de conexión. ● Argumentos (Entrada): Ninguno. ● Valor de Retorno: Objeto Connection que representa la conexión a la base de datos. Devuelve null en caso de error. setURL(String url) ● Descripción: Establece la URL de conexión a la base de datos. ● Argumentos (Entrada): ● String url: Nueva URL de conexión. ● Valor de Retorno: Ninguno. setUser(String user) ● Descripción: Establece el nombre de usuario para la conexión a la base de datos. ● Argumentos (Entrada): String user: Nuevo nombre de usuario. ● Valor de Retorno: Ninguno. setPass(String contrasena) ● Descripción: Establece la contraseña para la	Variables de la clase: ● URL: La URL de conexión a la base de datos MySQL. ● usuario: El nombre de usuario para la conexión a la base de datos. ● contrasena: La contraseña para la conexión a la base de datos. ● controler: El controlador JDBC utilizado para la conexión.

		conexión a la base de datos. ● Argumentos (Entrada): String contrasena: Nueva contraseña. ● Valor de Retorno: Ninguno.	
Ventana Venta	La clase VentanaVenta representa la interfaz gráfica de una ventana de venta en un sistema de punto de venta. Esta ventana permite agregar productos, calcular el subtotal y total de la venta, así como realizar el cobro y actualizar la cantidad de productos en la base de datos. La clase tiene métodos para agregar productos, realizar el cobro y actualizar la cantidad de productos en la base de datos.	Constructor: VentanaVenta(Conexion Conexion) ● Descripción:Constructor de la clase VentanaVenta. Inicializa la ventana de venta con los componentes gráficos y establece la conexión a la base de datos. ● Argumentos:Conexion Conexion: Objeto de conexión a la base de datos. ● Valores de Retorno:Ninguno. initComponets() ● Descripción:Inicializa y configura los componentes gráficos de la ventana de venta, como paneles, etiquetas, botones y la tabla de productos. ● Argumentos:Ninguno. ● Valores de Retorno:Ninguno. crearTabla() ● Descripción:Crea y configura la tabla de productos donde se mostrarán los detalles de la venta. ● Argumentos:Ninguno. ● Valores de Retorno:Ninguno. agregarPaneles() ● Descripción:Agrega los paneles y botones necesarios a la interfaz gráfica de la ventana de venta. ● Argumentos:Ninguno. ● Valores de Retorno:Ninguno. cobrar() ● Descripción:Realiza el cobro de la venta, actualizando la cantidad de productos en la base de datos y registrando la transacción en la tabla 'dinero'. ● Argumentos:Ninguno. ● Valores de Retorno:Ninguno. agregarProducto() ● Descripción: Agrega un producto a la tabla de productos en la venta, verificando la existencia y disponibilidad del mismo en la base de datos. ● Argumentos: Ninguno. ● Valores de Retorno: Ninguno.	Clase: EscuchadorBotones ● Descripción:La clase EscuchadorBotones implementa la interfaz ActionListener para manejar eventos de botones en la clase VentanaVenta. ● Método: actionPerformed(ActionEvent evt) ● Descripción:Maneja los eventos de los botones en la interfaz gráfica de la ventana de venta. ● Argumentos: ActionEvent evt: Evento del botón. ● Valores de Retorno: Ninguno.

		obtenerCobro() ● Descripción: Calcula y muestra el subtotal y total de la venta en la interfaz gráfica. ● Argumentos: Ninguno. ● Valores de Retorno: Ninguno. ExisteProducto(String id, Connection cn, PreparedStatement pst) ● Descripción: ● Verifica la existencia y disponibilidad de un producto en la base de datos. ● Argumentos:String id: Identificador del producto, Connection cn: Objeto de conexión a la base de datos,PreparedStatement pst: Objeto de declaración preparada. ● Valores de Retorno: ○ true: Si el producto existe y está disponible. ○ false: Si el producto no existe o no está disponible. actualizarDinero() ● Descripción: Actualiza la tabla 'dinero' en la base de datos con la cantidad de dinero correspondiente al subtotal de la venta. ● Argumentos:Ninguno. ● Valores de Retorno:Ninguno.	
Ventana Compra		2.1 Constructor VentanaCompra(Conexion Conexion): ● Descripción: Constructor de la clase que inicializa la ventana de compra. ● Argumentos: ● Conexion Conexion: Objeto que representa la conexión a la base de datos. ● Valores de Retorno: Ninguno. realizarCompra(): ● Descripción: Realiza una compra al proveedor e imprime un mensaje indicando que la compra fue realizada. ● Argumentos: cant: Un entero que representa la cantidad de productos a comprar. ● Valor de retorno: Un valor booleano que siempre es true para indicar que la compra fue realizada con éxito. SetImageButton(): ● Descripción: Configura la imagen de un botón utilizando la ruta de la imagen proporcionada.	

		• Argumentos:labelName: Un objeto JButton al cual se le configurará la imagen, root: Una cadena de texto que representa la ruta de la imagen. • Valor de retorno: Ninguno . calcularSubtotal(): • Definición: Este método realiza el cálculo del subtotal, el IVA y el total de la compra con base en los datos proporcionados en una tabla. Utiliza un modelo de tabla (jTable2) para obtener la cantidad y el precio de cada producto, multiplicándolos y acumulándolos para obtener el subtotal. Luego, calcula el IVA como el 16% del subtotal y suma el IVA al subtotal para obtener el total. Finalmente, actualiza tres campos de texto en la interfaz de usuario (jTextFieldSubtotal, jTextFieldIVA, jTextFieldTotal) con los resultados calculados. • Argumentos:Ninguno • Valores de Retorno:Ninguno • Manejo de Excepciones:Si la cantidad o el precio no son valores numéricos válidos, se muestra un mensaje de error indicando la fila problemática. llenarComboBoxProveedores(): • Descripción: Llena un JComboBox con nombres de proveedores obtenidos desde la base de datos. • Argumentos: Ninguno • Valores de Retorno: Ninguno Método : llenarComboBoxProducto() • Descripción: Llena un JComboBox con nombres de productos obtenidos desde la base de datos. • Argumentos: Ninguno • Valores de Retorno: Ninguno agregarProveedor(String proveedor, String nombre, int precio): • Descripción: Agrega un nuevo proveedor a la base de datos, verificando previamente si el producto ya existe. Actualiza modelos de JComboBox y la tabla de productos. • Argumentos:proveedor (String): Nombre del proveedor,nombre (String): Nombre del producto,precio (int): Precio del producto. • Valores de Retorno: Ninguno llenarTabla(): • Descripción: Llena una tabla con información de proveedores, productos y precios obtenidos desde la base de datos.	

		● Argumentos: Ninguno ● Valores de Retorno: Ninguno llenarTabla2(String sql): ● Descripción: Llena una segunda tabla con información específica obtenida mediante una consulta SQL. ● Argumentos:sql (String): Consulta SQL para obtener datos específicos. ● Valores de Retorno: Ninguno sumarCantidadesEnBaseDeDatos(String proveedor, String producto, float cantidadComprada): ● Descripción: Registra la cantidad comprada en la base de datos, actualizando la tabla "dinero". ● Argumentos:proveedor (String): Nombre del proveedor,producto (String): Nombre del producto,cantidadComprada (float): Cantidad comprada del producto. ● Valores de Retorno: Ninguno actualizarTablaSQL(): ● Descripción: Actualiza la tabla de productos en la interfaz gráfica con datos de la base de datos. ● Argumentos: Ninguno ● Valores de Retorno: Ninguno restarDineroEnBaseDeDatos(): ● Descripción: Este método realiza la operación de restar una cantidad específica de dinero de la base de datos, asociada a un proveedor y un producto. Verifica que haya suficiente dinero en la base de datos antes de realizar la operación y muestra mensajes de éxito o fracaso según el resultado. ● Argumentos:String proveedor: El nombre del proveedor asociado a la transacción,String producto: El nombre del producto asociado a la transacción,float total: La cantidad de dinero que se restará de la base de datos. ● Valores de Retorno: Este método no tiene un valor de retorno explícito. Muestra mensajes de éxito o fracaso mediante ventanas emergentes (JOptionPane). obtenerSumaActual(): ● Descripción: Este método obtiene la suma actual de la columna Cantidad_de_Dinero en la base de datos. ● Argumentos:Connection conexion: La conexión a la base de datos.	

		• Valores de Retorno: float sumaActual: La suma actual de la columna Cantidad_de_Dinero en la base de datos.