HITO 2 1ER TRIMESTRE ACCESO A DATOS

Realizado por: Ángel Gallego Felipe

20/11/2024

Campus FP Getafe

Desarrollo de aplicaciones multiplataforma

Contenido

ESPECIFICACIONES

Te acaban de contratar como desarrollador junior en la empresa Tech Solutions S.A., una compañía dedicada a ofrecer soluciones tecnológicas personalizadas para clientes de diferentes sectores. Recientemente, uno de sus clientes principales, BookWorld, una librería con un amplio catálogo de libros y artículos relacionados ha solicitado el desarrollo de una nueva aplicación para gestionar de manera eficiente sus inventarios, ventas y clientes.

Requisitos del cliente:

1. Gestión de inventario.
2. Gestión de clientes.
3. Ventas y facturación.
4. Consultas y reportes.

Tu Tarea:

Como parte del equipo de desarrollo, te han asignado el diseño y la implementación de una aplicación basada en bases de datos relacionales que satisfaga las necesidades descritas por el cliente.

Para lograrlo, deberás seguir los siguientes pasos:

1. Diseño de la BD: Diseña una base de datos (diseños conceptual, lógico y físico) que sirva para satisfacer los requisitos de usuario dentro del escenario que has planteado. Carga datos de prueba.
2. Aplicación: Realiza una aplicación completa funcional (CRUD) en Java para dos de los requisitos del cliente. No es necesario que implementes la aplicación completa. Puedes elegir uno de los tres primeros requisitos y el requisito nº 4.
3. Pruebas: Realiza una batería de pruebas para buscar posibles errores, documenta esta batería de pruebas. Después corrige los errores localizados. Documenta el proceso.

INTRODUCCIÓN

En un entorno empresarial en constante evolución, las organizaciones buscan herramientas tecnológicas que optimicen sus procesos internos y fortalezcan la experiencia del cliente. En este contexto, BookWorld, una librería con un extenso catálogo de libros y artículos relacionados, ha planteado la necesidad de una solución digital para gestionar de forma eficiente sus operaciones principales: inventarios, clientes, ventas y generación de reportes.

Este proyecto tiene como objetivo el desarrollo de una aplicación basada en bases de datos relacionales que permita a BookWorld integrar y automatizar estas áreas clave. Para su implementación, se han seleccionado tecnologías robustas y ampliamente utilizadas en el sector del desarrollo de software. Entre ellas destacan:

Java como lenguaje principal para la lógica del sistema, debido a su versatilidad y potencia en aplicaciones empresariales.

MySQL como sistema gestor de bases de datos, ofreciendo un enfoque estructurado y escalable para la organización de la información.

JDBC (Java Database Connectivity) para establecer la comunicación entre la aplicación y la base de datos.

La solución se estructura en cuatro módulos principales:

Gestión de Inventario: Permite el registro, actualización y consulta de los productos disponibles, asegurando un control detallado de los recursos.

Gestión de Clientes: Facilita la administración de información de los clientes, incluyendo datos personales y de contacto.

Ventas y facturación: Aquí tenemos toda la información referente a las ventas realizadas con todo los detalles del pedido para llevar una contabilidad excelente.

Consultas y Reportes: Proporciona a los usuarios reportes personalizados sobre el estado del inventario, historial de ventas y datos relevantes para la toma de decisiones.

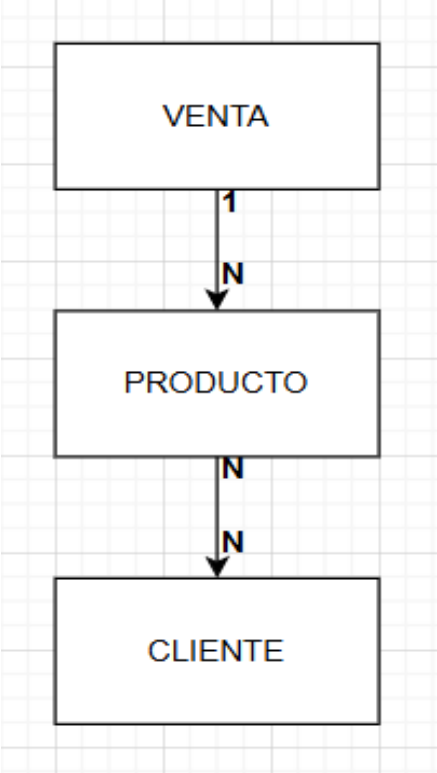
Además, se ha diseñado una base de datos siguiendo un enfoque metodológico que incluye los modelos conceptual, lógico y físico, garantizando una estructura clara y funcional. El sistema incluye operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para las entidades principales, asegurando la flexibilidad necesaria para cumplir con las expectativas del cliente.

A lo largo del desarrollo, se documentan las partes fundamentales del código, destacando la implementación de las capas de la aplicación, como la conexión a la base de datos mediante JDBC y los procesos de consulta y manipulación de datos. Por último, se realizarán pruebas exhaustivas para garantizar que el sistema funcione sin errores, documentando los hallazgos y las correcciones necesarias.

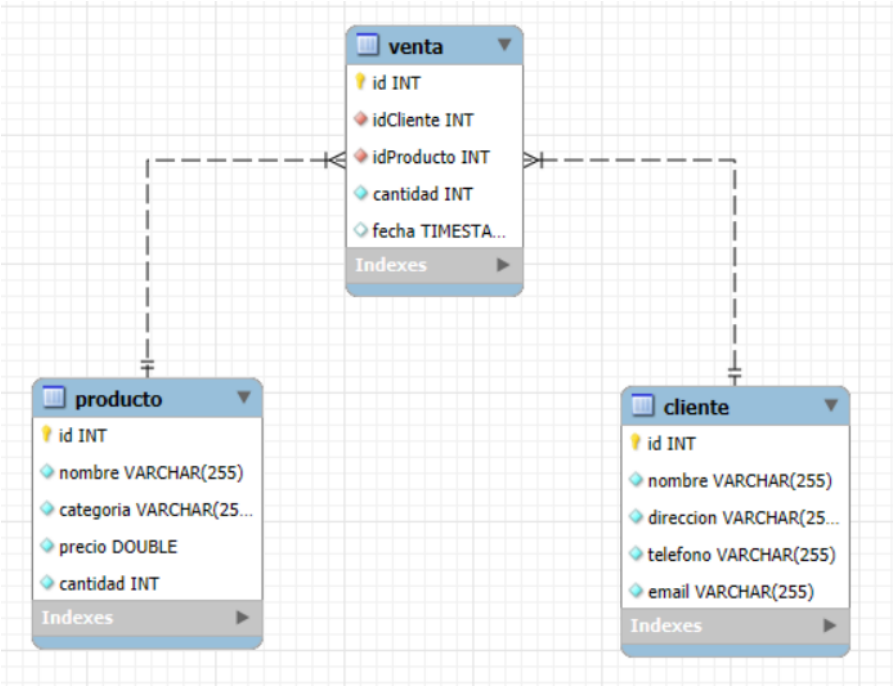
Este proyecto no solo busca satisfacer las necesidades de BookWorld, sino también servir como base para futuras ampliaciones, aportando una solución sólida, escalable y alineada con las mejores prácticas del desarrollo de software.

DISEÑO LOGICO Y CONCEPTUAL

- Diseño conceptual:



- Diseño lógico:



EXPLICACION DEL CODIGO MAS RELEVANTE

A continuación, procederé a explicar las partes más relevantes del código y la funcionalidad de la app, poniendo especial énfasis en las partes más interesantes del mismo.

En primer lugar, explicare los componentes fundamentales, luego las operaciones específicas de datos, y finalmente la interfaz de usuario que integra todo.

-ConexiónBaseDatos:

```
3   import java.sql.Connection;
4   import java.sql.DriverManager;
5   import java.sql.SQLException;
6
7   public class ConexionBaseDatos {  14 usages
8       private static final String URL = "jdbc:mysql://localhost:3306/Tienda";
9       private static final String USUARIO = "root";  1 usage
10      private static final String CONTRASEÑA = "curso";  1 usage
11
12      public static Connection obtenerConexion() throws SQLException {  14 usage
13          return DriverManager.getConnection(URL, USUARIO, CONTRASEÑA);
14      }
15  }
```

El método obtenerConexion se encarga de conectarse a la base de datos especificada en la constante URL. Para ello:

- Utiliza el controlador JDBC (DriverManager).
- Proporciona las credenciales (usuario y contraseña) necesarias.

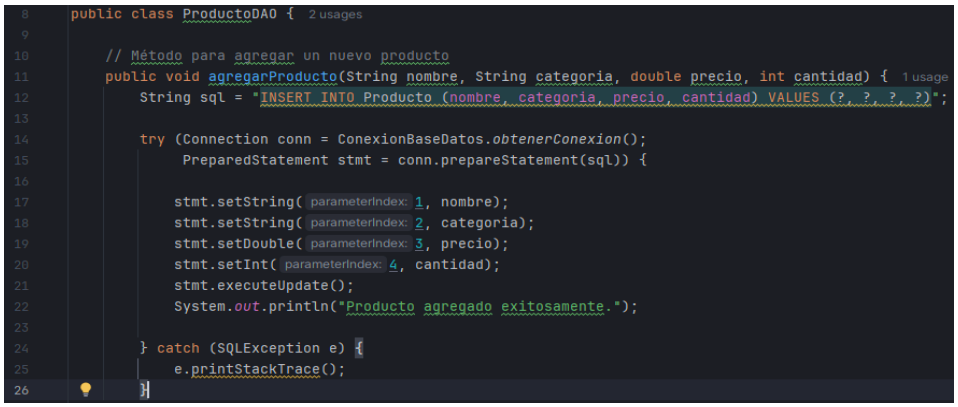
- ProductoDAO:

La clase ProductoDAO permite agregar, actualizar, eliminar y listar productos en la base de datos mediante una serie de métodos.

Cada uno de estos métodos interactúa con la base de datos a través de sentencias SQL preparadas, lo que proporciona una forma segura y eficiente de manipular los datos.

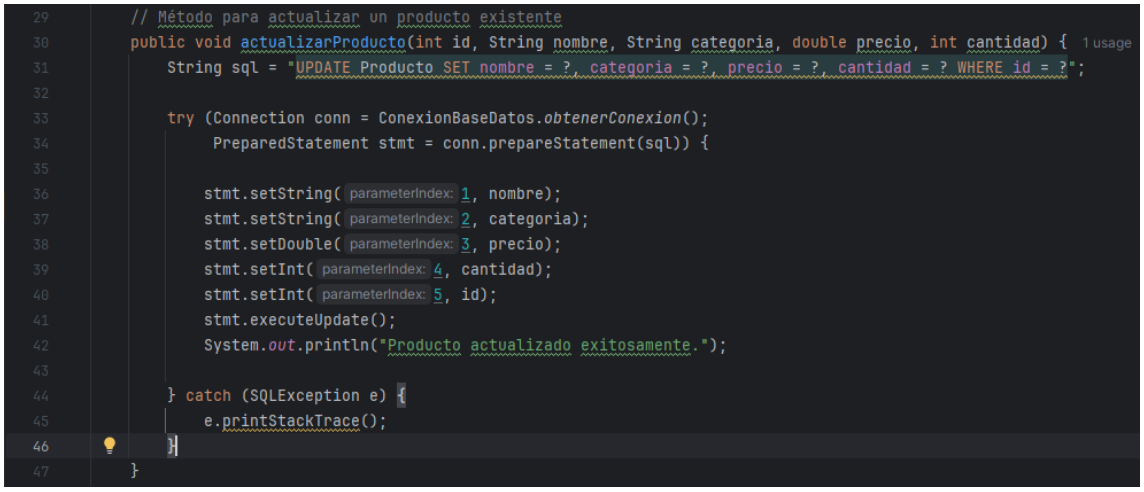
A continuación, aportare las imágenes del código que nos permiten realizar estas funciones de CRUD con esta clase.

Método “agregarProducto”: Este método agrega un nuevo producto a la tabla Producto



```
public class ProductoDAO {  2 usages

    // Método para agregar un nuevo producto
    public void agregarProducto(String nombre, String categoria, double precio, int cantidad) {  1 usage
        String sql = "INSERT INTO Producto (nombre, categoria, precio, cantidad) VALUES (?, ?, ?, ?)";

        try (Connection conn = ConexionBaseDatos.obtenerConexion();
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString( parameterIndex: 1, nombre);
            stmt.setString( parameterIndex: 2, categoria);
            stmt.setDouble( parameterIndex: 3, precio);
            stmt.setInt( parameterIndex: 4, cantidad);
            stmt.executeUpdate();
            System.out.println("Producto agregado exitosamente.");

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

Método “actualizarProducto”: Este método actualiza un producto existente, identificado por su id, en la tabla Producto. La consulta SQL utiliza los parámetros para los valores de los campos a actualizar.



```
    // Método para actualizar un producto existente
    public void actualizarProducto(int id, String nombre, String categoria, double precio, int cantidad) {  1 usage
        String sql = "UPDATE Producto SET nombre = ?, categoria = ?, precio = ?, cantidad = ? WHERE id = ?";

        try (Connection conn = ConexionBaseDatos.obtenerConexion();
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString( parameterIndex: 1, nombre);
            stmt.setString( parameterIndex: 2, categoria);
            stmt.setDouble( parameterIndex: 3, precio);
            stmt.setInt( parameterIndex: 4, cantidad);
            stmt.setInt( parameterIndex: 5, id);
            stmt.executeUpdate();
            System.out.println("Producto actualizado exitosamente.");

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

Método “eliminarProducto”: Este método elimina un producto de la tabla Producto

```
    // Método para eliminar un producto
    public void eliminarProducto(int id) {  1 usage
        String sql = "DELETE FROM Producto WHERE id = ?";

        try (Connection conn = ConexionBaseDatos.obtenerConexion();
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt( parameterIndex: 1, id);
            stmt.executeUpdate();
            System.out.println("Producto eliminado exitosamente.");

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

- ClienteDAO:

La clase ClienteDAO gestiona las operaciones relacionadas con los clientes en la base de datos. Se centra en tres funcionalidades principales: agregar, actualizar y eliminar clientes. Cada uno de estos métodos interactúa con la base de datos a través de sentencias SQL preparadas, lo que proporciona una forma segura y eficiente de manipular los datos.

Método “agregarCliente”: Este método agrega un cliente de la tabla cliente.

```
10      // Método para agregar un nuevo cliente
11      public void agregarCliente(String nombre, String direccion, String telefono, String email) {  1 usage
12          String sql = "INSERT INTO Cliente (nombre, direccion, telefono, email) VALUES (?, ?, ?, ?)";
13
14          try (Connection conn = ConexionBaseDatos.obtenerConexion();
15               PreparedStatement stmt = conn.prepareStatement(sql)) {
16
17              stmt.setString( parameterIndex: 1, nombre);
18              stmt.setString( parameterIndex: 2, direccion);
19              stmt.setString( parameterIndex: 3, telefono);
20              stmt.setString( parameterIndex: 4, email);
21              stmt.executeUpdate();
22              System.out.println("Cliente agregado exitosamente.");
23
24          } catch (SQLException e) {
25              e.printStackTrace();
26          }
```

Método “actualizarCliente”: Este método actualiza un cliente de la tabla cliente.

```
29      // Método para actualizar un cliente existente
30      public void actualizarCliente(int id, String nombre, String direccion, String telefono, String email) {  1 usage
31          String sql = "UPDATE Cliente SET nombre = ?, direccion = ?, telefono = ?, email = ? WHERE id = ?";
32
33          try (Connection conn = ConexionBaseDatos.obtenerConexion();
34               PreparedStatement stmt = conn.prepareStatement(sql)) {
35
36              stmt.setString( parameterIndex: 1, nombre);
37              stmt.setString( parameterIndex: 2, direccion);
38              stmt.setString( parameterIndex: 3, telefono);
39              stmt.setString( parameterIndex: 4, email);
40              stmt.setInt( parameterIndex: 5, id);
41              stmt.executeUpdate();
42              System.out.println("Cliente actualizado exitosamente.");
43
44          } catch (SQLException e) {
45              e.printStackTrace();
46          }
```

Método “eliminar clientes”: Este método elimina un cliente de la tabla cliente.

```
49      // Método para eliminar un cliente
50      public void eliminarCliente(int id) throws SQLException {  1 usage
51          Connection conexion = ConexionBaseDatos.obtenerConexion();
52          try {
53              // Eliminar filas relacionadas en la tabla Venta
54              String sqlEliminarVentas = "DELETE FROM Venta WHERE idCliente = ?";
55              PreparedStatement stmtEliminarVentas = conexion.prepareStatement(sqlEliminarVentas);
56              stmtEliminarVentas.setInt( parameterIndex: 1, id);
57              stmtEliminarVentas.executeUpdate();
58
59              // Eliminar el Cliente
60              String sqlEliminarCliente = "DELETE FROM Cliente WHERE id = ?";
61              PreparedStatement stmtEliminarCliente = conexion.prepareStatement(sqlEliminarCliente);
62              stmtEliminarCliente.setInt( parameterIndex: 1, id);
63              stmtEliminarCliente.executeUpdate();
64              System.out.println("Cliente eliminado exitosamente.");
65          } finally {
66              if (conexion != null) {
67                  conexion.close();
68              }
```

Método “listarClietnes”: Este método nos permite mostrar todos los clientes registrados en la base de datos.

```java
// Método para listar todos los clientes
public void listarClientes() {  1 usage
    String sql = "SELECT * FROM Cliente";

    try (Connection conn = ConexionBaseDatos.obtenerConexion();
         PreparedStatement stmt = conn.prepareStatement(sql);
         ResultSet rs = stmt.executeQuery()) {

        System.out.println("\n=== Listado de Clientes ===");
        while (rs.next()) {
            int id = rs.getInt( columnLabel: "id");
            String nombre = rs.getString( columnLabel: "nombre");
            String direccion = rs.getString( columnLabel: "direccion");
            String telefono = rs.getString( columnLabel: "telefono");
            String email = rs.getString( columnLabel: "email");

            System.out.printf("ID: %d, Nombre: %s, Dirección: %s, Teléfono: %s, Email: %s%n",
                    id, nombre, direccion, telefono, email);
        }

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

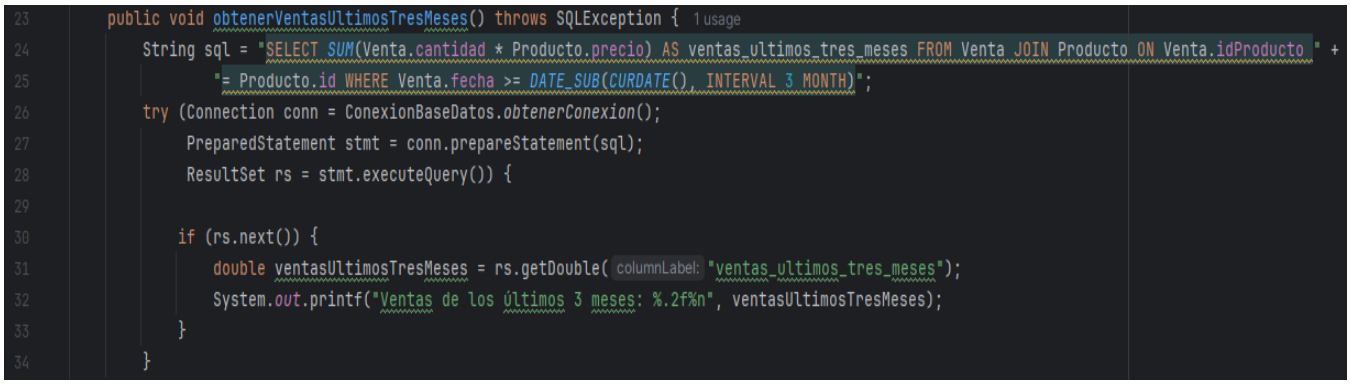
- VentaDAO:

La clase VentaDAO proporciona métodos para gestionar las operaciones relacionadas con las ventas en la base de datos. Esto incluye registrar ventas, obtener estadísticas y listar transacciones. A continuación, se detalla la funcionalidad de cada método:

Método “obtenerVentasTotales”: Calcula el total de ingresos generados por todas las ventas.

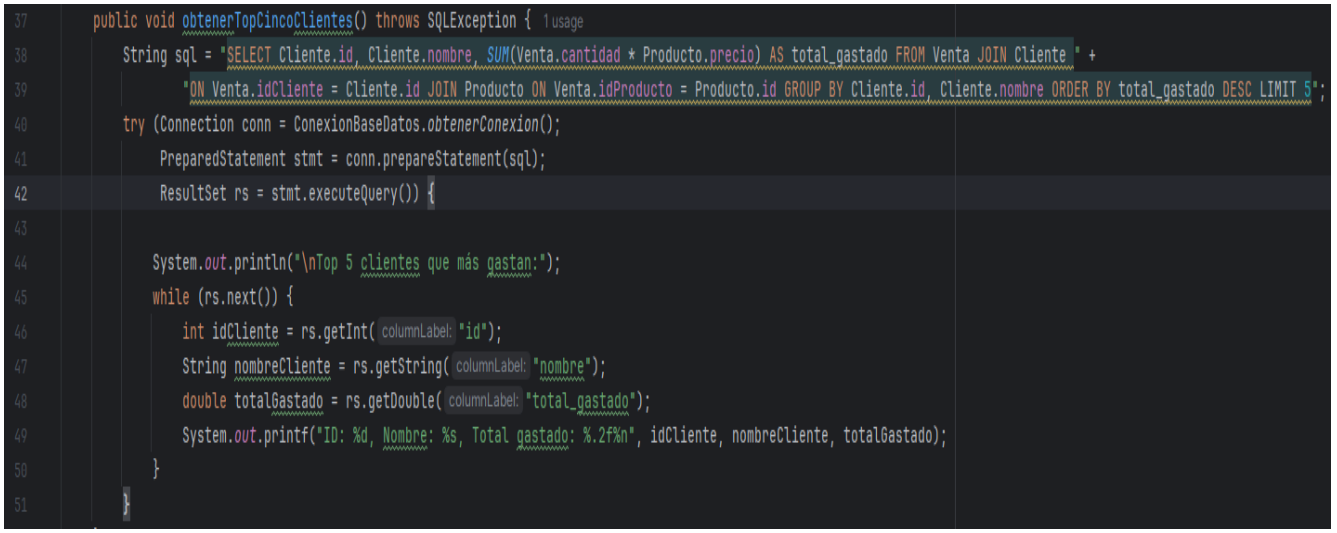
```java
public void obtenerVentasTotales() throws SQLException {  1 usage
    String sql = "SELECT SUM(Venta.cantidad * Producto.precio) AS ventas_totales FROM Venta JOIN Producto ON Venta.idProducto = Producto.id";
    try (Connection conn = ConexionBaseDatos.obtenerConexion();
         PreparedStatement stmt = conn.prepareStatement(sql);
         ResultSet rs = stmt.executeQuery()) {

        if (rs.next()) {
            double ventasTotales = rs.getDouble( columnLabel: "ventas_totales");
            System.out.printf("Ventas totales: %.2f%n", ventasTotales);
        }
    }
```

Método “obtenerVentasUltimosTresMeses”: Obtiene el total de ingresos generados en los últimos 3 meses. Filtra las ventas realizadas desde una fecha calculada usando DATE_SUB(CURDATE(), INTERVAL 3 MONTH).

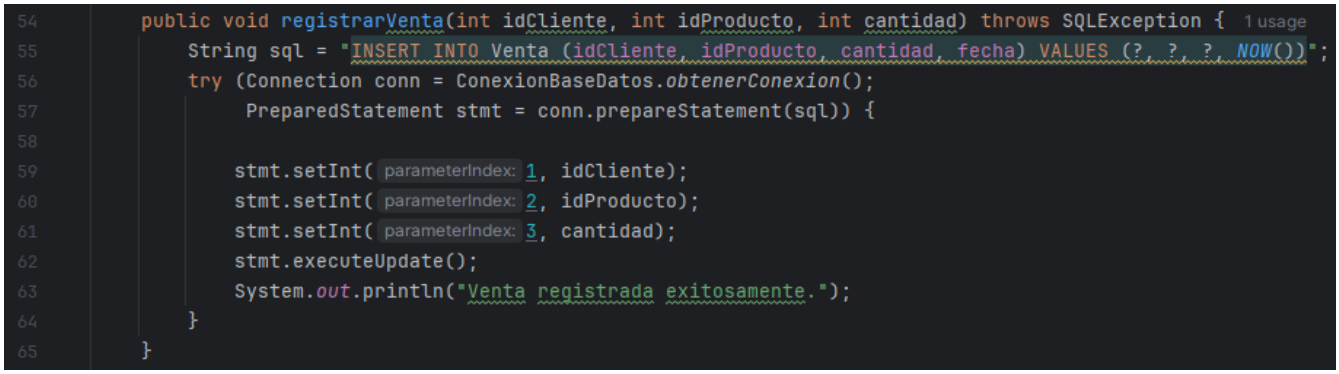


Método “obtenerTopCincoClientes”: Obtiene los 5 clientes que han gastado más en total, ordenados de mayor a menor gasto, combina las tablas Venta, Cliente y Producto.

Agrupa por cliente y calcula el total gastado usando SUM(Venta.cantidad * Producto.precio)



Método “registrarVenta”: Registra una nueva venta en la tabla Venta. Inserta el cliente (idCliente), producto (idProducto), cantidad vendida y la fecha actual (NOW()).



Método “listarVentas”: Lista todas las ventas realizadas, mostrando información como: ID de la venta, nombre del cliente, nombre del producto, cantidad vendida, fecha de la venta.

```java
public void listarVentas() throws SQLException {  1 usage
    String sql = "SELECT Venta.id, Cliente.nombre AS nombre_cliente, Producto.nombre AS nombre_producto, Venta.cantidad, " +
            "Venta.fecha FROM Venta JOIN Cliente ON Venta.idCliente = Cliente.id JOIN Producto ON Venta.idProducto = Producto.id";
    try (Connection conn = ConexionBaseDatos.obtenerConexion();
         PreparedStatement stmt = conn.prepareStatement(sql);
         ResultSet rs = stmt.executeQuery()) {

        System.out.println("\n=== Listado de Ventas ===");
        while (rs.next()) {
            int id = rs.getInt( columnLabel: "id");
            String nombreCliente = rs.getString( columnLabel: "nombre_cliente");
            String nombreProducto = rs.getString( columnLabel: "nombre_producto");
            int cantidad = rs.getInt( columnLabel: "cantidad");
            String fecha = rs.getString( columnLabel: "fecha");

            System.out.printf("ID: %d, Cliente: %s, Producto: %s, Cantidad: %d, Fecha: %s%n",
                    id, nombreCliente, nombreProducto, cantidad, fecha);
        }
```

- MenuPrincipal:

Paquete y clases DAO: ProductoDAO, ClienteDAO, y VentaDAO: Son clases encargadas de realizar operaciones con la base de datos (CRUD: Crear, Leer, Actualizar y Eliminar) para productos, clientes y ventas respectivamente. Estas clases interactúan con la capa de datos, pero el código específico de estas clases no está incluido aquí.

MenuPrincipal: Clase principal que proporciona un menú interactivo para el usuario, organizando las funcionalidades en diferentes secciones.

Uso de Scanner: Permite capturar entradas desde el teclado para interactuar con el usuario en tiempo real.

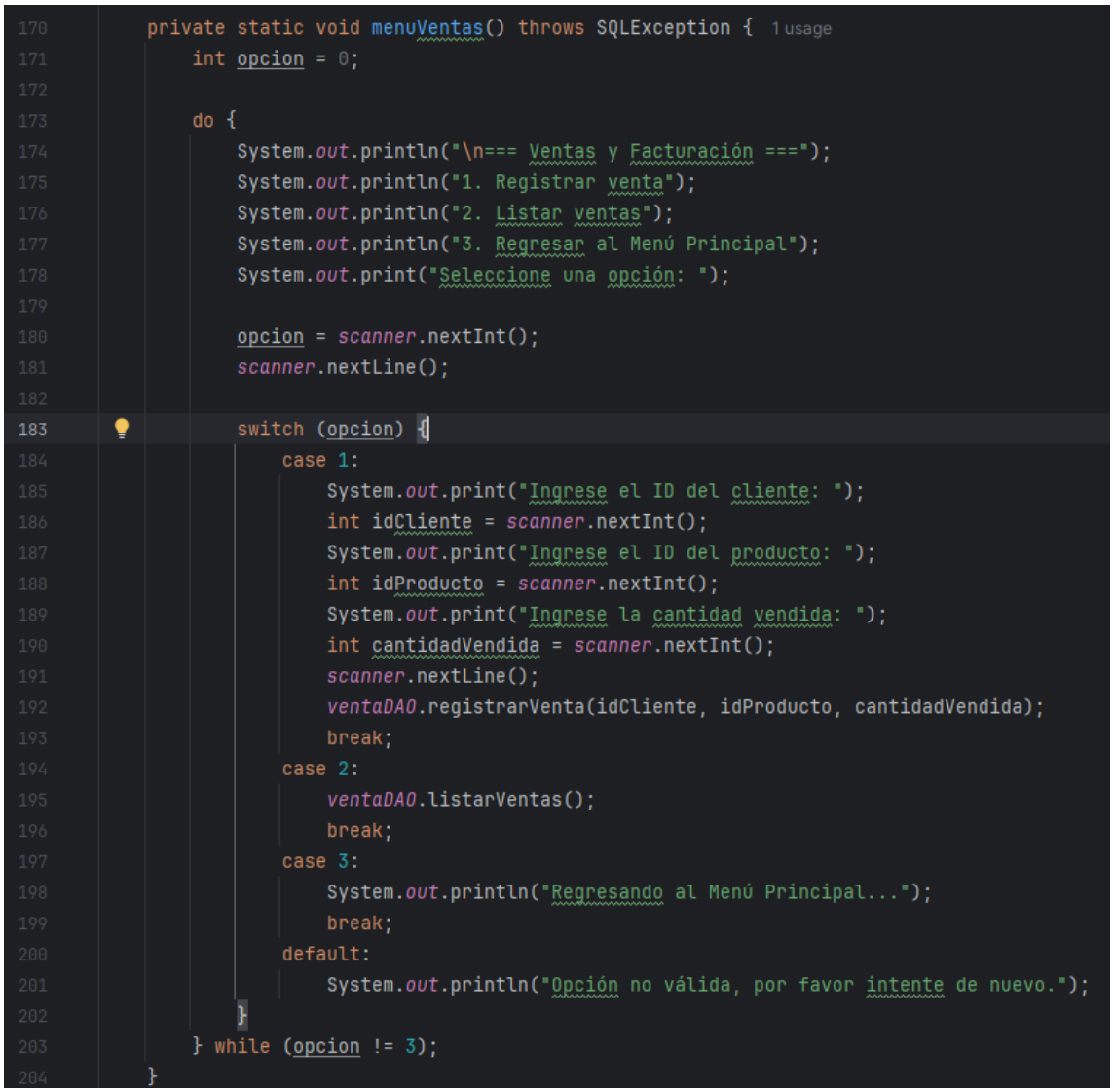
Manejo de excepciones: Los métodos manejan potenciales errores que puedan surgir durante la interacción con la base de datos mediante SQLException

Este fragmento de código declara e inicializa cuatro variables estáticas que son utilizadas a lo largo de la clase para manejar operaciones específicas relacionadas con productos, clientes, ventas, y para capturar entradas desde el teclado.

```java
private static ProductoDAO productoDAO = new ProductoDAO();  4 usages
private static ClienteDAO clienteDAO = new ClienteDAO();  4 usages
private static VentaDAO ventaDAO = new VentaDAO();  5 usages
private static Scanner scanner = new Scanner(System.in);  40 usages
```

El menú principal este compuesto de varios submenús todos con la misma estructura o muy parecida, a continuación explicaré uno de ellos.

- menuVentas: El método menuVentas es una parte del sistema que gestiona la funcionalidad relacionada con las ventas y la facturación en la aplicación. Permite al usuario realizar operaciones específicas, como registrar nuevas ventas o listar las ventas existentes.



```
170     private static void menuVentas() throws SQLException {  1 usage
171         int opcion = 0;
172
173         do {
174             System.out.println("\n=== Ventas y Facturación ===");
175             System.out.println("1. Registrar venta");
176             System.out.println("2. Listar ventas");
177             System.out.println("3. Regresar al Menú Principal");
178             System.out.print("Seleccione una opción: ");
179
180             opcion = scanner.nextInt();
181             scanner.nextLine();
182
183             switch (opcion) {
184                 case 1:
185                     System.out.print("Ingrese el ID del cliente: ");
186                     int idCliente = scanner.nextInt();
187                     System.out.print("Ingrese el ID del producto: ");
188                     int idProducto = scanner.nextInt();
189                     System.out.print("Ingrese la cantidad vendida: ");
190                     int cantidadVendida = scanner.nextInt();
191                     scanner.nextLine();
192                     ventaDAO.registrarVenta(idCliente, idProducto, cantidadVendida);
193                     break;
194                 case 2:
195                     ventaDAO.listarVentas();
196                     break;
197                 case 3:
198                     System.out.println("Regresando al Menú Principal...");
199                     break;
200                 default:
201                     System.out.println("Opción no válida, por favor intente de nuevo.");
202             }
203         } while (opcion != 3);
204     }
```

LA APLICACIÓN

- Menu principal:

Ahora procederé a mostrar imágenes de la aplicación funcionando y realizando varias acciones por los diferentes menús que la componen.

Al ejecutar la aplicación desde el la clase “MenuPrincipal” esta seria el menú que vemos en primer lugar con las 5 posibles opciones a escoger en función de nuestras necesidades.

```
=== Menú Principal ===
1. Gestión de Inventario
2. Gestión de Clientes
3. Ventas y Facturación
4. Consultas y Reportes
5. Salir
Seleccione una opción:
```

1. Gestion de inventario:

```
=== Gestión de Inventario ===
1. Agregar producto
2. Actualizar producto
3. Eliminar producto
4. Listar productos
5. Regresar al Menú Principal
Seleccione una opción:
```

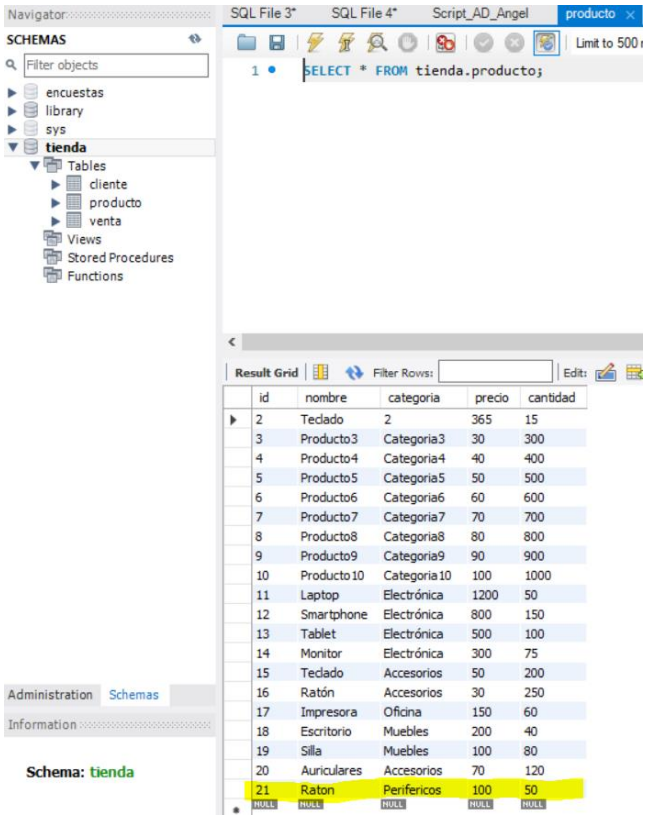
1.1 Agregar un producto a nuestro inventario

```
=== Gestión de Inventario ===
1. Agregar producto
2. Actualizar producto
3. Eliminar producto
4. Listar productos
5. Regresar al Menú Principal
Seleccione una opción: 1
Ingrese el nombre del producto: Raton
Ingrese la categoría del producto: Perifericos
Ingrese el precio del producto: 100
Ingrese la cantidad del producto: 50
Producto agregado exitosamente.
```

Y listamos los productos para comprobar que se añadió correctamente

```
ID: 19, Nombre: Silla, Categoría: Muebles, Precio: 100,00, Cantidad: 80
ID: 20, Nombre: Auriculares, Categoría: Accesorios, Precio: 70,00, Cantidad: 120
ID: 21, Nombre: Raton, Categoría: Perifericos, Precio: 100,00, Cantidad: 50
```

También podemos comprobarlo en la base de datos.



1.2 Actualizar un producto:

Actualizaremos ese mismo producto, cambiándole por ejemplo el precio y las unidades del stock.

```
=== Gestión de Inventario ===
1. Agregar producto
2. Actualizar producto
3. Eliminar producto
4. Listar productos
5. Regresar al Menú Principal
Seleccione una opción: 2
Ingrese el ID del producto a actualizar: 21
Ingrese el nuevo nombre del producto: Raton
Ingrese la nueva categoría del producto: Perifericos
Ingrese el nuevo precio del producto: 250
Ingrese la nueva cantidad del producto: 100
Producto actualizado exitosamente.
```

Y al volver a listar los productos de nuestro stock, vemos como efectivamente el producto fue actualizado.

```
ID: 19, Nombre: Silla, Categoría: Muebles, Precio: 100,00, Cantidad: 80
ID: 20, Nombre: Auriculares, Categoría: Accesorios, Precio: 70,00, Cantidad: 120
ID: 21, Nombre: Raton, Categoría: Perifericos, Precio: 250,00, Cantidad: 100
```

1.3 Eliminar un producto:

Ahora eliminaremos el producto que creamos anteriormente que es el ratón.

```
=== Gestión de Inventario ===
1. Agregar producto
2. Actualizar producto
3. Eliminar producto
4. Listar productos
5. Regresar al Menú Principal
Seleccione una opción: 3
Ingrese el ID del producto a eliminar: 21
Producto eliminado exitosamente.
```

En esta imagen podemos comprobar como el raton ya no se encuentra disponible en el stock al listar los productos

```
ID: 18, Nombre: Escritorio, Categoría: Muebles, Precio: 200,00, Cantidad: 40
ID: 19, Nombre: Silla, Categoría: Muebles, Precio: 100,00, Cantidad: 80
ID: 20, Nombre: Auriculares, Categoría: Accesorios, Precio: 70,00, Cantidad: 120
```

1.4 Listar productos:

Y por último en este menú, tenemos la opción “listar productos” que lo que hace es mostrar una lista de todos los productos disponibles en el stock

```
=== Gestión de Inventario ===
1. Agregar producto
2. Actualizar producto
3. Eliminar producto
4. Listar productos
5. Regresar al Menú Principal
Seleccione una opción: 4

=== Listado de Productos ===
ID: 2, Nombre: Teclado, Categoría: 2, Precio: 365,00, Cantidad: 15
ID: 3, Nombre: Producto3, Categoría: Categoria3, Precio: 30,00, Cantidad: 300
ID: 4, Nombre: Producto4, Categoría: Categoria4, Precio: 40,00, Cantidad: 400
ID: 5, Nombre: Producto5, Categoría: Categoria5, Precio: 50,00, Cantidad: 500
ID: 6, Nombre: Producto6, Categoría: Categoria6, Precio: 60,00, Cantidad: 600
ID: 7, Nombre: Producto7, Categoría: Categoria7, Precio: 70,00, Cantidad: 700
ID: 8, Nombre: Producto8, Categoría: Categoria8, Precio: 80,00, Cantidad: 800
ID: 9, Nombre: Producto9, Categoría: Categoria9, Precio: 90,00, Cantidad: 900
ID: 10, Nombre: Producto10, Categoría: Categoria10, Precio: 100,00, Cantidad: 1000
ID: 11, Nombre: Laptop, Categoría: Electrónica, Precio: 1200,00, Cantidad: 50
ID: 12, Nombre: Smartphone, Categoría: Electrónica, Precio: 800,00, Cantidad: 150
ID: 13, Nombre: Tablet, Categoría: Electrónica, Precio: 500,00, Cantidad: 100
ID: 14, Nombre: Monitor, Categoría: Electrónica, Precio: 300,00, Cantidad: 75
ID: 15, Nombre: Teclado, Categoría: Accesorios, Precio: 50,00, Cantidad: 200
ID: 16, Nombre: Ratón, Categoría: Accesorios, Precio: 30,00, Cantidad: 250
ID: 17, Nombre: Impresora, Categoría: Oficina, Precio: 150,00, Cantidad: 60
ID: 18, Nombre: Escritorio, Categoría: Muebles, Precio: 200,00, Cantidad: 40
ID: 19, Nombre: Silla, Categoría: Muebles, Precio: 100,00, Cantidad: 80
ID: 20, Nombre: Auriculares, Categoría: Accesorios, Precio: 70,00, Cantidad: 120
```

2. Gestión de clientes:

Para realizar todas las operaciones que tengan que ver con los clientes, aquí una imagen del menú.

```
=== Gestión de Clientes ===
1. Agregar cliente
2. Actualizar cliente
3. Eliminar cliente
4. Listar clientes
5. Regresar al Menú Principal
Seleccione una opción:
```

2.1 Agregar cliente:

Haremos como con el menú anterior de ver todos los menus, comenzando por “agregar cliente”

```
=== Gestión de Clientes ===
1. Agregar cliente
2. Actualizar cliente
3. Eliminar cliente
4. Listar clientes
5. Regresar al Menú Principal
Seleccione una opción: 1
Ingrese el nombre del cliente: Oscar
Ingrese la dirección del cliente: calle buenavista Ciempozuelos Madrid
Ingrese el teléfono del cliente: 123123123
Ingrese el email del cliente: oscarbuenavista@hotmail.com
Cliente agregado exitosamente.
```

Comprobación de que el cliente fue agregado correctamente.

```
ID: 20, Nombre: Pedro Díaz, Dirección: Calle Nube 606, Teléfono: 555-7654, Email: pedro.diaz@example.com
ID: 21, Nombre: Sofía Ramírez, Dirección: Boulevard Arcoíris 707, Teléfono: 555-8765, Email: sofia.ramirez@example.com
ID: 22, Nombre: Oscar, Dirección: calle buenavista Ciempozuelos Madrid, Teléfono: 123123123, Email: oscarbuenavista@hotmail.com
```

2.2 Actualizar cliente:

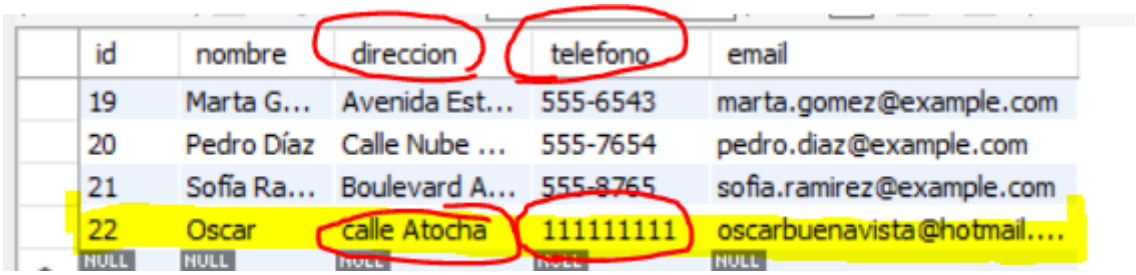
Ahora actualizaremos el cliente cambiándole por ejemplo la dirección y el teléfono.

```
=== Gestión de Clientes ===
1. Agregar cliente
2. Actualizar cliente
3. Eliminar cliente
4. Listar clientes
5. Regresar al Menú Principal
Seleccione una opción: 2
Ingrese el ID del cliente a actualizar: 22
Ingrese el nuevo nombre del cliente: Oscar
Ingrese la nueva dirección del cliente: calle Atocha
Ingrese el nuevo teléfono del cliente: 111111111
Ingrese el nuevo email del cliente: oscarbuenavista@hotmail.com
Cliente actualizado exitosamente.
```

Y ahora mostrare la comprobación de que se actualizo correctamente

```
ID: 20, Nombre: Pedro Díaz, Dirección: Calle Nube 606, Teléfono: 555-7654, Email: pedro.diaz@example.com
ID: 21, Nombre: Sofía Ramírez, Dirección: Boulevard Arcoíris 707, Teléfono: 555-8765, Email: sofia.ramirez@example.com
ID: 22, Nombre: Oscar, Dirección: calle Atocha, Teléfono: 111111111, Email: oscarbuenavista@hotmail.com
```

Y en la base de datos también se actualiza correctamente.



id	nombre	direccion	telefono	email
19	Marta G...	Avenida Est...	555-6543	marta.gomez@example.com
20	Pedro Díaz	Calle Nube ...	555-7654	pedro.diaz@example.com
21	Sofía Ra...	Boulevard A...	555-8765	sofia.ramirez@example.com
22	Oscar	calle Atocha	111111111	oscarbuenavista@hotmail....
NULL	NULL	NULL	NULL	NULL

2.3 Eliminar cliente:

Ahora le eliminaremos.

```
=== Gestión de Clientes ===
1. Agregar cliente
2. Actualizar cliente
3. Eliminar cliente
4. Listar clientes
5. Regresar al Menú Principal
Seleccione una opción: 3
Ingrese el ID del cliente a eliminar: 22
Cliente eliminado exitosamente.
```

2.4 Lista clientes:

Marcaremos la opción 4 del menú para que nos liste los clientes y efectivamente comprobar que se eliminó de manera correcta.

```
ID: 19, Nombre: Marta Gómez, Dirección: Avenida Estrella 505, Teléfono: 555-6543, Email: marta.gomez@example.com
ID: 20, Nombre: Pedro Díaz, Dirección: Calle Nube 606, Teléfono: 555-7654, Email: pedro.diaz@example.com
ID: 21, Nombre: Sofía Ramírez, Dirección: Boulevard Arcoíris 707, Teléfono: 555-8765, Email: sofia.ramirez@example.com
```

3. Ventas y Facturación:

En este apartado esta toda la información relacionada con los pedidos y el detalle de la venta, podemos registrar una venta nueva o mostrar las ventas realziadas.

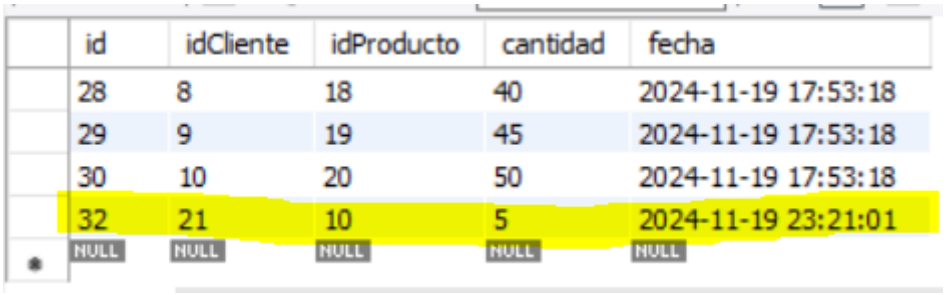
```
=== Ventas y Facturación ===
1. Registrar venta
2. Listar ventas
3. Regresar al Menú Principal
Seleccione una opción:
```

3.1 Registrar venta:

A continuación, procederemos a registrar una venta nueva.

```
=== Ventas y Facturación ===
1. Registrar venta
2. Listar ventas
3. Regresar al Menú Principal
Seleccione una opción: 1
Ingrese el ID del cliente: 21
Ingrese el ID del producto: 10
Ingrese la cantidad vendida: 5
Venta registrada exitosamente.
```

Adjuntamos una imagen de la base de datos para comprobar que la venta se registro correctamente.



id	idCliente	idProducto	cantidad	fecha
28	8	18	40	2024-11-19 17:53:18
29	9	19	45	2024-11-19 17:53:18
30	10	20	50	2024-11-19 17:53:18
32	21	10	5	2024-11-19 23:21:01
NULL	NULL	NULL	NULL	NULL

4. Consultas y reportes:

Aquí podemos ver una serie de datos bastante interesantes en cuanto a las ventas de la empresa.

```
=== Consultas y Reportes ===
1. Ventas totales
2. Ventas de los últimos 3 meses
3. Top 5 clientes que más gastan
4. Regresar al Menú Principal
Seleccione una opción:
```

4.1 Ventas totales:

```
=== Consultas y Reportes ===
1. Ventas totales
2. Ventas de los últimos 3 meses
3. Top 5 clientes que más gastan
4. Regresar al Menú Principal
Seleccione una opción: 1
Ventas totales: 96700,00
```

4.2 Ventas los últimos 3 meses:

```
=== Consultas y Reportes ===
1. Ventas totales
2. Ventas de los últimos 3 meses
3. Top 5 clientes que más gastan
4. Regresar al Menú Principal
Seleccione una opción: 2
Ventas de los últimos 3 meses: 96700,00
```

4.3 Top 5 clientes que más gastan:

```
Top 5 clientes que más gastan:
ID: 2, Nombre: Diana, Total gastado: 21300,00
ID: 8, Nombre: Cliente8, Total gastado: 14400,00
ID: 10, Nombre: Clinete11, Total gastado: 13500,00
ID: 9, Nombre: Cliente9, Total gastado: 12600,00
ID: 7, Nombre: Cliente7, Total gastado: 10150,00
```

Con esto terminaríamos de mostrar la funcionalidad de nuestra aplicación de consola a través de sus menús y sus diferentes opciones.

BIBLIOGRAFIA

- DuBois, S. (2016). Java for Absolute Beginners: Learn to Program the Fundamentals the Java 9+ Way. Apress.
Este libro proporciona una base sólida sobre Java, centrándose en su uso en aplicaciones empresariales y destacando conceptos clave como el manejo de bases de datos y la conexión mediante JDBC.

- MySQL. (n.d.). MySQL 8.0 Reference Manual. Recuperado de https://dev.mysql.com/doc/
Manual oficial de referencia para MySQL, que describe las funcionalidades y buenas prácticas para la creación y gestión de bases de datos relacionales.

- Oracle. (n.d.). Java Database Connectivity (JDBC) Overview. Recuperado de https://docs.oracle.com/javase/
Documentación oficial de Oracle sobre JDBC, que explica detalladamente cómo conectar aplicaciones Java con bases de datos relacionales.

- Farquhar, M., & Cotten, T. (2020). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media.
Este libro aborda conceptos esenciales sobre el diseño de sistemas de bases de datos, incluyendo la estructura y mantenimiento de datos en entornos escalables.

- Techopedia. (n.d.). CRUD Operations. Recuperado de https://www.techopedia.com/
Explicación sobre las operaciones CRUD y su importancia en aplicaciones empresariales, destacando su implementación en sistemas modernos.