# Prueba de Caja Negra

***"Panadería Asistencias"***


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


**V4.1**

**Fecha: 2025-01-09**

# Índice

## Historial de Revisión

| Fecha | Versión | Descripción | Autores |
|---|---|---|---|
| 19/Noviembre/2024 | 1 | Versión inicial | Gerald Astudillo<br>Henry Chalcualan<br>Lenin Erazo<br>Henry Suin |
| 11/Diciembre/2024 | 2 | Corrección de requisitos funcionales en caja negra | Gerald Astudillo<br>Henry Chalcualan<br>Lenin Erazo<br>Henry Suin |
| 18/Diciembre/2024 | 3 | Se añadió una descripción de cada requisito | Gerald Astudillo<br>Henry Chalcualan<br>Lenin Erazo<br>Henry Suin |
| 24/Enero/2025 | 4 | Se finalizo el documento con todas las pruebas de caja negra | Gerald Astudillo<br>Henry Chalcualan<br>Lenin Erazo<br>Henry Suin |

## Partición de clases equivalentes

## 1. Inicio de sesión

| HISOTIRA 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:<br>• **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:<br>• **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");
```

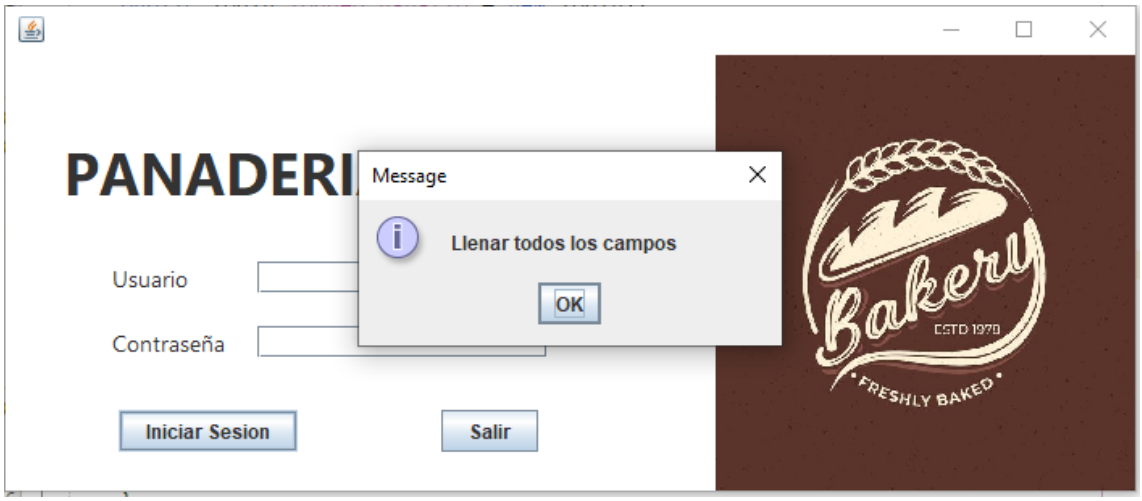
```
            if(nivel.equalsIgnoreCase("admin")){
               admin fa= new admin();
               fa.setVisible(true);
               this.setVisible(false);
            }
            else if(nivel.equalsIgnoreCase("usuario")){
               usuario fu= new usuario();
               fu.setVisible(true);
               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


PANADERIA RUBIO
Usuario
Contraseña
Iniciar Sesion
Salir


PANADERI
Message
Llenar todos los campos
OK
Usuario
Contraseña
Iniciar Sesion
Salir
Bakery
CSTD 1970
FRESHLY BAKED


PANADERI
Message
Usuario o contraseña incorrectos
OK
Usuario
Contraseña
Iniciar Sesion
Salir
Bakery
CSTD 1970
FRESHLY BAKED


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

| HISOTIRA 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:<br>• **El administrador podrá registrar nuevos usuarios en el sistema.**<br>• **Se debe validar que todos los campos sean ingresados correctamente.**<br>• **No se permitirá registrar un usuario con una cédula ya existente.** | |
| Validación:<br>• **Un usuario se registra correctamente si todos los datos son válidos.**<br>• **Se mostrará un mensaje de error si falta información o si la cédula ya está registrada.** | |

## 2.Codigo Fuente

### 2.1.Codigo java

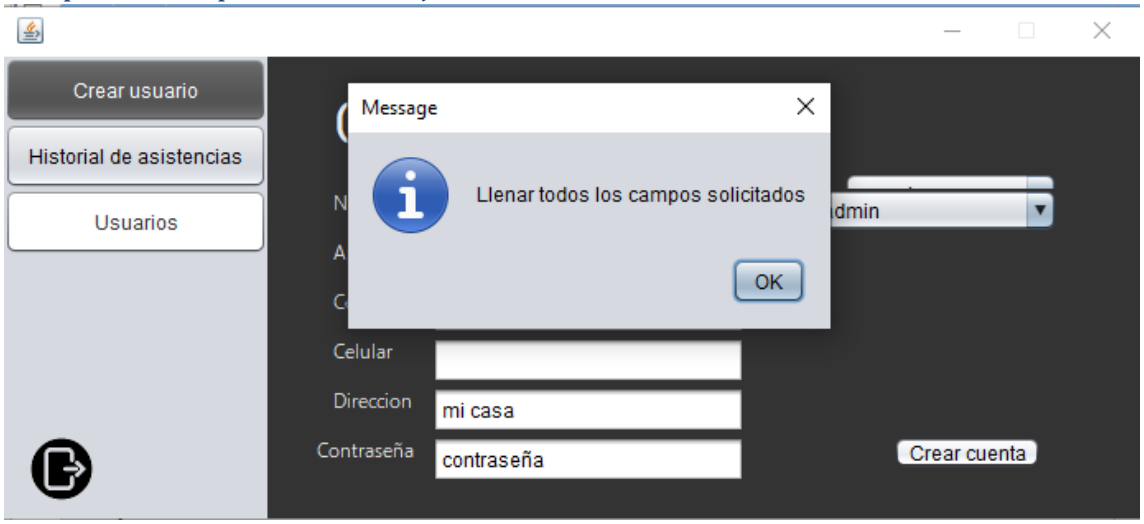
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String nombre,apellido,nivel,direc,contraseña,ceduV;
    int cedu,celular;
    nombre = nom.getText();
    apellido=apell.getText();
    nivel=level.getSelectedItem().toString();
    direc=direccion.getText();
    contraseña=pass.getText();
    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.");
     }
```

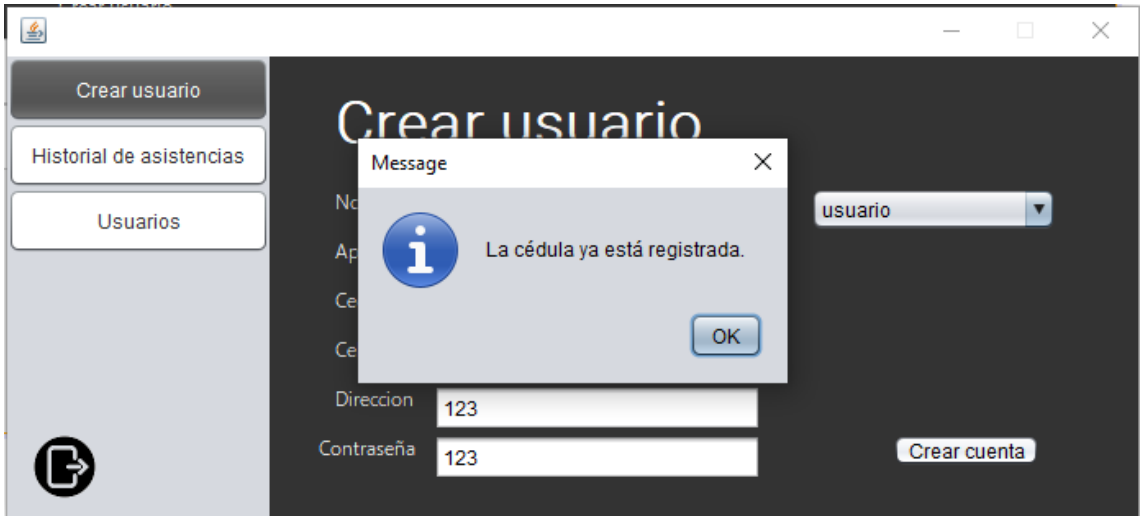
```
    else{
     try{
        cedu=Integer.parseInt(cedula.getText());
        celular=Integer.parseInt(telf.getText());
        PreparedStatement ps = cn.prepareStatement("INSERT INTO `usuarios`(`nombre`,
`apellido`, `cedula`, `celular`, `direccion`, `contraseña`, `nivel`) VALUES (?, ?, ?, ?, ?, ?, ?)");
        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){}
    }
  }
```

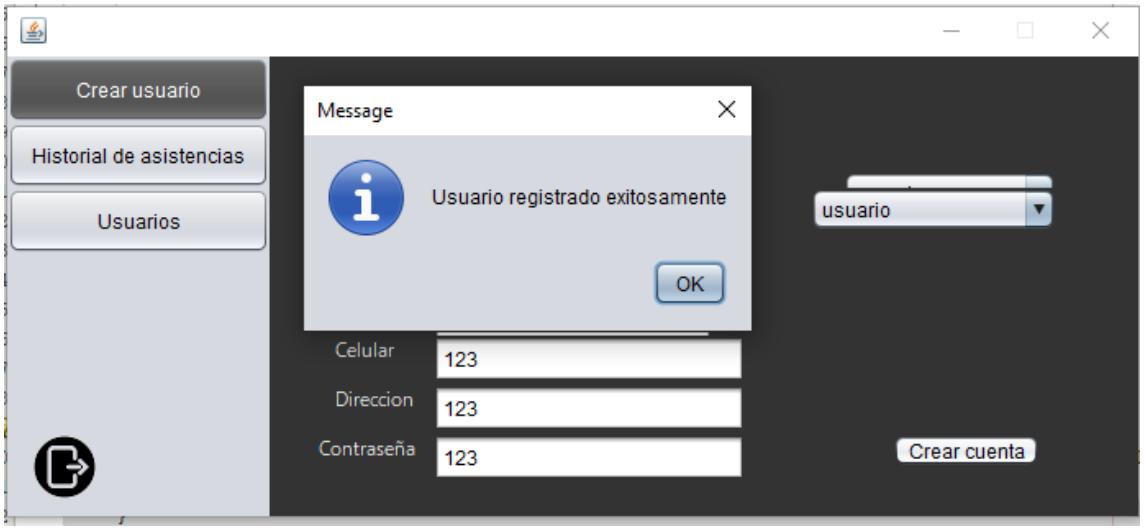
## 3.Particion de clases equivalentes

| Variable | Clase de Equivalencia | Estado | Representante |
|---|---|---|---|
| **Nombre** | EC1: nombre != vacío | Válido | "Juan" |
| | EC2: nombre = vacío | No válido | "" |
| **Apellido** | EC1: apellido != vacío | Válido | "Pérez" |
| | EC2: apellido = vacío | No válido | "" |
| **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 | "" |
| **Dirección** | EC1: dirección != vacío | Válido | "Av. Siempre Viva 742" |
| | EC2: dirección = vacío | No válido | "" |
| **Nivel** | EC1: nivel válido ("admin"/"usuario") | Válido | "usuario" |
| | EC2: nivel inválido | No válido | "admin" |

## 4.Captura de la pantalla de la ejecución



## 1. Registro de asistencias

| HISOTIRA 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:<br>• **Un usuario o subadministrador podrá registrar su asistencia al sistema.**<br>• **Se registrará la hora de entrada y la hora de salida.**<br>• **No se permitirá registrar una salida sin haber registrado una entrada previamente.** | |
| Validación:<br>• **Se validará que el usuario exista en la base de datos.**<br>• **Se verificará que la fecha y hora se almacenen correctamente.**<br>• **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) {

    try{

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

        ResultSet rs=ps.executeQuery();

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

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

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        String fechaHora = now.format(formatter);


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

        PreparedStatement psAsistencia = cn.prepareStatement(sqlAsistencia);


        psAsistencia.setString(1, usuario);

        psAsistencia.setString(2, fechaHora.split(" ")[0]);

        psAsistencia.setString(3, fechaHora.split(" ")[1]);

        psAsistencia.setString(4, fechaHora.split(" ")[1]);

        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
```

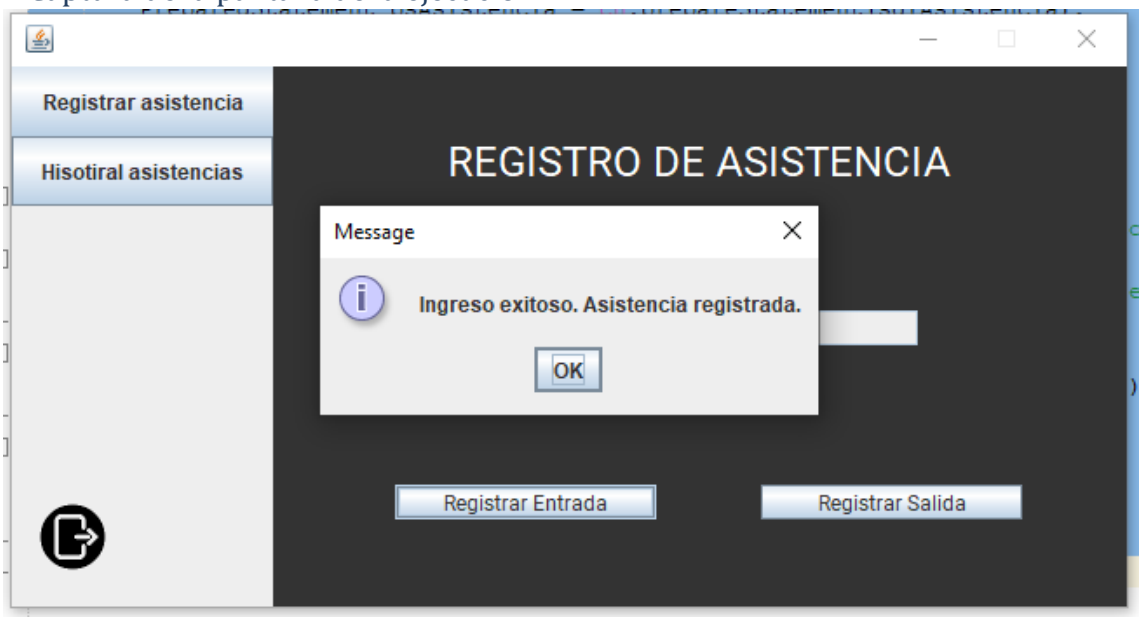
```
        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter dateFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern("HH:mm:ss");

        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());
```

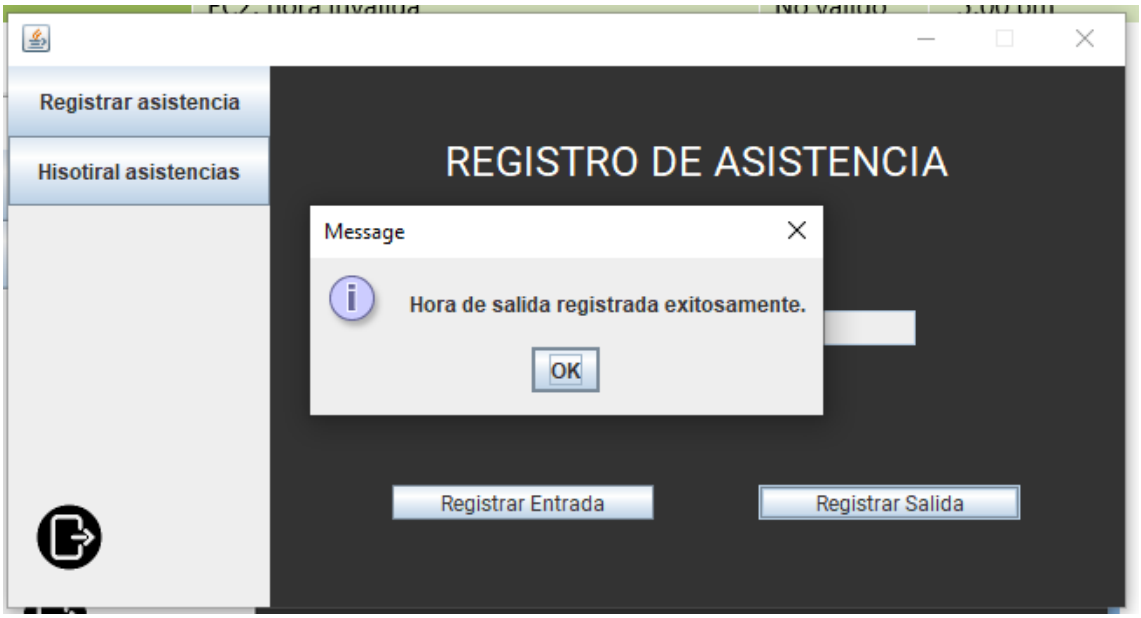
```
}

  }
```

## 3.Particion de clases equivalentes

| **Registro de Entrada** | | | |
|---|---|---|---|
| **Variable** | **Clase de Equivalencia** | **Estado** | **Representante** |
| **Usuario** | EC1: usuario existe y tiene nivel válido | Válido | "1723456789" |
| | EC2: usuario no registrado | No válido | "1111111111" |
| | EC3: usuario con contraseña incorrecta | No válido | "1723456789" con "claveerrada" |
| **Fecha** | EC1: fecha actual válida | Válido | "2025-02-10" |
| | EC2: fecha en formato incorrecto | No válido | "10-02-2025" |
| **Hora** | EC1: hora en formato HH:mm:ss | Válido | "08:30:00" |
| | EC2: hora con formato incorrecto | No válido | "8:30 am" |
| **Registro de Salida** | | | |
| **Variable** | **Clase de Equivalencia** | **Estado** | **Representante** |
| **Usuario** | EC1: usuario registrado con entrada previa | Válido | "1723456789" |
| | EC2: usuario sin entrada previa | No válido | "1111111111" |
| **Fecha** | EC1: fecha coincide con la entrada | Válido | "2025-02-10" |
| | EC2: fecha sin entrada registrada | No válido | "2025-02-11" |
| **Hora salida** | EC1: hora válida | Válido | "17:00:00" |
| | EC2: hora inválida | No válido | "5:00 pm" |

## 4.Captura de la pantalla de la ejecución






## 1. Administrar usuarios

| HISOTIRA DE USUARIO | |
|---|---|
| Número REQ 004 | **Usuario:** admin |
| Nombre Historia: **Administrar usuarios** | |
| Prioridad en negocio: **Alta** | **Riesgo de desarrollo:** medio |
| Iteración Asignada: **1** | |
| Programador responsable: **Gerald Astudillo** | |
| Descripción:<br>• **El administrador podrá modificar o eliminar cuentas de usuarios.**<br>• **Solo los administradores podrán cambiar los datos de los usuarios.**<br>• **Se pedirá confirmación antes de eliminar una cuenta.** | |
| Validación:<br>• **Se validará que la cédula ingresada corresponda a un usuario existente.**<br>• **Se confirmará la eliminación de la cuenta antes de proceder.** | |

## 2.Codigo Fuente

### 2.1.Codigo java Actualizar cuenta

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

    String nivel;

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

    String cedulaIngresada = B_cedula.getText();

    if (cedulaIngresada.isEmpty()) {

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

  }

    try{

      PreparedStatement ps = cn.prepareStatement("UPDATE usuarios SET nivel = ?
WHERE cedula = ?");

      ps.setString(1, nivel);

      ps.setString(2, cedulaIngresada);

      int rowsUpdated = ps.executeUpdate();

      if (rowsUpdated > 0) {

      JOptionPane.showMessageDialog(null, "Nivel actualizado exitosamente.");

    }

    }catch(Exception e){}

  }
```

### 2.1.Codigo java eliminar cuenta

```
  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 {
```

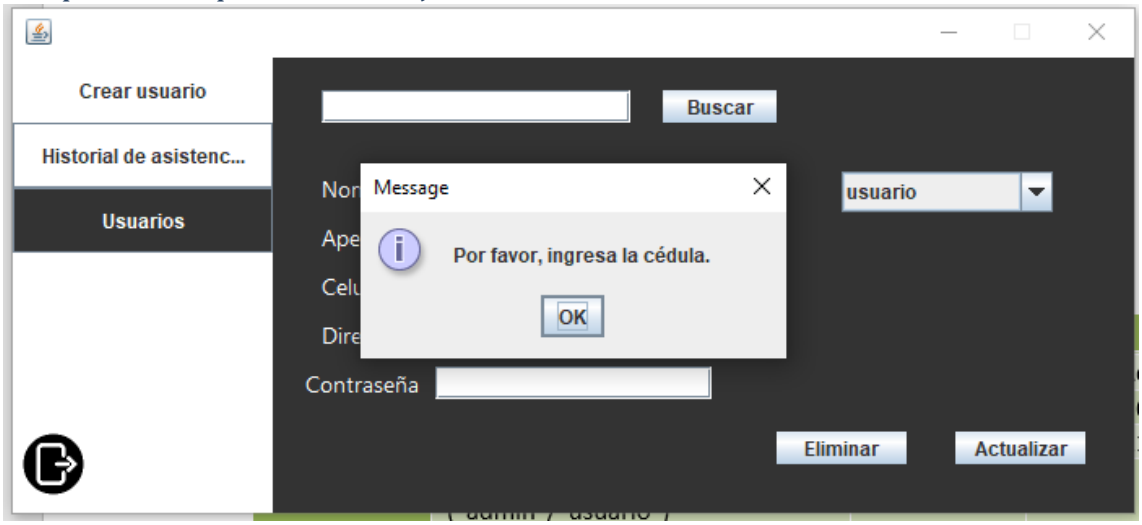
```
   var selection = JOptionPane.showOptionDialog(null, "¿Esta seguro que desea eliminar
este usuario?", "Mensaje!", 0, 3, null, options, options[0]);

   if (selection == 0) {

    try {

        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_direccion.setText("");

           jButton5.setEnabled(false);

           jButton6.setEnabled(false);

        }

     } catch (Exception e) {

        JOptionPane.showMessageDialog(null, "Error al eliminar el usuario: " +
e.getMessage());

     }

   }

   if (selection == 1) { }}}
```

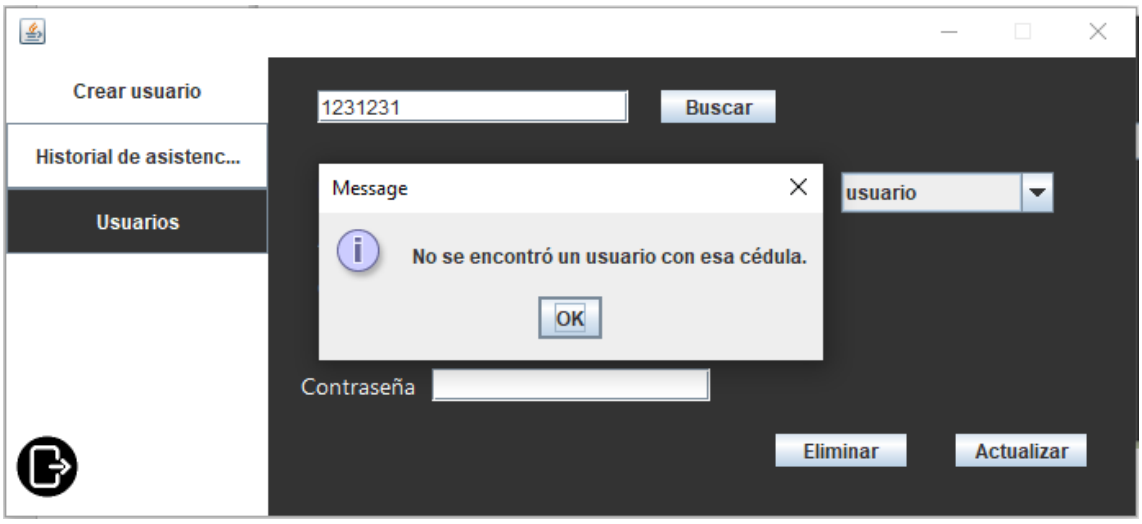
## 3.Particion de clases equivalentes

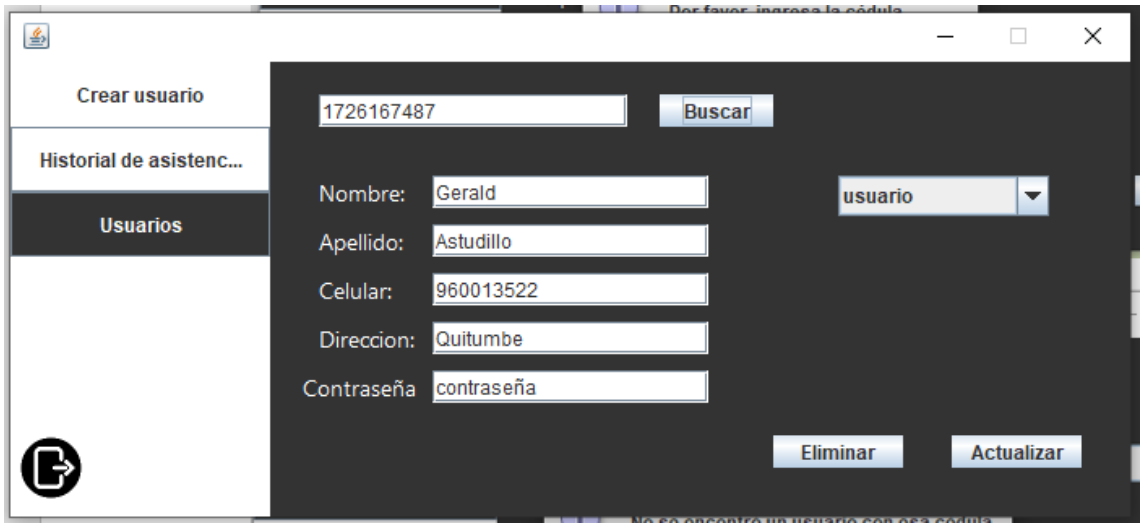
| **Actualizar Nivel de Cuenta** | | | |
|---|---|---|---|
| **Variable** | **Clase de Equivalencia** | **Estado** | **Representante** |

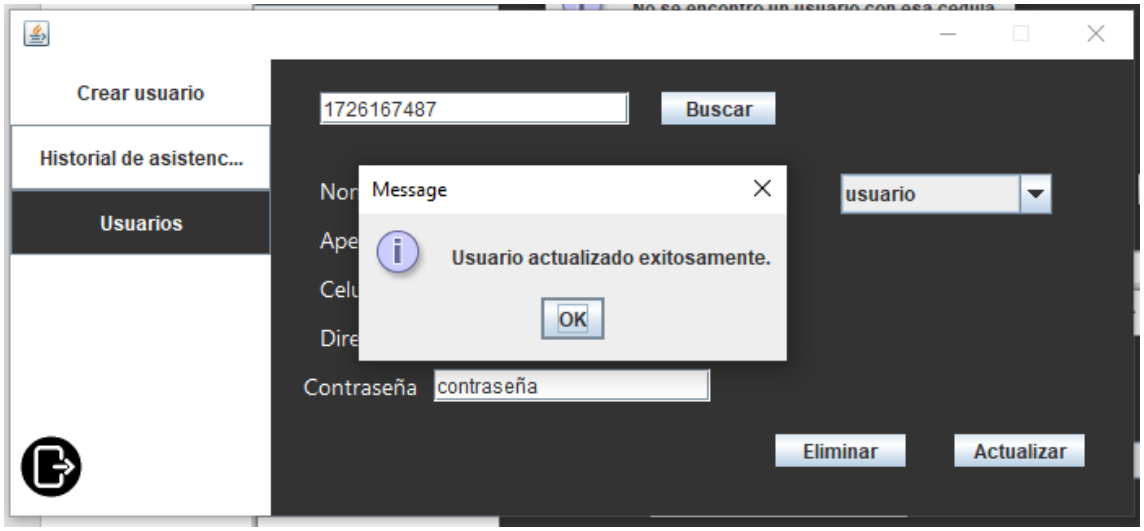
| | | | |
|---|---|---|---|
| **Cédula** | EC1: cédula existe en la BD | Válido | "1723456789" |
| | EC2: cédula no registrada | No válido | "1111111111" |
| **Nivel** | EC1: nivel permitido ("admin"/"usuario") | Válido | "admin" |
| | EC2: nivel no permitido | No válido | "moderador" |
| **Eliminar Cuenta** | | | |
| **Variable** | **Clase de Equivalencia** | **Estado** | **Representante** |
| **Cédula** | EC1: cédula existe en la BD | Válido | "1723456789" |
| | EC2: cédula no registrada | No válido | "1111111111" |
| **Confirmación** | EC1: usuario confirma eliminación | Válido | "Sí" |
| | EC2: usuario cancela eliminación | No válido | "No" |

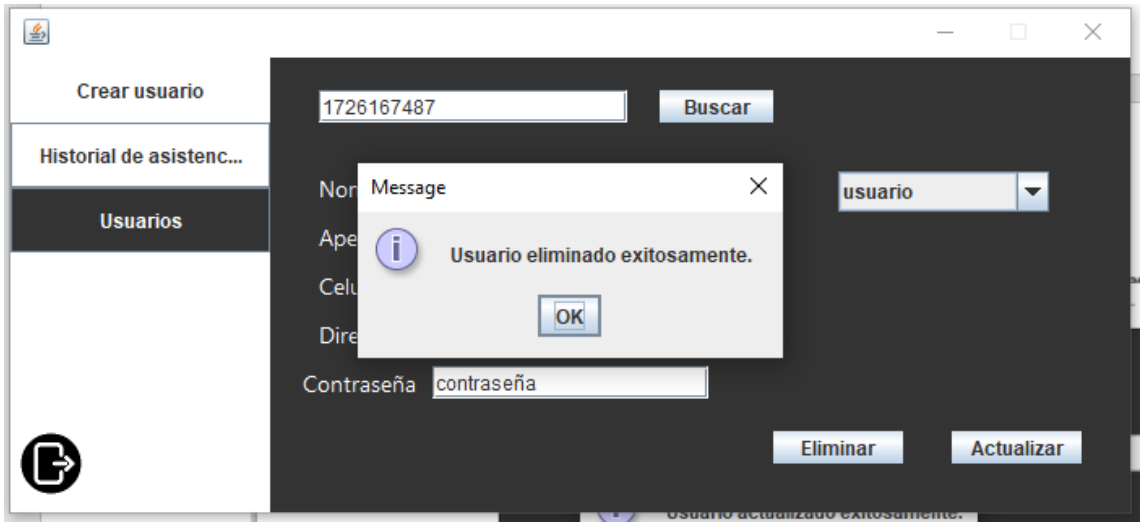
## 4.Captura de la pantalla de la ejecución




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


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


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

## 1. Historial de asistencias

| HISOTIRA DE USUARIO | |
|---|---|
| Número REQ 004 | **Usuario:** admin |
| Nombre Historia: **Historial de asistencias** | |
| Prioridad en negocio: **Alta** | **Riesgo de desarrollo:** medio |
| Iteración Asignada: **1** | |
| Programador responsable: **Gerald Astudillo** | |
| Descripción:<br>• **El administrador podrá tener acceso al historial de asistencia de los usuarios.**<br>• **El administrador podrá filtrar por usuario y fecha**<br>• **El administrador podrá exportar los datos a un archivo excel** | |
| Validación:<br>• **Se validará que la cédula ingresada corresponda a un usuario existente.**<br>• **Se verificará que el usuario tenga asistencias registradas en la fecha especificada**<br>• **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()) {
```

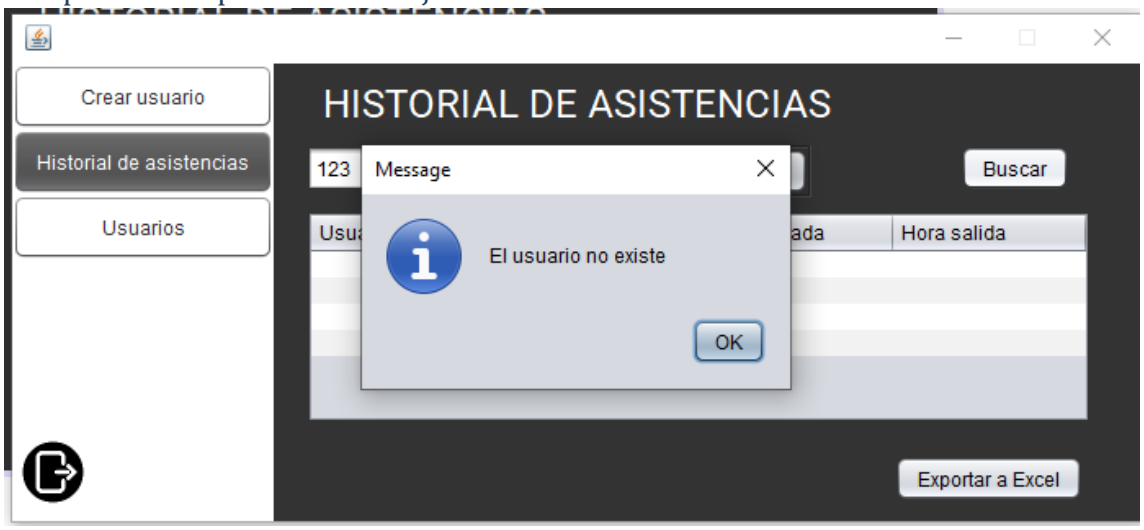
```
        // Caso 3: Un usuario específico en todas las fechas
        sql = "SELECT nombre, fecha, hora, hora_salida FROM asistencias WHERE usuario = ?";
        ps = cn.prepareStatement(sql);
        ps.setString(1, usuario_registro);
    } else {
        // Caso 4: Un usuario específico en una fecha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

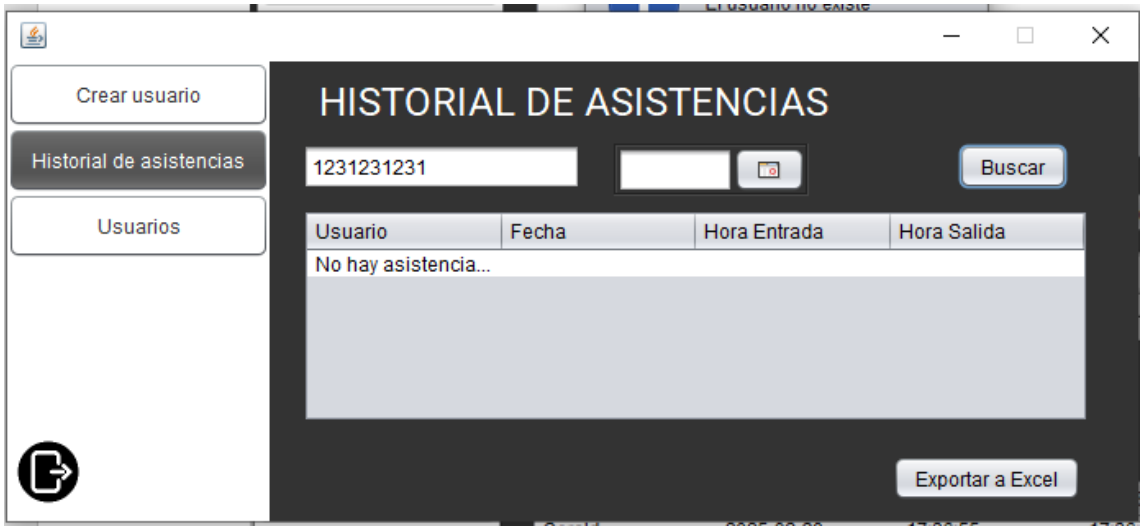
| Variable | Clase de Equivalencia (Condición de Código) | Estado | Representante |
|---|---|---|---|
| **usuario** | if (usuario_registro.length() == 10 && usuario_registro.matches("\\d+")) | Válido | "1723456789" |
| | if (usuario_registro.isEmpty()) | Válido | "" |
| | if (usuario_registro.length() != 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" |

## 3.Particion de clases equivalentes

## 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
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
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