Prueba de Caja Negra

"Panadería Asistencias"

Integrantes: Gerald Astudillo, Henry Chalcualan, Isaac Erazo, Henry Suin

V5.0

Fecha: 2025-02-26

Historial de Revisión

Fecha	Versión	Descripción	Autores
19/Noviembre/2024	1	Versión inicial	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
11/Diciembre/2024	2	Corrección de requisitos funcionales en caja negra	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
18/Diciembre/2024	3	Se añadió una descripción de cada requisito	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
24/Enero/2025	4	Se finalizo el documento con todas las pruebas de caja negra	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
26/Febrero/2025	5	Actualización de prueba 4, se añaden pruebas 5, 6 y 7	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin

Índice

Partición de clases equivalentes 4

1. Inicio de sesión 4

2.Codigo Fuente 4

3.Particion de clases equivalentes 5

4.Captura de la pantalla de la ejecución 6

1. Registro de usuarios 8

2.Codigo Fuente 8

3.Particion de clases equivalentes 9

4.Captura de la pantalla de la ejecución 10

1. Registro de asistencias 11

2.Codigo Fuente 11

4.Captura de la pantalla de la ejecución 15

1. Administrar usuarios 16

2.Codigo Fuente 16

3.Particion de clases equivalentes 18

4.Captura de la pantalla de la ejecución 18

1. Historial de asistencias 20

2.Codigo Fuente 20

3.Particion de clases equivalentes 23

4.Captura de la pantalla de la ejecución 24

1. Consulta de datos 26

2.Codigo Fuente 26

3.Particion de clases equivalentes 27

4.Captura de la pantalla de la ejecución 27

1. Eliminar usuarios 29

2.Codigo Fuente 29

3.Particion de clases equivalentes 31

4.Captura de la pantalla de la ejecución 31

Partición de clases equivalentes

1. Inicio de sesión

HISTORIA DE USUARIO	
Número REQ 001	**Usuario:** usuario/admin
Nombre Historia: **Inicio de sesión**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** Alto
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **Este código implementa la validación de credenciales ingresadas por el usuario. Se valida que el usuario y contraseña consten en la base de datos y se verifica el nivel del sujeto para darle acceso al respectivo formulario**	
Validación: • **Los datos son validos sin constan en la base de datos**	

2.Codigo Fuente

2.1.Codigo java

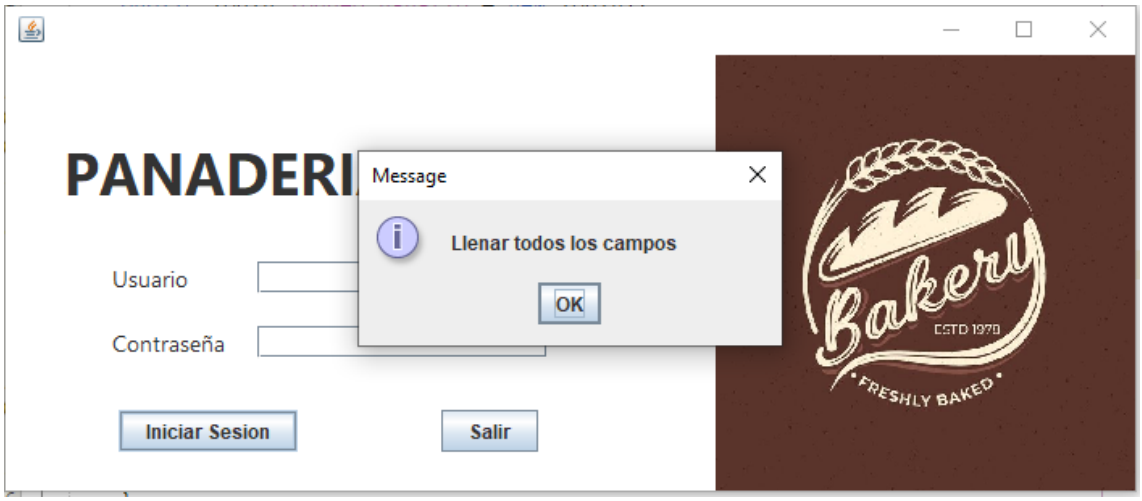
```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    usuario = user.getText();
    contraseña = new String(pass.getPassword());
    if(usuario.equals("")||contraseña.equals("")){
      JOptionPane.showMessageDialog(null,"Llenar todos los campos");
    }
    else{
      try{
        PreparedStatement ps=cn.prepareStatement("SELECT nivel FROM
usuarios WHERE cedula='"+usuario+"' AND contraseña='"+contraseña+"'");
        ResultSet rs=ps.executeQuery();
        if(rs.next()){
          String nivel=rs.getString("nivel");
```

```
            if(nivel.equalsIgnoreCase("admin")){
               admin fa= new admin();
               fa.setVisible(true);
               this.setVisible(false);
            }
            else if(nivel.equalsIgnoreCase("usuario")){
               usuario fu= new usuario();
               fu.setVisible(true);
               this.setVisible(false);
            }
          }
          else{
          JOptionPane.showMessageDialog(null, "Usuario o contraseña
incorrectos");
          }
        }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
        }}
  }
```

3.Particion de clases equivalentes

VARIABLE	CLASE DE EQUIVALENCIA	ESTADO	REPRESENTANTE
Usuario	EC1:usuario=cedula	Valido	1726167487
	EC1:usuario!= cedula	No válido	/////***°
	EC1:usuario!= cedula	No válido	asdasasdasd
	EC1:usuario!= cedula	No válido	vacío
contraseña	EC!: contraseña= contraseña	Válido	adminspass
	EC!: contraseña!= contraseña	No válido	vacío
	EC!: contraseña!= contraseña	No válido	asdasdasda
	EC!: contraseña!= contraseña	No válido	/////***°

4.Captura de la pantalla de la ejecución




Registrar asistencia
Hisotiral asistencias
REGISTRO DE ASISTENCIA
Nombre:
123
Registrar Entrada
Registrar Salida


Crear usuario
Historial de asistencias
Usuarios
Crear usuario
Nombre
Apellido
Cedula
Celular
Direccion
Contraseña
usuario
Crear cuenta

1. Registro de usuarios

HISTORIA DE USUARIO	
Número REQ 002	**Usuario:** admin
Nombre Historia: **Registro de usuarios**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** Medio
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **El administrador podrá registrar nuevos usuarios en el sistema.** • **Se debe validar que todos los campos sean ingresados correctamente.** • **No se permitirá registrar un usuario con una cédula ya existente.**	
Validación: • **Un usuario se registra correctamente si todos los datos son válidos.** • **Se mostrará un mensaje de error si falta información o si la cédula ya está registrada.**	

2.Codigo Fuente

2.1.Codigo java

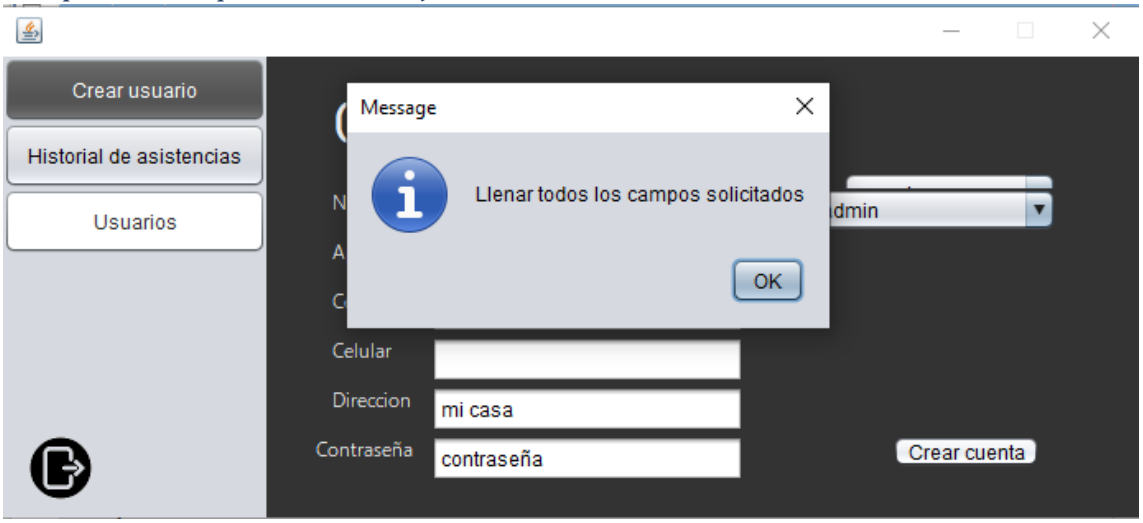
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String nombre,apellido,nivel,direc,contraseña,ceduV;
    int cedu,celular;
    nombre = nom.getText();
    apellido=apell.getText();
    nivel=level.getSelectedItem().toString();
    direc=direccion.getText();
    contraseña=pass.getText();
    ceduV = cedula.getText();
    if (nombre.isEmpty() || apellido.isEmpty() || cedula.getText().isEmpty() ||
telf.getText().isEmpty() || direc.isEmpty() || contraseña.isEmpty()){
    JOptionPane.showMessageDialog(null,"Llenar todos los campos solicitados");
    }
    else if (ceduV.length() != 10) {
     JOptionPane.showMessageDialog(null, "La cédula debe tener exactamente 10
dígitos.");
     }
```

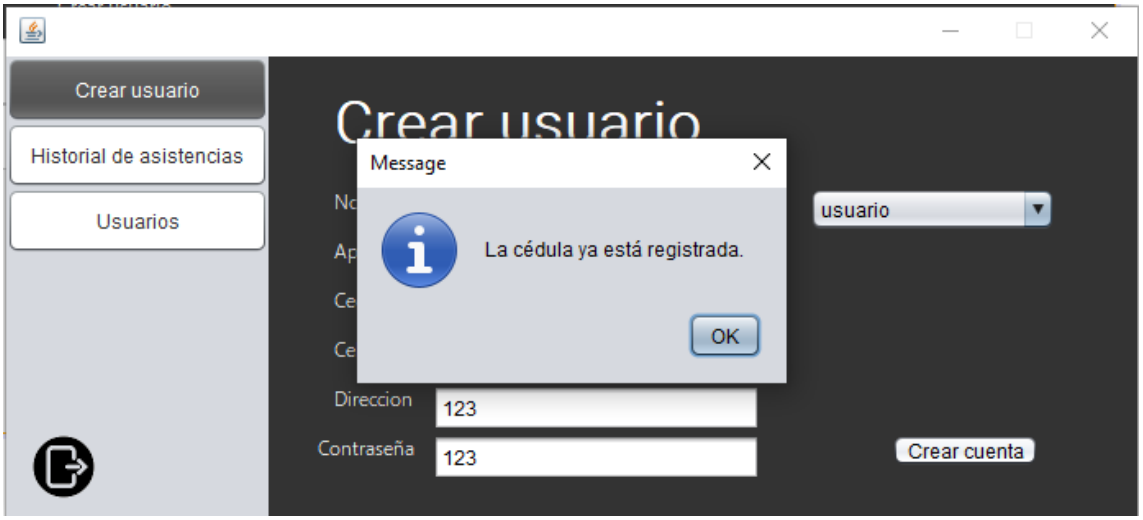
```
    else{
     try{
        cedu=Integer.parseInt(cedula.getText());
        celular=Integer.parseInt(telf.getText());
        PreparedStatement ps = cn.prepareStatement("INSERT INTO `usuarios`(`nombre`,
`apellido`, `cedula`, `celular`, `direccion`, `contraseña`, `nivel`) VALUES (?, ?, ?, ?, ?, ?, ?)");
        ps.setString(1, nombre);
        ps.setString(2, apellido);
        ps.setInt(3, cedu);
        ps.setInt(4, celular);
        ps.setString(5, direc);
        ps.setString(6, contraseña);
        ps.setString(7, nivel);
        int rowsInserted = ps.executeUpdate();
        if (rowsInserted > 0) {
        JOptionPane.showMessageDialog(null, "Usuario registrado exitosamente");}
        else {
        JOptionPane.showMessageDialog(null, "No se pudo registrar el usuario");}
        }catch(java.sql.SQLIntegrityConstraintViolationException e){
        JOptionPane.showMessageDialog(null, "La cédula ya está registrada.");
        }
        catch(Exception e){}
    }}
```

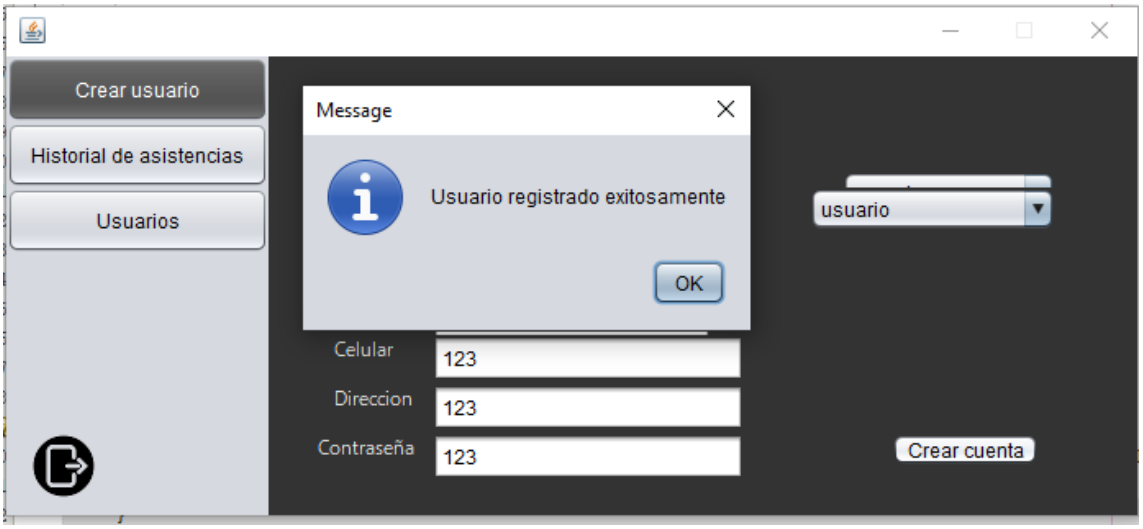
3.Particion de clases equivalentes

Variable	Clase de Equivalencia	Estado	Representante
Nombre	EC1: nombre != vacío	Válido	"Juan"
	EC2: nombre = vacío	No válido	vacío
Apellido	EC1: apellido != vacío	Válido	"Pérez"
	EC2: apellido = vacío	No válido	Vacío
Cédula	EC1: cédula = 10 dígitos numéricos	Válido	"1723456789"
	EC2: cédula < 10 o > 10 dígitos	No válido	"1234567"
	EC3: cédula con caracteres no numéricos	No válido	"abc123xyz"
Teléfono	EC1: teléfono = numérico	Válido	"0987654321"
	EC2: teléfono con caracteres alfanuméricos	No válido	"09a7b6543c"
Contraseña	EC1: contraseña != vacío	Válido	"Clave123"
	EC2: contraseña = vacío	No válido	Vacío
Dirección	EC1: dirección != “ ”	Válido	"Av. Siempre Viva 742"
	EC2: dirección == “ ”	No válido	vacío

4.Captura de la pantalla de la ejecución







1. Registro de asistencias

HISTORIA DE USUARIO	
Número REQ 003	**Usuario:** usuario
Nombre Historia: **Registro de asistencias**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** Alto
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **Un usuario o subadministrador podrá registrar su asistencia al sistema.** • **Se registrará la hora de entrada y la hora de salida.** • **No se permitirá registrar una salida sin haber registrado una entrada previamente.**	
Validación: • **Se validará que el usuario exista en la base de datos.** • **Se verificará que la fecha y hora se almacenen correctamente.** • **Se impedirá registrar una salida sin una entrada previa.**	

2.Codigo Fuente

2.1.Codigo java registro de entrada

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

    nombre.setText(usuario);


   try {

   // Primero se valida si el usuario tiene un registro de ingreso para hoy

   PreparedStatement psCheck = cn.prepareStatement("SELECT * FROM
asistencias WHERE usuario = ? AND fecha = ?");

   LocalDateTime now = LocalDateTime.now();
```

```
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd");

            String fecha = now.format(formatter);  // Solo la fecha, sin la hora


            psCheck.setString(1, usuario);

            psCheck.setString(2, fecha);


            ResultSet rsCheck = psCheck.executeQuery();


            // Si ya existe un registro, no se inserta nuevo registro de asistencia

            if (rsCheck.next()) {

               JOptionPane.showMessageDialog(null, "Ya has registrado tu ingreso hoy.");

            } else {

               // Si no existe, se registra el nuevo ingreso

               String sqlSelectNivel = "SELECT nivel FROM usuarios WHERE cedula=? AND
contraseña=?";

               PreparedStatement ps = cn.prepareStatement(sqlSelectNivel);

               ps.setString(1, usuario);

               ps.setString(2, contraseña);

               ResultSet rs = ps.executeQuery();


               if (rs.next()) {

                  // Registramos la asistencia

                  String sqlAsistencia = "INSERT INTO asistencias (usuario, fecha, hora,
hora_salida,salida,nombre) VALUES (?, ?, ?, ?,?,?)";

                  PreparedStatement psAsistencia = cn.prepareStatement(sqlAsistencia);
```

```
                String horaEntrada =
now.format(DateTimeFormatter.ofPattern("HH:mm:ss"));



                psAsistencia.setString(1, usuario);

                psAsistencia.setString(2, fecha);

                psAsistencia.setString(3, horaEntrada);  // Hora de entrada

                psAsistencia.setString(4, horaEntrada);

                psAsistencia.setInt(5, 0);

                psAsistencia.setString(6, nombre_user);

                int rowsInserted = psAsistencia.executeUpdate();

                if (rowsInserted > 0) {

                   JOptionPane.showMessageDialog(null, "Ingreso exitoso. Asistencia
registrada.");

                } else {

                   JOptionPane.showMessageDialog(null, "No se pudo registrar la
asistencia.");

                }

              }

            }

                }catch(Exception e){}
}
```

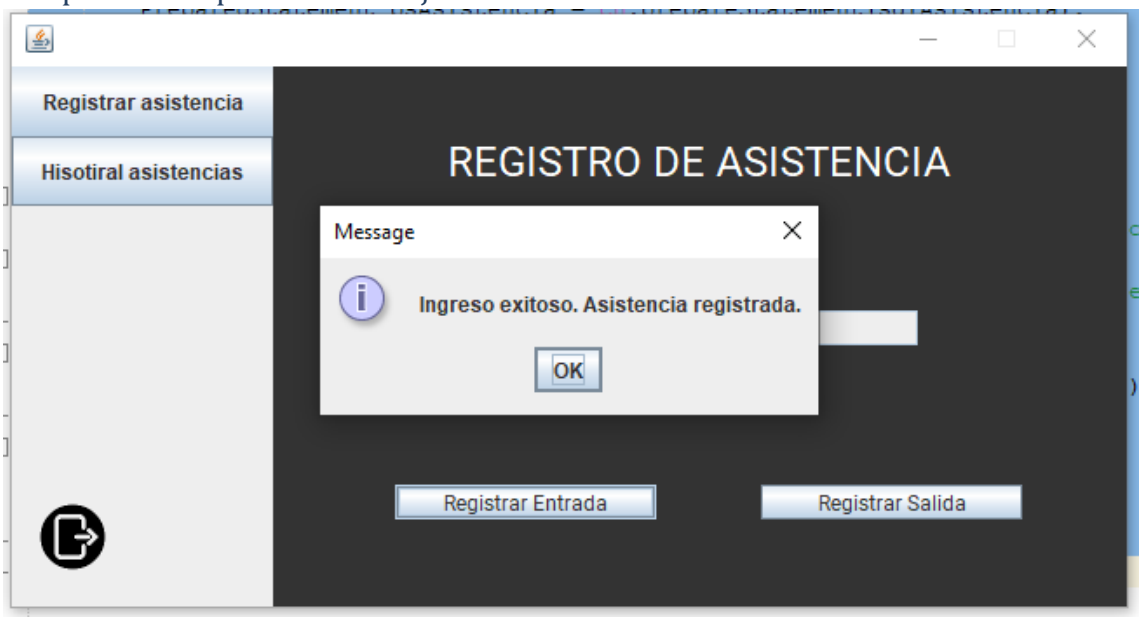
2.3.Codigo java registro de salida

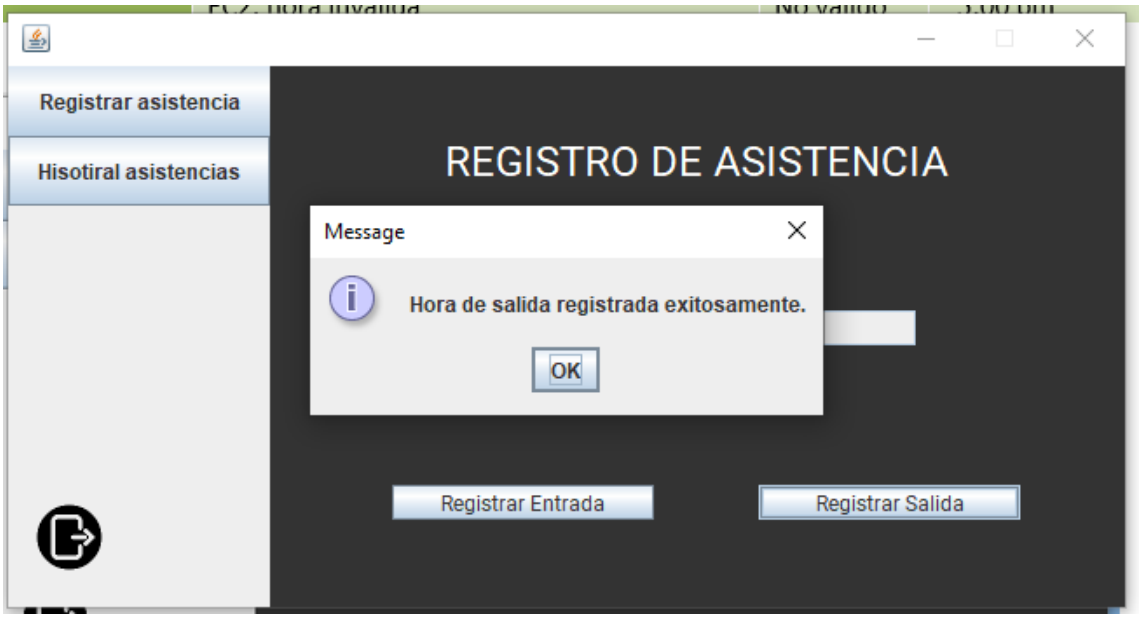
```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    try{

        PreparedStatement ps=cn.prepareStatement("SELECT nivel FROM usuarios
WHERE cedula='"+usuario+"' AND contraseña='"+contraseña+"'");

        ResultSet rs=ps.executeQuery();

        if (rs.next()) {
```

```
        // Obtener la fecha y hora actuales
        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern("HH:mm:ss");

        String fechaActual = now.format(dateFormatter); // Fecha actual
        String horaSalida = now.format(timeFormatter); // Hora actual

        // Actualizar la hora de salida en la fila correspondiente
        String sqlAsistencia = "UPDATE asistencias SET hora_salida=? WHERE
usuario="+usuario+" AND fecha=?";
        PreparedStatement psAsistencia = cn.prepareStatement(sqlAsistencia);

        psAsistencia.setString(1, horaSalida); // Establecer la hora de salida
        psAsistencia.setString(2, fechaActual); // Establecer la fecha actual

        int rowsUpdated = psAsistencia.executeUpdate();
        if (rowsUpdated > 0) {
          JOptionPane.showMessageDialog(null, "Hora de salida registrada exitosamente.");
        } else {
          JOptionPane.showMessageDialog(null, "No se encontró un registro de entrada
pendiente para este usuario en la fecha actual.");
        }
      } else {
        JOptionPane.showMessageDialog(null, "Usuario o contraseña incorrectos.");
      }
    } catch (Exception e) {
      e.printStackTrace();
```

```
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());
}
    }
```

4.Captura de la pantalla de la ejecución





1. Administrar usuarios

HISTORIA DE USUARIO	
Número REQ 004	**Usuario:** admin
Nombre Historia: **Edición de cuentas**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** medio
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **El administrador podrá modificar las cuentas de usuarios.** • **Solo los administradores podrán cambiar los datos de los usuarios.**	
Validación: • **Se validará que la cédula ingresada corresponda a un usuario existente.**	

2.Codigo Fuente

2.1.Codigo java Actualizar cuenta

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

    String nivel,celular,direccion,contrasena,nombre,apellido;

    nivel=B_nivel.getSelectedItem().toString();

    String cedulaIngresada = B_cedula.getText();

    if (cedulaIngresada.isEmpty()) {

    JOptionPane.showMessageDialog(null, "Por favor, ingresa la cédula.");

  }

    try{

       PreparedStatement verificar = cn.prepareStatement("SELECT nombre, apellido,
cedula, celular, direccion, contraseña, nivel FROM usuarios WHERE cedula = ?");
```

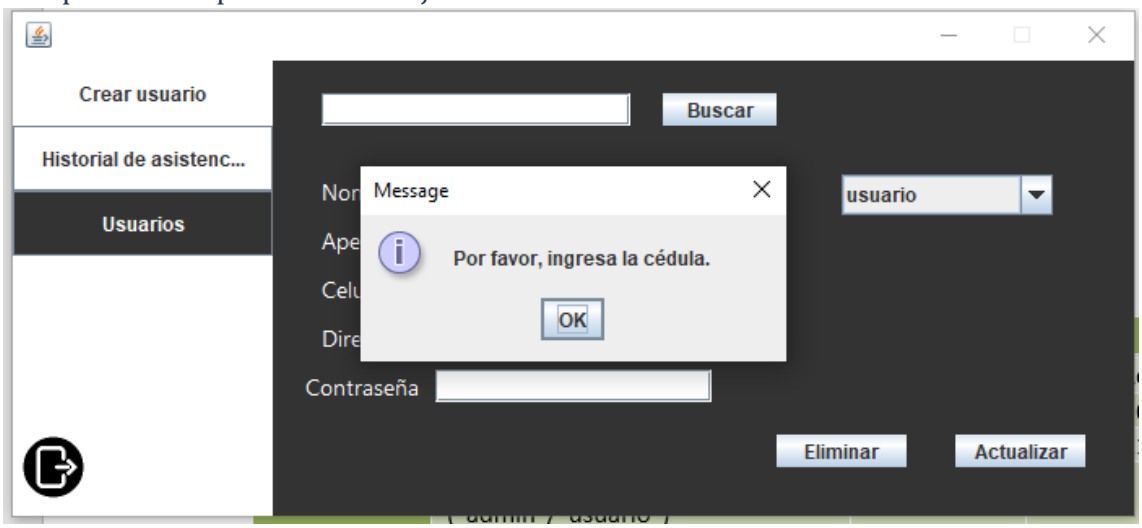
```
verificar.setString(1, cedulaIngresada);
ResultSet verificarrs = verificar.executeQuery();

if (verificarrs.next()) {
String usuario_consultado = verificarrs.getString("cedula");

// Verificar si se intenta editar al usuario actual
if (usuario_consultado.equals(usuario)) {
   JOptionPane.showMessageDialog(null, "No se puede editar los datos del usuario actual.");
}
else{
celular = B_celular.getText();
direccion = B_direccion.getText();
contrasena = B_contraseña.getText();
nombre =  B_nombre.getText();
apellido = B_apellido.getText();

PreparedStatement ps = cn.prepareStatement("UPDATE usuarios SET nivel = ?,celular = ?,direccion = ?, contraseña = ?,nombre = ?,apellido = ? WHERE cedula = ?");
ps.setString(1, nivel);
ps.setString(2, celular);
ps.setString(3, direccion);
ps.setString(4, contrasena);
ps.setString(5,nombre);
ps.setString(6,apellido);
ps.setString(7, cedulaIngresada);
int rowsUpdated = ps.executeUpdate();
```

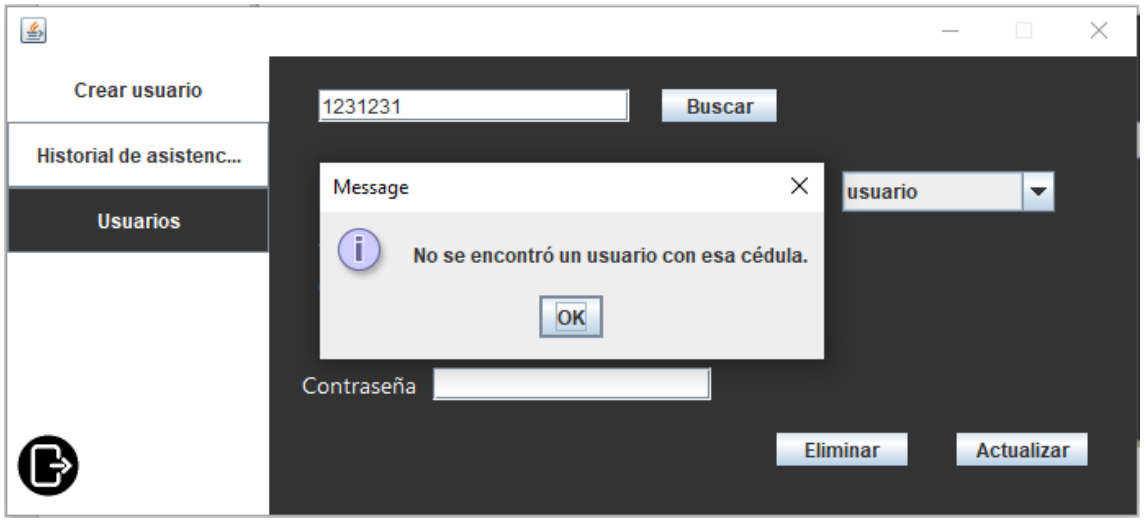
```
        if (rowsUpdated > 0) {
        JOptionPane.showMessageDialog(null, "Usuario actualizado exitosamente.");
    }}}
    }catch(Exception e){
    JOptionPane.showMessageDialog(null, e);}
}
```

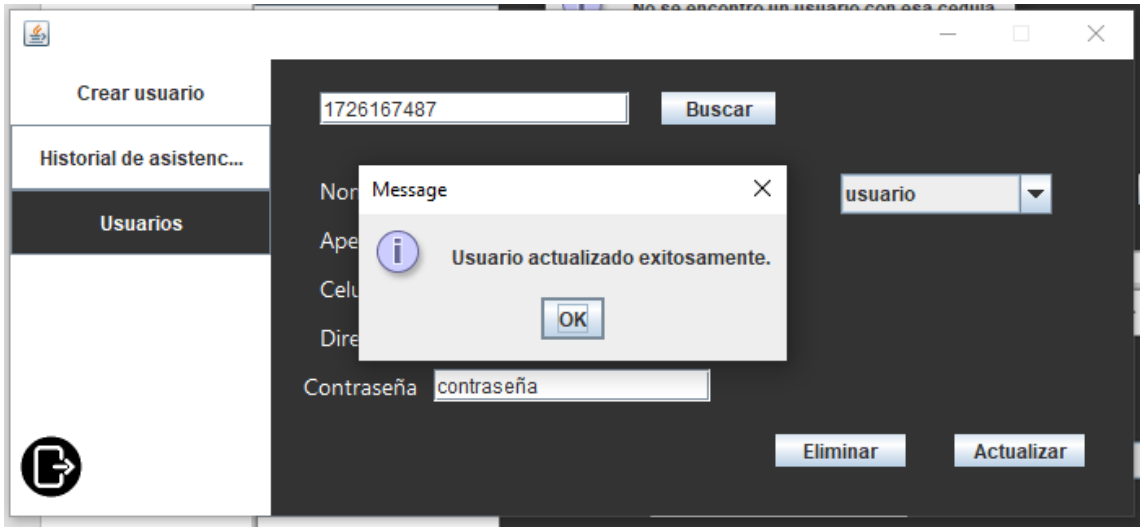
3.Particion de clases equivalentes

Actualizar Nivel de Cuenta			
Variable	**Clase de Equivalencia**	**Estado**	**Representante**
B_cedula	EC1: B_cedula==cedula	Válido	"1723456789"
	EC2: B_cedula!=cedula	No válido	"1111111111"
	EC1: B_cedula==usuario	No válido	"1714797303"
	EC2: B_cedula!=usuario	Válido	"1726167487"

4.Captura de la pantalla de la ejecución




Crear usuario
Historial de asistenc...
Usuarios
1231231
Buscar
Message
No se encontró un usuario con esa cédula.
OK
usuario
Contraseña
Eliminar
Actualizar


Crear usuario
Historial de asistenc...
Usuarios
1726167487
Buscar
Message
Usuario actualizado exitosamente.
OK
usuario
Contraseña
contraseña
Eliminar
Actualizar

1. Historial de asistencias

HISTORIA DE USUARIO	
Número REQ 005	**Usuario:** admin
Nombre Historia: **Historial de asistencias**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** medio
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **El administrador podrá tener acceso al historial de asistencia de los usuarios.** • **El administrador podrá filtrar por usuario y fecha** • **El administrador podrá exportar los datos a un archivo excel**	
Validación: • **Se validará que la cédula ingresada corresponda a un usuario existente.** • **Se verificará que el usuario tenga asistencias registradas en la fecha especificada** • **Se verificara que la tabla tenga datos antes de ser exportada como Excel**	

2.Codigo Fuente

```
    private void B_buscarActionPerformed(java.awt.event.ActionEvent evt) {

  String usuario_registro = H_buscar.getText().trim();

  String nombre_user = "";

  Date date = date_f.getDate();

  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

  String fechaSeleccionada = (date != null) ? sdf.format(date) : "";


try {

  // Obtener nombre del usuario si se ingresa una cédula específica

  if (!usuario_registro.isEmpty()) {

     String sql = "SELECT nombre FROM usuarios WHERE cedula = ?";

     PreparedStatement ps = cn.prepareStatement(sql);
```

```
        ps.setString(1, usuario_registro);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            nombre_user = rs.getString("nombre");
        } else {
            JOptionPane.showMessageDialog(null, "El usuario no existe");
        }
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error al obtener usuario: " + e.getMessage());
}

try {
    String sql;
    PreparedStatement ps;

    // Construir la consulta según los filtros
    if (usuario_registro.isEmpty() && fechaSeleccionada.isEmpty()) {
        // Caso 1: Todos los registros
        sql = "SELECT nombre, fecha, hora, hora_salida FROM asistencias";
        ps = cn.prepareStatement(sql);
    } else if (usuario_registro.isEmpty()) {
        // Caso 2: Todos los usuarios en una fecha específica
        sql = "SELECT nombre, fecha, hora, hora_salida FROM asistencias WHERE fecha = ?";
        ps = cn.prepareStatement(sql);
        ps.setString(1, fechaSeleccionada);
    } else if (fechaSeleccionada.isEmpty()) {
```

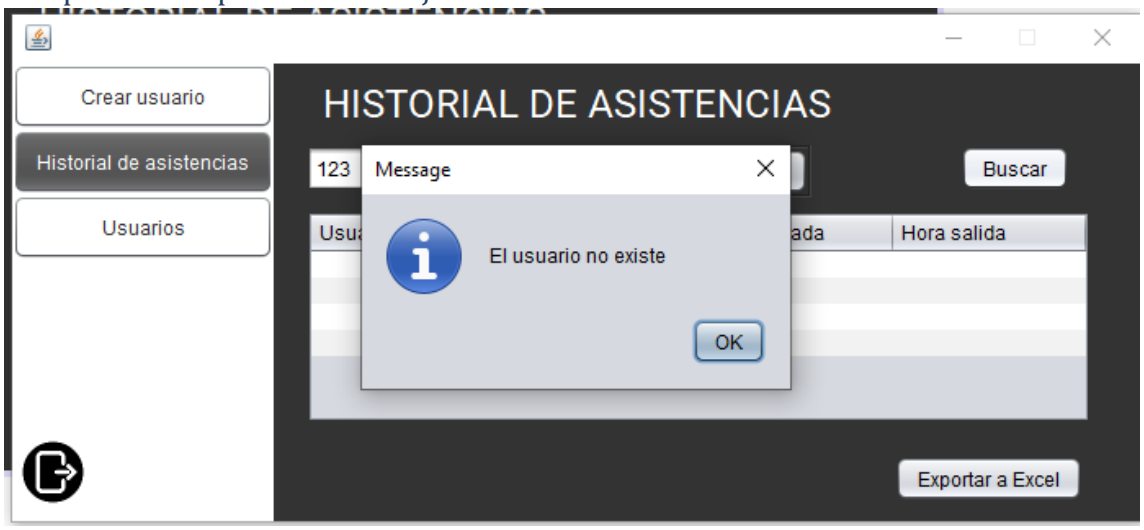
```
        // Caso 3: Un usuario específico en todas las fechas
        sql = "SELECT nombre, fecha, hora, hora_salida FROM asistencias WHERE usuario = ?";
        ps = cn.prepareStatement(sql);
        ps.setString(1, usuario_registro);
    } else {
        // Caso 4: Un usuario específico en una fecha específica
        sql = "SELECT nombre, fecha, hora, hora_salida FROM asistencias WHERE usuario = ? AND fecha = ?";
        ps = cn.prepareStatement(sql);
        ps.setString(1, usuario_registro);
        ps.setString(2, fechaSeleccionada);
    }

    ResultSet prs = ps.executeQuery();

    // Definir las columnas del JTable
    String[] columnNames = {"Usuario", "Fecha", "Hora Entrada", "Hora Salida"};
    DefaultTableModel model = new DefaultTableModel(columnNames, 0);

    // Agregar los registros al JTable
    while (prs.next()) {
        String usuario = prs.getString("nombre");
        String fecha = prs.getString("fecha");
        String hora = prs.getString("hora");
        String hora_salida = prs.getString("hora_salida");
        model.addRow(new Object[]{usuario, fecha, hora, hora_salida});
    }
```

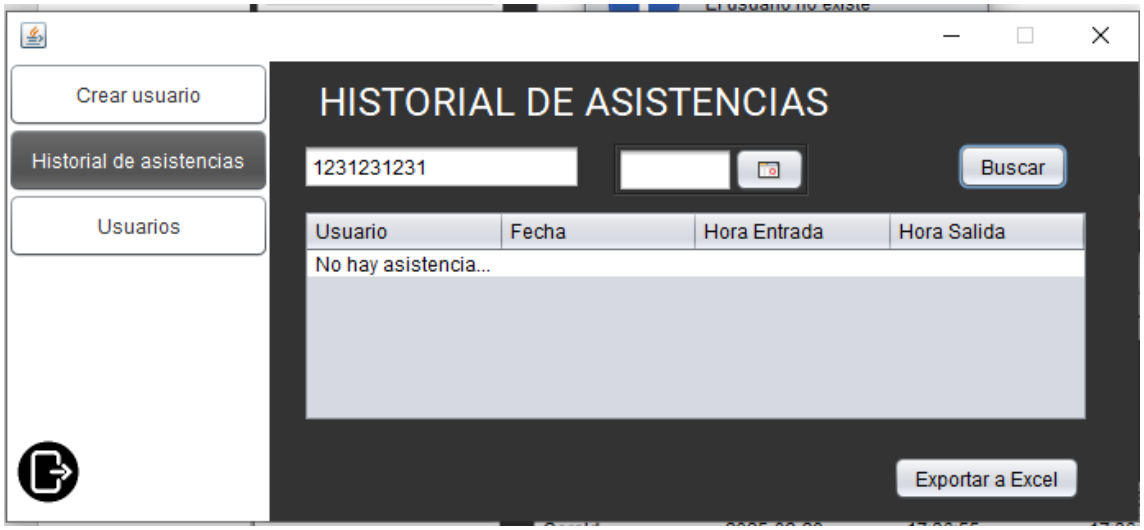
```
    // Si no hay registros
    if (model.getRowCount() == 0) {
        model.addRow(new Object[]{"No hay asistencias registradas.", "", "", ""});
    }

    // Establecer el modelo en el JTable
    Lista_asistencias.setModel(model);

} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Error al obtener el historial de asistencias: " +
e.getMessage());
}
    }
```

3.Particion de clases equivalentes

Variable	Clase de Equivalencia (Condición de Código)	Estado	Representante
usuario	if (usuario_registro.length() == 10 && usuario_registro.matches("\\d+"))	Válido	"1723456789"
	if (usuario_registro.isEmpty())	Válido	""
	if (usuario_registro.length() != 10)	No válido	"12345"
	if (!usuario_registro.matches("\\d+"))	No válido	"abc123xyz"
Date	if (Date("\\d{4}-\\d{2}-\\d{2}"))	Válido	"2024-05-15"
	if (Date.isEmpty())	Válido	""
	if (!Date ("\\d{4}-\\d{2}-\\d{2}"))	No válido	"15/05/2024"

4.Captura de la pantalla de la ejecución






Crear usuario
Historial de asistencias
Usuarios
HISTORIAL DE ASISTENCIAS
Buscar
Usuario
Fecha
Hora Entrada
Hora Salida
asd
2025-02-20
17:36:36
17:36:38
leandro
2025-02-20
17:36:55
17:36:56
leandro
2025-02-23
18:33:33
18:33:55
Exportar a Excel


Crear usuario
Historial de asistencias
Usuarios
HISTORIAL DE ASISTENCIAS
1726167487
!0/02/2025
Buscar
Usuario
Fecha
Hora Entrada
Hora Salida
Gerald
2025-02-20
17:36:55
17:36:56
Exportar a Excel


Crear usuario
Historial de asistencias
Usuarios
HISTORIAL DE ASISTENCIAS
!0/02/2025
Buscar
Usuario
Fecha
Hora Entrada
Hora Salida
asd
2025-02-20
17:36:36
17:36:38
leandro
2025-02-20
17:36:55
17:36:56
Exportar a Excel

1. Consulta de datos

HISTORIA DE USUARIO	
Número REQ 006	**Usuario:** admin
Nombre Historia: **Consulta de datos**	
Prioridad en negocio: **Media**	**Riesgo de desarrollo:** medio
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **El administrador ingresara la cedula del usuario que desea consultar** • **El administrador consultar los datos de las cuentas registradas.**	
Validación: • **Se mostrara los datos del usuario consultado.**	

2.Codigo Fuente

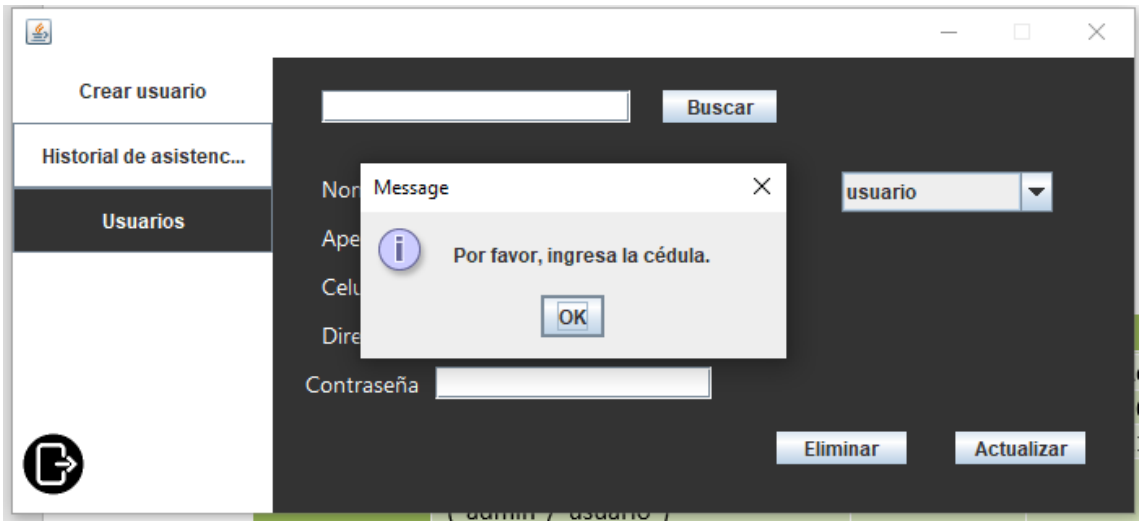
```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

      String cedulaIngresada = B_cedula.getText();

  if (cedulaIngresada.isEmpty()) {

     JOptionPane.showMessageDialog(null, "Por favor, ingresa la cédula.");

  }

  try {

     PreparedStatement ps = cn.prepareStatement("SELECT nombre, apellido, cedula,
celular, direccion,contraseña, nivel FROM usuarios WHERE cedula = ?");

     ps.setString(1, cedulaIngresada);

     ResultSet rs = ps.executeQuery();

     if (rs.next()) {

        B_nombre.setText(rs.getString("nombre"));

        B_apellido.setText(rs.getString("apellido"));

        B_celular.setText(rs.getString("celular"));

        B_direccion.setText(rs.getString("direccion"));
```

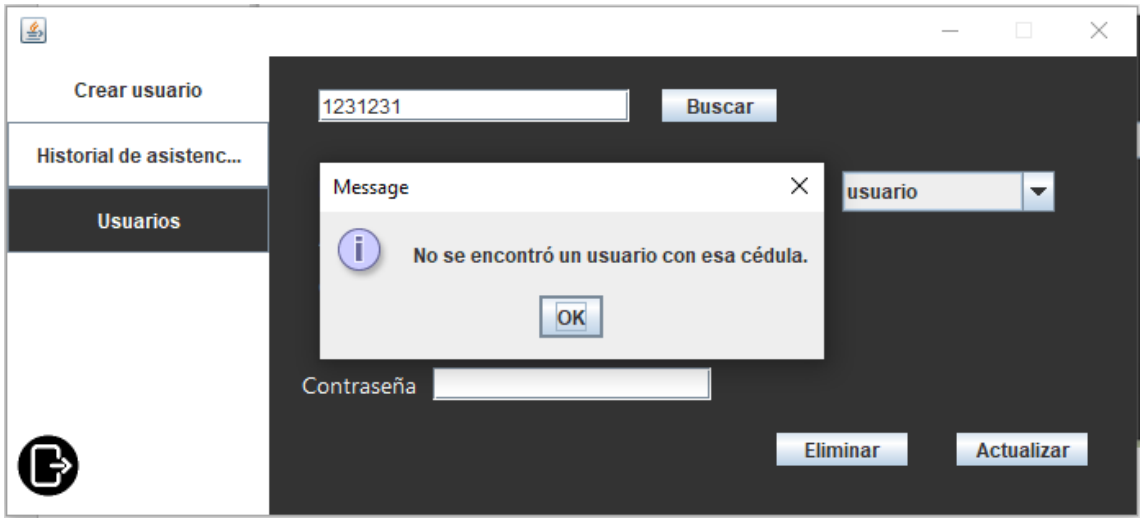
```
            B_contraseña.setText(rs.getString("contraseña"));
            String nivelUsuario = rs.getString("nivel");
            B_nivel.setSelectedItem(nivelUsuario);
            jButton5.setEnabled(true);
            jButton6.setEnabled(true);
        } else {
            JOptionPane.showMessageDialog(null, "No se encontró un usuario con esa cédula.");
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al buscar el usuario: " + e.getMessage());
    }
}
```

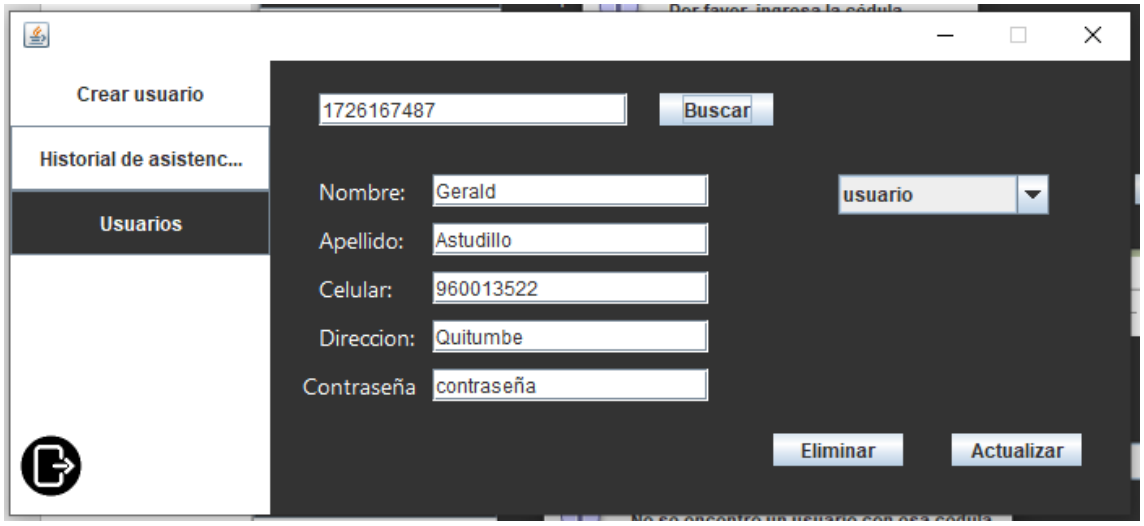
3.Particion de clases equivalentes

Consulta de datos			
Variable	**Clase de Equivalencia**	**Estado**	**Representante**
B_cedula	EC1: B_cedula==cedula	Válido	"1723456789"
	EC2: B_cedula!=cedula	No válido	"1111111111"
	EC1: B_cedula==usuario	No válido	"1714797303"
	EC2: B_cedula!=usuario	Válido	"1726167487"

4.Captura de la pantalla de la ejecución




Crear usuario
Historial de asistenc...
Usuarios
1231231
Buscar
Message
No se encontró un usuario con esa cédula.
OK
usuario
Contraseña
Eliminar
Actualizar


Crear usuario
Historial de asistenc...
Usuarios
1726167487
Buscar
Nombre:
Gerald
Apellido:
Astudillo
Celular:
960013522
Direccion:
Quitumbe
Contraseña
contraseña
usuario
Eliminar
Actualizar

1. Eliminar usuarios

HISTORIA DE USUARIO	
Número REQ 007	**Usuario:** admin
Nombre Historia: **Eliminar usuarios**	
Prioridad en negocio: **Alta**	**Riesgo de desarrollo:** medio
Iteración Asignada: **1**	
Programador responsable: **Gerald Astudillo**	
Descripción: • **El administrador podrá eliminar las cuentas de los trabajadores.** • **Se pedirá confirmación al administrador para eliminar la cuenta.**	
Validación: • **Se validará que la cédula ingresada corresponda a un usuario existente.** • **Se mostrara una mensaje de que la cuenta fue eliminada exitosamente**	

2.Codigo Fuente

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

   String cedulaIngresada = B_cedula.getText();

   String[] options = { "si", "no"};

   if (cedulaIngresada.isEmpty()) {

   JOptionPane.showMessageDialog(null, "Por favor, ingresa la cédula para eliminar el
usuario.");}

   else {

   var selection = JOptionPane.showOptionDialog(null, "¿Esta seguro que desea eliminar
este usuario?", "Mensaje!", 0, 3, null, options, options[0]);

   if (selection == 0) {

     try {

        PreparedStatement verificar = cn.prepareStatement("SELECT nombre, apellido,
cedula, celular, direccion, contraseña, nivel FROM usuarios WHERE cedula = ?");
```

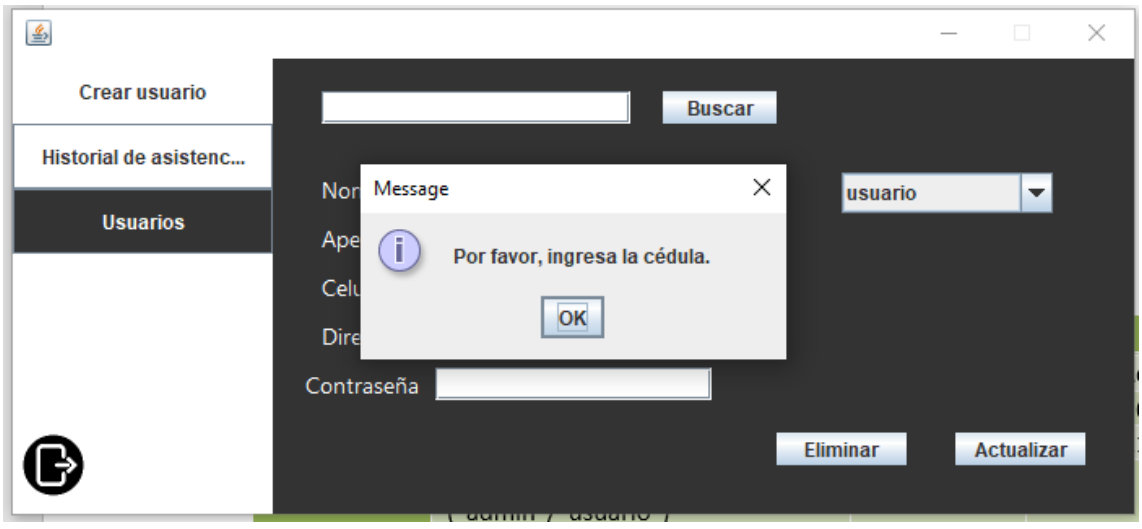
```
verificar.setString(1, cedulaIngresada);
ResultSet verificarrs = verificar.executeQuery();

if (verificarrs.next()) {
String usuario_consultado = verificarrs.getString("cedula");

// Verificar si se intenta editar al usuario actual
if (usuario_consultado.equals(usuario)) {
   JOptionPane.showMessageDialog(null, "No se puede eliminar al usuario actual.");
}
else{
String sql = "DELETE FROM usuarios WHERE cedula = ?";
PreparedStatement ps = cn.prepareStatement(sql);
ps.setString(1, cedulaIngresada);

int rowsAffected = ps.executeUpdate();

if (rowsAffected > 0) {
   JOptionPane.showMessageDialog(null, "Usuario eliminado exitosamente.");
   B_nombre.setText("");
   B_apellido.setText("");
   B_cedula.setText("");
   B_celular.setText("");
   B_contraseña.setText("");
   B_direccion.setText("");
   jButton5.setEnabled(false);
   jButton6.setEnabled(false);
```

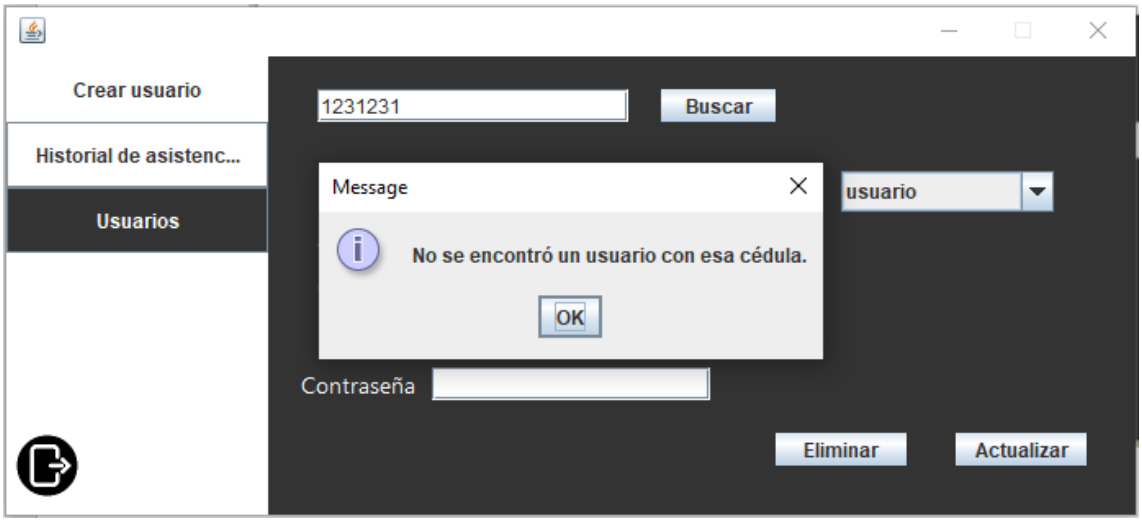
```
        }}}
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error al eliminar el usuario: " +
e.getMessage());
    }
  }
  if (selection == 1) {
  }
  }
  }
```

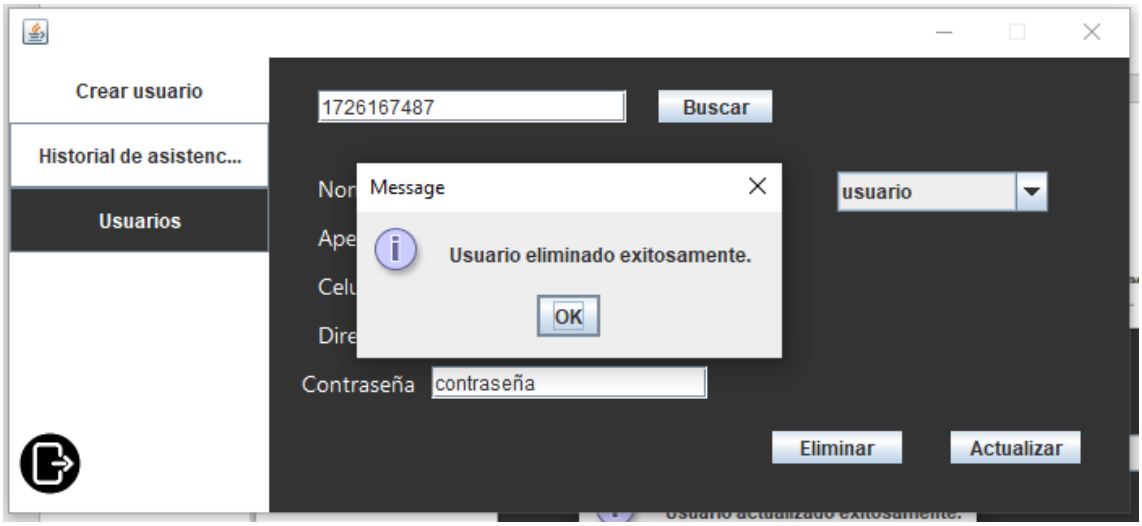
3.Particion de clases equivalentes

Eliminar Cuenta			
Variable	**Clase de Equivalencia**	**Estado**	**Representante**
B_cedula	EC1: B_cedula==cedula	Válido	"1723456789"
	EC2: B_cedula!=cedula	No válido	"1111111111"
	EC1: B_cedula==usuario	No válido	"1714797303"
	EC2: B_cedula!=usuario	Válido	"1726167487"

4.Captura de la pantalla de la ejecución




Crear usuario
Historial de asistenc...
Usuarios
1231231
Buscar
Message
No se encontró un usuario con esa cédula.
OK
usuario
Contraseña
Eliminar
Actualizar


Crear usuario
Historial de asistenc...
Usuarios
1726167487
Buscar
Message
Usuario eliminado exitosamente.
OK
usuario
Contraseña
contraseña
Eliminar
Actualizar