Prueba de Caja Blanca

"Panadería Asistencias"

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

V1.0

Fecha: 2025-01-09

INDICE

Historial de Revisión

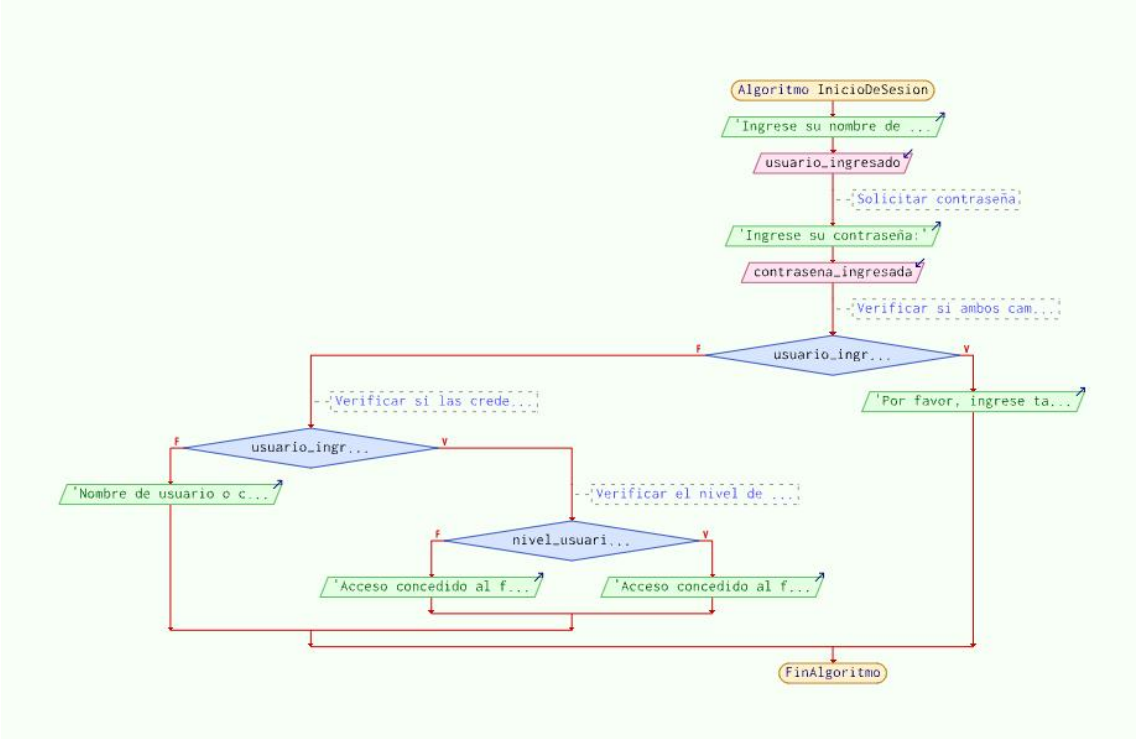
Fecha	Versión	Descripción	Autores
19/Noviembre/2024	1	Versión inicial	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
11/Diciembre/2024	2	Correcciones de los grafos de flujo	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin
18/Diciembre/2024	3	Se finaliza las pruebas de caja blanca de los requisitos	Gerald Astudillo Henry Chalcualan Lenin Erazo Henry Suin

Prueba caja blanca de Inicio de Sesión

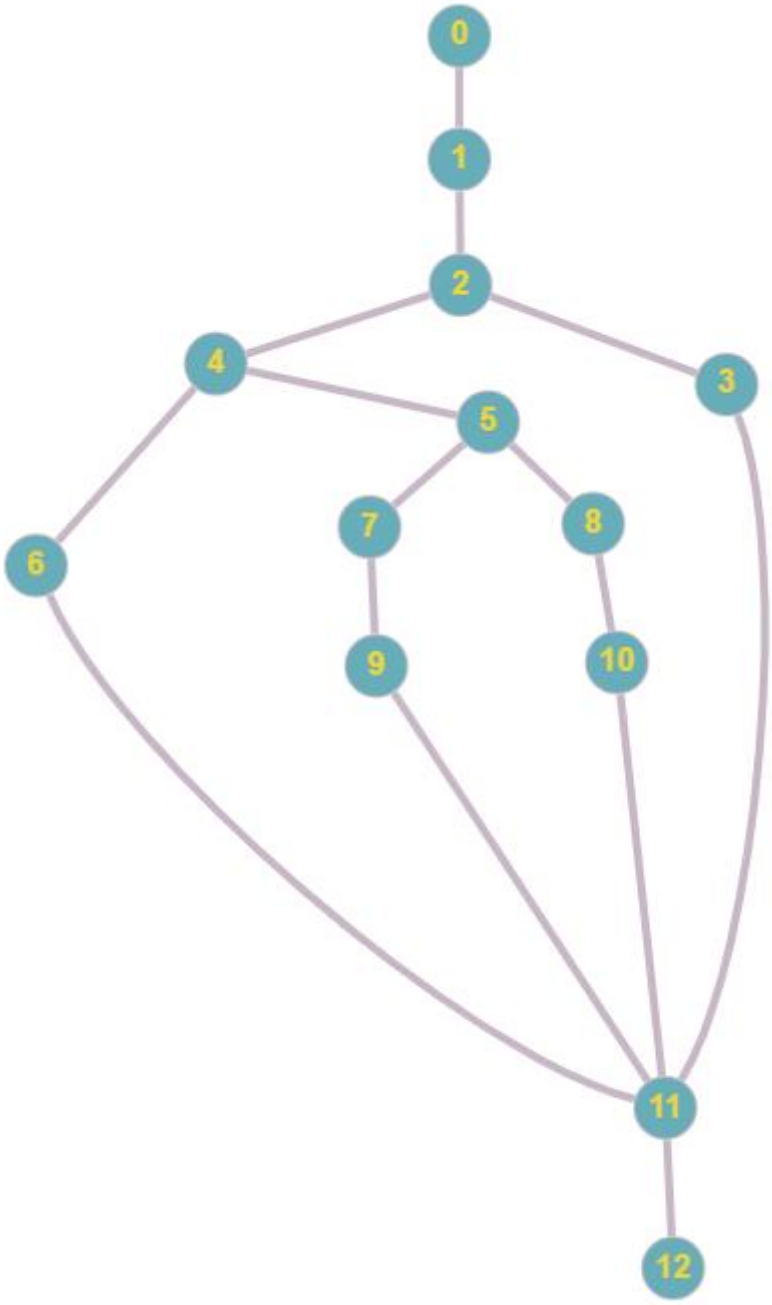
1. CÓDIGO FUENTE

```
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);
          }}
   }

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
   }
```

2. DFD


Algoritmo InicioDeSesion
'Ingrese su nombre de ...
usuario_ingresado
Solicitar contraseña
'Ingrese su contraseña: '
contrasena_ingresada
Verificar si ambos cam...
usuario_ingr...
F
V
'Por favor, ingrese ta...
Verificar si las crede...
usuario_ingr...
F
V
'Nombre de usuario o c...
Verificar el nivel de ...
nivel_usuari...
F
V
'Acceso concedido al f...
'Acceso concedido al f...
FinAlgoritmo

3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1 → 2 → 3 → 11 → Fin
R2: 1 → 2 → 4 → 6 → 11 → Fin
R3: 1 → 2 → 4 → 5 → 7 → 9 → 11 → Fin
R4: 1 → 2 → 4 → 5 → 8 → 10 → 11 → Fin

COMPLEJIDAD CICLOMÁTICA

Fórmula 1: V(G)=P+1
P=3 (nodos de decisión).
V(G)=3+1=4
Fórmula 2: V(G)=A−N+2

A=13 (aristas).
N=11(nodos).
V(G)=13−11+2=4

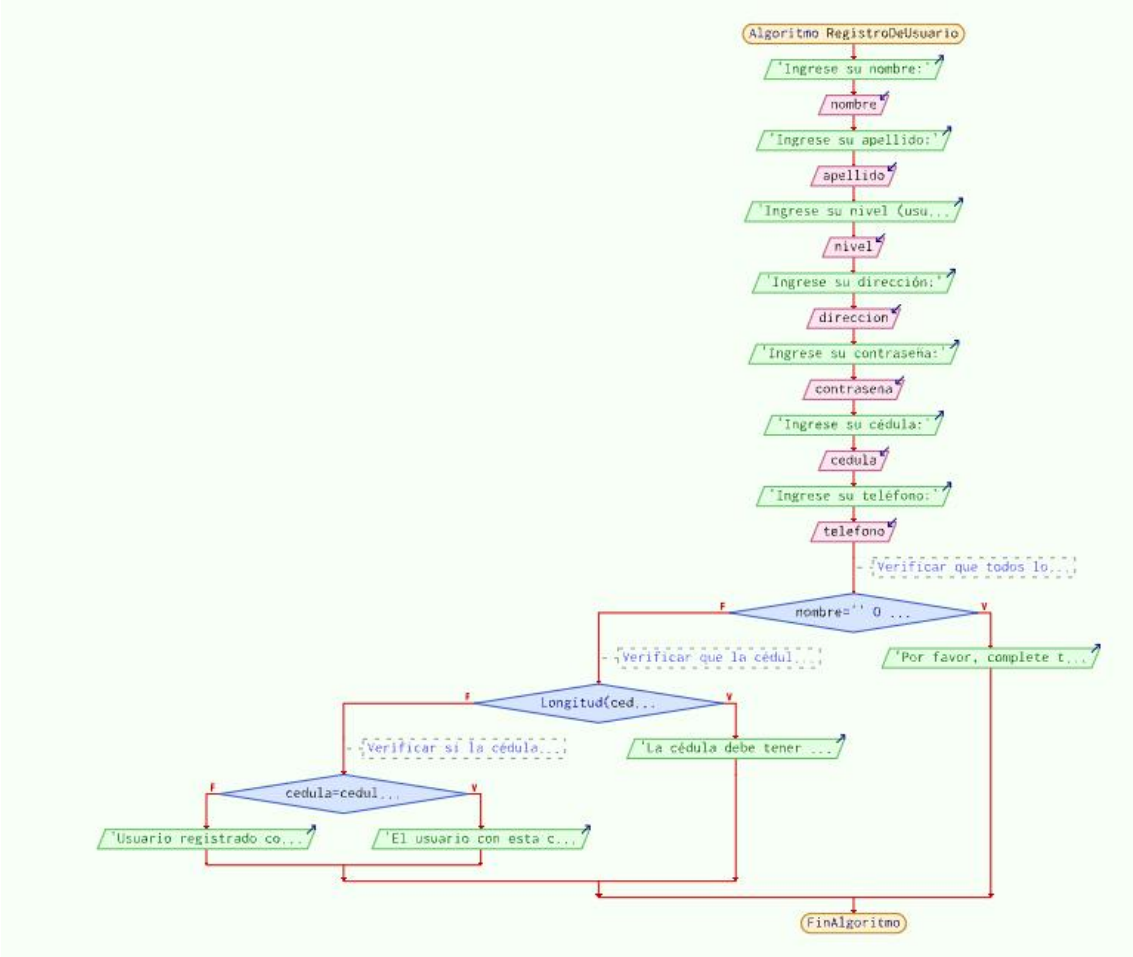
Prueba caja blanca de Registro de Usuarios

1. CÓDIGO FUENTE

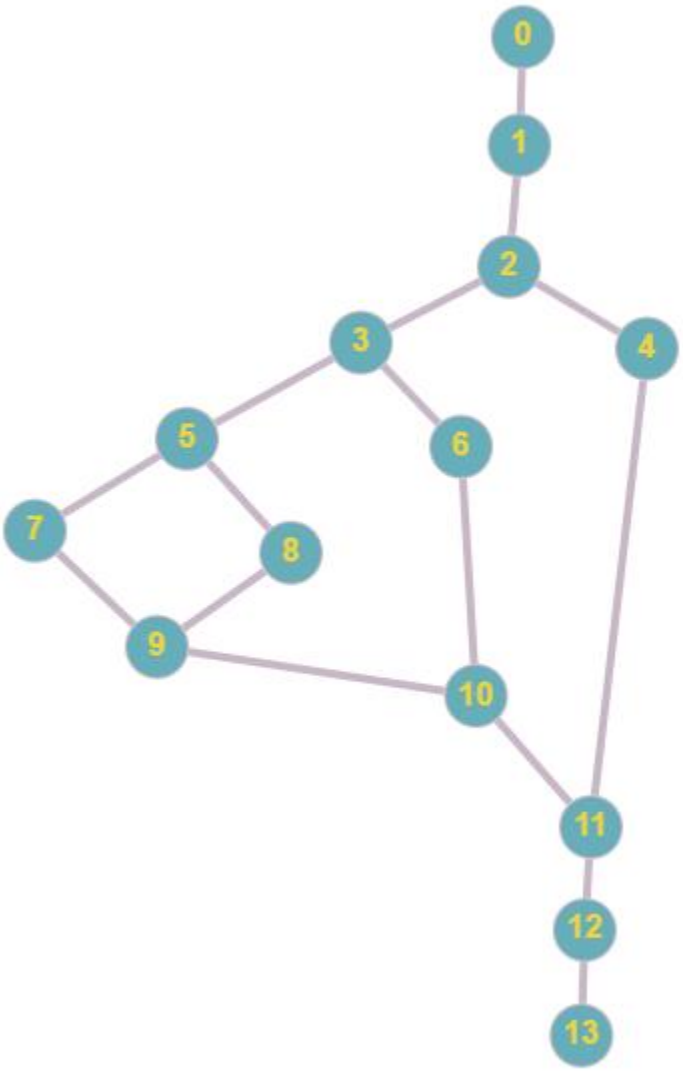
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      String nombre,apellido,nivel,direc,contraseña,ceduV;
      int cedu,celular;
      nombre = nom.getText();
      apellido=apell.getText();
      nivel=level.getSelectedItem().toString();
      direc=jTextField6.getText();
      contraseña=jTextField1.getText();
      ceduV = jTextField4.getText();
if (nombre.isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar el nombre.");
} else if (apellido.isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar el apellido.");
} else if (ceduV.isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar la cédula.");
} else if (jTextField5.getText().isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar el teléfono.");
} else if (direc.isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar la dirección.");
} else if (contraseña.isEmpty()) {
   JOptionPane.showMessageDialog(null, "Falta ingresar la contraseña.");
}
      else if (ceduV.length() != 10) {
       JOptionPane.showMessageDialog(null, "La cédula debe tener exactamente 10
dígitos.");
       }
      else{
       try{
          cedu=Integer.parseInt(jTextField4.getText());
          celular=Integer.parseInt(jTextField5.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){}
    }
}
```

2. DFD



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1 → 2 → 4 → 11 → 12 → Fin
R2: 1 → 2 → 3 → 6 → 10 → 11 → 12 → Fin
R3: 1 → 2 → 3 → 5 → 8 → 9 → 10 → 11 → 12 → Fin
R4: 1 → 2 → 3 → 5 → 7 → 9 → 10 → 11 → 12 → Fin

COMPLEJIDAD CICLOMÁTICA

Fórmula 1: V(G)=P+1
P=3 (nodos de decisión).
V(G)=3+1=4
Fórmula 2: V(G)=A−N+2
A=14 (aristas).
N=12(nodos).
V(G)=14−12+2=4

Prueba caja blanca de Registrar Asistencia

1. **CÓDIGO FUENTE**

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();
```

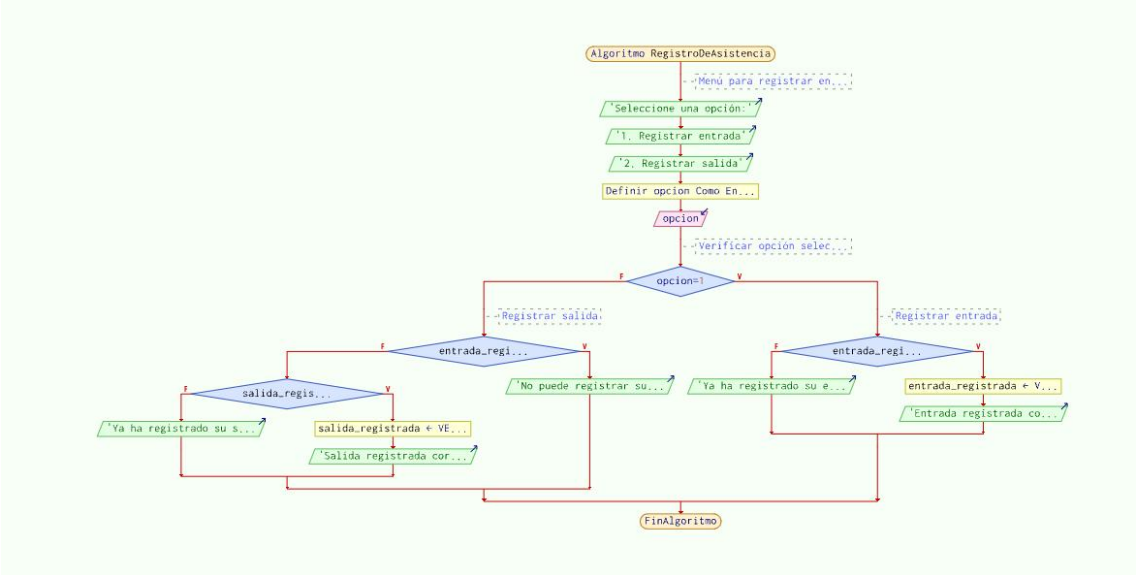
```
            if (rowsInserted > 0) {
                JOptionPane.showMessageDialog(null, "Ingreso exitoso. Asistencia registrada.");
            } else {
                JOptionPane.showMessageDialog(null, "No se pudo registrar la asistencia.");
            }
        }
    }
            }catch(Exception e){}
  }
```

Salida:

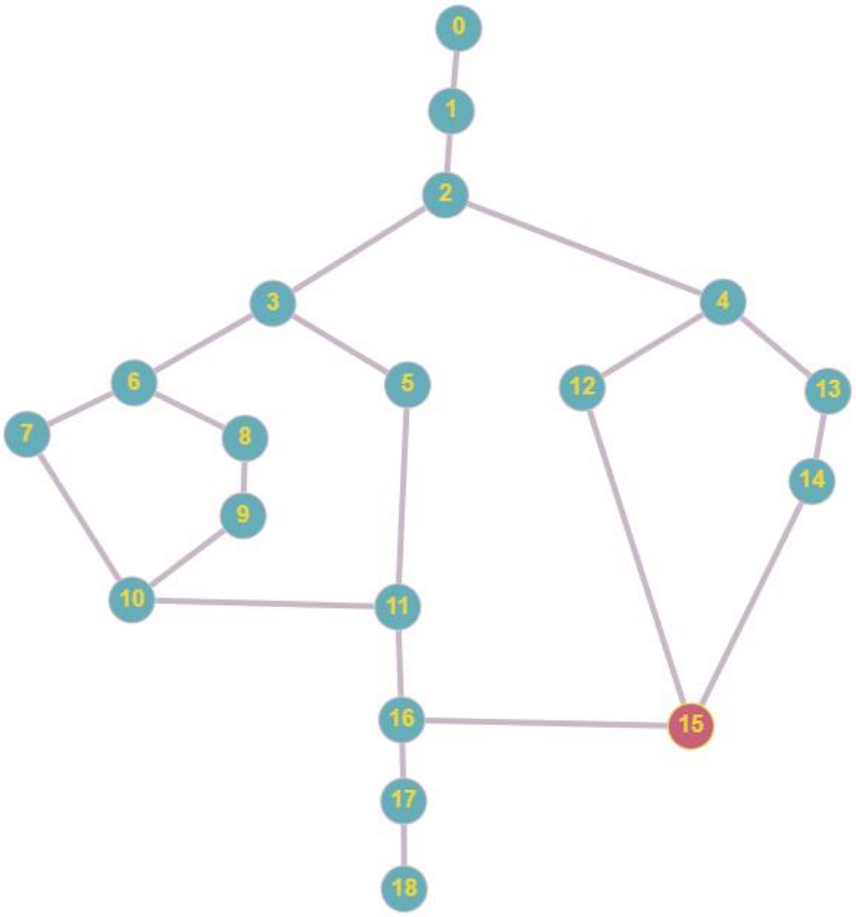
```
  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
  // Verifica si ya se registró la salida para hoy
  PreparedStatement psCheck = cn.prepareStatement("SELECT * FROM asistencias WHERE usuario = ? AND fecha = ?");
  LocalDateTime now = LocalDateTime.now();
  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
  String fecha = now.format(formatter);  // Solo la fecha, sin la hora

  psCheck.setString(1, usuario);  // Establecer usuario
  psCheck.setString(2, fecha);    // Establecer fecha actual

  ResultSet rsCheck = psCheck.executeQuery();

  // Verifica si ya existe un registro de salida para el usuario
  if (rsCheck.next()) {
      int salida = rsCheck.getInt("salida");
      if (salida == 1) {
          JOptionPane.showMessageDialog(null, "Ya has registrado tu salida hoy.");
      } else {
          // Usuario autenticado, actualiza la hora de salida
          PreparedStatement ps = cn.prepareStatement("SELECT nivel FROM usuarios WHERE cedula = ? AND contraseña = ?");
          ps.setString(1, usuario);
          ps.setString(2, contraseña);
          ResultSet rs = ps.executeQuery();

          if (rs.next()) {
              DateTimeFormatter timeFormatter = DateTimeFormatter.ofPattern("HH:mm:ss");
              String horaSalida = now.format(timeFormatter); // Hora de salida actual

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

              psAsistencia.setString(1, horaSalida); // Hora de salida
              psAsistencia.setInt(2, 1); // Marca la salida como registrada (1)
              psAsistencia.setString(3, usuario); // Establece el usuario
              psAsistencia.setString(4, fecha); // Establece 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.");
            }
        }
    } else {
        JOptionPane.showMessageDialog(null, "No se encontró un registro de asistencia para este usuario en la fecha actual.");
    }

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

2. DFD



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1 → 2 → 4 → 13 → 14 → 15 → 16 → 17 → Fin
R2: 1 → 2 → 4 → 12 → 15 → 16 → 17 → Fin
R3: 1 → 2 → 3 → 5 → 11 → 16 → 17 → Fin
R4: 1 → 2 → 3 → 6 → 8 → 9 → 10 → 11 → 16 → 17 → Fin
R5: 1 → 2 → 3 → 6 → 7 → 10 → 11 → 16 → 17 → Fin

COMPLEJIDAD CICLOMÁTICA

Fórmula 1: V(G)=P+1
P=4 (nodos de decisión).
V(G)=4+1=5
Fórmula 2: V(G)=A−N+2
A=20 (aristas).
N=17(nodos).
V(G)=20−17+2=5

Prueba caja blanca de Administrar usuarios

1. CÓDIGO FUENTE

Actualizar

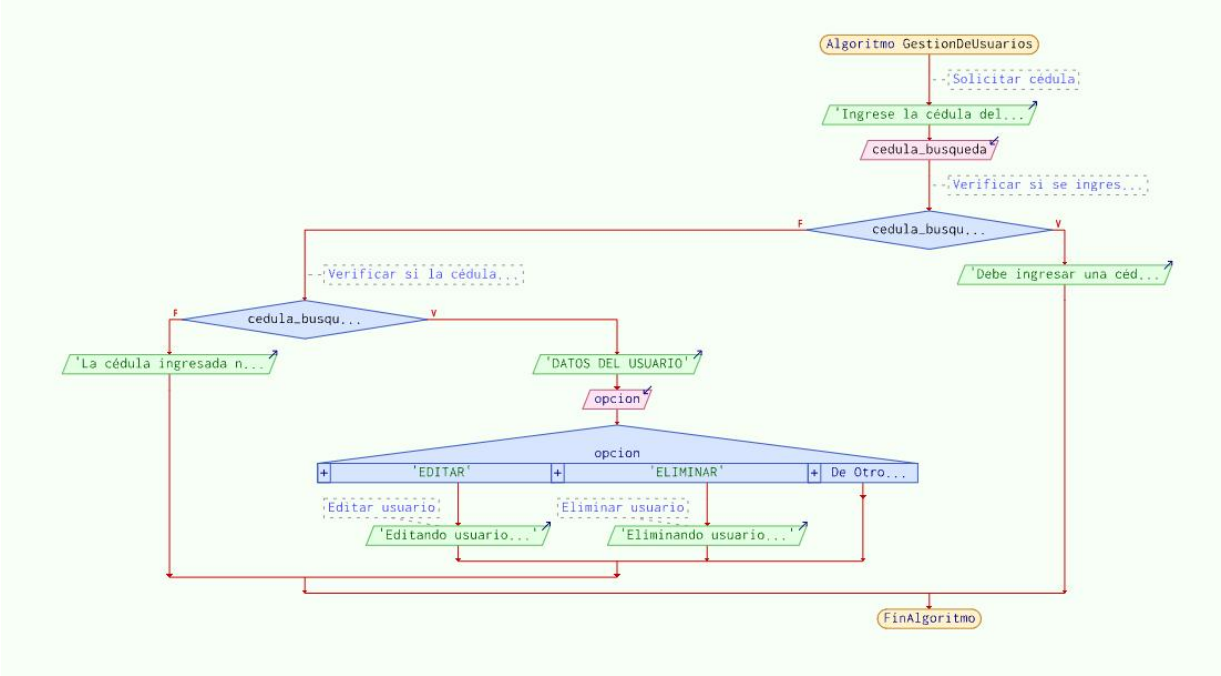
```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        String nivel,celular,direccion,contrasena;
        nivel=B_nivel.getSelectedItem().toString();
        String cedulaIngresada = B_cedula.getText();
        if (cedulaIngresada.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Por favor, ingresa la cédula.");
    }
        try{
            celular = B_celular.getText();
            direccion = B_direccion.getText();
            contrasena = B_contraseña.getText();
            PreparedStatement ps = cn.prepareStatement("UPDATE usuarios SET nivel
= ?,celular = ?,direccion = ?, contraseña = ? WHERE cedula = ?");
            ps.setString(1, nivel);
            ps.setString(2, celular);
            ps.setString(3, direccion);
            ps.setString(4, contrasena);
            ps.setString(5, cedulaIngresada);
            int rowsUpdated = ps.executeUpdate();
            if (rowsUpdated > 0) {
            JOptionPane.showMessageDialog(null, "Usuario actualizado exitosamente.");
        }
        }catch(Exception e){}
    }
```

Eliminar

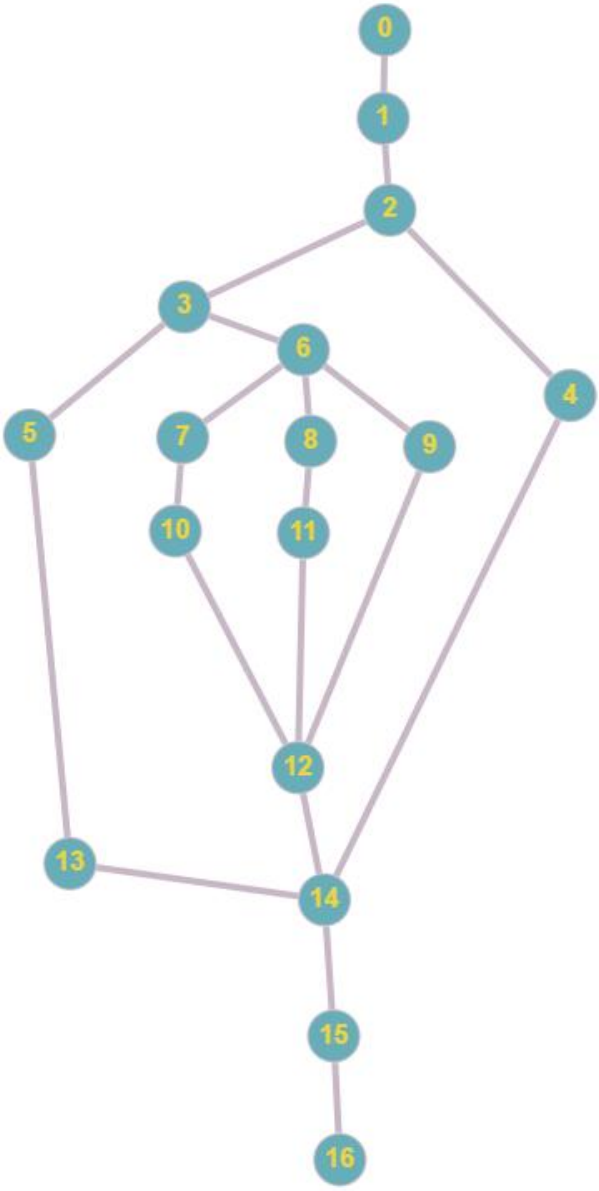
```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    String cedulaIngresada = B_cedula.getText();
    String[] options = { "si", "no"};
    if (cedulaIngresada.isEmpty()) {
    JOptionPane.showMessageDialog(null, "Por favor, ingresa la cédula para eliminar el
usuario.");}
    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_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) {
    }}}
```

2. DFD



3. GRAFO DE FLUJO



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1 → 2 → 4 → 14 → 15 → Fin
R2: 1 → 2 → 3 → 5 → 13 → 14 → 15 → Fin
R3: 1 → 2 → 3 → 6 → 9 → 12 → 14 → 15 → Fin
R4: 1 → 2 → 3 → 6 → 8 → 11 → 12 → 14 → 15 → Fin
R5: 1 → 2 → 3 → 5 → 13 → 14 → 15 → Fin

COMPLEJIDAD CICLOMÁTICA

Fórmula 1: V(G)=P+1
P=4 (nodos de decisión).
V(G)=4+1=5
Fórmula 2: V(G)=A−N+2
A=18 (aristas).
N=15(nodos).

$$V(G)=18-15+2=5$$

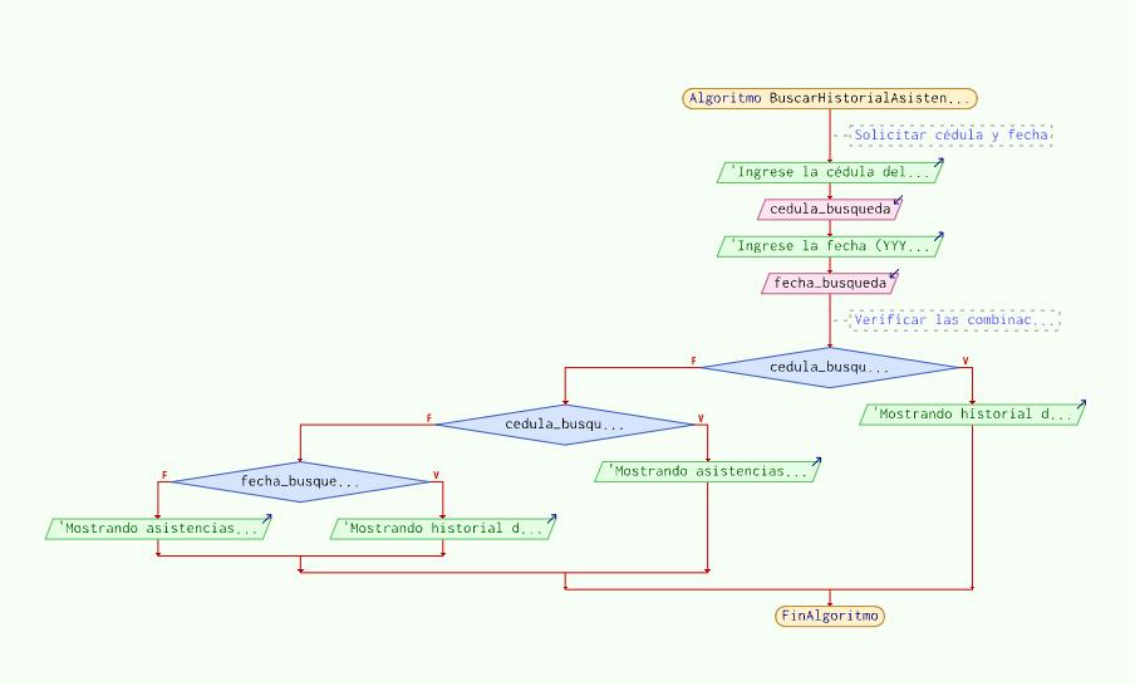
Prueba caja blanca de Historial de asistencias

1. CÓDIGO FUENTE

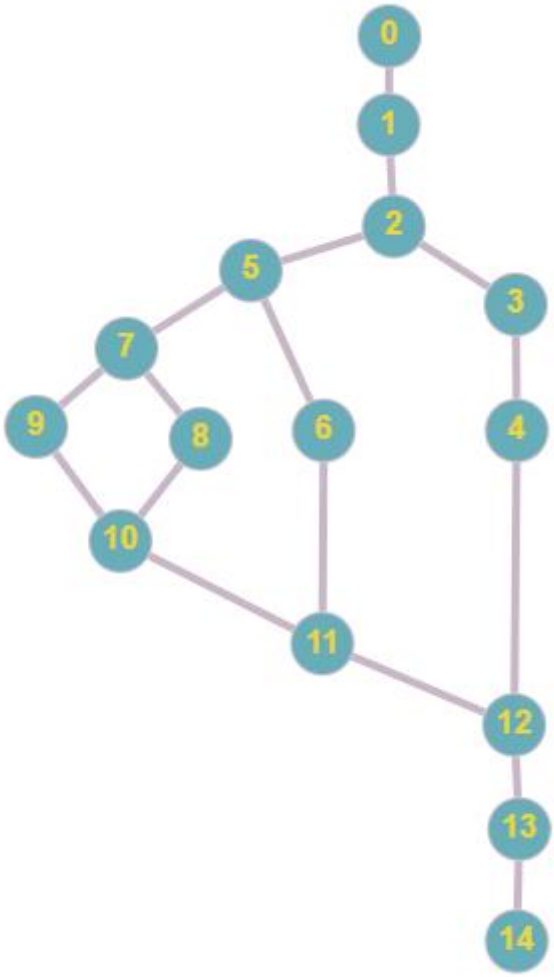
```
private void B_buscarActionPerformed(java.awt.event.ActionEvent evt) {
    String usuario_registro = H_buscar.getText().trim();
    String nombre_user = "";
    Date date = date_f.getDate();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String fechaSeleccionada = (date != null) ? 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 rs = 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 (rs.next()) {
        String usuario = usuario_registro.isEmpty() ? rs.getString("nombre") : nombre_user;
        String fecha = rs.getString("fecha");
        String hora = rs.getString("hora");
        String hora_salida = rs.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());
}

    }
```

2. DFD


Algoritmo BuscarHistorialAsisten...
Solicitar cédula y fecha
'Ingrese la cédula del...
cedula_busqueda
'Ingrese la fecha (YYY...
fecha_busqueda
Verificar las combinac...
cedula_busqu...
F
V
'Mostrando historial d...
cedula_busqu...
F
V
'Mostrando asistencias...
fecha_busque...
F
V
'Mostrando asistencias...
'Mostrando historial d...
FinAlgoritmo

3. GRAFO DE FLUJO



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1 → 2→ 3 → 4 → 12 → 13 → Fin
R2: 1 → 2 → 5 → 6 → 11 → 12 → 13 → Fin
R3: 1 → 2 → 5 → 7 → 8 → 10 → 11 → 12 → 13 → Fin
R4: 1 → 2 → 5 → 7 → 9 → 10 → 11 → 12 → 13 → Fin

COMPLEJIDAD CICLOMÁTICA

Fórmula 1: V(G)=P+1V
P=4 (nodos de decisión).
V(G)=3+1=4
Fórmula 2: V(G)=A−N+2
A=15 (aristas).
N=13 (nodos).
V(G)=15−13+2=4