UNIVERSIDAD DE SONORA

Departamento de Ingeniería Industrial

"El saber de mis hijos hará mi grandeza"

Desarrollo de Sistemas II

Impartido por el Ing. Sánchez Schmitz, Gerardo

Modelo RBAC

López Corrales, Andrés - 222211626
Mendoza Dórame, Ingrid Zareth - 222221197

23 de Mayo del 2022

Hermosillo, Sonora. México

Introducción

Para comenzar nuestro ejemplo de Role Based Access Control (RBAC), nosotros propusimos un sistema funcional que controla el acceso y los permisos de un zoológico. La idea principal es que los usuarios del sistema esten divididos en administradores, cuidadores,visitantes, y los respectivos animales que conforman el establecimiento. Nuestra propuesta para seguir un modelo de RBAC, fue predefinir una base de datos que contenga los usuarios y contraseñas de cada usuario humano, y algunas características de los animales. De esta manera, será posible limitar el acceso de cada usuario, primeramente verificando que su contraseña esté correcta, y así pudiendo darle opciones de control y métodos, dependiendo de su rol dentro del sistema. Esta es la estructura general de nuestro proyecto, pero a continuación se darán más detalles de nuestra base de datos y de los permisos de cada usuario.

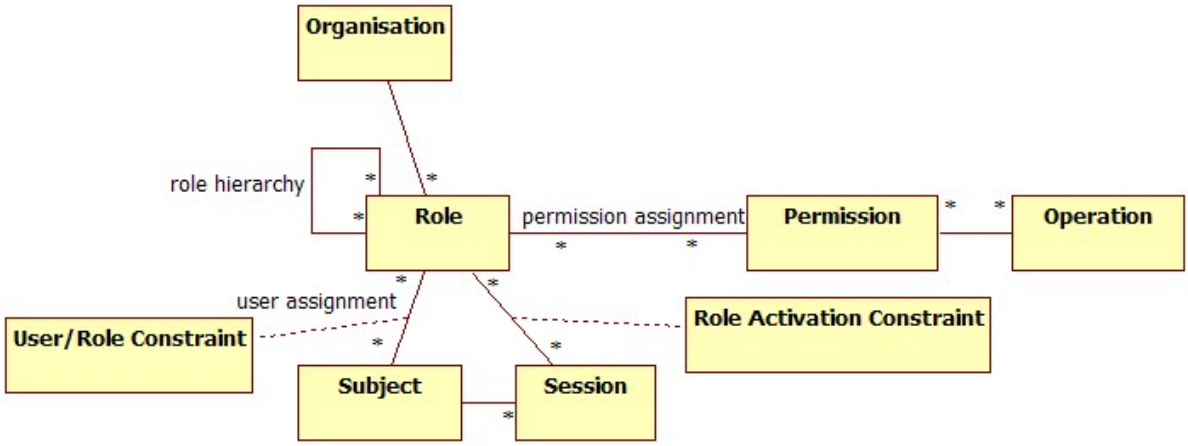
Fases de un Sistema de Software aplicadas

Identificación del problema:
Teníamos que crear un control de usuarios basado en roles, donde cada usuario tuviera roles distintos y en base a esos roles poder realizar diferentes funciones en base a qué rol perteneciera. Debíamos conectar una base de datos a nuestro programa de java para que dentro de nuestra aplicación, cuando el usuario ingresará con su respectivo rol, pudiera realizar acciones que tenía permitido.

Recopilación de información:
La información más importante que se tuvo que recopilar para poder empezar con el programa fue conocer un poco sobre el lenguaje de sql. Unos comandos que se tuvieron que investigar para poder hacer lo que se tenía planeado fueron los siguientes:

- SELECT se utiliza cuando quieres leer (o seleccionar) tus datos.
- INSERT se utiliza cuando quieres añadir (o insertar) nuevos datos.
- UPDATE se utiliza cuando quieres cambiar (o actualizar) datos existentes.
- DELETE se utiliza cuando quieres eliminar (o borrar) datos existentes.

También, vimos lo que era hacer un control de usuarios basado en roles, por lo que se investigó la estructura general de cómo es que se debía de ver un programa de este tipo, el diagrama siguiente representa la estructura:



Análisis de información:
Ya una vez con la información necesaria, concluimos que primero debíamos de crear una base de datos con los usuarios y los roles de cada uno, luego, dentro de java debíamos de hacer el primer paso para la conexión de la base de datos hacía INTELLIJ. Ya con el primer paso de la conexión, pudimos empezar a pensar en el diseño y la funcionalidad que le queríamos dar a nuestra aplicación.

Requisitos:
Los requisitos básicos principales fueron que la aplicación se conectara a una base de datos, que tuviera una interfaz gráfica, que se le implementaran botones, "labels", "textfields", etc. a esa interfaz gráfica y que cuando se ingresará a la aplicación con tu usuario, pudiera realizar las operaciones que su rol solo le permitiera.

Diseño:
El diseño que le dimos a la aplicación fue una interfaz gráfica de 720x720 de largo y de ancho, donde primeramente, se mostraría un botón de "empezar" y de "salir". El botón de "salir" funciona para salirte de la aplicación de una, pero si se da al de "empezar" te manda a otro frame donde te pide tu usuario y tu contraseña. En base con que usuario ingresas, va a mandarte a otro frame con los permisos que tiene ese usuario para realizar. Si se entra como "Administrador" tendrás permiso de ver, agregar y eliminar cuidadores del zoológico. Si entras con "Cuidadores" podrás ver, agregar y eliminar animales del zoológico. Y si entras como visitante, nomas podrás ver los animales del zoológico.

Programación:
Lo primero que se realizó fue la conexión a la base de datos por medio de la url, el usuario y la contraseña del HeidiSQL. Ya teniendo los métodos establecidos de estos 3 elementos, pudimos ir

creando la interfaz gráfica de la misma forma como se explicó en “Diseño”. Durante la programación se debieron de crear “Statements” para poder especificar qué es lo que se quería hacer a la base de datos de HeidiSQL para luego poder ejecutar el “Statement” y actualizar el hecho en la base de datos.

Implementación:
Más adelante, esta aplicación podría migrar hacía un servidor externo donde más personas puedan utilizarlo desde la comodidad de su trabajo o casa por medio de internet.

Mantenimiento:
Cada 6 meses se podría dar un mantenimiento a esta aplicación apara poder entregarle mejoras e irle agregando más funciones o incluso creando nuevos roles que sean necesario sus implementación.

Matanza del sistema:
El sistema cerrará cuando todos los usuarios sean obsoletos o se requiera de algo más avanzado y más necesario que lo que se tiene actualmente.

Base de Datos

Primeramente se definieron los usuarios de los administradores, a los cuales se les asignaron tres usuarios con sus respectivas contraseñas (por motivos de testeo, se les asignó la misma contraseña). La intención de estos roles es que tengan acceso a más actividades aparte de simplemente ver el resto de los usuarios. Además de tener acceso de vista general, los administradores también pueden añadir y eliminar otros usuarios, tales como los que conforman a los cuidadores y los visitantes. Para lograr hacer esto, se hizo una conexión con nuestra base de datos y se asignaron permisos.

desarrollo.administradores: 3 filas en total (aproximadamente)

id	user	password
1	RaulAdmin	Xenoblade05
2	BrissAdmin	Xenoblade05
3	KevinAdmin	Xenoblade05

De manera similar, definimos los usuarios y contraseñas de cuidadores, los cuales tienen acceso a los animales y a sus alimentos. Las mismas características de contraseñas se les asignaron a los cuidadores, nuevamente por motivos de testeo. Para los cuidadores, se crearon cinco usuarios de manera preliminar, los cuales pueden ser gestionados por los administradores.

desarrollo.cuidadores: 5 filas en total (aproximadamente)

id	user	password
1	SteveCuid	Xenoblade05
2	MarthaCuid	Xenoblade05
3	RamonCuid	Xenoblade05
4	RoggieCuid	Xenoblade05
5	HelpbertoCuid	Xenoblade05

Para nuestro sistema, también decidimos crear el rol de visitantes, el cual se creó con la intención de también poder ser gestionado por los administradores. Tiene una estructura muy parecida a la de los cuidadores, con la excepción de que estos pueden ver a los animales.

desarrollo.visitantes: 8 filas en total (aproximadamente)

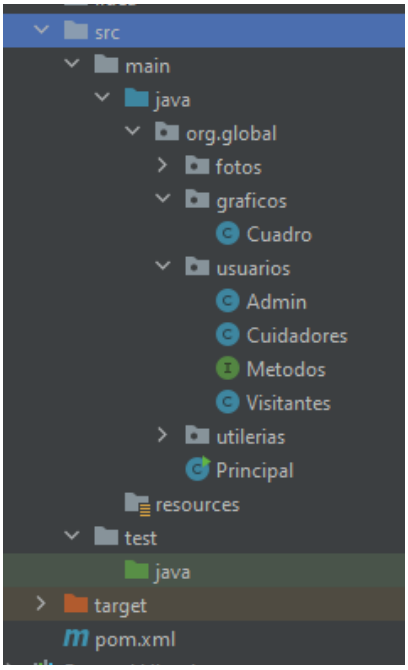
id	user	password
1	KarenV	Xenoblade05
2	JudeV	Xenoblade05
3	MikeV	Xenoblade05
4	ClayV	Xenoblade05
5	MirandaV	Xenoblade05
6	GinaV	Xenoblade05
7	ReaganV	Xenoblade05
8	ChecoV	Xenoblade05

La última tabla en nuestra base de datos es para almacenar a los animales del zoológico. Una de las características de estos es la comida que consumen, por lo que también se consideró esta propiedad. A diferencia de los roles humanos, los animales no tienen usuario ni contraseña para su acceso al sistema, sino que sirven como interacción con los cuidadores y visitantes.

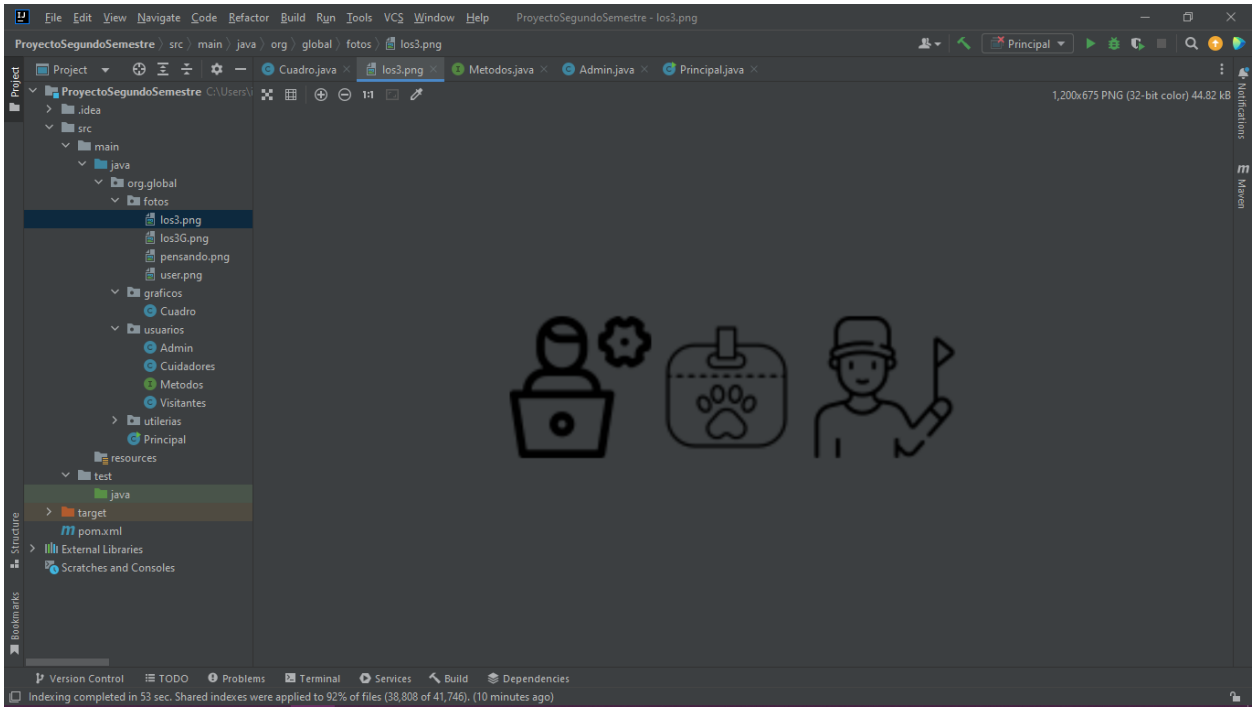
id	nombre	comida
1	Hippo	Plantas
2	León	Carne
3	Mono	Platano

Documentación del Código

Para la estructura principal de nuestro código, optamos por hacer divisiones por paquetes; fotos, graficos, usuarios y utilerias. El trabajo principal estuvo en hacer la interfaz gráfica para el usuario, y conectar el código a la base de datos mediante *pom*. A continuación se muestra la estructura general del código.



El paquete de fotos únicamente se utilizó para almacenar los recursos gráficos utilizados en los JFrames.



Por otra parte, en el paquete de Cuadros se trabajaron los JFrames para la interacción con el usuario. Dentro de esta porción es donde hicimos llamar a la base de datos, para poder acceder a las tablas y sus componentes. Estas secciones están indicadas en amarillo.

```
package org.global.graficos;

import org.global.usuarios.Admin;
import org.global.usuarios.Cuidadores;
import org.global.usuarios.Visitantes;
import org.global.utilerias.ConectorBD;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;

public class Cuadro extends JFrame { //clase de gráficos donde esta toda la
interfaz gráfica
   Admin admin = new Admin(); //Instancia clase Admin
   Cuidadores cuidadores = new Cuidadores(); //Instancia Clase Cuidadores
   Visitantes visitantes = new Visitantes(); //Instancia Clase Visitantes
   ConectorBD conectorBD = new ConectorBD(); //Instancia clase ConectorBD para
conectar a la base de datos
   //------------------------
   //Toda esta sección para la primera ventana que te sale cuando corres el
programa
   JPanel panelPrincipal;
   ImageIcon userFoto;
   JLabel fotoUsu;
   JButton empezarBoton;
   JButton salirBoton;
   //------------------------
   //Segundo frame, aquí es donde ingresas poniendo el usuario y contraseña
   JPanel panelIngresaAdmin;
   JFrame frameIngresaAdmin;
   ImageIcon imagenAdmin;
   JLabel fotoImagenAdmin;
   JLabel userAdmin;
   JLabel contraAdmin;
   JTextField userFieldAdmin;
   JPasswordField contraFieldAdmin;
   JButton ingresarAdmin;
   //------------------------
   String usu; //Usuario - Variable que es la que va a tomar valor cuando se
ingresa con este usuario
   String pass; //Contraseña - Variable que es la que va a tomar valor cuando
se ingresa con este usuario
```

```
   Connection con;//Con esto se creará una conexión hacia la base de datos


   public Cuadro() { //Primer frame de la aplicación
       panelPrincipal = new JPanel();
       panelPrincipal.setLayout(null);//Layout null
       add(panelPrincipal);
       setSize(720,720);//720x720
       setResizable(false);
       setLocationRelativeTo(null);
       setDefaultCloseOperation(3);
       primeraParteDelFrame();//Contenido que tiene este primer frame
       setVisible(true);
   }

   public void primeraParteDelFrame(){//Contenido que va a tener el primer
frame
       fotoUsu = new JLabel();
       userFoto = new
ImageIcon("src\\main\\java\\org\\global\\fotos\\user.png");
       fotoUsu.setLocation(266,0);
       fotoUsu.setSize(300,200);
       fotoUsu.setIcon(new
ImageIcon(userFoto.getImage().getScaledInstance(150,150, Image.SCALE_SMOOTH)));
       fotoUsu.setVisible(true);
       panelPrincipal.add(fotoUsu);

       empezarBoton = new JButton();
       empezarBoton.setBounds(270,260,150,50);
       empezarBoton.setVisible(true);
       empezarBoton.setText("Empezar");
       empezarBoton.setFont(new Font("Times new roman",Font.BOLD,20));
       ActionListener cuandoEmpezar = new ActionListener() {
           @Override
           public void actionPerformed(ActionEvent e) {
               dispose();
               ingresaAdmin();//lo que se hace cuando clickeas a empezar

           }
       };empezarBoton.addActionListener(cuandoEmpezar);

       panelPrincipal.add(empezarBoton);

       salirBoton = new JButton();
       salirBoton.setBounds(270,460,150,50);
       salirBoton.setVisible(true);
       salirBoton.setText("Salir");
       salirBoton.setFont(new Font("Times new roman",Font.BOLD,20));
       ActionListener darSalir = new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {//Lo que hace cuando
clickeas "Salir"
                String[] adios = {"Salir","Cancelar"};
                int salir = JOptionPane.showOptionDialog(null,"¿Quiere
salir?","SALIR",JOptionPane.DEFAULT_OPTION,JOptionPane.QUESTION_MESSAGE,new
ImageIcon("src\\main\\java\\org\\global\\fotos\\pensando.png"),adios,adios[0]);
                if (salir == 0) {
                    dispose();
                }
            }
        };salirBoton.addActionListener(darSalir);

        panelPrincipal.add(salirBoton);
    }

    public void ingresaAdmin(){ //Segundo frame - aquí se inicia con usuario y
contraseña
        panelIngresaAdmin = new JPanel();
        frameIngresaAdmin = new JFrame();
        panelIngresaAdmin.setLayout(null);
        frameIngresaAdmin.add(panelIngresaAdmin);
        frameIngresaAdmin.setSize(720,720);
        frameIngresaAdmin.setResizable(false);
        frameIngresaAdmin.setLocationRelativeTo(null);
        frameIngresaAdmin.setDefaultCloseOperation(3);
        elementosIngresaAdmin(); //Llamada a función de los elementos que tendrá
este segundo frame
        frameIngresaAdmin.setVisible(true);
    }

    public void elementosIngresaAdmin(){ //Elementos que tendrá el segundo frame
        imagenAdmin = new
ImageIcon("src\\main\\java\\org\\global\\fotos\\los3G.png");
        fotoImagenAdmin = new JLabel();
        fotoImagenAdmin.setBounds(180,30,300,200);
        fotoImagenAdmin.setIcon(new
ImageIcon(imagenAdmin.getImage().getScaledInstance(350,350,
Image.SCALE_SMOOTH)));
        fotoImagenAdmin.setVisible(true);

        userFieldAdmin = new JTextField();
        userFieldAdmin.setBounds(270,230,300,50);
        userFieldAdmin.setVisible(true);

        contraFieldAdmin = new JPasswordField();
        contraFieldAdmin.setBounds(270,360,300,50);
        contraFieldAdmin.setVisible(true);
```

```
        userAdmin = new JLabel("User: ");
        userAdmin.setBounds(200,230,150,50);
        userAdmin.setFont(new Font("New Times Roman",Font.BOLD,20));
        userAdmin.setVisible(true);

        contraAdmin = new JLabel("Password: ");
        contraAdmin.setBounds(150,360,150,50);
        contraAdmin.setFont(new Font("New Times Roman",Font.BOLD,20));
        contraAdmin.setVisible(true);

        ingresarAdmin = new JButton("Ingresar");
        ingresarAdmin.setBounds(270,490,150,50);
        ingresarAdmin.setFont(new Font("Times New Roman",Font.BOLD,20));
        ActionListener actionIngr = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                try {
                    Class.forName("com.mysql.cj.jdbc.Driver");
                    con =
DriverManager.getConnection(conectorBD.paraConectarUrl(),

conectorBD.paraConectarUsername(),conectorBD.paraConectarPass()); //Connection
a la base de datos

                    usu = userFieldAdmin.getText();
                    pass = contraFieldAdmin.getText();

                    //Entrará en base si ingresaste como un admin, un cuidador o
un visitante
                    Statement sta = con.createStatement();
                    String sql = "select * from administradores where
user='"+usu+"' and password='"+pass+"'";
                    ResultSet rs = sta.executeQuery(sql);

                    Statement sta1 = con.createStatement();
                    String sql1 = "select * from cuidadores where user='"+usu+"'
and password='"+pass+"'";
                    ResultSet rs1 = sta1.executeQuery(sql1);

                    Statement sta2 = con.createStatement();
                    String sql2 = "select * from visitantes where user='"+usu+"'
and password='"+pass+"'";
                    ResultSet rs2 = sta2.executeQuery(sql2);

                    if (rs.next()){ //Si todo bien, se ingresará como admin
                         frameIngresaAdmin.dispose();
                        admin.decirQnEntro("Entró Admin "+usu);//Admin - si es
que se conecta a la base de datos, esta es la acción que realizará
```

```
                        cuadroFinalAdmin();

                    } else if (rs1.next()) {//Si todo bien, se ingresará como
cuidador
                        frameIngresaAdmin.dispose();
                        cuidadores.decirQnEntro("Entró Cuidador "+
usu);//Cuidadores - si es que se conecta a la base de datos, esta es la acción
que realizará
                        cuadroFinalCuidadores();

                    } else if (rs2.next()) {//Si todo bien, se ingresará como
visitante
                        frameIngresaAdmin.dispose();
                        visitantes.decirQnEntro("Entró visitante
"+usu);//Visitantes - si es que se conecta a la base de datos, esta es la
acción que realizará
                        cuadroFinalVisitantes();

                    }else {
                        JOptionPane.showMessageDialog(null,"MAL...");
                        userFieldAdmin.setText("");
                        contraFieldAdmin.setText("");
                    }

                }catch (Exception exe){
                    System.out.println(exe.getMessage());
                }
            }
        };ingresarAdmin.addActionListener(actionIngr);

        ingresarAdmin.setVisible(true);
        panelIngresaAdmin.add(fotoImagenAdmin);
        panelIngresaAdmin.add(userFieldAdmin);
        panelIngresaAdmin.add(contraFieldAdmin);
        panelIngresaAdmin.add(userAdmin);
        panelIngresaAdmin.add(contraAdmin);
        panelIngresaAdmin.add(ingresarAdmin);
    }

    JPanel panelFinalVisi;
    JFrame frameFinalVisi;
    JLabel holaVisit;
    JButton verAnimalesVisit;
    JTextArea dondeVerAnimVisi;
    public void cuadroFinalVisitantes(){ //Frame que entrará si entramos como
visitante
        panelFinalVisi = new JPanel();
        frameFinalVisi = new JFrame();
        panelFinalVisi.setLayout(null);
```

```
        frameFinalVisi.add(panelFinalVisi);
        frameFinalVisi.setSize(720,720);
        frameFinalVisi.setResizable(false);
        frameFinalVisi.setLocationRelativeTo(null);
        frameFinalVisi.setDefaultCloseOperation(3);
        elementosFinalVisitante();//Llamado al metodo de los elementos que
tendrá el frame de visitantes
        frameFinalVisi.setVisible(true);
    }

    public void elementosFinalVisitante(){//Elementos del frame de visitantes
        holaVisit = new JLabel("Hola Visitante "+ usu);
        holaVisit.setBounds(95,0,800,200);
        holaVisit.setFont(new Font("Times New Roman",Font.BOLD,40));

        dondeVerAnimVisi = new JTextArea();
        dondeVerAnimVisi.setLineWrap(true);
        dondeVerAnimVisi.setEditable(false);
        JScrollPane scroll = new
JScrollPane(dondeVerAnimVisi,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.
HORIZONTAL_SCROLLBAR_NEVER);
        scroll.setBounds(110,300,500,300);

        verAnimalesVisit = new JButton("Ver Animales");
        verAnimalesVisit.setBounds(250,200,200,50);
        verAnimalesVisit.setFont(new Font("times new roman",Font.ITALIC,30));
        ActionListener darBotonVerAnimales = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { //Cuando de le da a
este botón realizará la acción de ver los elementos de la base de datos de los
animales
                dondeVerAnimVisi.setText("");
                Statement sta = null;
                        try {
                            sta = con.createStatement();
                        } catch (SQLException ex) {
                            throw new RuntimeException(ex);
                        }
                        String sql = "SELECT nombre from animales"; //Nomas
podrá ver los animales de la base de datos
                        try {
                            ResultSet rs = sta.executeQuery(sql);
                            ResultSetMetaData rsmd = rs.getMetaData();
                            int columnas = rsmd.getColumnCount();

                            while (rs.next()) {
                                for (int i = 1; i <= columnas; i++) {
                                    dondeVerAnimVisi.append(rs.getString(i) +
"\t");
```

```
                                }
                                dondeVerAnimVisi.append("\n");
                    }
                } catch (SQLException ex) {
                    throw new RuntimeException(ex);
                }
            }
        };verAnimalesVisit.addActionListener(darBotonVerAnimales);
        panelFinalVisi.add(holaVisit);
        panelFinalVisi.add(verAnimalesVisit);
        panelFinalVisi.add(scroll);
    }

    JPanel panelFinalCuid;
    JFrame frameFinalCuid;
    JLabel holaCuid;
    JButton verComidaCuid;
    JButton agregarAnimal;
    JButton eliminarAnimal;
    JTextArea dondeVerComidaCuid;

    public void cuadroFinalCuidadores(){//Frame que entrará si entramos como
cuidadores
        panelFinalCuid = new JPanel();
        frameFinalCuid = new JFrame();
        panelFinalCuid.setLayout(null);
        frameFinalCuid.add(panelFinalCuid);
        frameFinalCuid.setSize(720,720);
        frameFinalCuid.setResizable(false);
        frameFinalCuid.setLocationRelativeTo(null);
        frameFinalCuid.setDefaultCloseOperation(3);
        elementosFinalCuidador();//Llamado al metodo de los elementos que tendrá
el frame de cuidadores
        frameFinalCuid.setVisible(true);
    }

    public void elementosFinalCuidador(){//Elementos que tendrá el frame de
cuidadores
        dondeVerComidaCuid = new JTextArea();
        dondeVerComidaCuid.setLineWrap(true);
        dondeVerComidaCuid.setEditable(false);
        JScrollPane scroll = new
JScrollPane(dondeVerComidaCuid,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPan
e.HORIZONTAL_SCROLLBAR_NEVER);
        scroll.setBounds(110,300,500,300);


        holaCuid = new JLabel("Hola Cuidador "+ usu);
        holaCuid.setBounds(95,0,800,200);
```

```
        holaCuid.setFont(new Font("Times New Roman",Font.BOLD,40));

        verComidaCuid = new JButton("Ver Comida");
        verComidaCuid.setBounds(50,200,200,50);
        verComidaCuid.setFont(new Font("times new roman",Font.ITALIC,20));

        agregarAnimal = new JButton("Agregar Animal");
        agregarAnimal.setBounds(250,200,200,50);
        agregarAnimal.setFont(new Font("times new roman",Font.ITALIC,20));

        eliminarAnimal = new JButton("Eliminar Animal");
        eliminarAnimal.setBounds(450,200,200,50);
        eliminarAnimal.setFont(new Font("times new roman",Font.ITALIC,20));


        ActionListener botonesCuidador = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (e.getSource() == verComidaCuid){//Boton de "ver" para ver la
comida de los animales
                    dondeVerComidaCuid.setText("");
                    Statement sta = null;
                    try {
                        sta = con.createStatement();
                    } catch (SQLException ex) {
                        throw new RuntimeException(ex);
                    }
                    String sql = "SELECT nombre, comida from animales";
                    try {
                        ResultSet rs = sta.executeQuery(sql);
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int columnas = rsmd.getColumnCount();

                        while (rs.next()) {
                            for (int i = 1; i <= columnas; i++) {
                                dondeVerComidaCuid.append(rs.getString(i) +
"\t");
                            }
                            dondeVerComidaCuid.append("\n");
                        }
                    } catch (SQLException ex) {
                        throw new RuntimeException(ex);
                    }
                }
                if (e.getSource() == agregarAnimal){//Botón para ingresar a un
nuevo animal a la base de datos
                    try {
                        String agregarAni = "INSERT INTO animales (nombre,
comida) VALUES (?, ?)";
```

```
                        PreparedStatement sta =
con.prepareStatement(agregarAni); //Statement para ir haciendo una conexión con
la variable "AgregarAni" que tiene el comando de sql

                        String nombreAni = "";
                        String comidaAni = "";

                        boolean parar = false;
                        boolean parar1 = false;


                            do {
                                nombreAni = JOptionPane.showInputDialog("Nombre
del nuevo animal: ");
                                if (nombreAni == null){
                                    nombreAni = "";
                                }
                                if (nombreAni.equals("")) {
                                    JOptionPane.showMessageDialog(null,
"MAL...Agrega un animal bien");
                                    parar = false;
                                } else {
                                    parar = true;
                                }
                            } while (!parar);

                            do {
                                comidaAni = JOptionPane.showInputDialog("Comida
del nuevo animal");
                                if (comidaAni == null){
                                    comidaAni = "";
                                }
                                if (comidaAni.equals("")){

JOptionPane.showMessageDialog(null,"MAL...Agrega una comida bien");
                                }else{
                                    JOptionPane.showMessageDialog(null, "Se
agregó " + nombreAni);
                                    parar1 = true;
                                }
                            }while (!parar1);

                        //Se coloca la información dentro de las variables del
statement
                        sta.setString(1, nombreAni);
                        sta.setString(2, comidaAni);
```

```
                        sta.executeUpdate();//Se ejecuta y se actualiza en la
base de datos

                        sta.close();//Se cierra el statement
                    } catch (SQLException exe) {
                        exe.printStackTrace();
                    }
                }

                if (e.getSource() == eliminarAnimal){//Boton para eliminar a un
animal de la base de datos
                    try {
                        String eliminarAni = "DELETE FROM animales WHERE nombre
= ?";

                        PreparedStatement sta =
con.prepareStatement(eliminarAni);

                        String eliminarNombreAni = "";

                        boolean parar = false;

                        try {
                            do {// el do while funciona para poder realizar la
acción entre los parentesis del 'do' hasta que la condición inicial cambie
                                eliminarNombreAni =
JOptionPane.showInputDialog("Nombre del animal que eliminar: ");
                                if (eliminarNombreAni.equals("")) {
                                    JOptionPane.showMessageDialog(null,
"MAL...No se eliminó nada porque no pusiste nada");
                                    parar = false;
                                } else {
                                    JOptionPane.showMessageDialog(null,
"Eliminaste a " + eliminarNombreAni);
                                    parar = true;
                                }
                            } while (!parar);
                        }catch (NullPointerException npe){
                            JOptionPane.showMessageDialog(null,"Se canceló la
operación");
                        }

                        sta.setString(1,eliminarNombreAni);

                        sta.executeUpdate();

                        sta.close();
                    } catch (SQLException exe) {
                        exe.printStackTrace();
```

```
                    }
                }
            }

};verComidaCuid.addActionListener(botonesCuidador);agregarAnimal.addActionListe
ner(botonesCuidador);
        eliminarAnimal.addActionListener(botonesCuidador);

        panelFinalCuid.add(holaCuid);
        panelFinalCuid.add(verComidaCuid);
        panelFinalCuid.add(agregarAnimal);
        panelFinalCuid.add(eliminarAnimal);
        panelFinalCuid.add(scroll);

    }

    JPanel panelFinalAdmin;
    JFrame frameFInalAdmin;
    JLabel holaAdmin;
    JButton verCuidAdmin;
    JButton agregarCuidAdmin;
    JButton eliminarCuidAdmin;
    JTextArea dondeVerCuidAdmin;

    public void cuadroFinalAdmin(){ //Frame si se ingresa como admin
        panelFinalAdmin = new JPanel();
        frameFInalAdmin = new JFrame();
        panelFinalAdmin.setLayout(null);
        frameFInalAdmin.add(panelFinalAdmin);
        frameFInalAdmin.setSize(720,720);
        frameFInalAdmin.setResizable(false);
        frameFInalAdmin.setLocationRelativeTo(null);
        frameFInalAdmin.setDefaultCloseOperation(3);
        elementosFinalAdmin();//Llamado al metodo de los elementos que tendrá el
frame de administrador
        frameFInalAdmin.setVisible(true);
    }

    public void elementosFinalAdmin(){//Elementos que tendrá el frame de
administrador
        holaAdmin = new JLabel("Hola Admin "+usu);
        holaAdmin.setBounds(95,0,800,200);
        holaAdmin.setFont(new Font("Times New Roman",Font.BOLD,40));

        dondeVerCuidAdmin = new JTextArea();
        dondeVerCuidAdmin.setLineWrap(true);
        dondeVerCuidAdmin.setEditable(false);
```

```
        JScrollPane scroll = new
JScrollPane(dondeVerCuidAdmin,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane
.HORIZONTAL_SCROLLBAR_NEVER);
        scroll.setBounds(110,300,500,300);

        verCuidAdmin = new JButton("Ver Cuid");
        verCuidAdmin.setBounds(50,200,200,50);
        verCuidAdmin.setFont(new Font("times new roman",Font.ITALIC,20));

        agregarCuidAdmin = new JButton("Agregar Cuid");
        agregarCuidAdmin.setBounds(250,200,200,50);
        agregarCuidAdmin.setFont(new Font("times new roman",Font.ITALIC,20));

        eliminarCuidAdmin = new JButton("Eliminar Cuid");
        eliminarCuidAdmin.setBounds(450,200,200,50);
        eliminarCuidAdmin.setFont(new Font("times new roman",Font.ITALIC,20));

        ActionListener botonesAdmin = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (e.getSource() == verCuidAdmin){ //Botón para ver los
cuidadores de la base de datos
                    dondeVerCuidAdmin.setText("");
                    Statement sta = null;
                    try {
                        sta = con.createStatement();
                    } catch (SQLException ex) {
                        throw new RuntimeException(ex);
                    }
                    String sql = "SELECT user, password FROM cuidadores";
                    try {
                        ResultSet rs = sta.executeQuery(sql);
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int columnas = rsmd.getColumnCount();

                        while (rs.next()) {
                            for (int i = 1; i <= columnas; i++) {
                                dondeVerCuidAdmin.append(rs.getString(i) +
"\t");
                            }
                            dondeVerCuidAdmin.append("\n");
                        }
                    } catch (SQLException ex) {
                        throw new RuntimeException(ex);
                    }
                }
                if (e.getSource() == agregarCuidAdmin){//Botón para poder
ingresar cuidadores a la base de datos
```

```
                    try {
                        String agregarCuidador = "INSERT INTO cuidadores (user,
password) VALUES (?, ?)";

                        PreparedStatement sta =
con.prepareStatement(agregarCuidador);

                        JDialog cerrar = new JDialog();
                        String nombreCuid = "";

                        boolean parar = false;

                            do {
                                nombreCuid =
JOptionPane.showInputDialog(cerrar,"Nombre del nuevo cuidador: ");
                                if (nombreCuid == null){
                                    nombreCuid = "";
                                }
                                if (nombreCuid.equals("")) {
                                    JOptionPane.showMessageDialog(null,
"MAL...Agrega un nombre bien");
                                    parar = false;
                                } else {
                                    JOptionPane.showMessageDialog(null, "Se
agregó " + nombreCuid);
                                    parar = true;
                                }

                            } while (!parar);

                        String contrasena = "Xenoblade05";

                        sta.setString(1, nombreCuid);
                        sta.setString(2, contrasena);

                        sta.executeUpdate();

                        sta.close();
                    } catch (SQLException exe) {
                        exe.printStackTrace();
                    }
                }
                if (e.getSource() == eliminarCuidAdmin){ //Botón para eliminar a
un cuidador de la base de datos
                    try {
                        String eliminarCuidador = "DELETE FROM cuidadores WHERE
user = ?";
```

```
                        PreparedStatement sta =
con.prepareStatement(eliminarCuidador);

                        String eliminarCuid = "";

                        boolean parar = false;

                        try {
                            do {// el do while funciona para poder realizar la
acción entre los parentesis del 'do' hasta que la condición inicial cambie
                                eliminarCuid =
JOptionPane.showInputDialog("Nombre del cuidador que eliminar: ");
                                if (eliminarCuid.equals("")) {
                                    JOptionPane.showMessageDialog(null,
"MAL...No se eliminó nada porque no pusiste nada");
                                    parar = false;
                                } else {
                                    JOptionPane.showMessageDialog(null,
"Eliminaste a " + eliminarCuid);
                                    parar = true;
                                }
                            } while (!parar);
                        }catch (NullPointerException npe){
                            JOptionPane.showMessageDialog(null,"Se canceló la
operación");
                        }

                        sta.setString(1,eliminarCuid);

                        sta.executeUpdate();

                        sta.close();
                    } catch (SQLException exe) {
                        exe.printStackTrace();
                    }
                }
            }

};verCuidAdmin.addActionListener(botonesAdmin);agregarCuidAdmin.addActionListen
er(botonesAdmin);
        eliminarCuidAdmin.addActionListener(botonesAdmin);

        panelFinalAdmin.add(holaAdmin);
        panelFinalAdmin.add(verCuidAdmin);
        panelFinalAdmin.add(agregarCuidAdmin);
        panelFinalAdmin.add(eliminarCuidAdmin);
        panelFinalAdmin.add(scroll);
    }
}
```

Dentro del paquete Usuarios, se definieron tres clases y una interfaz. La interfaz utilizada fue la siguiente, y se implementó como recurso de verificación del método `decirQnEntro`, para asegurarnos del rol del usuario.

```
package org.global.usuarios;

public interface Metodos {//Interfaz para agregarlas a las clases de Admin,
Cuidadores y Visitantes

   void decirQnEntro(String usuario);

}
```

El siguiente paquete, Utilerias, fue uno de los más importantes, pues es lo que permite la conexión directa con nuestra base de datos. Aquí definimos la ruta de la base, al igual que la contraseña que utilizamos para su acceso.

```
package org.global.utilerias;

public class ConectorBD { //Clase donde se definen las variables para usarlas
cuando se requiera conectarse a la base de datos
   public final static String url = "jdbc:mysql://localhost:3306/desarrollo";
   public final static String username = "root";
   public final static String password = "Xenoblade04";
   public String paraConectarUrl(){String mandar = url+"?useSSL=false";return
mandar;
   }
   public String paraConectarUsername(){return username;
   }
   public String paraConectarPass(){
       return password;
   }
}
```

Finalmente, en la clase Principal, mandamos llamar una nueva instancia de nuestra clase Cuadro, para correr los JFrames del paquete Graficos.

```
package org.global;

import org.global.graficos.Cuadro;

public class Principal {
```

```
   public static void main(String[] args) {//Main
       Cuadro cuadro = new Cuadro();//Se activa la clase cuadro que es la de la
interfaz gráfica
   }
}
```

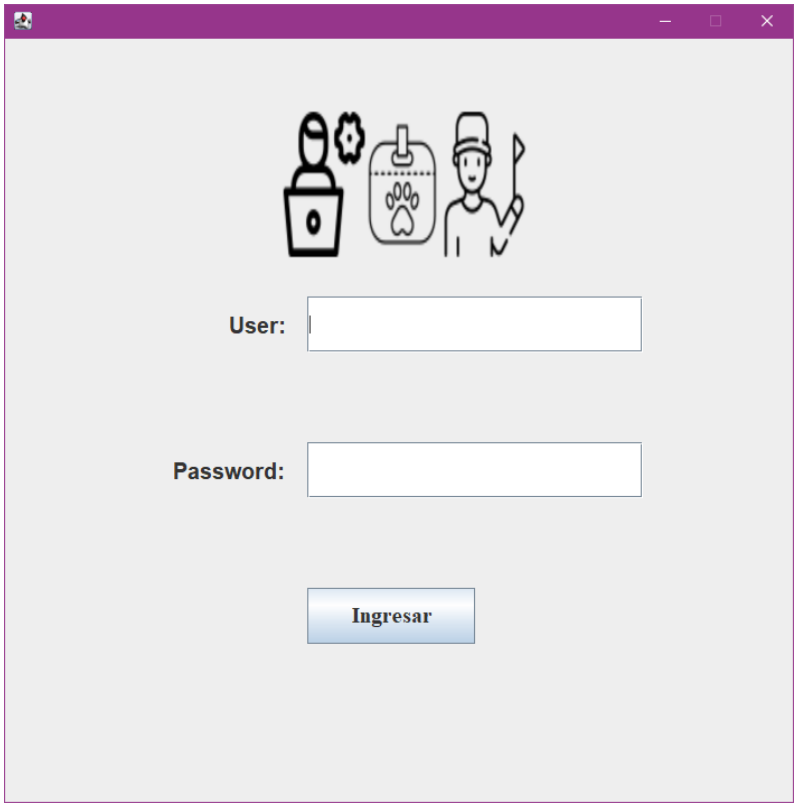
Como nota adicional, para la conexión con pom, utilizamos la siguiente estructura preexistente.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>org.example</groupId>
   <artifactId>BD</artifactId>
   <version>1.0-SNAPSHOT</version>

   <properties>
       <maven.compiler.source>17</maven.compiler.source>
       <maven.compiler.target>17</maven.compiler.target>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>
   <dependencies>
       <dependency>
           <groupId>mysql</groupId>
           <artifactId>mysql-connector-java</artifactId>
           <version>8.0.32</version>
       </dependency>
       <dependency>
           <groupId>org.mariadb.jdbc</groupId>
           <artifactId>mariadb-java-client</artifactId>
           <version>2.7.2</version>
       </dependency>
   </dependencies>

</project>
```

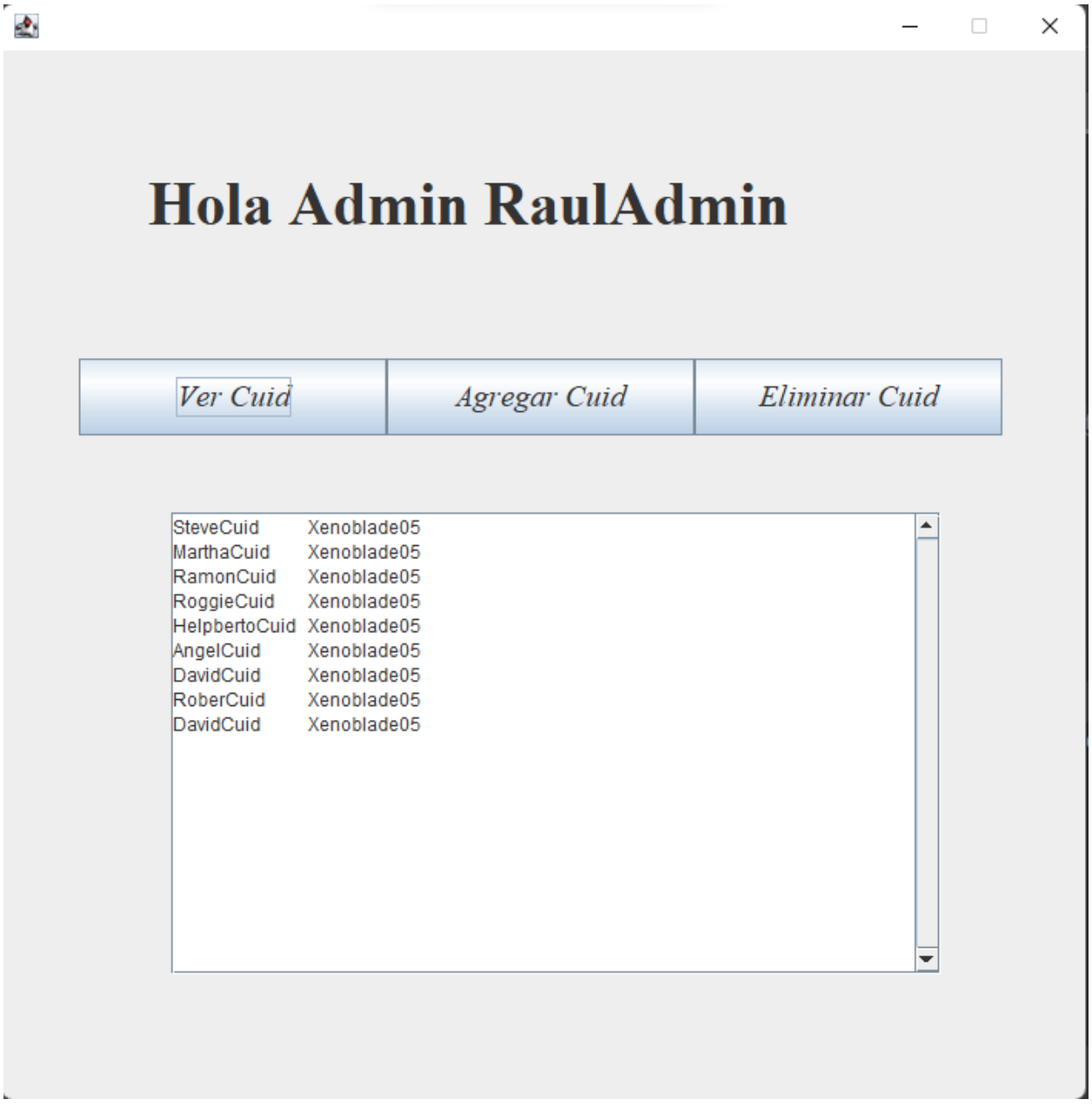
Manual de Uso

Al comenzar, tenemos el frame principal.

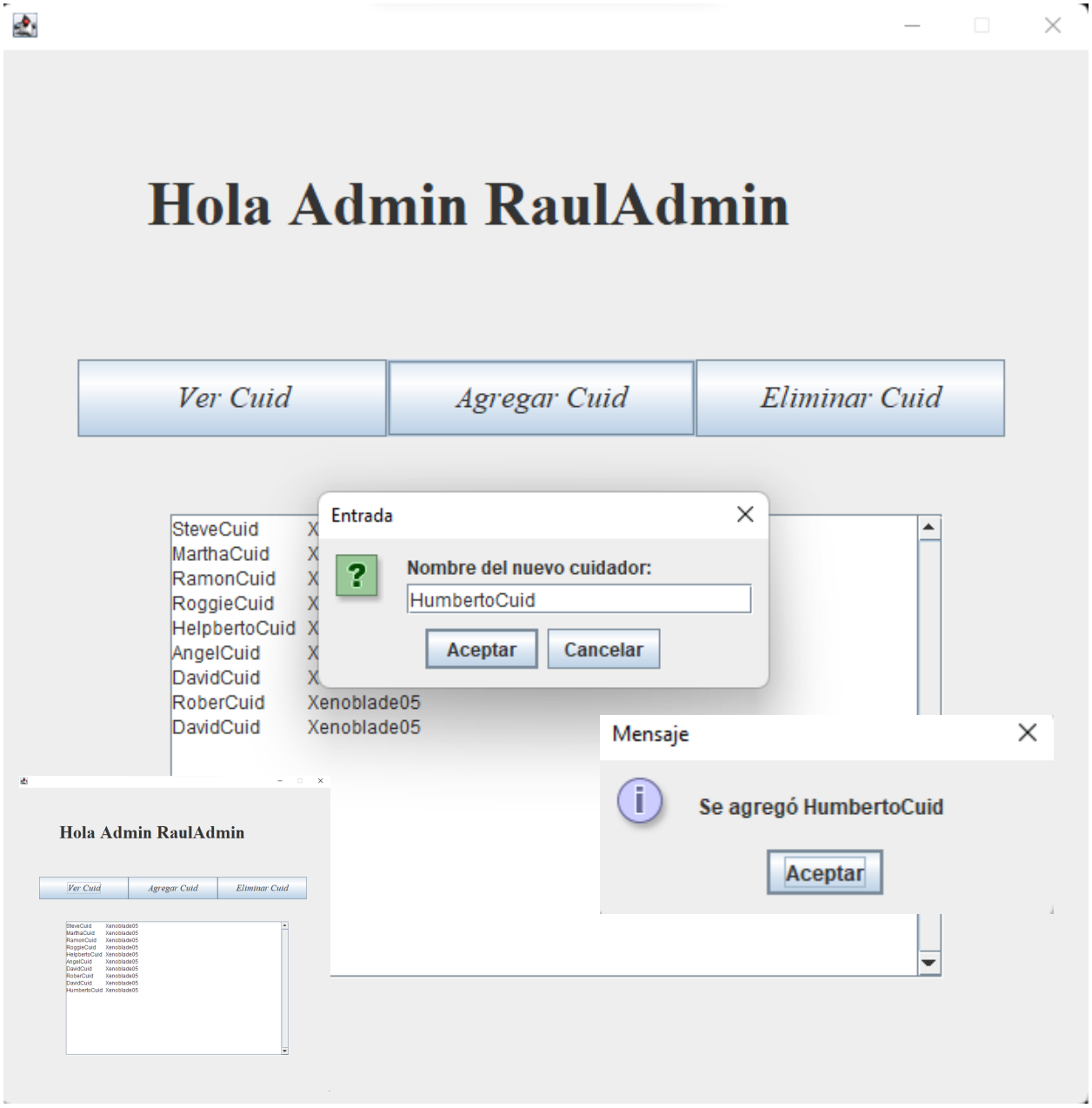
Después de dar Empezar, tenemos la siguiente pantalla, la cual te pide validar por medio de tu usuario y contraseña que seas parte del sistema.



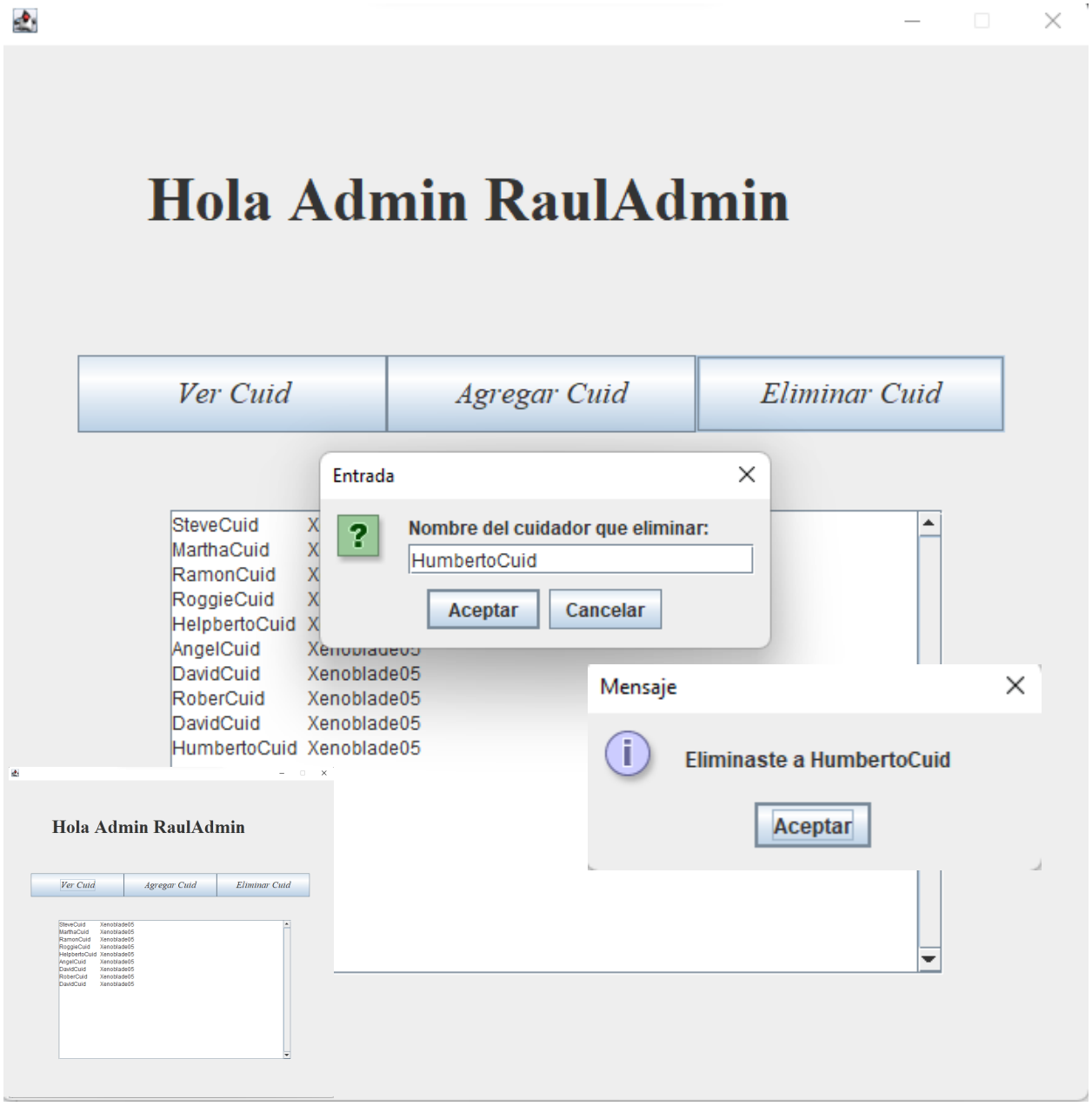
El usuario y contraseña deben de coincidir con uno de los existentes en la base de datos. De no ser el caso, le pedirá al usuario intentar de nuevo. La idea principal de nuestro sistema, es controlar el tipo de acceso a distintos métodos y acciones, dependiendo del rol que inicie sesión. Es decir, si se accede con un rol de cuidador, este no tendrá acceso a los métodos de un administrador, y viceversa.
Cuando se entra con el administrador, tendrás las opciones de ver la base de datos de los cuidadores y la opción de agregar y eliminar cuidadores de esta misma base de datos.



Para ver a los cuidadores



Agregar a los cuidadores



Eliminar a los cuidadores

Cuando se entra con el cuidador, tendrás las opciones de ver la base de datos de los animales y la comida que le toca a cada uno de ellos y la opción de agregar y eliminar animales y su comida de esta misma base de datos

Ver animales y comida de la base de datos

Agregar animales y su comida a la base de datos

Eliminar animales de la base de datos

Cuando se entra con el visitante, tendrás las opciones de ver solamente a los animales de la base de datos, no tendrá opción de agregar ni eliminar.

Solamente ver animales de la base de datos

En el video adjuntado a continuación, esto se muestra con claridad. El video también incluye una demostración de los permisos que tiene cada rol, mostrando las acciones que pueden realizar mediante los métodos establecidos.

Video: ***https://www.youtube.com/watch?v=t2w05rXCAOw&ab_channel=casAndres***

Referencias - Recopilación de información:

- MariaDB. (s.f.). Comandos SQL Básicos. https://mariadb.com/kb/es/basic-sql-statements/
- Wikipedia. (s.f.). Role-based access control. https://en.wikipedia.org/wiki/Role-based_access_control