MANUAL TÉCNICO.

Nombre: Fredy José Gabriel Herrera Funes. No. Carné: 202130478

-Descripción: programa dirigido a una entidad bancaria, que desea controlar las tarjetas de crédito que se solicitan, activan, desactivan y los movimientos que se realizan con ellas.

-Herramientas utilizadas:

- Desarrollado en sistema operativo Ubunto linux, versión 24.04.
- IDE utilizado: Apache NetBeans 22.
- Servidor para base de datos: MySQL.
- Lenguaje de programación utilizado: Java (Programación orientada a objetos).
- Versión Java: 17.

-Explicación del código:

- **Backend:**

-Clase Solicitudes: se utiliza para manejar solicitudes en una aplicación. Permite crear una nueva solicitud, asegurarse de que el número de solicitud sea único, guardar la solicitud en la base de datos, y actualizar el estado de la solicitud. La clase maneja la conexión a la base de datos y proporciona métodos para interactuar con la tabla solicitud_nueva.

La gestión de errores se realiza mediante mensajes de diálogo que informan al usuario sobre problemas en la conexión a la base de datos o en las operaciones de inserción y actualización.

GenerarNumeroSolicitud(): genera un número de solicitud único de 6 dígitos utilizando la clase Random. Este número se utiliza para identificar de forma única cada solicitud.

verificarNumeroSolicitud(int numeroSolicitud): verifica en la base de datos si el número de solicitud generado ya existe. Retorna true si el número es único (no existe en la base de datos) y false en caso contrario.

GuardarSolicitud(): genera un número de solicitud único, verifica su disponibilidad, y luego inserta la solicitud en la base de datos con los datos proporcionados. Si hay un error al insertar, se muestra un mensaje de error.

actualizarEstadoSolicitud(boolean nuevoEstado): actualiza el estado de una solicitud en la base de datos según el número de solicitud actual. Si hay un error durante la actualización, se muestra un mensaje de error.

-Clase Administrador: permite gestionar tarjetas de crédito en una aplicación. Incluye funcionalidades para:

Autorizar tarjetas basadas en solicitudes y verificar saldos y límites.

Realizar y registrar movimientos en tarjetas.

Actualizar el estado de las tarjetas y gestionar solicitudes asociadas.
La clase interactúa con una base de datos MySQL para almacenar y recuperar información sobre tarjetas y solicitudes. Los mensajes informativos se muestran mediante JOptionPane en caso de errores o estados especiales.

autorizarTarjetas(double salario, String tipo, String numeroSoli):: calcula el límite de la tarjeta como el 60% del salario. Verifica si el límite es suficiente para el tipo de tarjeta y el estado de la solicitud. Si todo es válido, crea una nueva tarjeta y actualiza el estado de la solicitud a aprobado. Muestra mensajes de información según el resultado.

verificarEstadoSolicitud(String numeroSoli): consulta la base de datos para verificar el estado de la solicitud especificada por numeroSoli. Retorna true si la solicitud está pendiente (no aprobada ni rechazada).

verificarSalario(double salario, double limite, String tipo): verifica si el límite calculado para la tarjeta es suficiente según el tipo de tarjeta (Internacional, Regional, Nacional). Retorna true si el límite es adecuado, false en caso contrario.

buscarSolicitud(String numeroSolicitud): busca una solicitud en la base de datos por su número. Si se encuentra, autoriza la tarjeta utilizando la información de la solicitud.

establecerMovimiento(String numeroTarjeta, String descripcion, String fecha, String codEstablecimiento, String monto, String tipo): realiza un movimiento (cargo o abono) en la tarjeta especificada. Verifica el estado de la tarjeta y el saldo antes de realizar la operación. Muestra mensajes si el movimiento no es válido.

obtenerInformacionTarjeta(String numeroTarjeta): consulta la base de datos para obtener la información de la tarjeta especificada (saldo, límite, estado). Muestra los detalles de la tarjeta en la consola.

actualizarEstadoTarjeta(String numeroTarjeta): actualiza el estado de la tarjeta a inactiva si no hay deuda pendiente. Si hay deuda, muestra un mensaje indicando el monto pendiente.

mostrarMensaje(String mensaje, String titulo, int tipoMensaje): muestra un mensaje emergente con el texto proporcionado.

VerificarDeuda(): verifica si la tarjeta tiene deuda pendiente (es decir, si el saldo es menor o igual al límite). Retorna true si no hay deuda.

-Clase Tarjeta se utiliza para: generar números únicos de tarjetas de crédito basados en el tipo de tarjeta.

Almacenar la información de la tarjeta en la base de datos MySQL.

Proporcionar acceso al número de la tarjeta a través de un método getter.

La generación del número de tarjeta asegura que cada tarjeta tenga un número único, utilizando un sistema de prefijos y un contador. La conexión con la base de datos se maneja a través del objeto Connection proporcionado por la clase Administrador. Los errores en la conexión o en las operaciones de base de datos se informan mediante mensajes en la consola.

obtenerNumeroTarjeta(String tipo): genera un nuevo número de tarjeta basado en el tipo de tarjeta (Nacional, Internacional, Regional). Utiliza un prefijo específico para cada tipo y añade un dígito variable y un contador para asegurar la unicidad del número. Si el contador supera 9999, incrementa el dígito variable.

obtenerUltimoNumeroTarjeta(String prefix): consulta la base de datos para obtener el último número de tarjeta que comienza con el prefijo especificado. Ordena los resultados en orden descendente para obtener el número más reciente.

GuardarTarjeta(): guarda la nueva tarjeta en la base de datos. Inserta el número de tarjeta generado, el número de solicitud, la fecha, el límite, el saldo y el estado de la tarjeta. Muestra un mensaje de éxito si la tarjeta se guarda correctamente, o un mensaje de error en caso contrario.

-Clase Consultor se utiliza para:

Obtener información detallada sobre una tarjeta de crédito específica.

Actualizar el estado de una tarjeta en la base de datos.

Verificar si una tarjeta tiene deuda basada en el límite y el saldo.

Mostrar mensajes de error o información al usuario.

Cerrar la conexión a la base de datos de forma segura.

Cada método maneja operaciones específicas relacionadas con la base de datos y proporciona mecanismos para gestionar y mostrar mensajes de error, lo que facilita la interacción con el usuario y el manejo de errores en el proceso.

obtenerInformacionTarjeta(String numeroTarjeta): consulta la base de datos para obtener la información completa de una tarjeta específica usando el número de tarjeta proporcionado. Rellena los atributos de la clase (numeroTarjeta, limite, estadoTarjeta, tipo, nombre, direccion) con la información obtenida. Retorna true si la tarjeta se encuentra y se obtiene información, o false si no se encuentra.

actualizarEstadoTarjeta(String numeroTarjeta, boolean nuevoEstado): actualiza el estado de una tarjeta en la base de datos (estado_tarjeta) y establece la fecha de actualización a la fecha actual. Retorna true si se actualizó al menos una fila en la base de datos, o false en caso contrario. Muestra un mensaje de error si ocurre una excepción.

TieneDeuda(): determina si la tarjeta tiene deuda comparando el límite con el saldo actual de la tarjeta. Retorna true si el límite es mayor que el saldo, indicando que hay deuda.

ObtenerSaldo(): obtiene el saldo de la tarjeta desde la base de datos. Utiliza el número de tarjeta almacenado en el atributo numeroTarjeta para realizar la consulta. Retorna el saldo obtenido o 0.0 en caso de error.

mostrarMensaje(String mensaje, String titulo, int tipo): muestra un mensaje utilizando un cuadro de diálogo (JOptionPane) con el texto, título y tipo especificados.

Close(): cierra la conexión con la base de datos si está abierta. Lanza una excepción SQLException si ocurre un error durante el cierre.

-Clase Movimiento: se utiliza para gestionar las operaciones financieras en tarjetas de crédito, como los cargos y abonos. La clase maneja las siguientes tareas:

Realizar Cargos y Abonos: Calcula el nuevo saldo después de realizar una operación y actualiza el saldo en la base de datos.

Registrar Movimientos: Inserta detalles sobre cada movimiento (cargo o abono) en la base de datos.

Manejo de Errores: Muestra mensajes de error si ocurre algún problema al actualizar el saldo o al registrar un movimiento.

realizarCargo(double saldo): calcula el nuevo saldo después de realizar un cargo (disminuir el saldo). Llama al método actualizarSaldo para actualizar el saldo en la base de datos. Muestra un mensaje de error si no se puede actualizar el saldo.

realizarAbono(double saldo): calcula el nuevo saldo después de realizar un abono (aumentar el saldo). Llama al método actualizarSaldo para actualizar el saldo en la base de datos. Muestra un mensaje de error si no se puede actualizar el saldo.

actualizarSaldo(String numeroTarjeta, double nuevoSaldo): actualiza el saldo de la tarjeta especificada en la base de datos. Retorna true si la actualización fue exitosa, o false en caso contrario. Muestra un mensaje de error si ocurre una excepción durante la actualización.

GuardarMovimiento(): inserta un nuevo registro de movimiento en la tabla movimientos de la base de datos.
Los valores se insertan en los campos correspondientes (numero_tarjeta, fecha_movimiento, tipo_movimiento, descripcion, establecimiento, monto).
Muestra un mensaje de éxito si el movimiento se registra correctamente, o un mensaje de error si ocurre un problema durante la inserción.

-Clase ConsultarListaTarjetas
Esta clase maneja la conexión con una base de datos MySQL y permite realizar consultas para obtener y mostrar datos sobre tarjetas bancarias en una tabla gráfica (Jtable).

obtenerDatos(JTable table, JCheckBox jcbRangoFecha, JCheckBox jcbLimite, JCheckBox jcbNombre, JComboBox<String> jcbTipo, JComboBox<String> jcbEstado, JTextField jtfFechaInicio, JTextField jtfFechaFin, JTextField jtfLimite, JTextField jtfNombre): construye una consulta SQL dinámica basada en los filtros seleccionados en los componentes de la interfaz gráfica (JCheckBox, JComboBox, JTextField).
Ejecuta la consulta y llena la tabla JTable con los resultados.

Muestra mensajes de error si ocurre algún problema con los datos o la consulta.

ajustarAnchoYAltura(JTable table): ajusta el ancho de las columnas y la altura de las filas de la tabla JTable en función del contenido de las celdas y encabezados, con márgenes adicionales para mejorar la presentación.

-Clase ConsultarListaSolicitudes: esta clase se encarga de manejar la conexión con una base de datos MySQL y permite realizar consultas para obtener y mostrar datos sobre solicitudes en una tabla gráfica (Jtable).

obtenerDatos(JTable table, JCheckBox jcbRangoFecha, JCheckBox jcbSalario, JComboBox<String> jcbTipo, JComboBox<String> jcbEstado, JTextField jtfFechaInicio, JTextField jtfFechaFin, JTextField jtfSalario): construye una consulta SQL dinámica basada en los filtros seleccionados en los componentes de la interfaz gráfica (JCheckBox, JComboBox, JtextField). Ejecuta la consulta y llena la tabla JTable con los resultados.
Muestra mensajes de error si hay problemas con los datos o con la consulta.

ajustarAnchoYAltura(JTable table): ajusta el ancho de las columnas y la altura de las filas de la tabla JTable según el contenido de las celdas y los encabezados, con márgenes adicionales para una mejor presentación.

-Clase ConsultarEstadoDeCuenta: esta clase gestiona la conexión a una base de datos MySQL para obtener y mostrar datos sobre tarjetas de crédito, movimientos, saldos e intereses en una tabla gráfica (Jtable).

obtenerDatos(JTable table, JCheckBox jcbNoTarjeta, JCheckBox jcbTipo, JCheckBox jcbSaldo, JCheckBox jcbInteres, JTextField jtfNumeroTarjeta, JComboBox<String> jcbTipos, JTextField jtfMontoSaldo, JTextField jtfMontoInteres): construye una consulta SQL dinámica basada en los filtros seleccionados en la interfaz gráfica (JCheckBox, JComboBox, JtextField). Ejecuta la consulta y llena la tabla JTable con los resultados.
Usa un DecimalFormat para formatear los intereses a dos decimales.
Muestra mensajes de error si hay problemas con los datos o la consulta.

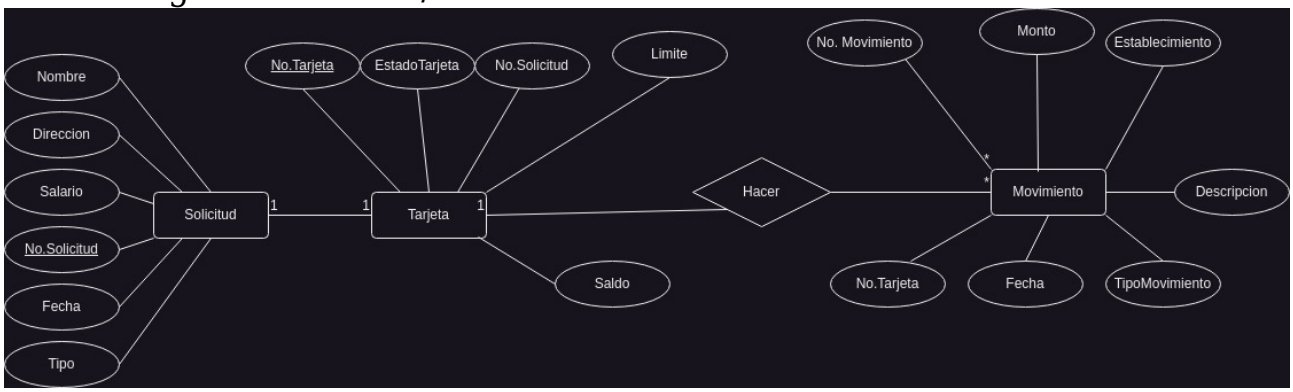
ajustarAnchoYAltura(JTable table): ajusta el ancho de las columnas y la altura de las filas de la tabla JTable basándose en el contenido de las celdas y los encabezados.
Añade márgenes extra para una mejor presentación.

- **Fronted:**

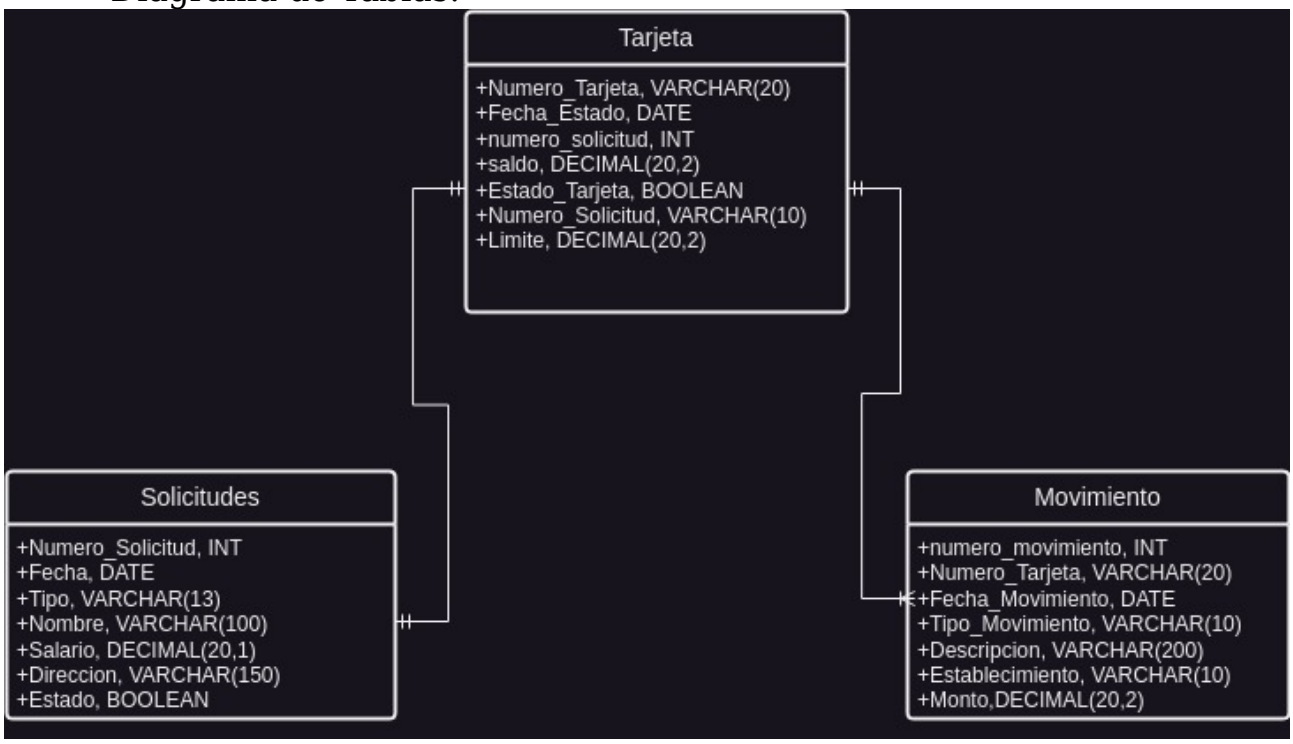
Se utilizó un frame principal, con un JdesktopPanel, y sus InternalFrame, cuales son:
-AutorizacionTarjetas.
-CancelarTarjetas.
-ConsultarTarjeta.
-InterlFrameEstadoCuenta
-InterlFrameLisTarjetas
-InterlFrameListaSolicitudes
-InterlFrameMovimiento
-InterlFrameSolicitud.

-Diagramas:

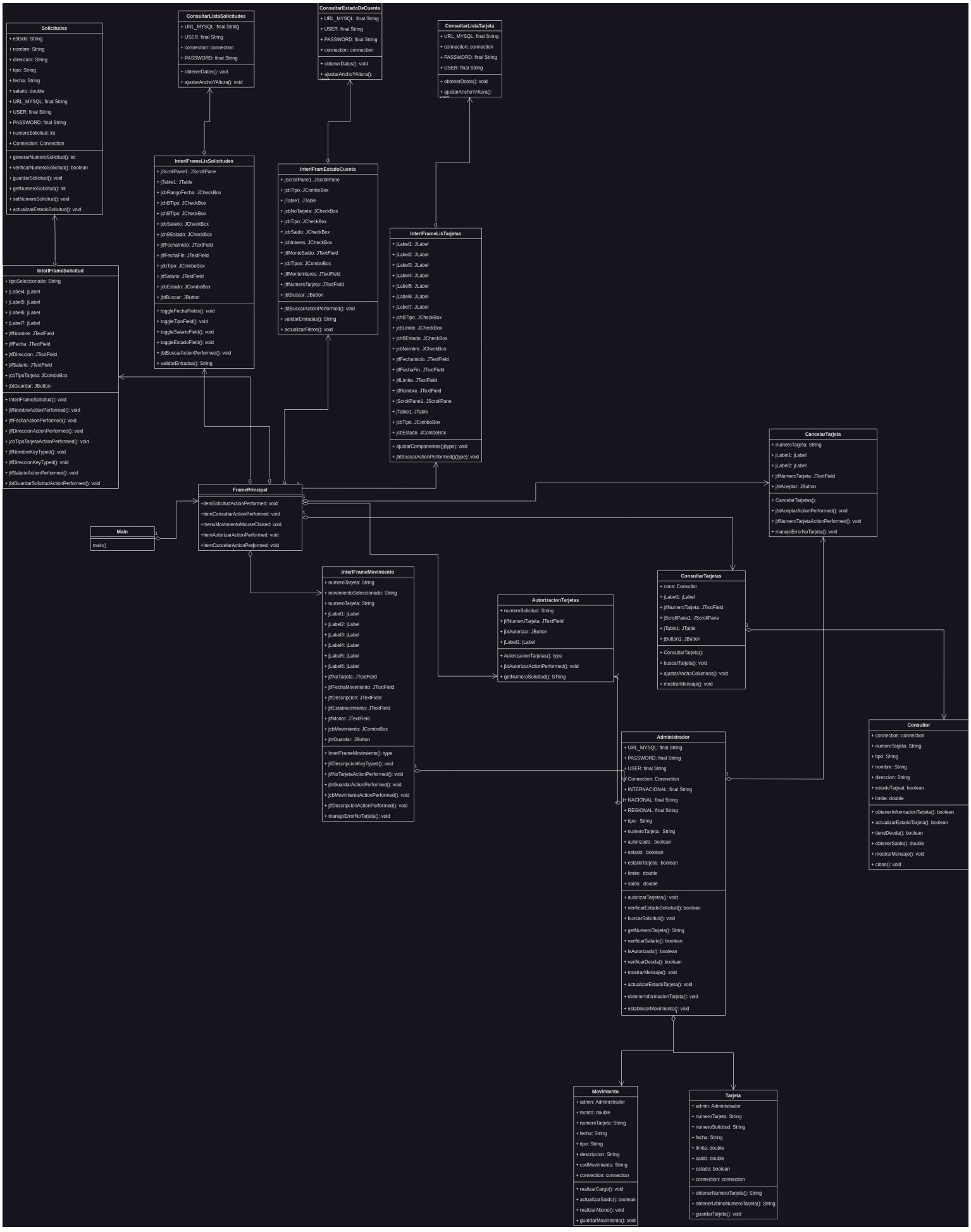
-Diagrama Entidad/Relación:



-Diagrama de Tablas:



-Diagrama Clases:



-Mapeo Físico de la Base de Datos:

```
CREATE SCHEMA control_entidad_bancaria;
USE control_entidad_bancaria;
CREATE TABLE solicitud (
    numero_solicitud INT NOT NULL,
    fecha_solicitud DATE,
    tipo VARCHAR(13) NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    salario DECIMAL(20,2) NOT NULL,
    direccion VARCHAR(150) NOT NULL,
    estado_solicitud BOOLEAN NULL,
    CONSTRAINT PK_SOLICITUD PRIMARY KEY (numero_solicitud)
);


CREATE TABLE tarjetas (
    numero_tarjeta VARCHAR(20) NOT NULL,
    fecha_estado DATE,
    numero_solicitud INT NOT NULL,
    limite DECIMAL(20,2) NOT NULL,
    saldo DECIMAL(20,2) NOT NULL,
    estado_tarjeta BOOLEAN NOT NULL,
    CONSTRAINT PK_TARJETA PRIMARY KEY (numero_tarjeta),
    CONSTRAINT FK_SOLICITUD_IN_NUMERO_SOLICITUD
            FOREIGN KEY (numero_solicitud) REFERENCES solicitud_nueva
(numero_solicitud)
);


CREATE TABLE movimientos (
    numero_movimiento INT AUTO_INCREMENT PRIMARY KEY,
    numero_tarjeta VARCHAR(20) NOT NULL,
    fecha_movimiento DATE NOT NULL,
    tipo_movimiento VARCHAR(10) NOT NULL,
    descripcion VARCHAR(200) NOT NULL,
    establecimiento VARCHAR(10) NOT NULL,
    monto DECIMAL(20,2) NOT NULL,
    CONSTRAINT FK_NUMERO_TARJETA
                    FOREIGN KEY (numero_tarjeta) REFERENCES tarjetas
(numero_tarjeta)

);
```