

```
private void jLabel7MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int dialog = JOptionPane.YES_NO_OPTION;
    int result = JOptionPane.showConfirmDialog(null, "Desea salir del login?", "Exit", dialog);

    if(result == 0){
        System.exit(0);
    }
}
```

Se declara una variable “dialog” de tipo entera que es la que captura la opción que el usuario de clic ya sea si es si o no para después en la variable result de tipo entera obtenga el valor y además muestre un mensaje el cual dirá si desea salir del login para luego ser evaluado el valor obtenido y si es igual a cero que es el valor que asigna cuando se cliquea “no” cerrara la aplicación.

```
private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.setState(Login.ICONIFIED);
}
```

Al label se le crea el evento mouseClicked el cual al darle clic con la función setState se le manda como paramatro el nombre del formulario a minimizar.

```

private void btnIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    if(!txtEmail.getText().matches("^\\w{5,8}\\w{3,}(\\.\\w{1,4}){1,3}$") || IsNumeric(txtEmail.getText())) && (!txtEmail.getText().equals("root")){
        JOptionPane.showMessageDialog(this, "Email Ingresado es Invalido o Numerico");
        return;
    }
    if(txtPassword.getText().equals("") || IsNumeric(txtPassword.getText())){
        JOptionPane.showMessageDialog(this, "Contraseña no puede estar vacio o ser numerico");
        return;
    }
    try{
        Connection conn = Conexion.Conectarse(); //Obtenemos la conexion
        if(conn == null){
            throw new Exception("No se Pudo Conectar");
        }
        CallableStatement proc; //Declara un objeto de CallableStatement
        proc = conn.prepareCall("{call loguearse (?,?)}"); //Se encierra entre { la instruccion call y el procedimiento}
        proc.setString(1,txtEmail.getText()); //Segun los ? se le asigna sus valores siguiendo el orden y su tipo
        proc.setString(2, txtPassword.getText()); //x2
        if(txtEmail.getText().equals("root") && txtPassword.getText().equals("root")){
            ResultSet rs = proc.executeQuery(); //Si el procedimiento es un select se guarda en un rs y se executeQuery()
            if(rs.next()){
                rs.first();
                boolean haveto = false;
                if(DefaultPass(rs.getString(2)).equals(txtPassword.getText())){
                    haveto = true;
                    System.out.println("Tiene que cambiar el Pass");
                }
                System.out.println(rs.getInt(1));
                switch (rs.getString(3)){
                    case "Administrador":
                        AdministradorMain AM = new AdministradorMain();
                        AM.setidEmpleado(rs.getInt(1));
                        AM.setNombreUser(rs.getString(2));
                        AM.setidDepartament(rs.getInt(4));
                        AM.setNombreDepartamento(rs.getString(5));
                        AM.setHaveToChangePass(haveto);
                        AM.setVisible(true);
                        break;
                    case "Empleado":
                        TesterMain TM = new TesterMain();
                        TM.setidEmpleado(rs.getInt(1));
                        TM.setNombreUser(rs.getString(2));
                        TM.setidDepartament(rs.getInt(4));
                        TM.setNombreDepartamento(rs.getString(5));
                        TM.setHaveToChangePass(haveto);
                        TM.setVisible(true);
                        break;
                    case "Programador":
                        ProgramadorMain EM2 = new ProgramadorMain();
                        EM2.setidEmpleado(rs.getInt(1));
                        EM2.setNombreUser(rs.getString(2));
                        EM2.setidDepartament(rs.getInt(4));
                        EM2.setNombreDepartamento(rs.getString(5));
                        EM2.setHaveToChangePass(haveto);
                        EM2.setVisible(true);
                }
                this.dispose();
            }else{
                JOptionPane.showMessageDialog(null, "Usuario y/o Contraseña Incorrecto");
            }
        }catch (Exception e){
            JOptionPane.showMessageDialog(null, "Ha Ocurrido un error al Conectarse");
            System.out.println(e.getMessage());
        }
    }
}

```

En el botón de iniciar sesión lo que se realiza es la validación del correo que no sea numérica, que cumpla con la expresión regular y además que sea diferente de root luego procedemos a validar la contraseña para que el campo no esté vacío y además que no sea numérica.

Después dentro del try creamos la conexión a la base luego se verifica si es de tipo null lo que significa que no se pudo realizar una conexión a la base de datos para luego declara un objeto de tipo CallableStatement el cual nos permitirá mandar a llamar los procedimientos almacenado logearse que recibe dos parámetros que es el email y la contraseña donde ambos son de tipo string por lo cual a ambos se les antepone setString la posición que recibe y la información que obtiene de txt luego se verifica si el email es root y la contraseña igual, continuando con el flujo del sistema si el procedimiento es un select este se almacena en un rs de tipo ResultSet y se ejecuta el Query.

Luego se verifica si la contraseña es igual a que se asigna por defecto la cual según la función DefaultPass dice que la contraseña tiene que ser el nombre del empleado al revés seguido del número 503 y si la contraseña esta de esa manera le solicita que cambie la contraseña.

Luego evalúa los casos dependiendo del tipo de rol de cada usuario el cual nos sirve para que de esa manera pueda acceder a su correspondiente Main sabiendo el rol y al departamento que pertenece como por ejemplo el rol 1 que es el de administrador le accedera al main administrador donde podrá agregar, eliminar, modificar y buscar los departamentos, los empleados y los roles y si sucesivamente con cada uno de los roles designados por el administrador a cada empleado.

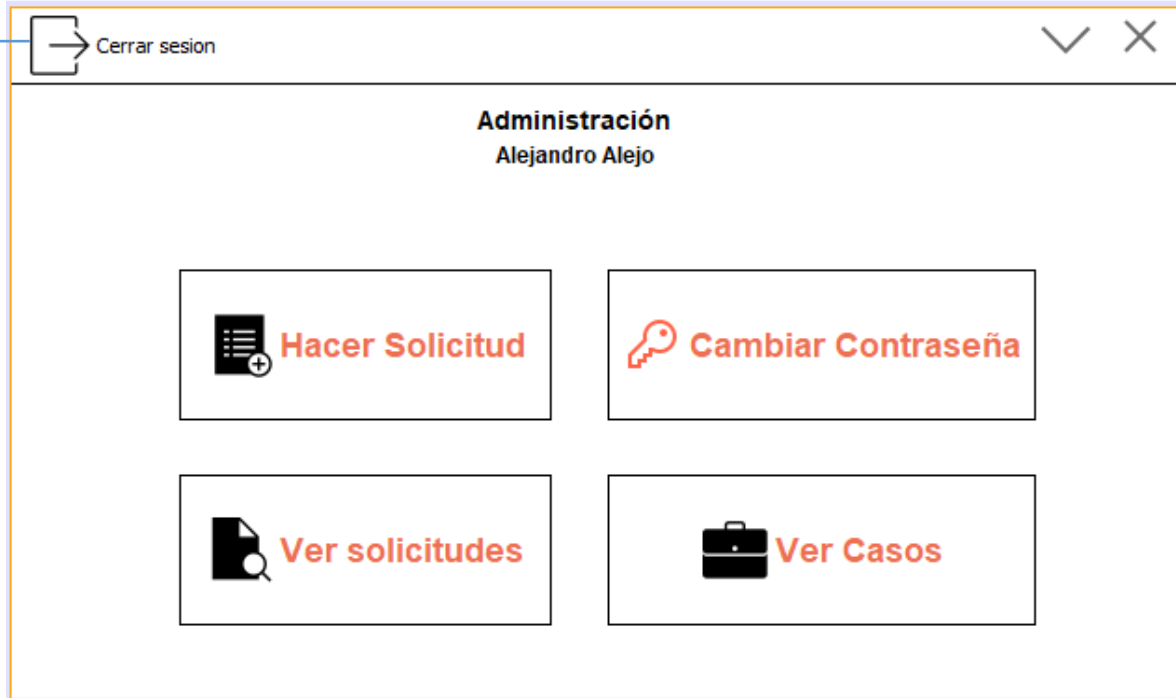
Ya para finalizar el botón iniciar sesión está el catch el cual se encarga de capturar los errores durante la ejecución.

```
private boolean IsNumeric(String x){
    try{
        int y = Integer.parseInt(x);
        return true;
    }catch(Exception e){
        return false;
    }
}
```

Función que nos permite validar que los datos ingresados en los textbox no sean numéricos la cual estará en todos los formularios donde sea necesario validar que no se introduzcan datos numéricos.

```
private String DefaultPass(String x){
    String reverse = ""; // "Victor".toLowerCase() + "503";
    String nombre = x.toLowerCase();
    for(int i = nombre.length() - 1; i >= 0; i--){
        reverse = reverse + nombre.charAt(i);
    }
    reverse += "503";
    System.out.println(reverse);
    return reverse;
}
```

Esta función nos sirve para verificar que la contraseña ingresada sea el nombre del empleado al revés y que tenga el numero 503



```
private int idDepartamento;
private int idEmpleado;
private String NombreDepartamento;
private String NombreUser;
private boolean HaveToChangePass;

public void setidEmpleado(int id){
    this.idEmpleado = id;
}

public void setidDepartament(int idDepartamento){
    this.idDepartamento = idDepartamento;
}

public void setNombreDepartamento(String NombreDepartamento){
    this.NombreDepartamento = NombreDepartamento;
}

public void setNombreUser(String NombreUser){
    this.NombreUser = NombreUser;
}

private int getidDepartamento(){
    return this.idDepartamento;
}

public void setHaveToChangePass(boolean haveto){
    this.HaveToChangePass = haveto;
}
```

Se declaran las variables de tipo private las cuales obtiene el nombre el departamento el id para lograr saber qué tipo de usuario a ingresado y así poder enviar de la misma manera en que se recibieron dichas variables a los diferentes mantenimientos donde sea necesaria cada variable.

Con ese JLabel le permitirá cerrar la sesión iniciada y lo enviará al login en dado caso que dese cambiarse de cuenta o quiera hacer un proceso diferente.



## Hacer Solicitud

```
private void btnSolicitudesActionPerformed(java.awt.event.ActionEvent evt) {  
    Solicitudes s = new Solicitudes();  
    s.setidDepartamento(idDepartamento);  
    s.setVisible(true);  
}
```

En esta opción abrirá el formulario Solicitudes el cual le permitirá crea una solicitud de sistema y además se le envía el id del departamento que del usuario que ha iniciado sesión para poder saber de qué departamento se ha realizado la solicitud.



## Cambiar Contraseña

```
private void btnContraseña3ActionPerformed(java.awt.event.ActionEvent evt) {  
    ChangePass();  
}
```

```
private void ChangePass() {  
    String newPass = JOptionPane.showInputDialog(this, "Ingrese nueva contraseña");  
    if(newPass == null) {  
        JOptionPane.showMessageDialog(this, "Decidio no actualizar su contraseña\nSe le preguntara el siguiente Login");  
        return;  
    }  
    try {  
        Connection conn = Conexion.Conectarse();  
        if(conn == null) {  
            JOptionPane.showMessageDialog(this, "Fallo al conectarse");  
            return;  
        }  
        CallableStatement proc = conn.prepareCall("{ call actualizar_contraseña (?,?)}");  
        proc.setInt(1, idEmpleado);  
        proc.setString(2, newPass);  
        proc.execute();  
        System.out.println(idEmpleado + " " + newPass);  
        JOptionPane.showMessageDialog(this, "Se actualizo la contraseña");  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(this, e.getMessage());  
    }  
}
```

En esta opción mandara a llamar una función la cual le solicita al usuario cambiar su contraseña ya que tiene la contraseña por defecto que sería el nombre del empleado al revés seguido del número 503 por ejemplo el empleado de nombre Matthew su contraseña por defecto sería wehttam503 entonces al tener esta contraseña se le solicita al empleado que la cambia para luego ser actualizada en la base de datos por medio de la función ChangePass en la cual se manda a llamar el procedimiento actualizar\_contraseña el cual recibe dos parámetros el id y la contraseña nueva donde el id es el obtenido por defecto cuando recién se loguea el empleado.



## Ver solicitudes

```
private void btnSolicitudes1ActionPerformed(java.awt.event.ActionEvent evt) {  
    GestionSolicitudes e = new GestionSolicitudes();  
    e.setidDepartamento(getidDepartamento());  
    e.setVisible(true);  
}
```

Esta opción le permitiría al jefe de Área visualizar las solicitudes que ha realizado y poder cancelar la solicitud que ya no quiere que sea aprobada.



**Ver Casos**

```
private void btnCasosActionPerformed(java.awt.event.ActionEvent evt) {  
    GestionCasos c = new GestionCasos();  
    c.setidDepartamento(idDepartamento);  
    c.setVisible(true);  
}
```

En esta opción le permitirá al empleado acceder al formulario ver casos y visualizar el programador que está realizando el sistema que ha solicitado y podrá adicionar observaciones.

**Crear solicitud**

**Nombre:** \_\_\_\_\_ **Descripcion:** \_\_\_\_\_

**Enviar**

```
private void jLabel6MouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    this.dispose();  
}
```

Este label con el evento mouseclicked le permitirá al empleado regresar al menú de jefe de área después de haber realizado una solicitud o si quiere acceder a otra opción de las que tiene disponible.

```

private int idDepartamento;
private boolean IsNumeric(String x){
    try{
        int y = Integer.parseInt(x);
        return true;
    }catch(NumberFormatException e){
        return false;
    }
}

public void setidDepartamento(int idDepartamento){
    this.idDepartamento = idDepartamento;
}

```

Se declara la variable idDepartamento de tipo global la cual almacenara el id del departamento del empleado que ha iniciado sesión para ser utilizada en el botón de realizar solicitud y la función IsNumeric nos servirá para validar si los datos ingresados no sean numéricos.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtNombre.getText().equals("") || IsNumeric(txtNombre.getText())){
        JOptionPane.showMessageDialog(null, "El nombre no puede quedar vacio");
        return;
    }
    if(txtDescripcion.getText().equals("") || IsNumeric(txtDescripcion.getText())){
        JOptionPane.showMessageDialog(null, "La descripcion no puede quedar vacia");
        return;
    }

    try{
        Connection con = Conexion.Conectarse();
        if(con == null){
            throw new Exception("No se pudo Conectar");
        }
        CallableStatement proc;
        proc = con.prepareCall("{call realizar_solicitud (?,?,?) }");
        proc.setString(1, txtNombre.getText());
        proc.setString(2, txtDescripcion.getText());
        proc.setInt(3, idDepartamento);
        ResultSet rs = proc.executeQuery();
        rs.next();
        System.out.println(rs.getString(1));
        JOptionPane.showMessageDialog(this, rs.getString(1));
        this.dispose();
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(this, e.getMessage());
        System.out.println(e.getMessage());
    }
}




```

El botón enviar nos permitirá enviar la solicitud que el empleado necesite en el cual se valida si los datos ingresados no están vacíos y además que no sean numéricos para luego proceder al try en el cual se realizara la conexión con la base para luego declarar un objeto de tipo CallableStatement el cual permitirá realizar la ejecución del procedimiento almacenado realizar\_solicitud el cual solo recibe tres parámetros por que dicha solicitud media vez es creada por defecto el estado que tiene En espera de respuesta el cual el jefe de desarrollo lo revisara y el decidirá si lo aprueba o no y en caso que lo apruebe será asignado a un programador.

Además, como el procedimiento almacenado recibe tres parámetros se le envía lo obtenido en el txt de nombre como primer parámetro como segundo el dato obtenido en el txt descripción y como tercer parámetro el parámetro que se obtiene cuando el empleado se logue el cual es el

idDepartamento luego se ejecuta el query y por ultimo está el catch el cual recibe la excepción de los problemas encontrados en el proceso de ejecución.

## Gestión de Solicitudes



**:::Solicitudes:::**

**Información Solicitud**

**Nombre:**

**Descripción:**

**Acciones**

Cancelar

Anterior

Siguiente



```

private void btnSiguienteActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        btnAnterior.setEnabled(true);
        if(Data.next()){
            txtid.setText(Data.getString(1));
            txtNombre.setText(Data.getString(2));
            txtDescripcion.setText(Data.getString(3));
            if(!Data.next()){
                btnSiguiente.setEnabled(false);
            }
            Data.previous();
        }
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
}

```

```

private void btnAnteriorActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        btnSiguiente.setEnabled(true);
        if(Data.previous()){
            txtid.setText(Data.getString(1));
            txtNombre.setText(Data.getString(2));
            txtDescripcion.setText(Data.getString(3));
            if(!Data.previous()){
                btnAnterior.setEnabled(false);
            }
            Data.next();
        }
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
}

```

El botón siguiente al cargar el formulario se carga la primera solicitud que el empleado a realizado en dado caso que el haya realizado muchas más al presionar el botón siguiente el podrá ir viendo una por una las solicitudes hechas y en caso que dese regresar a la anterior el presiona el botón anterior los cuales obtienen el dato anterior de Data por medio de previous() o el siguiente de la misma manera solo que se utiliza el next() y se llenan los campos del formulario con los datos obtenidos y de esa manera poder cancelar una solicitud al dar clic en cancelar que manda a llamar el procedimiento almacenado eliminar\_solicitud el cual recibe un parámetro que es el id de la solicitud consultada.



**:::Casos:::**

**Información Caso**

**Nombre:**

**Codigo:**

**Descripción:**

**Fecha Limite:**

**Programador:**

-- Seleccione Uno

**Tester:**

-- Seleccione Uno

**Observaciones:**

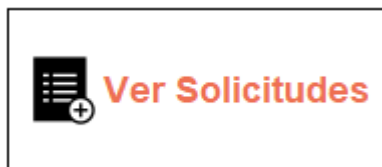
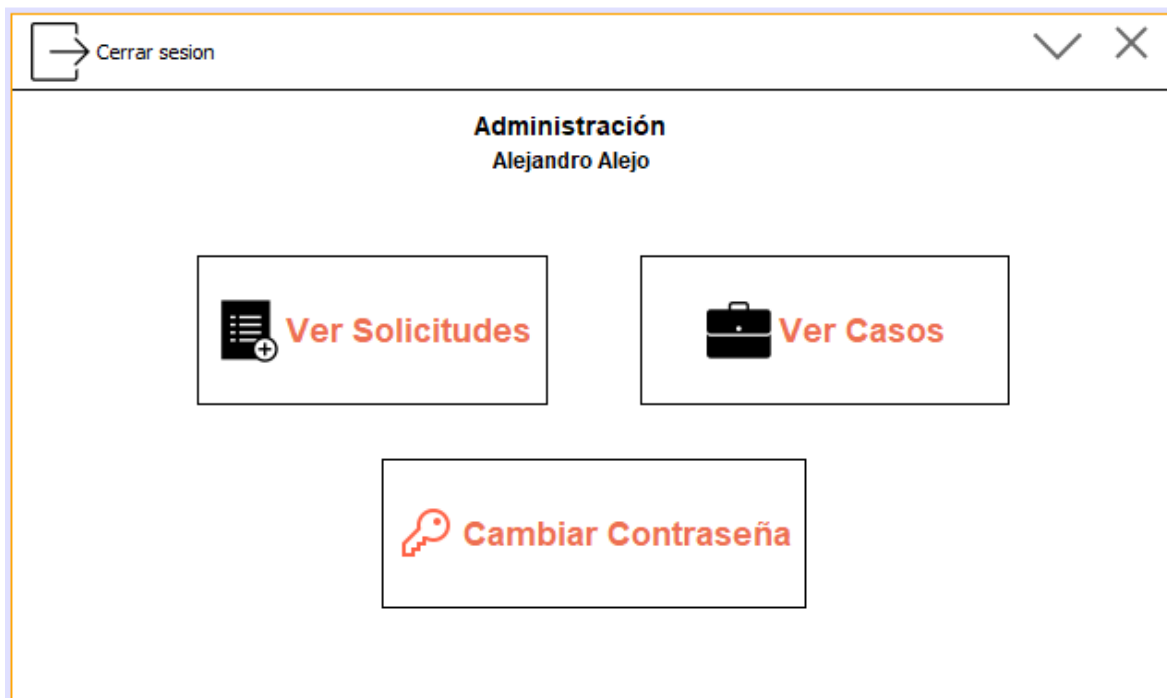
(Opcional)

**Acciones**

Anterior

Siguiente

En esta función el empleado solicitante podrá gestionar los casos de las diferentes solicitudes que él ha realizado y poder dar observacions visualizar el encargado de realizar el sistema y el encargado de probar el sistema antes de ser entregado para la prueba final.



```
private void btnSolicitudesActionPerformed(java.awt.event.ActionEvent evt) {
    GestionSolicitudes e = new GestionSolicitudes();
    e.setidDepartamento(getidDepartamento());
    e.setVisible(true);
}
```

El jefe de desarrollo por medio de esta opción podrá visualizar el formulario de las solicitudes hechas por los jefes de área enviando como parámetro el id del departamento que se obtienen del logueo.



```
private void btnCasosActionPerformed(java.awt.event.ActionEvent evt) {
    GestionCasos c = new GestionCasos();
    c.setidDepartamento(idDepartamento);
    c.setVisible(true);
}
```

En esta opción el jefe de desarrollo podrá acceder al formulario de ver caso y así poder visualizar los avances del proyecto que se le ha asignado a cada programador.

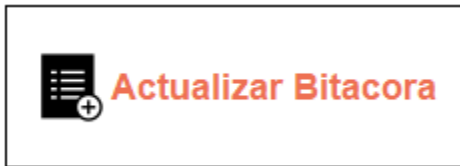
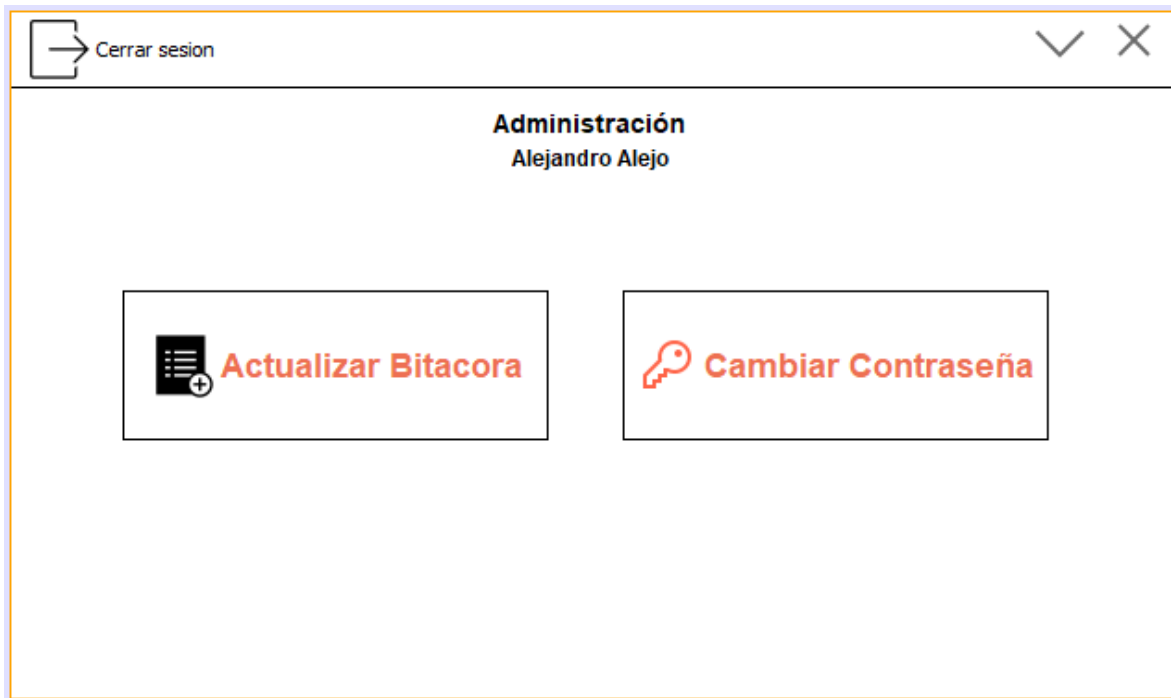


## Cambiar Contraseña

```
private void btnContraseña3ActionPerformed(java.awt.event.ActionEvent evt) {  
    ChangePass();  
}
```

```
private void ChangePass() {  
    String newPass = JOptionPane.showInputDialog(this, "Ingrese nueva contraseña");  
    if(newPass == null) {  
        JOptionPane.showMessageDialog(this, "Decidio no actualizar su contraseña\nSe le preguntara el siguiente Login");  
        return;  
    }  
    try {  
        Connection conn = Conexion.Conectarse();  
        if(conn == null) {  
            JOptionPane.showMessageDialog(this, "Fallo al conectarse");  
            return;  
        }  
        CallableStatement proc = conn.prepareCall("{ call actualizar_contraseña (?,?) }");  
        proc.setInt(1, idEmpleado);  
        proc.setString(2, newPass);  
        proc.execute();  
        System.out.println(idEmpleado + " " + newPass);  
        JOptionPane.showMessageDialog(this, "Se actualizo la contraseña");  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(this, e.getMessage());  
    }  
}
```

En esta opción mandara a llamar una función la cual le solicita al usuario cambiar si contraseña ya que tiene la contraseña por defecto que sería el nombre del empleado al revés seguido del número 503 por ejemplo el empleado de nombre Matthew su contraseña por defecto seria wehttam503 entonces al tener esta contraseña se le solicita al empleado que la cambia para luego ser actualizada en la base de datos por medio de la función ChangePass en la cual se manda a llamar el procedimiento actualizar\_contraseña el cual recibe dos parámetros el id y la contraseña nueva donde el id es el obtenido por defecto cuando recién se loguea el empleado.



```
private void btnBitacoraActionPerformed(java.awt.event.ActionEvent evt) {  
    Bitacora b = new Bitacora();  
    b.setidEmpleado(idEmpleado);  
    b.setVisible(true);  
}
```

En esta opción abrirá el formulario actualizar bitácora y se le enviará como parámetro el id del programador que está accediendo a la actualización de la bitácora.



**:::Bitacoras:::**

**Información Solicitud**

**Descripción**

**Porcentaje Avanzado**

1 ▾

**Caso**

**Nombre del caso**

**Acciones**

Actualizar

Limpiar

En este formulario se le cargara la información del proyecto asignado para que pueda ir actualizando la bitácora dependiendo de los avances que vaya realizando poniendo la descripción y el porcentaje de avance que lleva el empleado.