UNIVERSIDAD DE LAS AMÉRICAS

Integrantes:

Pullas Carlos

Materia:

PROGRAMACION II

Tarea:

Proyecto final

Contenido

1.- Enunciado del problema

La empresa “Importaciones RJ” necesita de un sistema la cual tenga un medio de seguridad que le permita ingresar al gerente y una vez dentro le permita crear usuarios para empleados. El sistema tiene que ayudar a organizar su mercadería la cual le tendrá que permitir registrar, borrar, y actualizar el stock. De igual manera tiene que permitir llevar la lista de sus clientes con información de los mismos como cedula, teléfono, dirección; El sistema tiene que tener un apartado en el cual le permita generar un archivo pdf para ser impreso ya que ellos cuentan con notas de ventas físicas por estar categorizados en RIMPE negocio popular.

Objetivo General:

Desarrollar un sistema informático integral que permita a la empresa "Importaciones RJ" gestionar eficazmente su inventario de mercadería, registrar y mantener la información de los clientes, y generar facturas en formato PDF, al mismo tiempo que garantiza la seguridad en el acceso al sistema.

Objetivos Específicos:

1. Implementar un Sistema de Seguridad: Desarrollar un sistema de autenticación seguro que permita al gerente acceder de forma segura al sistema, protegiendo la información confidencial.

2. Gestión de Usuarios Eficiente: Facilitar la creación, modificación y eliminación de cuentas de usuario para los empleados, permitiendo al gerente gestionar quién tiene acceso a las diferentes funciones del sistema.

3. Optimizar la Gestión de Inventario y Clientes: Diseñar un sistema que permita registrar, actualizar y eliminar productos en el inventario, así como gestionar la información detallada de los clientes, incluyendo cédula, teléfono y dirección.

2. Identificación del problema (Objetivo de la solución informática)

Cliente:

- Registrar y gestionar información de clientes, incluyendo cédula, teléfono y dirección.

- Generar facturas en formato PDF para imprimir en notas de ventas físicas.

Usuario:

- Permitir al gerente ingresar al sistema de manera segura.

- Crear y gestionar cuentas de usuario para los empleados.

Requerimiento funcional:

1. Seguridad de Acceso:

- Implementar un sistema de autenticación seguro que permita al gerente acceder al sistema.

2. Gestión de Usuarios:

- Permitir al gerente crear, eliminar y actualizar cuentas de usuario para los empleados.

3. Inventario de Mercadería:

- Registrar la información de la mercadería, incluyendo nombre, cantidad en stock y detalles.

- Permitir al gerente agregar, eliminar y actualizar productos en el inventario.

4. Gestión de Clientes:

- Registrar y mantener información detallada de los clientes, incluyendo cédula, teléfono y dirección.

- Facilitar la búsqueda y edición de la información de los clientes.

5. Generación de Nota de Venta en formato PDF:

- Proporcionar una función que permita al gerente generar notas de ventas en formato PDF.

- Las facturas deben incluir detalles de la compra y la información del cliente.

Mundo del problema:

La empresa "Importaciones RJ" se dedica a la importación y venta de productos. Necesita un sistema que le permita gestionar su inventario de mercadería, mantener un registro de sus clientes y generar notas de venta en formato PDF para su uso en

notas de ventas físicas. Además, requiere un sistema de gestión de usuarios para sus empleados y garantizar la seguridad en el acceso al sistema.

Requerimiento no funcional:

1. Seguridad:

- El sistema debe ser seguro y proteger la información confidencial de la empresa, como datos de clientes y el inventario.

2. Usabilidad:

- El sistema debe ser fácil de usar para que el gerente y los empleados puedan operarlo sin dificultad.

3. Rendimiento:

- El sistema debe funcionar de manera eficiente, incluso cuando se maneje una gran cantidad de datos.

4. Escalabilidad:

- El sistema debe ser escalable para permitir futuras expansiones del negocio y un aumento en la cantidad de usuarios y productos.

5. Disponibilidad:

- El sistema debe estar disponible en todo momento para garantizar la continuidad de las operaciones de la empresa.

6. Generación de PDF:

- La generación de notas de ventas en formato PDF debe ser rápida y de alta calidad, asegurando que las notas de ventas sean legibles y profesionales.

7. Mantenimiento y Soporte:

- Debe existir un plan de mantenimiento y soporte para garantizar que el sistema funcione sin interrupciones y se puedan solucionar problemas rápidamente.

3. Identidades involucradas en el problema

1. Empresa "Importaciones RJ"

- Descripción: La entidad principal es la empresa "Importaciones RJ", que se dedica a la importación y venta de productos. La empresa busca implementar un sistema informático para gestionar su inventario, registrar información de clientes y generar facturas en formato PDF.

2. Gerente de la Empresa

- Descripción: El gerente de la empresa es una identidad clave que necesita acceso al sistema. Su función principal es supervisar las operaciones y tomar decisiones estratégicas. El gerente es el usuario principal que ingresará al sistema y gestionará a los empleados.

3. Empleados de la Empresa

- Descripción: Los empleados de la empresa son usuarios que requieren cuentas en el sistema para realizar tareas relacionadas con la gestión del inventario, atención a clientes y generación de facturas. Los empleados incluyen a vendedores, encargados de inventario y personal de atención al cliente.

4. Clientes de la Empresa

- Descripción: Los clientes son individuos o empresas que compran productos a "Importaciones RJ". La empresa necesita mantener un registro de información detallada de los clientes, incluyendo cédula o RUC, teléfono y dirección, para proporcionar un servicio más personalizado y generar facturas.

5. Productos en el Inventario

- Descripción: Los productos en el inventario son los artículos que "Importaciones RJ" importa y vende. Cada producto tiene detalles como nombre, cantidad en stock y descripción. La entidad del producto es fundamental para la gestión del inventario y la generación de facturas.

6. Notas de Ventas en Formato PDF

- Descripción: Las notas de ventas en formato PDF son documentos generados por el sistema que se utilizan para imprimir en notas de ventas físicas legalmente autorizadas por el SRI. Contienen detalles de la compra, información del cliente y se generan como parte del proceso de ventas de la empresa.

Desarrollo

package controlador(Categoria)
Este código es una parte de un programa en Java que forma parte de un controlador para la gestión de categorías en una base de datos.

1. `Ctrl_Categoria` es una clase que parece ser el controlador para gestionar operaciones relacionadas con categorías en una base de datos.

2. El método `guardar(Categoria objeto)` se utiliza para guardar una nueva categoría en la base de datos. Toma un objeto de tipo `Categoria` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_categoria` con tres columnas: `id`, `descripcion`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `Categoria`.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

3. El método `existeCategoria(String categoria)` se utiliza para verificar si una categoría ya existe en la base de datos. Toma el nombre de la categoría como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Crea una sentencia SQL que busca una categoría en la tabla `tb_categoria` con la descripción especificada.

- Ejecuta la sentencia SQL y devuelve `true` si se encuentra al menos una coincidencia.

4. El método `actualizar(Categoria objeto, int idCategoria)` se utiliza para actualizar una categoría existente en la base de datos. Toma un objeto de tipo `Categoria` y el ID de la categoría como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que actualiza la descripción de la categoría en la tabla `tb_categoria` con el ID proporcionado.

- Establece el valor de la descripción en la sentencia SQL con los datos del objeto `Categoria`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

5. El método `eliminar(int idCategoria)` se utiliza para eliminar una categoría de la base de datos. Toma el ID de la categoría como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de eliminación que elimina la categoría de la tabla `tb_categoria` con el ID proporcionado.

- Ejecuta la sentencia SQL y devuelve `true` si la eliminación se realizó con éxito.

En resumen, este código implementa operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para gestionar categorías en una base de datos utilizando Java y SQL. Se encarga de la inserción, consulta de existencia, actualización y eliminación de registros de categorías en la tabla `tb_categoria`. Además maneja la conexión a la base de datos.

package controlador;(Cliente)
forma parte de un controlador para gestionar operaciones relacionadas con clientes en una base de datos.

1. `Ctrl_Cliente` es una clase que parece ser el controlador para gestionar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) relacionadas con clientes en una base de datos.

2. El método `guardar(Cliente objeto)` se utiliza para guardar un nuevo cliente en la base de datos. Toma un objeto de tipo `Cliente` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_cliente` con siete columnas: `id`, `nombre`, `apellido`, `cedula`, `telefono`, `direccion`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `Cliente`.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

3. El método `existeCliente(String cedula)` se utiliza para verificar si un cliente ya está registrado en la base de datos. Toma la cédula del cliente como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Crea una sentencia SQL que busca un cliente en la tabla `tb_cliente` con la cédula especificada.

- Ejecuta la sentencia SQL y devuelve `true` si se encuentra al menos una coincidencia.

4. El método `actualizar(Cliente objeto, int idCliente)` se utiliza para actualizar un cliente existente en la base de datos. Toma un objeto de tipo `Cliente` y el ID del cliente como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que actualiza los campos del cliente en la tabla `tb_cliente` con el ID proporcionado.

- Establece los valores de la sentencia SQL con los datos del objeto `Cliente`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

5. El método `eliminar(int idCliente)` se utiliza para eliminar un cliente de la base de datos. Toma el ID del cliente como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de eliminación que elimina al cliente de la tabla `tb_cliente` con el ID proporcionado.

- Ejecuta la sentencia SQL y devuelve `true` si la eliminación se realizó con éxito.

En resumen, este código implementa operaciones CRUD para gestionar clientes en una base de datos. Realiza la inserción, consulta de existencia, actualización y eliminación de registros de clientes en la tabla `tb_cliente`.

package controlador;(Producto)
forma parte de un controlador para gestionar operaciones relacionadas con productos en una base de datos.

1. `Ctrl_Producto` es una clase que parece ser el controlador para gestionar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) relacionadas con productos en una base de datos.

2. El método `guardar(Producto objeto)` se utiliza para guardar un nuevo producto en la base de datos. Toma un objeto de tipo `Producto` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_producto` con ocho columnas: `id`, `nombre`, `cantidad`, `precio`, `descripcion`, `porcentajeIva`, `idCategoria`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `Producto`.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

3. El método `existeProducto(String producto)` se utiliza para verificar si un producto ya está registrado en la base de datos. Toma el nombre del producto como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Crea una sentencia SQL que busca un producto en la tabla `tb_producto` con el nombre especificado.

- Ejecuta la sentencia SQL y devuelve `true` si se encuentra al menos una coincidencia.

4. El método `actualizar(Producto objeto, int idProducto)` se utiliza para actualizar un producto existente en la base de datos. Toma un objeto de tipo `Producto` y el ID del producto como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que actualiza los campos del producto en la tabla `tb_producto` con el ID proporcionado.

- Establece los valores de la sentencia SQL con los datos del objeto `Producto`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

5. El método `eliminar(int idProducto)` se utiliza para eliminar un producto de la base de datos. Toma el ID del producto como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de eliminación que elimina el producto de la tabla `tb_producto` con el ID proporcionado.

- Ejecuta la sentencia SQL y devuelve `true` si la eliminación se realizó con éxito.

6. El método `actualizarStock(Producto object, int idProducto)` se utiliza para actualizar la cantidad en stock de un producto en la base de datos. Toma un objeto de tipo `Producto` y el ID del producto como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que actualiza la cantidad en stock del producto en la tabla `tb_producto` con el ID proporcionado.

- Establece el valor de la cantidad en stock en la sentencia SQL con los datos del objeto `Producto`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

En resumen, este código implementa operaciones CRUD para gestionar productos en una base de datos. Realiza la inserción, consulta de existencia, actualización, eliminación y actualización de stock de registros de productos en la tabla `tb_producto`. También maneja la conexión a la base de datos

package controlador;(RegistroVenta)

forma parte de un controlador para gestionar operaciones relacionadas con la venta de productos en una base de datos.

1. `Ctrl_RegistrarVenta` es una clase que parece ser el controlador para gestionar las operaciones relacionadas con la venta de productos, tanto para la cabecera de la venta como para los detalles de la venta.

2. La variable `idCabeceraRegistrada` se utiliza para almacenar el ID de la cabecera de la venta recién registrada. Esta variable se utilizará posteriormente al registrar los detalles de la venta.

3. El método `guardar(CabeceraVenta objeto)` se utiliza para guardar la cabecera de una venta en la base de datos. Toma un objeto de tipo `CabeceraVenta` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_cabecera_venta` con cinco columnas: `id`, `idCliente`, `valorPagar`, `fechaVenta`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `CabeceraVenta`.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

- También obtiene el ID generado de la cabecera de la venta recién registrada y lo almacena en `idCabeceraRegistrada`.

4. El método `guardarDetalle(DetalleVenta objeto)` se utiliza para guardar los detalles de una venta en la base de datos. Toma un objeto de tipo `DetalleVenta` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_detalle_venta` con diez columnas: `id`, `idCabeceraVenta`, `idProducto`, `cantidad`, `precioUnitario`, `subTotal`, `descuento`, `iva`, `totalPagar`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `DetalleVenta`, incluyendo el `idCabeceraVenta` que se obtuvo previamente.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

5. El método `actualizar(CabeceraVenta objeto, int idCabeceraVenta)` se utiliza para actualizar la información de la cabecera de una venta existente en la base de datos. Toma un objeto de tipo `CabeceraVenta` y el ID de la cabecera de la venta como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que modifica los campos de la cabecera de venta en la tabla `tb_cabecera_venta` con el ID proporcionado.

- Establece los valores de la sentencia SQL con los datos del objeto `CabeceraVenta`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

En resumen, este código implementa operaciones para gestionar ventas de productos en una base de datos. Permite registrar la cabecera de la venta, los detalles de la venta y actualizar la información de la cabecera de venta. También maneja la conexión a la base de datos

package controlador;(Usuario)
forma parte de un controlador para gestionar operaciones relacionadas con usuarios en una base de datos.

1. `Ctrl_Usuario` es una clase que parece ser el controlador para gestionar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) relacionadas con usuarios en una base de datos.

2. El método `guardar(Usuario objeto)` se utiliza para guardar un nuevo usuario en la base de datos. Toma un objeto de tipo `Usuario` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos utilizando la clase `Conexion`.

- Prepara una sentencia SQL de inserción para la tabla `tb_usuario` con siete columnas: `id`, `nombre`, `apellido`, `usuario`, `password`, `telefono`, y `estado`.

- Establece los valores de la sentencia SQL con los datos del objeto `Usuario`.

- Ejecuta la sentencia SQL y devuelve `true` si la inserción se realizó con éxito.

3. El método `existeUsuario(String usuario)` se utiliza para verificar si un usuario ya está registrado en la base de datos. Toma el nombre de usuario como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Crea una sentencia SQL que busca un usuario en la tabla `tb_usuario` con el nombre de usuario especificado.

- Ejecuta la sentencia SQL y devuelve `true` si se encuentra al menos una coincidencia.

4. El método `loginUser(Usuario objeto)` se utiliza para validar el inicio de sesión de un usuario en la base de datos. Toma un objeto de tipo `Usuario` como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Crea una sentencia SQL que verifica si existe un usuario con el nombre de usuario y contraseña proporcionados.

- Ejecuta la sentencia SQL y devuelve `true` si se encuentra un usuario con las credenciales proporcionadas.

5. El método `actualizar(Usuario objeto, int idUsuario)` se utiliza para actualizar la información de un usuario existente en la base de datos. Toma un objeto de tipo `Usuario` y el ID del usuario como argumentos y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de actualización que modifica los campos del usuario en la tabla `tb_usuario` con el ID proporcionado.

- Establece los valores de la sentencia SQL con los datos del objeto `Usuario`.

- Ejecuta la sentencia SQL y devuelve `true` si la actualización se realizó con éxito.

6. El método `eliminar(int idUsuario)` se utiliza para eliminar un usuario de la base de datos. Toma el ID del usuario como argumento y realiza lo siguiente:

- Establece una conexión a la base de datos.

- Prepara una sentencia SQL de eliminación que elimina al usuario de la tabla `tb_usuario` con el ID proporcionado.

- Ejecuta la sentencia SQL y devuelve `true` si la eliminación se realizó con éxito.

En resumen, este código implementa operaciones CRUD para gestionar usuarios en una base de datos. Realiza la inserción, consulta de existencia, actualización y eliminación de registros de usuarios en la tabla `tb_usuario`. También maneja la

conexión a la base de datos y captura excepciones de SQL, además de proporcionar una función para validar el inicio de sesión de un usuario.

package controlador(VentaPDF)
Este código es una parte de una aplicación Java que genera un archivo PDF de factura de venta a partir de los datos de una transacción de venta

1. La clase `VentaPDF` se encarga de generar un archivo PDF de factura de venta con información detallada sobre la venta, incluyendo los datos del cliente, los productos comprados y el monto total a pagar.

2. El método `DatosCliente(int idCliente)` se utiliza para obtener los datos del cliente a partir de su ID en la base de datos. Esto incluye el nombre, cédula, teléfono y dirección del cliente.

3. El método `generarFacturaPDF()` se encarga de crear el archivo PDF de la factura de venta. Realiza lo siguiente:

- Obtiene la fecha actual y la formatea.

- Define un nombre de archivo PDF basado en el nombre del cliente y la fecha actual.

- Abre un archivo de salida para escribir el PDF.

- Crea un documento PDF y lo abre para escritura.

- Agrega una imagen al documento que puede ser un logotipo o encabezado.

- Crea un encabezado de factura con información como el número de factura y la fecha.

- Luego, agrega datos del cliente, como nombre, cédula, teléfono y dirección.

- Agrega un espacio en blanco.

- Agrega una tabla con información detallada sobre los productos comprados, incluyendo cantidad, descripción, precio unitario y precio total para cada producto.

- Calcula y agrega el total a pagar.

- Agrega un espacio en blanco y una línea de firma.

- Agrega un mensaje de agradecimiento.

- Cierra el documento y el archivo.

- Abre el archivo PDF generado automáticamente para que el usuario pueda verlo.

En resumen, este código genera un archivo PDF de factura de venta con información detallada sobre la transacción de venta, incluyendo los datos del cliente y los productos comprados. El PDF resultante se abre automáticamente para su visualización.

Librerías Utilizadas

1. java.sql: Esta librería es parte del paquete estándar de Java y proporciona clases para la gestión de bases de datos y la realización de consultas SQL. Se utiliza en los códigos para conectarse a una base de datos, ejecutar consultas SQL y trabajar con resultados de consultas.

2. com.itextpdf.text: Esta librería se utiliza para crear y manipular documentos PDF en Java. Se utiliza para generar documentos PDF en el código `VentaPDF` y dar formato a los documentos, incluyendo la inserción de imágenes, tablas y texto.

3. java.awt: Este paquete proporciona clases para crear interfaces gráficas de usuario (GUI) en Java. En el código `VentaPDF`, se utiliza para trabajar con imágenes y la creación de documentos PDF.

4. java.io: Esta librería proporciona clases para trabajar con entrada/salida (I/O) en Java. Se utiliza para crear y escribir archivos, como en el código `VentaPDF` para guardar el archivo PDF generado.

5. javax.swing: Este paquete proporciona clases y componentes gráficos para crear aplicaciones de interfaz de usuario en Java. Aunque no se muestra explícitamente en los códigos que proporcionaste, se usa en muchas aplicaciones Java para crear interfaces gráficas de usuario interactivas.

6. java.util: Esta librería proporciona clases y utilidades para trabajar con colecciones, fechas, y otros objetos en Java. No se muestra directamente en los códigos proporcionados, pero generalmente se utiliza en aplicaciones Java para tareas de utilidad.

7. java.text: Este paquete se utiliza para formatear y analizar texto, incluyendo fechas y números. Puede ser utilizado para dar formato a fechas y horas, aunque no está explícitamente presente en los códigos proporcionados.

Estas librerías son esenciales para realizar tareas específicas en los códigos proporcionados, como la gestión de bases de datos, la generación de documentos PDF, la manipulación de imágenes y la interacción con el usuario a través de interfaces gráficas de usuario. Cada una de estas librerías cumple un propósito particular en el desarrollo de aplicaciones en Java.

Clase de conexión a una base de datos MySQL en Java

1. java.sql: Esta librería proporciona clases y funcionalidades para trabajar con bases de datos SQL en Java. En este código, se utiliza para establecer y gestionar la conexión a una base de datos MySQL.

2. java.sql.Connection: La interfaz `Connection` es parte de la librería `java.sql` y se utiliza para representar una conexión a una base de datos. En este código, se crea una instancia de `Connection` para establecer la conexión a la base de datos MySQL.

3. java.sql.DriverManager: Esta clase de la librería `java.sql` se utiliza para obtener una conexión a la base de datos. En el código, se utiliza para establecer la conexión a una base de datos MySQL local.

El código realiza lo siguiente:

- Define una clase llamada `Conexion` que proporciona un método estático llamado `conectar` para obtener una conexión a una base de datos MySQL.

- El método `conectar` intenta establecer una conexión a una base de datos MySQL local (en el mismo equipo) con la URL de conexión, nombre de usuario y contraseña proporcionados.

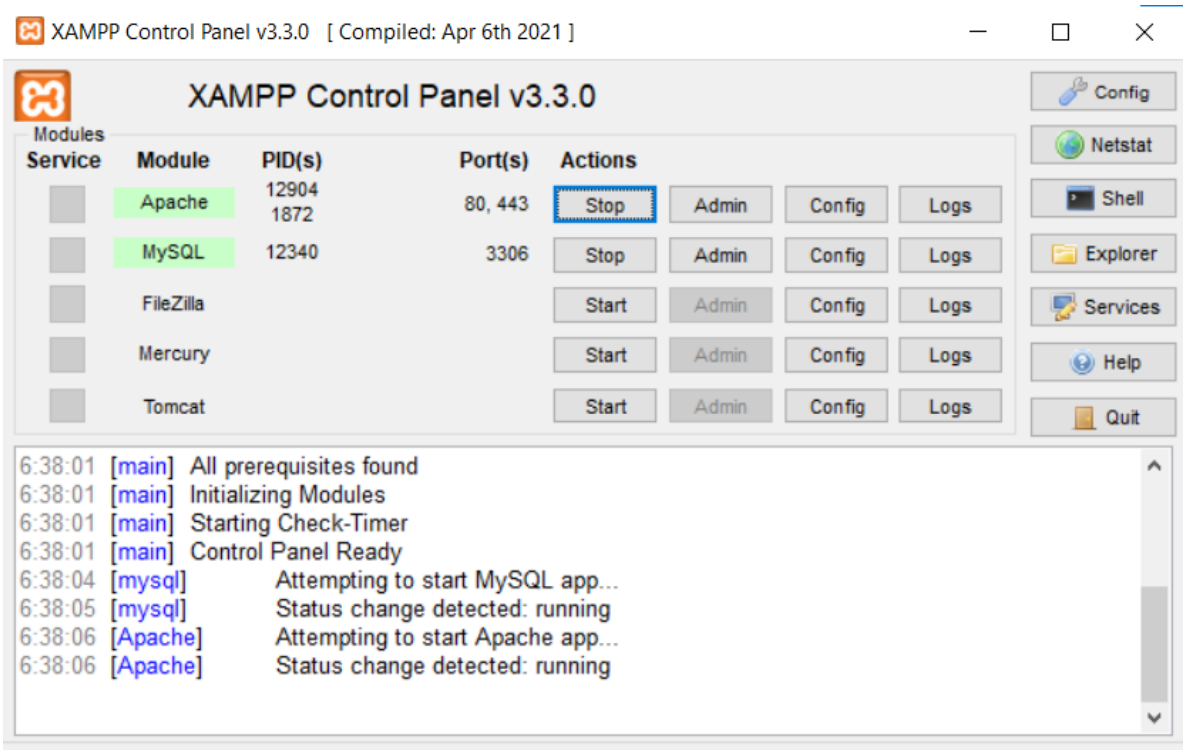
- Si la conexión se establece con éxito, el método devuelve un objeto `Connection`. Si hay un error en la conexión, se captura y se imprime en la consola.

este código es una clase de utilidad que facilita la conexión a una base de datos MySQL local y utiliza las librerías estándar de Java para lograrlo. La conexión a la base de datos es un paso esencial para interactuar con la base de datos y realizar operaciones como consultas y actualizaciones.

Base de datos Funcionamiento

En este caso estamos utilizando lo que es xampp control panel a continuación dejare el link de descarga.

Link: https://www.apachefriends.org/es/download.html



El siguiente paso sera descargar el mysql workbench de igual manera dejare el link de descarga a continuacion para su descarga de igual manera en el zip adjunto dejare el archivo en el cual se encuentra el código utilizado para la base de datos.

Link: https://dev.mysql.com/downloads/workbench/

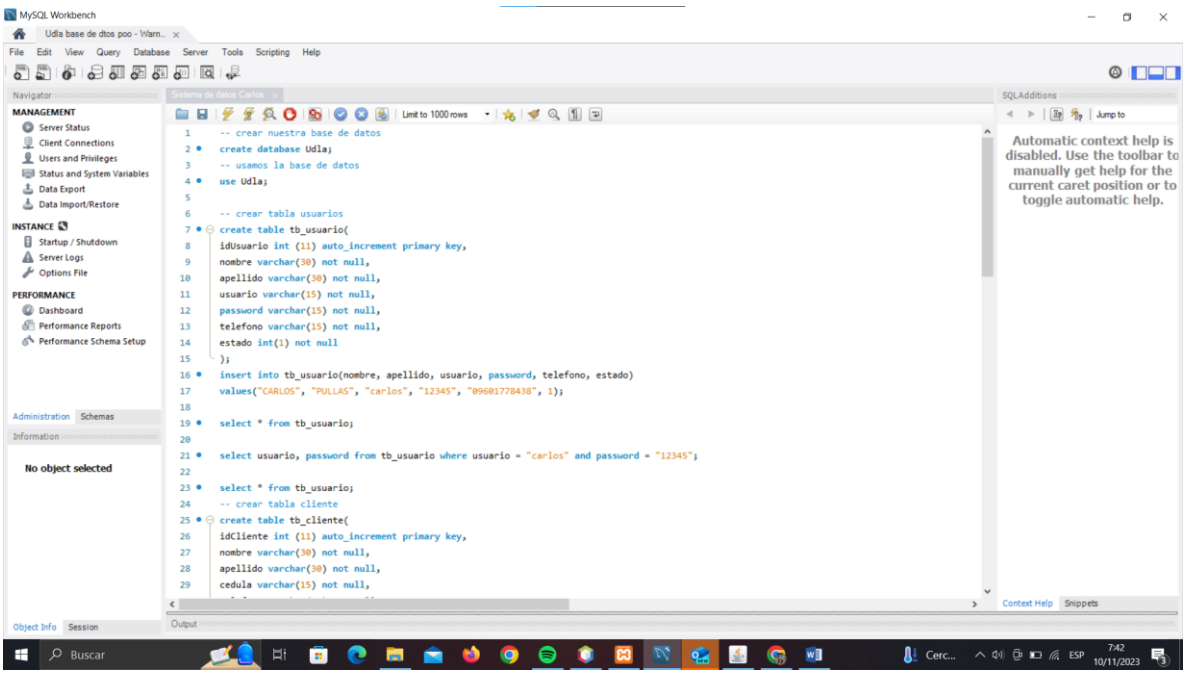
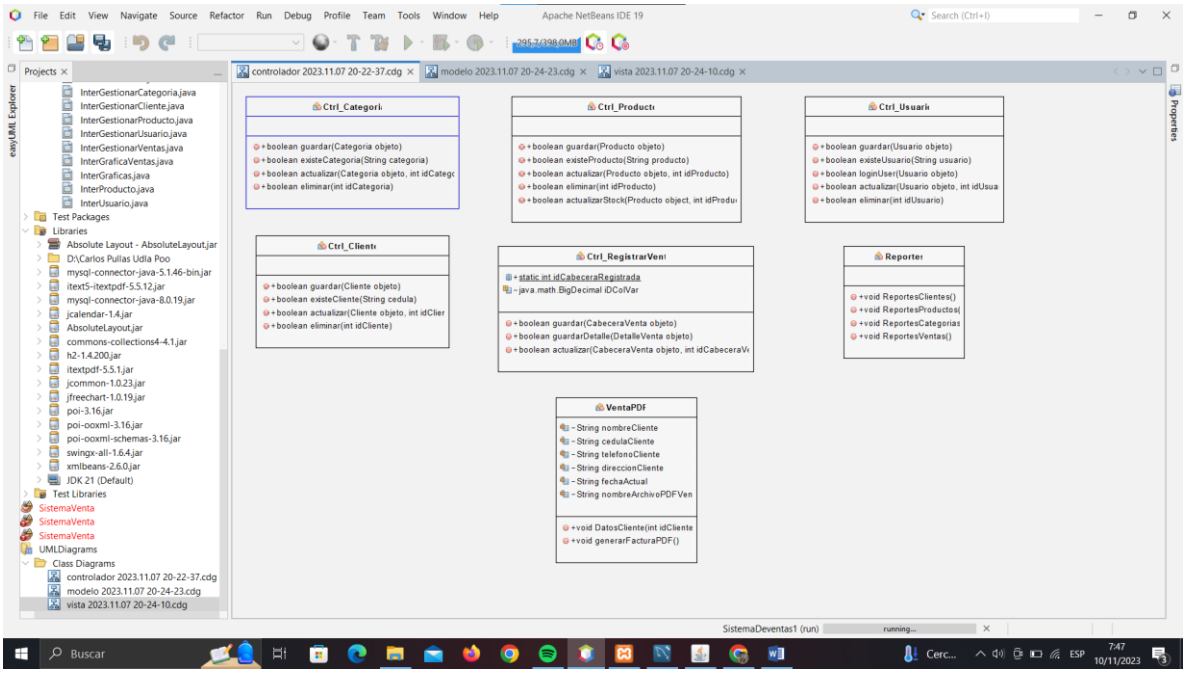
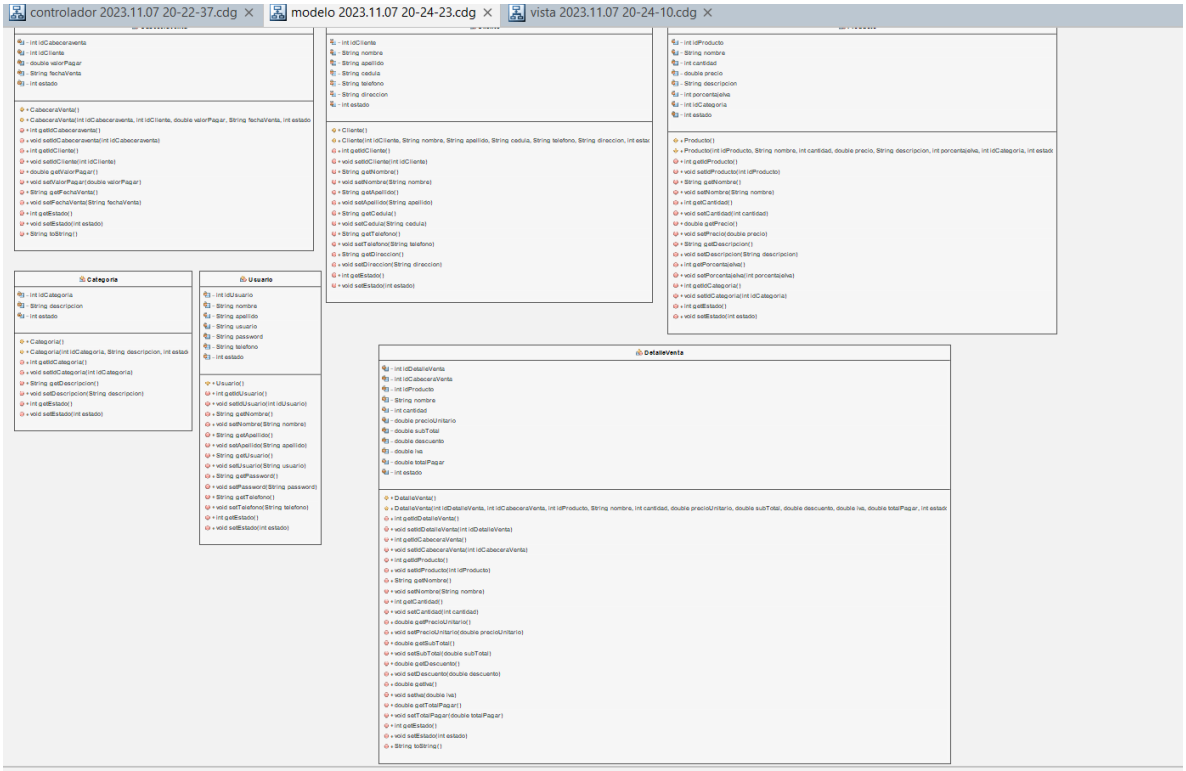
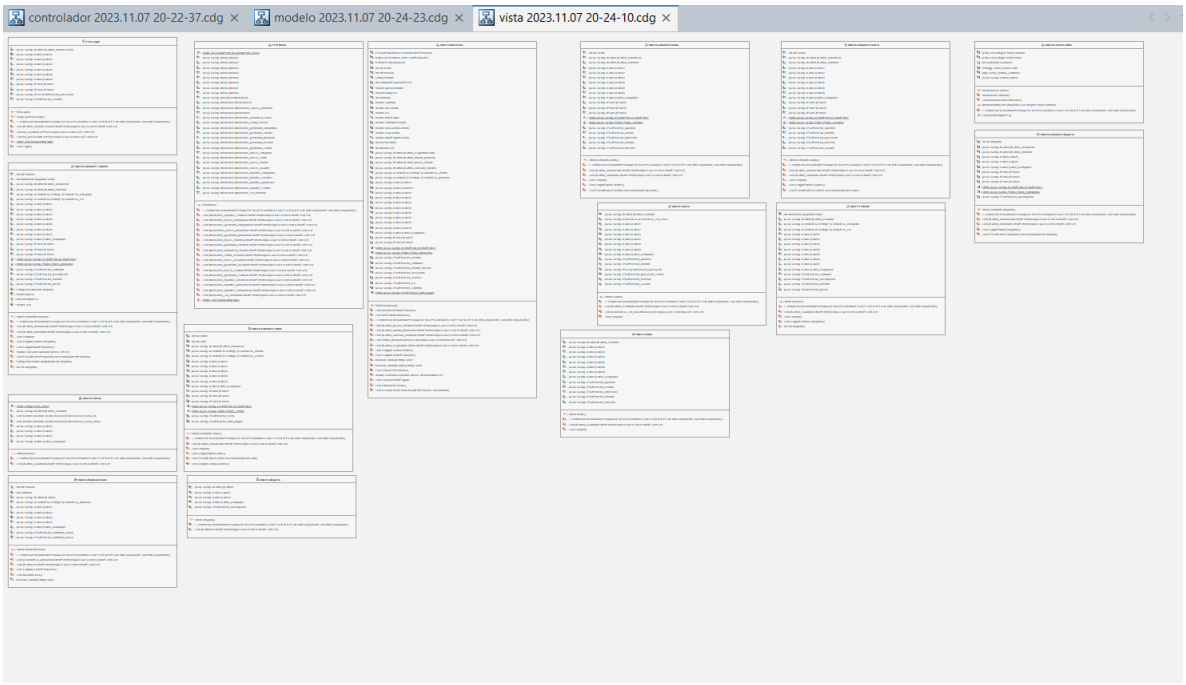


Diagrama de UML

En este caso de igual manera dejare una carpeta llamada Carlos_Pullas_103_03_E1_P1

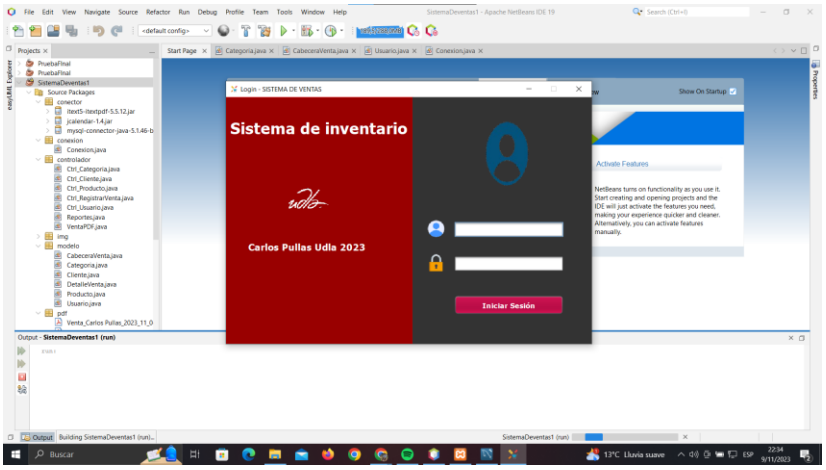

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Apache NetBeans IDE 19
Projects
controlador 2023.11.07 20-22-37.cdg
modelo 2023.11.07 20-24-23.cdg
vista 2023.11.07 20-24-10.cdg
Ctrl_Categori
Ctrl_Producto
Ctrl_Usuario
Ctrl_Client
Ctrl_RegistrarVent
Reporte
VentaPDF
Test Packages
Libraries
UMLDiagrams
Class Diagrams
SistemaDeventas1 (run)
running...
Buscar
ESP
7:47
10/11/2023


controlador 2023.11.07 20-22-37.cdg
modelo 2023.11.07 20-24-23.cdg
vista 2023.11.07 20-24-10.cdg
Categoria
Usuario
DetalleVenta



Anexos

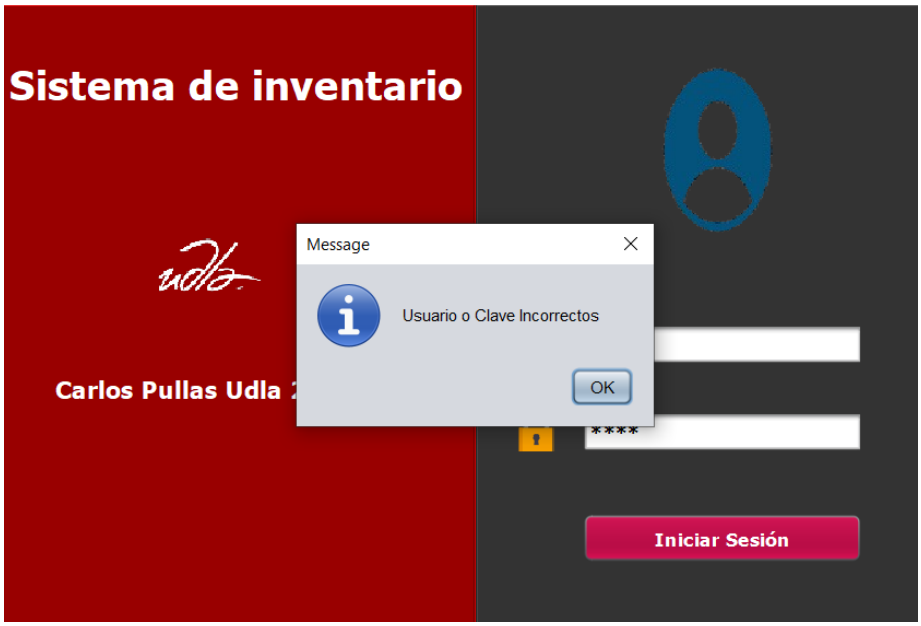
Inicio



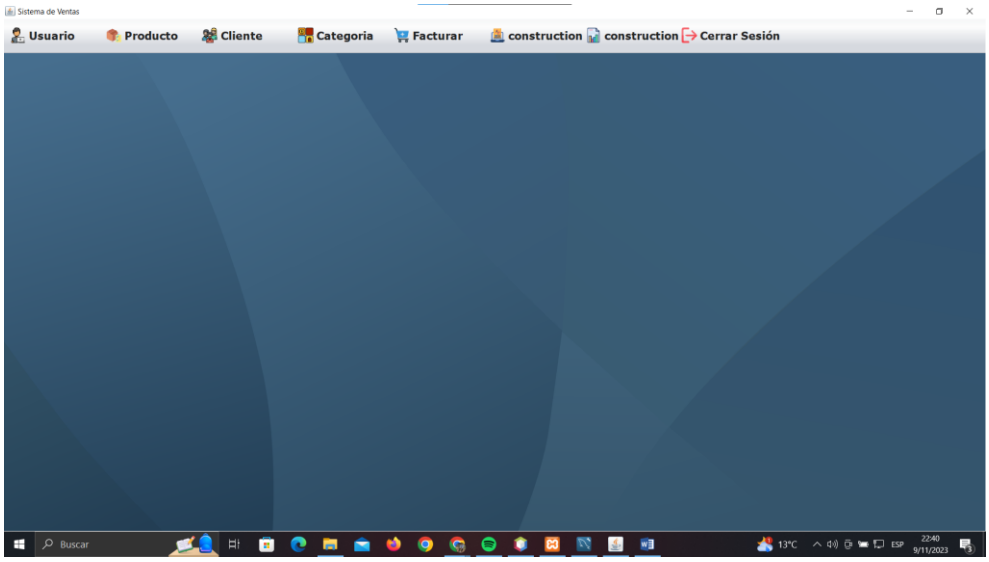
Validación de Login

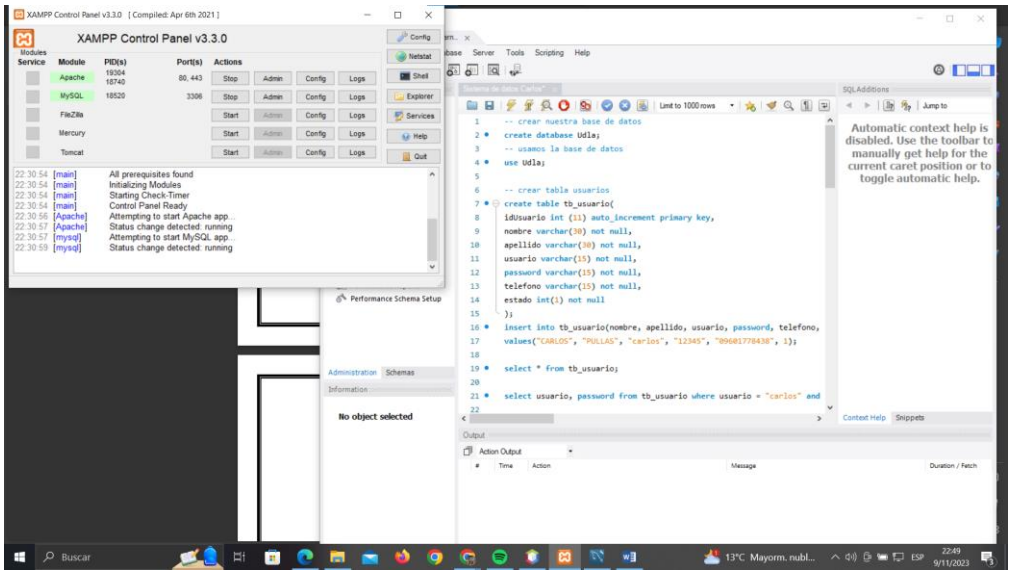
- Falta un campo de llenar

- Clave incorrecta



- Clave correcta ingreso a Menú





Conclusiones:

1. Con la implementación exitosa del sistema de seguridad, se garantiza que solo el gerente y usuarios tenga acceso al sistema, protegiendo así la información crítica de la empresa. La seguridad es un componente clave para el cumplimiento del objetivo general.

2. La eficiente gestión de usuarios permitirá al gerente controlar quién puede acceder a las diversas funciones del sistema, lo que facilita la administración y garantiza que solo las personas autorizadas puedan realizar cambios en el inventario y en la información de los clientes.

3. La optimización en la gestión del inventario y la información de los clientes asegura que la empresa pueda llevar un registro actualizado de sus productos y sus clientes. Esto facilita la toma de decisiones y la generación de notas de ventas en formato PDF, contribuyendo al cumplimiento del objetivo general de mejorar la eficacia de la empresa "Importaciones RJ".

4. Al encontrarse la empresa categorizada en el RIMPE Negocio Popular, no tiene obligatoriedad de emitir facturas electrónicas pero la implementación de un sistema informático que permita tener un registro de sus clientes y un control de su inventario le permite a la empresa unipersonal automatizar su facturación con rapidez y eficiencia.