MANUAL TÉCNICO
Proyecto Final

Nombre:

Dany Noé de León Santizo

Carnet:

201830006

Tecnologías Compatibles

- NetBeans, IntelliJ, Eclipse
- Java 8, 12, 15, 17, 21
- Windows XP, Windows 7, Windows 8, Windows 10 , Windows 11, Linux, MacOS

DESCRIPCIÓN MÉTODOS CREADOS

CLASE SOLICITAR

Hacemos uso de lo siguiente para asignar un límite a la tarjeta en base al tipo que fue solicitado

```
if (ENUMTipoTarjeta.values()[tipo] == ENUMTipoTarjeta.NACIONAL) {
    this.limiteTarjeta = 5000;
} else if (ENUMTipoTarjeta.values()[tipo] == ENUMTipoTarjeta.REGIONAL) {
    this.limiteTarjeta = 10000;
} else {
   this.limiteTarjeta = 20000;
}
this.estado = "TRAMITE";
```

Método PROCESAR TRAMITE

```
public void procesarTramite() {
    this.sql.getConnection();
    //Meter datos del usuario
    String comandoUsuario = "INSERT INTO usuario (nombre, direccion, salario) VALUES ('" + nombreUsuario
            + "', '" + direccion + "','" + salario + "');";

    //Meter datos a la tarjeta
    String comandoTarjeta = "INSERT INTO tarjeta (tipo, limite, estado) VALUES ('" + tipo
            + "', '" + limiteTarjeta + "','" + estado + "');";

    //Abajo muestra el último numero de solicitud
    numeroSolicitud = sql.escribirDatosSolicitudSQL(comandoUsuario, comandoTarjeta);
    if (!numeroSolicitud.equals("")) {
        txtNoSolicitud.setText("Su número de solicitud es: " + numeroSolicitud);
    }

    JOptionPane.showMessageDialog(null, "Solicitud Procesada", "", JOptionPane.PLAIN_MESSAGE);
}
```

En esta clase tenemos otro método el cual inserta los datos que el usuario nos da para solicitar su tarjeta, notar que ingresamos el dato de "direccion", "salario", "limiteTarjeta", "estado" y se devuelve un numero de solicitud.

CLASE AUTORIZACION

Método agregarFechasCreacionYModificacion

```java
public void agregarFechasCreacionYModificacion() throws ParseException {
    Date thisDate = new Date();
    java.sql.Date sqlDate = new java.sql.Date (thisDate.getTime());
    //SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    //this.fechaCreacion = dateFormat.format(thisDate);
    //Date date = dateFormat.parse(fechaCreacion);
    System.out.println("Cambiando fecha");
    //Los ingresa a la base de datos
    try {
        String comandoIngresarFechaCreacion = "UPDATE tarjeta set fecha_creacion = " + "\"" + sqlDate + "\""
                    + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
        Statement statementInsert = connection.createStatement();
        statementInsert.executeUpdate(comandoIngresarFechaCreacion);

        comandoIngresarFechaCreacion = "UPDATE tarjeta set fecha_ultima_modificacion = " + "\"" + sqlDate + "\""
                    + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
        statementInsert.executeUpdate(comandoIngresarFechaCreacion);

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

DESCRIPCION: es un método que nos permite ingresar a la base de datos en la tabla "tarjeta" la fecha_creacion y también fecha_ultima_modificacion

Método cambiarNumeroTarjeta

```java
public void cambiarNumeroTarjeta(String tipoTarjeta) {
    //Verifica que haya un valor asignado a tarjetas del mismo tipo, si no eixste lo asigna,
    //si existe encuentra el numero más grande según tipo de tarjeta (nacional, regional, internacional)
    try {
        String verificarNumeroMasGrande = "SELECT * FROM tarjeta WHERE tipo like " + "\"" + tipoTarjeta + "\"" + " AND numero_tarjeta is not null";
        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(verificarNumeroMasGrande);
        //Si no existe asigna un valor inicial
        if (!resultSet.next()) {
            //Asignar un valor inicial según el tipo de tarjeta (Nacional, Regional o Internacional)

            String comandoEscogerValorInicialTipo = "";
            if (tipoTarjeta.equals("NACIONAL")) {
                comandoEscogerValorInicialTipo = "UPDATE tarjeta set numero_tarjeta = " + "\"" + valorInicialNacional + "\""
                        + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
            } else if (tipoTarjeta.equals("REGIONAL")) {
                comandoEscogerValorInicialTipo = "UPDATE tarjeta set numero_tarjeta = " + "\"" + valorInicialRegional + "\""
                        + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
            } else {
                comandoEscogerValorInicialTipo = "UPDATE tarjeta set numero_tarjeta = " + "\"" + valorInicialInternacional + "\""
                        + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
            }

              String comandoUsuario = "INSERT INTO tarjeta where tipo like " + "\"" + tipoTarjeta + "\"" + " (numero_tarjeta) VALUES ('" + valorInicialNacional
              + "');";
            statementInsert.executeUpdate(comandoEscogerValorInicialTipo);
        } else {
            //Acá es cuando ya se asignó un número a las tarjetas, extrae el valor más grande y le sumar 1 con el método
            String numeroTarjeta = "select max(numero_tarjeta) from tarjeta";
            statementInsert = connection.createStatement();
            resultSet = statementInsert.executeQuery(numeroTarjeta);
            String numeroTarjetaAnterior = "";

            if (resultSet.next()) {
                numeroTarjetaAnterior = resultSet.getString("max(numero_tarjeta)");
            }

            //Escribir nuevo numero tarjeta
            String comandoIngresarNumeroTarjeta = "UPDATE tarjeta set numero_tarjeta = " + "\"" + nuevoNumeroTarjeta(numeroTarjetaAnterior) + "\""
                        + " WHERE numero_solicitud = " + "\"" + numeroSolicitud + "\"" + " ";
            statementInsert.executeUpdate(comandoIngresarNumeroTarjeta);
```

Descripción, a este método se le da un String que contiene el número más alto de tarjeta, si no hay un valor inicial de tarjeta entonces se le asignará un valor en base al tipo de tarjeta (como se ve abajo), en dado caso ya haya un valor de tarjeta entonces le sumará 1 unidad y luego se lo asignará a la tarjeta que se está autorizando

```
private final String valorInicialNacional = "4256 3102 6540 0000";
private final String valorInicialRegional = "4256 3102 6560 0000";
private final String valorInicialInternacional = "4256 3102 6580 0000";
```

Método tipoTarjeta

```
public String tipoTarjeta() {
    //Averigua el tipo de tarjeta
    String tipoTarjeta = "";
    try {
        String numeroTarjeta = "SELECT * FROM tarjeta WHERE numero_solicitud like " + numeroSolicitud + "";
        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(numeroTarjeta);
        if (resultSet.next()) {
            tipoTarjeta = resultSet.getString(("tipo"));
        }
    } catch (Exception e) {
        System.out.println("Error buscar tipo tarjeta");
        e.printStackTrace();
    }
    return tipoTarjeta;
}
```

DESCRIPCION: nos permite saber que tipo de tarjeta es la que estamos autorizando, esto nos servirá para definir el número de la tarjeta, como se vió más arriba

Método nuevoNumeroTarjeta

```
public String nuevoNumeroTarjeta(String numeroTarjetaAnterior) {
    String[] divNumeroTarjeta = numeroTarjetaAnterior.split("");
    //Abajo mete el número final es uno solo
    String numeroNuevo = divNumeroTarjeta[13] + divNumeroTarjeta[15] + divNumeroTarjeta[16] + divNumeroTarjeta[17]
            + divNumeroTarjeta[18];

    int numeroNuevoInt = Integer.parseInt(numeroNuevo);
    numeroNuevoInt++;

    String numeroNuevoString = String.valueOf(numeroNuevoInt);

    //Abajo agrega ceros que son eliminados al pasarlo a int
    if (numeroNuevoString.length() < 5) {
        int cerosPorAgregar = 5 - numeroNuevoString.length();
        for (int cerosAgregados = 0; cerosAgregados < cerosPorAgregar; cerosAgregados++) {
            numeroNuevoString = "0" + numeroNuevoString;
        }
    }

    String[] divNumeroNUevo = numeroNuevoString.split("");
    //Abajo mete el número nuevo de tarjeta
    divNumeroTarjeta[13] = divNumeroNUevo[0];
    divNumeroTarjeta[15] = divNumeroNUevo[1];
    divNumeroTarjeta[16] = divNumeroNUevo[2];
    divNumeroTarjeta[17] = divNumeroNUevo[3];
    divNumeroTarjeta[18] = divNumeroNUevo[4];
    String nuevoNumeroTarjeta = "";
    for (String numero : divNumeroTarjeta) {
        nuevoNumeroTarjeta += numero;
    }

    return nuevoNumeroTarjeta;
}
```

DESCRIPCION: en éste método lograremos sumarle una unidad al numero de tarjeta más grande que exista en la base de datos, esto nos servirá para asignar un número de tarjeta

CLASE MOVIMIENTO

Método procesarIntereses

```
public void procesarInteres() {
    //No puede superar el limite, si lo supera no se procesa el
    //Extraer el valor que haya en saldo
    String selectTarjetaMovimiento = "SELECT * FROM tarjeta WHERE numero_tarjeta like \"" + numeroTarjeta + "\"";

    try {
        //Extrae el valor del saldo
        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(selectTarjetaMovimiento);
        if (resultSet.next()) {
            tipoTarjeta = resultSet.getString("tipo");

            if (tipoTarjeta.equals("NACIONAL")) {
                interes = saldoTarjeta.add(monto).multiply(interesNacional);
            } else if (tipoTarjeta.equals("REGIONAL")) {
                interes = saldoTarjeta.add(monto).multiply(interesRegional);
            } else if (tipoTarjeta.equals("INTERNACIONAL")) {
                interes = saldoTarjeta.add(monto).multiply(interesInternacional);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

DESCRIPCIÓN: el método nos permite calcular los intereses en base al tipo de tarjeta que tengamos

Método formatoFechaAdecuado

```
public boolean formatoFechaAdecuado() {
    DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
    Date date;

    try {
        this.fechaMov = fechaMov.replace("/", "-");
        date = dateFormat.parse(fechaMov);
        fechaFormatoSQL = new java.sql.Date(date.getTime());

        return true;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "El formato de la fecha no es el adecuado", "Formato Fecha Incorrecto", JOptionPane.PLAIN_MESSAGE);
        return false;
    }
}
```

DESCRIPCIÓN: nos permite convertir la fecha dada por el usuario en una forma que pueda ser usada por el programa.

Método crearRelacionMovimiento

```java
public void crearRelacionMovimiento() throws ParseException {
    //Estraer datos usuario según el número de tarjeta, para esto extraer el numeor de solocitud
    int idUsuario = 0;
    int numeroSolicitud = 0;
    String selectTarjetaMovimiento = "SELECT * FROM tarjeta WHERE numero_tarjeta like \"" + numeroTarjeta + "\"";

    try {
        //Estraer datos usuario según el número de tarjeta, para esto extraer el numeor de solocitud
        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(selectTarjetaMovimiento);
        if (resultSet.next()) {
            idUsuario = resultSet.getInt("numero_solicitud");
            numeroSolicitud = idUsuario;
        }
       //Ingresa los datos a la tabla de RELACION MOVIMIENTO
       String comandoCrearRelacion = "INSERT INTO movimiento (id_usuario, numero_solicitud, fecha_movimiento,"
               + " descripcion_movimiento, establecimiento_movimiento, monto, tipoMov, interes) " + "VALUE ('" + idUsuario + "', '"
               + "" + numeroSolicitud + "','" + fechaFormatoSQL + "','" + descripcionMov + "','" + establecimientoMov + "', '"
               + "" + monto + "', '" + tipoMovmiento + "', '" + interes + "');";
        statementInsert = connection.createStatement();
        statementInsert.executeUpdate(comandoCrearRelacion);
    } catch (Exception e) {
        System.out.println("Error al ingresar datos de la relacion");
        e.printStackTrace();
    }
}
```

DESCRIPCIÓN: es uno de los métodos más importante ya que crea la relación entre lsa tablas definidas en MYSQL (tarjeta y usuario), acá usamos el comando que va a relacionar a ambas en la tabla "movimiento"

Método saldoDisponible

```java
public boolean saldoDisponible() {
    //Abajo el limite es según el tipo de tarjeta
    //Abajo en el if el -1 significa menor y el 0 igual al valor
    if ((saldoTarjeta.add(monto)).compareTo(limiteTarjeta) == -1 || (saldoTarjeta.add(monto)).compareTo(limiteTarjeta) == 0) {
        //Hay suficiente dinero para gastar
        return true;
    } else {
        //No hay dinero para gastar
        JOptionPane.showMessageDialog(null, "No hay suficiente saldo disponible para el monto seleccionado, saldo disponible \""+ saldoTarjeta
                + "\" ", "Saldo Insuficiente", JOptionPane.PLAIN_MESSAGE);
    }

    return false;
}
```

DESCRIPCIÓN: verifica que hay suficiente dinero para hacer un movimiento bancario

Métodos abonar y cargo

```java
public void cargoTarjeta() {
    String comandoCargo = "UPDATE tarjeta set saldo = " + "\"" + saldoTarjeta.add(monto) + "\""
            + " WHERE numero_tarjeta = " + "\"" + numeroTarjeta + "\"" + " ";
    try {
        Statement statementInsert = connection.createStatement();
        statementInsert.executeUpdate(comandoCargo);

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

public void abonarTarjeta() {
    String comandoAbono = "UPDATE tarjeta set saldo = " + "\"" + saldoTarjeta.subtract(monto) + "\""
            + " WHERE numero_tarjeta = " + "\"" + numeroTarjeta + "\"" + " ";
    try {
        Statement statementInsert = connection.createStatement();
        statementInsert.executeUpdate(comandoAbono);

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

DESCRIPCIÓN: estos dos métodos ejecutan los comandos que van a sumar o restar en el saldo del usuario

Las clases ESTADO DE CUENTA, LISTADO SOLICITUDES, LISTADO TARJETAS son bastante parecidos lo que cambia son los datos a ingresar

Método extraerDatosTabla

```
public void extraerDatosTabla() {
    DefaultTableModel tableModel = (DefaultTableModel) tabla.getModel();
    tableModel.setRowCount(0);

    try {
        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(comandoEstadoCuenta);
        //Existe el numero de tarjeta
        while (resultSet.next()) {
            String tipoTarjeta2 = resultSet.getString("tipo");
            String nombreUsuario = resultSet.getString("nombre");
            String direccionUsuario = resultSet.getString("direccion");
            String tipoMov = resultSet.getString("tipoMov");
            String fechaMov = resultSet.getString("fecha_movimiento");
            String descripcionMov = resultSet.getString("descripcion_movimiento");
            String establecimientoMov = resultSet.getString("establecimiento_movimiento");
            int montoMov = resultSet.getInt("monto");
            double interesesSaldo = resultSet.getDouble("interes");
            int saldo = resultSet.getInt("saldo");

            tableModel.addRow(new Object[]{numeroTarjeta, tipoTarjeta2, nombreUsuario, direccionUsuario,
                tipoMov, fechaMov, descripcionMov, establecimientoMov, montoMov, interesesSaldo, saldo});

        }
```

DESCRIPCIÓN: lo que hace es extraer de la DB los distintos datos filtrados y los mete a una tabla en la GUI

Método generarHTML, para los distintos apartados solo cambian los datos ingresados al HTML

```
public void generarHTML() throws IOException {
    File h;
    //No tarjeta, tipo tarjeta, limite, nombre, dirección, estado de tarjeta
    String html = ("<div><h1>Estado Cuenta Tarjeta \"" + numeroTarjeta + "\" </h1><p></p>");
    if (gestorArchivoBE.getPathDefinido()) {
        h = new File(gestorArchivoBE.getFile().getCanonicalPath() + "/EstadoCuenta.html");
    } else {
        h = new File("EstadoCuenta.html");
    }
    //File h = new File("Consulta.html");
    try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(h, true));
        bw.write(html);
        bw.write("<table bgcolor=\"black\">");
        bw.write("<tr bgcolor=\"grey\">");
        bw.write("<th>Número de Tajeta</th>");
        bw.write("<th>Tipo de Tarjeta</th>");
        bw.write("<th>Nombre Usuario</th>");
        bw.write("<th>Dirección Usuario</th>");
        bw.write("<th>Tipo Movimiento</th>");
        bw.write("<th>Fecha Movimiento</th>");
        bw.write("<th>Descripción Movimiento</th>");
        bw.write("<th>Establecimiento Movimiento</th>");
        bw.write("<th>Monto Movimiento</th>");
        bw.write("<th>Intereses</th>");
        bw.write("<th>Saldo</th>");

        bw.write("</tr>");

        Statement statementInsert = connection.createStatement();
        ResultSet resultSet = statementInsert.executeQuery(comandoEstadoCuenta);
        //Existe el numero de tarjeta
        while (resultSet.next()) {
```

```
while (resultSet.next()) {
    //numeroTarjeta, tipoTarjeta2, nombreUsuario, direccionUsuario,
          tipoMov, fechaMov, descripcionMov, establecimientoMov, montoMov, interesesSaldo, saldo

    String tipoTarjeta2 = resultSet.getString("tipo");
    String nombreUsuario = resultSet.getString("nombre");
    String direccionUsuario = resultSet.getString("direccion");
    String tipoMov = resultSet.getString("tipoMov");
    String fechaMov = resultSet.getString("fecha_movimiento");
    String descripcionMov = resultSet.getString("descripcion_movimiento");
    String establecimientoMov = resultSet.getString("establecimiento_movimiento");
    int montoMov = resultSet.getInt("monto");
    double interesesSaldo = resultSet.getDouble("interes");
    int saldo = resultSet.getInt("saldo");

    bw.write("<tr bgcolor=\"lightgrey\" align=\"center\">");
    bw.write("<td>" + numeroTarjeta + "</td>");
    bw.write("<td>" + tipoTarjeta2 + "</td>");
    bw.write("<td>" + nombreUsuario + "</td>");
    bw.write("<td>" + direccionUsuario + "</td>");
    bw.write("<td>" + tipoMov + "</td>");
    bw.write("<td>" + fechaMov + "</td>");
    bw.write("<td>" + descripcionMov + "</td>");
    bw.write("<td>" + establecimientoMov + "</td>");
    bw.write("<td>" + montoMov + "</td>");
    bw.write("<td>" + interesesSaldo + "</td>");
    bw.write("<td>" + saldo + "</td>");

}

bw.write("</tr>");

bw.write("</table>");
bw.write("<br>");
bw.write("</br>");
```

DESCRIPCIÓN: nos sirve para meter cada elemento de la tabla de la DB a su equivalente HTML, esto se hizo de manera manual

Método crearComandoParaFiltrar

```
public void crearComandoParaFiltrar() {
    comandoEstadoCuenta += " AND (numero_tarjeta = \"" + numeroTarjeta + "\")";

    if (filtrarTipoTarjeta) {
        comandoEstadoCuenta += " AND (tipo = \"" + tipoTarjeta + "\")";
    }
    if (filtrarSaldoMayorA) {
        comandoEstadoCuenta += " AND (saldo > " + saldoMayorA + ")";
    }
    if (filtrarInteresMayorA) {
        comandoEstadoCuenta += " AND (interes > " + interesMayorA + ")";
    }
    System.out.println("COmando creado: " + comandoEstadoCuenta);
}
```

DESCRIPCIÓN: este método nos permite crear un comando en base a los distintos filtros existentes