manual técnico

Sistema de aprovisionamiento vlan, vrf y direccionamiento

Contenido

[REQUISITOS 2](#_Toc536375139)

[RED DEL PROVEEDOR DE INTERNET 2](#_Toc536375140)

[**DIAGRAMA DE RED** 2](#_Toc536375141)

[**CONFIGURACION DE DISPOSITIVOS** 3](#_Toc536375142)

[- **Configuración Básica** 3](#_Toc536375143)

[- **Configuracion del router P** 3](#_Toc536375144)

[- **Configuración del router PE1-UIO** 4](#_Toc536375145)

[DESARROLLO DE LA INTERFAZ 5](#_Toc536375146)

[**Clase conectorDB** 5](#_Toc536375147)

[**Clase Consulta** 6](#_Toc536375148)

[**Clase Logs** 6](#_Toc536375149)

[**Clase Registro** 7](#_Toc536375150)

[**Clase SSHConnector** 10](#_Toc536375151)

[**Clase JFRAME Inicio** 11](#_Toc536375152)

[**Clase JFRAME Administración** 14](#_Toc536375153)

# REQUISITOS

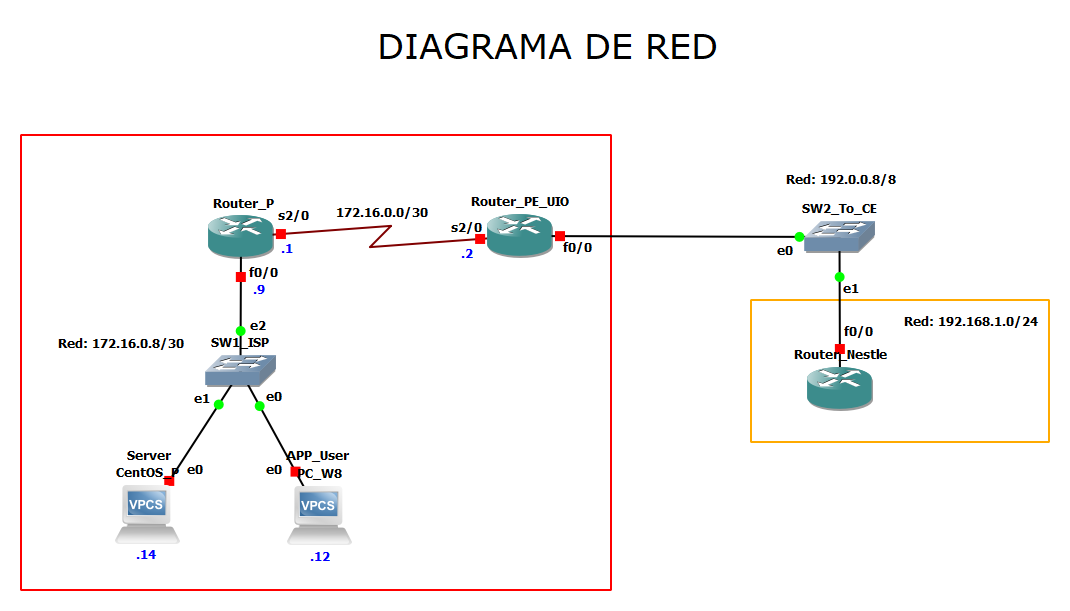
* GNS3 V1.2 o superiores (Opcional para simulaciones de red)
* NetBeans IDE 8.1

jdk 1.8 de java

* mysql-connector-java-5.1.47 (conector mysql)
* jsch-0.1.54 (Conector ssh)
* Oracle VM VirtualBox
* Windows 10 o 7
* Centos 7
* Router Cisco 2800 Series
* Switch Catalyst 2960 Series

# RED DEL PROVEEDOR DE INTERNET

## **DIAGRAMA DE RED**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DIRECCIONAMIENTO ISP** | | | | |
| **dispositivo** | **interfaz** | **Direccion IP** | **Masc.Subred** | **Gateway** |
| **P** | s0/2/0 | 172.16.0.1 | 255.255.255.252 | --- |
| lo0 | 1.1.1.1 | 255.255.255.255 | --- |
| f0/0 | 172.16.0.9 | 255.255.255.248 | --- |
| **PE1-UIO** | s0/2/0 | 172.16.0.2 | 255.255.255.252 | --- |
| lo0 | 2.2.2.2 | 255.255.255.255 | --- |
| **Server-centos** | eth0 | 172.16.0.14 | 255.255.255.248 | 172.16.0.9 |
| **PC-APP** | eth0 | 172.16.0.12 | 255.255.255.248 | 172.16.0.9 |

## **CONFIGURACION DE DISPOSITIVOS**

* **Configuración Básica**

la configuración básica en los dispositivos tanto el dispositivo PE como en el P permite dar algunos parametros de seguridad y habilitar la conexión ssh

enable

configure terminal

hostname PE1-UIO

ip domain-name cisco.com

line console 0

logging synchronous

exec-timeout 3 3

login local

exit

crypto key generate rsa

1024

line vty 0 15

logging synchronous

exec-timeout 3 3

login local

transport input ssh

exit

service password-encryption

banner motd #ACCESO AUTORIZADO SOLO A PERSONAL AUTORIZADO#

username admin privilege 15 secret admin

username monitoreo privilege 5 secret monitoreo

ip ssh version 2

ip ssh authentication-retries 3

ip ssh time-out 30

do wr

* **Configuracion del router P**

Se Configura las interfaces de acuerdo a lo preestablecido además se configura ospf para que detecte a los vecinos dentro del área 0, además de la interfaz virtual de loopback.

int s0/2/0

ip add 172.16.0.1 255.255.255.252

no shutdown

int lo0

ip add 1.1.1.1 255.255.255.255

int f0/0

ip add 172.16.0.9 255.255.255.248

no shut

exit

router ospf 1

network 172.16.0.0 0.0.0.3 area 0

network 172.16.0.4 0.0.0.3 area 0

network 172.16.0.8 0.0.0.7 area 0

network 1.1.1.1 0.0.0.0 area 0

exit

do wr

* **Configuración del router PE1-UIO**

Se configuro el router PE con el protocolo ospf para que detecte el vecino P y haya una conexión directa.

int s0/2/0

ip add 172.16.0.2 255.255.255.252

no shutdown

int lo0

ip add 2.2.2.2 255.255.255.255

exit

router ospf 1

network 172.16.0.0 0.0.0.3 area 0

network 2.2.2.2 0.0.0.0 area 0

exit

do wr

# DESARROLLO DE LA INTERFAZ

## **Clase conectorDB**

Esta clase se encargará de crear la conexion con la base de datos del servidor centos que esta en nuestra red(complemento obtenido de internet)

**public** **class** **ConectorDB** {

**private** **static** Connection conn;

**private** **static** **final** String driver = "com.mysql.jdbc.Driver";

**private** **static** **final** String user = "root";

**private** **static** **final** String password = "1234";

**private** **static** **final** String url = "jdbc:mysql://172.16.0.14:3306/respaldo";

**public** ConectorDB() {

conn = **null**;

**try** {

Class.forName(driver).newInstance();

conn =(Connection) DriverManager.getConnection(url, user, password);

**if** (conn != **null**) {

System.out.println("CONEXION ESTABLECIDA");

}

} **catch** (ClassNotFoundException | SQLException e) {

JOptionPane.showMessageDialog(**null**, "Falla en la conexión con el servidor, inténtelo más tarde");

System.out.println("Falla en la conexión con el servidor, inténtelo más tarde" + e);

JOptionPane.showMessageDialog(**null**, "Falla en la conexión con el servidor, inténtelo más tarde");

} **catch** (InstantiationException ex) {

Logger.getLogger(ConectorDB.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (IllegalAccessException ex) {

Logger.getLogger(ConectorDB.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

**public** Connection getConnection() {

**return** conn;

}

**public** void desconectar() {

**try** {

conn.close();

conn = **null**;

**if** (conn == **null**) {

System.out.println("CONEXION TERMINADA");

}

} **catch** (SQLException ex) {

Logger.getLogger(ConectorDB.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

}

## **Clase Consulta**

Esta clase se encarga de obtener el query correspondiente a la tabla correspondiente.

Tiene como parámetro de entrada la conexión de la base de datos y el nombre de la table a revisar el query y obtenemos como resultado un ResultSet similar a un buffer donde se almacena los registros de las tablas para hacer validaciones en el programa

**public** **class** **Consulta** {

**public** ResultSet getResultSetTabla(Connection con,String tabla){

ResultSet rs=**null**;

**try** {

String SQL = "SELECT \* FROM "+tabla+"";

Statement stmt = con.createStatement();

rs = stmt.executeQuery(SQL);

**return** rs;

} **catch** (SQLException ex) {

Logger.getLogger(Consulta.class.getName()).log(Level.SEVERE, **null**, ex);

}

**return** rs;

}

}

## **Clase Logs**

La clase logs permite registrar en un archivo de texto los logs de nuestra aplicación.

La función recibe como parametron el mensaje indicado, por ejemplo cuando se accede al Sistema segun la historia de usuario 1 entonces se verifica la fecha y hora del Sistema y se escribe en el archivo

**public** **class** **Logs** {

**public** Logs(String mensaje) {

*//Obtener la hora y fecha actual*

*//Fecha*

Date sistFecha = **new** Date();

SimpleDateFormat formatoFecha = **new** SimpleDateFormat("dd-MMM-YYYY");

*//Hora*

Date sistHora = **new** Date();

SimpleDateFormat formatoHora = **new** SimpleDateFormat("hh:mm:ss a");

Calendar hoy = Calendar.getInstance();

*//Guarda el registro de ingreso del usuario*

**try**{

File archivo;

FileWriter escribir;

PrintWriter linea;

archivo = **new** File("Registro.txt");

**if**(!archivo.exists()){

archivo.createNewFile();

}

escribir = **new** FileWriter(archivo, **true**);

linea = **new** PrintWriter(escribir);

*//escribir fecha y hora*

linea.println(formatoFecha.format(sistFecha)+" "+String.format(formatoHora.format(sistHora), hoy));

*//escribir usuario que ingreso al sistema*

linea.println(mensaje);

linea.close();

escribir.close();

}**catch**(Exception e){

JOptionPane.showMessageDialog(**null**, "Error al guaradar el registro");

}

}

}

## **Clase Registro**

La clase registro permite al usuario registrar en la base de datos los datos correspondiente a la creacion de vlan, vrf y direccionamiento de acuerdo a su campo y valor .

Además se encuentra la función modificar registro, que permite modificar un campo correspondiente. Dicha función recibira la conexión con la base, nombre de la table, el campo/valor identificador, el campo/valor a modificar.

**public** **class** **Registro** {

Statement stmt;

PreparedStatement psInsertar;

**public** void registrarVLAN(Connection con, int N\_vlan, String ciudad, String empresa) {

**try** {

String SQL = "insert into vlan (noRegistro,N\_vlan,ciudad,empresa)" + " values(?,?,?,?)";

stmt = con.createStatement();

psInsertar = con.prepareStatement(SQL);

psInsertar.setInt(1, 0);

psInsertar.setInt(2, N\_vlan);

psInsertar.setString(3, ciudad);

psInsertar.setString(4, empresa);

psInsertar.executeUpdate();

psInsertar.close();

stmt.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** void registrarVRF(Connection con, String nombreVRF, int N\_vlan, String ciudad, String empresa) {

String SQL = "insert into VRF (nombreVRF,empresa,ciudad,vlan)" + " values(?,?,?,?)";

**try** {

stmt = con.createStatement();

psInsertar = con.prepareStatement(SQL);

psInsertar.setString(1, nombreVRF);

psInsertar.setString(2, empresa);

psInsertar.setString(3, ciudad);

psInsertar.setInt(4, N\_vlan);

psInsertar.executeUpdate();

psInsertar.close();

stmt.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** void registrarDireccionamiento(Connection con, String dir\_red, String submask\_red, String empresa, String ciudad, int vlan, String nombreVRF, String nombrePE, int enlaces) {

String SQL = "insert into direccionamiento (dir\_red,submask\_red,empresa,ciudad,vlan,nombreVRF,nombrePE,enlaces)" + " values(?,?,?,?,?,?,?,?)";

**try** {

stmt = con.createStatement();

psInsertar = con.prepareStatement(SQL);

psInsertar.setString(1, dir\_red);

psInsertar.setString(2, submask\_red);

psInsertar.setString(3, empresa);

psInsertar.setString(4, ciudad);

psInsertar.setInt(5, vlan);

psInsertar.setString(6, nombreVRF);

psInsertar.setString(7, nombrePE);

psInsertar.setInt(8, enlaces);

psInsertar.executeUpdate();

psInsertar.close();

stmt.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** void modificartabla(Connection con, String tabla, String campoID, String registroID, String campoMOD, String registroMOD) {

String SQL = "update " + tabla + " set " + campoMOD + " = ?" + " where " + campoID + " = ?";

**try** {

stmt = con.createStatement();

psInsertar = con.prepareStatement(SQL);

**if** (Administracion.isNumeric(registroMOD)) {

psInsertar.setInt(1, Integer.parseInt(registroMOD));

} **else** {

psInsertar.setString(1, registroMOD);

}

psInsertar.setString(2, registroID);

psInsertar.executeUpdate();

psInsertar.close();

stmt.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

## **Clase SSHConnector**

La clase SSHConnector permitirá crear la conexión SSH con el router PE y ejecutar los commandos en el router

* La función **connect** pide parámetros de User con el cual esta ingresando la aplicación el user tiene un privilegio el cual le da la características a los permisos de configuración del router, el parámetro PE con el cual obtendremos la dirección ip de la interfaz de conexión via ssh, y el puerto que es un entero constante igual a 22;
* La función **executeCommand** pide como parámetro un String que es la configuración a realizar en el PE (una sola línea simulando los saltos o ENTERs como ’\n’)

**public** **class** **SSHConnector** {

**private** **static** **final** String ENTER\_KEY = "\n";

**private** Session session;

**public** void connect(User user, PE pe, int port)

**throws** JSchException, IllegalAccessException {

**if** (**this**.session == **null** || !**this**.session.isConnected()) {

JSch jsch = **new** JSch();

**this**.session = jsch.getSession(user.getUser\_router(), pe.getDireccionIP(), port);

**this**.session.setPassword(user.getPass\_router());

*// Parametro para no validar key de conexion.*

**this**.session.setConfig("StrictHostKeyChecking", "no");

**this**.session.connect();

System.out.println("nueva sesion");

} **else** {

**throw** **new** IllegalAccessException("Sesion SSH ya iniciada.");

}

}

**public** **final** String executeCommand(String comandos)

**throws** IllegalAccessException, JSchException, IOException {

**if** (**this**.session != **null** && **this**.session.isConnected()) {

String[] lineasC=comandos.split(ENTER\_KEY);

*// Abrimos un canal SSH. Es como abrir una consola.*

Channel channelExec = (Channel) **this**.session.openChannel("shell");

OutputStream out= channelExec.getOutputStream();

PrintStream commander = **new** PrintStream(out, **true**);

channelExec .setOutputStream(System.out, **true**);

channelExec.connect();

**for**(String z:lineasC){

commander.println(z);

}

commander.close();

String salida="";

**if**(!comandos.equals("")){

InputStream in = channelExec.getInputStream();

*// Obtenemos el texto impreso en la consola.*

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(in));

String linea;

**while** ((linea = reader.readLine()) != **null**) {

salida=salida+linea+"\n";

}

}

*// Cerramos el canal SSH.*

channelExec.disconnect();

*// Retornamos el texto impreso en la consola.*

**return** salida;

} **else** {

**throw** **new** IllegalAccessException("No existe sesion SSH iniciada.");

}

}

*/\*\**

*\* Cierra la sesión SSH.*

*\*/*

**public** **final** void disconnect() {

**this**.session.disconnect();

System.out.println("SESION TERMINADA");

}

**public** Session getSesion(){

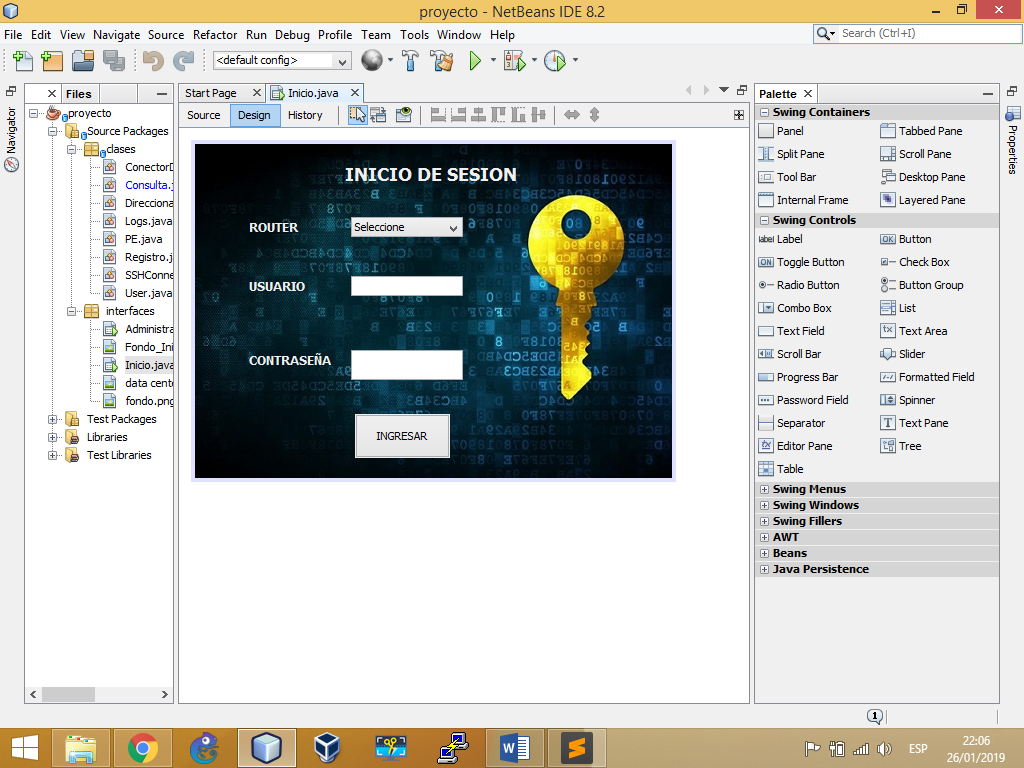
**return** **this**.session;

}

}

## **Clase JFRAME Inicio**

Esta clase muestra el primer formulario de inicio de sesión para acceder al sistema de aprovisionamiento (se tomara en cuenta las principales funciones dentro de la clase).:



* La función **jButton1ActionPerformed** permite la ejecución del código implementado al hacer clic en el botón ingresar para poder mostrar el segundo formulario
* La función **inicializaComboBoxPE** permite asignar los enrutadores asignados en la base de datos para ello crea la conexión con la base de datos y hace consulta de los PE y asignar el nombre al JComboBox
* La función **validaruser** permite verifica el usuario ingresado, implementar la función devuelve un entero

1. La función devuelve 1 entonces el usuario ingresado y la contraseña son correctos
2. La función devuelve 0 si el usuario ingresado existe pero su contraseña es incorrecta
3. La función devuelve -1 si el usuario no existe

**private** void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {*//GEN-FIRST:event\_jButton1ActionPerformed*

**try** {

**if** (!jComboBox1.getSelectedItem().toString().equals("Seleccione")) {

**if** (!(jTextField1.getText().equals("")) && !(jTextField2.getText().equals(""))) {

**switch** (validaruser(jTextField1.getText(), jTextField2.getText())) {

**case** 1:

sshConnector = **new** SSHConnector();

**try** {

System.out.println(pe.toString());

sshConnector.connect(user, pe, PUERTOSSH);

String inicioshow = sshConnector.executeCommand("");

System.out.println(inicioshow);

Logs logs = **new** Logs("Ingreso del usuario: " + jTextField1.getText());

adminForm = **new** Administracion();

adminForm.setPE(pe);

adminForm.setUser(user);

adminForm.setConectorDB(conDB);

adminForm.setSSHConector(sshConnector);

adminForm.setjLabelRouter(pe.getNombre());

adminForm.setVisible(**true**);

**this**.setVisible(**false**);

sshConnector.disconnect();

*//registro del usuario que ingresa*

} **catch** (JSchException E) {

JOptionPane.showMessageDialog(**null**, "problemas de conexion SSH, intentelo mas tarde");

}

**break**;

**case** 0:

JOptionPane.showMessageDialog(**null**, "Contraseña incorrecta inténtelo de nuevo");

**break**;

**case** -1:

JOptionPane.showMessageDialog(**null**, "El usuario ingresado no existe");

**break**;

**default**:

JOptionPane.showMessageDialog(**null**, "ERROR");

**break**;

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Falta información, inténtelo de nuevo");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Falta información, inténtelo de nuevo");

}

} **catch** (Exception ex) {

Logger.getLogger(Inicio.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

**public** void inicializaComboBoxPE() {

**try** {

conDB = **new** ConectorDB();

Connection reg = conDB.getConnection();

consulta = **new** Consulta();

ResultSet rs = consulta.getResultSetTabla(reg, "providerEdge");

**if** (rs != **null**) {

**while** (rs.next()) {

pes.add(**new** PE(rs.getString("nombrePE"), rs.getString("ciudad"), rs.getString("direccionIP"), rs.getString("int\_vlan")));

jComboBox1.addItem(rs.getString("nombrePE"));

}

} **else** {

System.out.println("ERROR");

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** int validaruser(String user, String pass) {

int op = -1;

**try** {

Connection reg = conDB.getConnection();

consulta = **new** Consulta();

ResultSet rs = consulta.getResultSetTabla(reg, "user");

**if** (rs != **null**) {

**while** (rs.next()) {

**if** (user.equals(rs.getString("usuario"))) {

**if** (pass.equals(rs.getString("contrasena"))) {

**this**.user = **new** User(user, pass, Integer.parseInt(rs.getString("privilegio")));

rs.close();

**return** 1;

}

op = 0;

}

}

} **else** {

System.out.println("ERROR");

}

rs.close();

} **catch** (Exception e) {

e.printStackTrace();

}

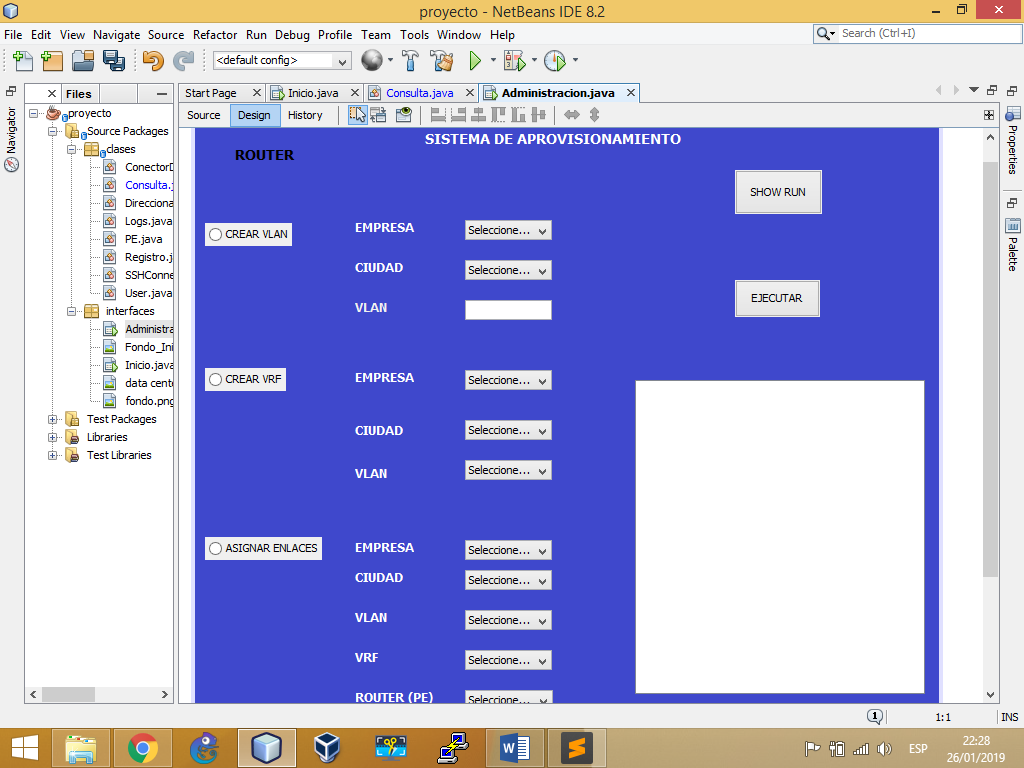
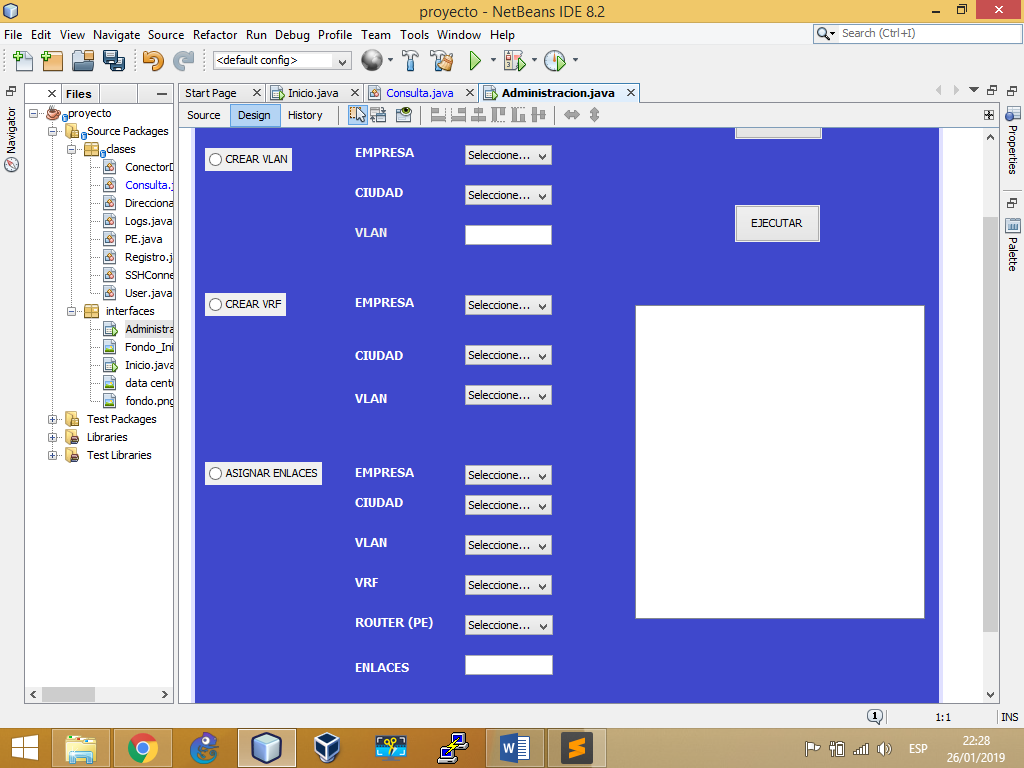
**return** op;

}

## **Clase JFRAME Administración**

Esta clase es la base de nuestro programa donde se centran las historias de usuario 2,3 y 4. Permitirá crear las vlans, vrf y asignar direccionamiento por vlan.

* Al hacer clic en cada **jOptionButton** se habilitara cada historia de usuario correspondiente ocultando los campos extras y actualizando los ResultSet de las tablas correspondientes y junto a ello los campos de elección como los **jComboBox**
* La función **asignaCampoJcombo** permite asignar los ítems a un jCombo cuyos parametros de entrada son la conexión,el jComboBox donde se desea asignar los items, el nombre de la tabla y el campo con los registros que se desea asignar al jCmoboBox
* La función **isNumeric** recibe como parámetro un string el cual retorna un boolean, si al tratar de convertirlo a número no presenta algún error retorna verdadero caso contrario retorna falso.
* La función **asignarEnlace** permitirá asignar las direcciones ip a las subinterfaces de vlan correspondiente a la empresa, esta función retorna un string con los datos de dirección de red, submascara de red, el numero de enlace y además un formato extra que indicara cuando se incrementa la mascara cuando la cantidad de enlaces sobremasa la cantidad de enlaces permitido por la mascara anterior
* La función **registrarvlanBD\_R** permite registrar en la base de datos y escribir los comandos en el router acerca de la historia de usuario 2 (registrar VLAN), esta función recibe como parámetro los campos de numero entero de vlan, la ciudad donde se desea registrar la vlan, la empresa que contrato la vlan, y una opción extra que permitirá registra los logs y mostrar mensajes de salida de acuerdo a las diferentes posibilidades presentadas en la historia de usuario 2
* La función **registrarVrfRoutBD** permite registrar en la base de datos y configurar en el PE los parámetros necesarios para la historia de usuario 3, la función recibe como parámetro el numero de vlan, la empresa y la ciudad el cual se configurar automáticamente con los datos previamente registrados
* La función **jButton2ActionPerformed** es la mas importante porque realizara todos los cambios de acuerdo a la asignación dada por los jOptionsButton, es decir realizará la acción seleccionada como crear vlan, crear vrf o asignar enlaces



**public** void asignaCampoJcombo(JComboBox jCombo, String nombreTabla, String nombreCampo) {

jCombo.removeAllItems();

jCombo.addItem("Seleccione...");

**try** {

Connection reg = conDB.getConnection();

consulta = **new** Consulta();

ResultSet rs = consulta.getResultSetTabla(reg, nombreTabla);

**if** (rs != **null**) {

**while** (rs.next()) {

jCombo.addItem(rs.getString(nombreCampo));

}

} **else** {

System.out.println("ERROR");

}

rs.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** boolean isNumeric(String cadena) {

boolean resultado;

**try** {

Integer.parseInt(cadena);

resultado = **true**;

} **catch** (NumberFormatException excepcion) {

resultado = **false**;

}

**return** resultado;

}

**public** String asignarEnlace() {

Direccionamiento dir = **null**;

int numero\_Enlaces\_Existentes = 0;

String lineassh = "";

String mensaje\_de\_aceptacion = "no";

**try** {

Connection reg = conDB.getConnection();

consulta = **new** Consulta();

ResultSet rs = consulta.getResultSetTabla(reg, "direccionamiento");

**if** (rs != **null**) {

**while** (rs.next()) {

**if** (jComboBox6.getSelectedItem().toString().equals(rs.getString("empresa"))) {

dir = **new** Direccionamiento(rs.getString("dir\_red"), rs.getString("submask\_red"), rs.getString("empresa"), rs.getString("ciudad"), rs.getInt("vlan"), rs.getString("nombreVRF"), rs.getString("nombrePE"), rs.getInt("enlaces"));

**break**;

}

}

} **else** {

System.out.println("ERROR");

}

rs.close();

**if** (dir == **null**) {

mensaje\_de\_aceptacion = "no";

} **else** {

mensaje\_de\_aceptacion = "si";

numero\_Enlaces\_Existentes = dir.getEnlaces();

}

*//La mascara del ISP puede ser 191.128.0.0/12*

*//obtencion de los 3 primeros octetos de la subred*

numeroEnlacesEmpresa = Integer.parseInt(jTextField2.getText());

vlanEmpresa = Integer.parseInt(jComboBox8.getSelectedItem().toString());

ArrayList<Integer> listaBinarioVlan = **new** ArrayList<Integer>();

*//funcion devuelve un arreglo de la vlan en binario*

listaBinarioVlan = binario\_list(vlanEmpresa, 13);

int primer\_octeto = 10;

int segundo\_octeto = listaBinarioVlan.get(0) + 2 \* listaBinarioVlan.get(1) + 4 \* listaBinarioVlan.get(2) + 8 \* listaBinarioVlan.get(3) + 16 \* listaBinarioVlan.get(4);

int tercer\_octeto = 128 \* listaBinarioVlan.get(5) + 64 \* listaBinarioVlan.get(6) + 32 \* listaBinarioVlan.get(7) + 16 \* listaBinarioVlan.get(8)

+ 8 \* listaBinarioVlan.get(9) + 4 \* listaBinarioVlan.get(10) + 2 \* listaBinarioVlan.get(11) + listaBinarioVlan.get(12);

int cuarto\_octeto = 0;

*//Mascara para la subred*

int primero\_mascara = 255, segundo\_mascara = 255, tercero\_mascara = 255;

int cuarto\_mascara = 255;

*//int numero\_Enlaces\_Existentes = dir.getEnlaces();//Valor sacado de la base de datos*

**for** (int i = 1; i < 9; i++) {

**if** ((numero\_Enlaces\_Existentes + 2) < Math.pow(2, i)) {

cuarto\_mascara = 256 - (int) Math.pow(2, i);

i = 9;

}

}

System.out.println(cuarto\_mascara);

int cuarto\_mascara\_temporal = cuarto\_mascara;

**for** (int i = 1; i < 9; i++) {

**if** ((numeroEnlacesEmpresa + 2 + numero\_Enlaces\_Existentes) < Math.pow(2, i)) {

cuarto\_mascara\_temporal = 256 - (int) Math.pow(2, i);

i = 9;

}

}

System.out.println(cuarto\_mascara\_temporal);

int nuevo\_numero\_de\_enlaces = numeroEnlacesEmpresa + numero\_Enlaces\_Existentes;

System.out.println("");

**if** ((nuevo\_numero\_de\_enlaces + 2) > 256) {

**return** "exedeEnlaces";

} **else** {

**if** (cuarto\_mascara\_temporal == cuarto\_mascara) {

**if** (mensaje\_de\_aceptacion.equals("si")) {

cuarto\_mascara = cuarto\_mascara\_temporal;

System.out.println("se agrego mas enlaces dentro de la misma subred");

**return** primer\_octeto + "." + segundo\_octeto + "." + tercer\_octeto + "." + cuarto\_octeto

+ "," + primero\_mascara + "." + segundo\_mascara + "." + tercero\_mascara + "."

+ cuarto\_mascara + "," + nuevo\_numero\_de\_enlaces + "," + "agregoensubred";

} **else** {

cuarto\_mascara = cuarto\_mascara\_temporal;

System.out.println("se agrego una nueva subred diferente no existente a la base de datos");

**return** primer\_octeto + "." + segundo\_octeto + "." + tercer\_octeto + "." + cuarto\_octeto

+ "," + primero\_mascara + "." + segundo\_mascara + "." + tercero\_mascara + "."

+ cuarto\_mascara + "," + nuevo\_numero\_de\_enlaces;

}

} **else** {

System.out.println("");

System.out.println(cuarto\_mascara + "," + cuarto\_mascara\_temporal);

System.out.println("");

**if** (mensaje\_de\_aceptacion.equals("si")) {

cuarto\_mascara = cuarto\_mascara\_temporal;

int closeSelected = JOptionPane.showConfirmDialog(**null**, "¿Desea asignar un nueva\n subred Si/No?", "Alerta", JOptionPane.YES\_NO\_OPTION);

**if** (closeSelected == JOptionPane.YES\_OPTION) {

System.out.println("se cambio la mascara cuando habia enlaces anteriores");

**return** primer\_octeto + "." + segundo\_octeto + "." + tercer\_octeto + "." + cuarto\_octeto

+ "," + primero\_mascara + "." + segundo\_mascara + "." + tercero\_mascara + "."

+ cuarto\_mascara + "," + nuevo\_numero\_de\_enlaces + "," + "cambiodemascara";

} **else** {

**return** "";

}

} **else** {

System.out.println("");

cuarto\_mascara = cuarto\_mascara\_temporal;

**return** primer\_octeto + "." + segundo\_octeto + "." + tercer\_octeto + "." + cuarto\_octeto

+ "," + primero\_mascara + "." + segundo\_mascara + "." + tercero\_mascara + "."

+ cuarto\_mascara + "," + nuevo\_numero\_de\_enlaces;

}

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** lineassh;

}

**public** void registrarvlanBD\_R(int vlan, String ciudad, String empresa, int option) {

Connection reg = conDB.getConnection();

Registro r = **new** Registro();

r.registrarVLAN(reg, vlan, ciudad, empresa);

**try** {

PE PE\_SSH = **null**;

sshConnector = **new** SSHConnector();

**for** (PE pe : Inicio.pes) {

**if** (ciudad.equals(pe.getCiudad())) {

PE\_SSH = **new** PE(pe.getNombre(), pe.getCiudad(), pe.getDireccionIP(), pe.getInt\_vlan());

**break**;

}

}

System.out.println(PE\_SSH.toString());

sshConnector.connect(user, PE\_SSH, Inicio.PUERTOSSH);

String result = sshConnector.executeCommand("conf t\n" + "int " + PE\_SSH + "." + vlan + "\n" + "encapsulation dot1q " + vlan + "\n" + "description VLAN-DE-EMPRESA-" + empresa + "\n" + "do wr \n\n");

jTextArea1.setText(result);

sshConnector.disconnect();

**if** (option == -1) {

JOptionPane.showMessageDialog(**null**, "VLAN " + vlan + " creada exitosamente en la ciudad " + ciudad + " para la empresa " + empresa);

Logs logs = **new** Logs("VLAN " + vlan + " creada exitosamente en la ciudad " + ciudad + " para la empresa " + empresa);

} **else** **if** (option == 1) {

JOptionPane.showMessageDialog(**null**, "Se asigno una VLAN nueva " + vlan + " en la ciudad " + ciudad + " para la empresa " + empresa);

Logs logs = **new** Logs("Se asigno una VLAN nueva " + vlan + " en la ciudad " + ciudad + " para la empresa " + empresa);

}

} **catch** (JSchException e) {

e.printStackTrace();

} **catch** (IllegalAccessException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (IOException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

**public** void registrarVrfRoutBD(int vlan, String empresa, String ciudad) {

Connection reg = conDB.getConnection();

Registro r = **new** Registro();

String nombreVRF = "VRF\_" + empresa + "\_" + vlan + "";

System.out.println(nombreVRF);

r.registrarVRF(reg, nombreVRF, vlan, ciudad, empresa);

**try** {

PE PE\_SSH = **null**;

sshConnector = **new** SSHConnector();

**for** (PE pe : Inicio.pes) {

**if** (ciudad.equals(pe.getCiudad())) {

PE\_SSH = **new** PE(pe.getNombre(), pe.getCiudad(), pe.getDireccionIP(), pe.getInt\_vlan());

**break**;

}

}

System.out.println(PE\_SSH.toString());

sshConnector.connect(user, PE\_SSH, Inicio.PUERTOSSH);

String result = sshConnector.executeCommand("config t\nip vrf " + nombreVRF + "\nrd 1:" + vlan + "\nroute-target export 1:" + vlan + "\nroute-target import 1:" + vlan + "\nint " + PE\_SSH.getInt\_vlan() + "." + vlan + "\nip vrf forwarding " + nombreVRF + "\ndo wr\n" + "" + "\ndo wr\n\n ");

jTextArea1.setText(result);

sshConnector.disconnect();

JOptionPane.showMessageDialog(**null**, "se creo" + nombreVRF + " para la empresa " + empresa + " con vlan " + vlan);

Logs logs = **new** Logs("se creo" + nombreVRF + " para la empresa " + empresa + " con vlan " + vlan);

} **catch** (JSchException e) {

e.printStackTrace();

} **catch** (IllegalAccessException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (IOException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

**private** void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {*//GEN-FIRST:event\_jButton2ActionPerformed*

*//la opcion de crear vlan*

**if** (jRadioButton1.isSelected()) {

*//valida si los campos son correctamente llenados y seleccionados*

**if** (!(jComboBox1.getSelectedItem().toString().equals("Seleccione...")) && !(jTextField1.getText().equals("")) && !(jComboBox5.getSelectedItem().toString().equals("Seleccione..."))) {

*//verifica si la vlan es un numero*

**if** (isNumeric(jTextField1.getText())) {

*//verifica si la vlan es un numero entre 1-4096*

**if** ((Integer.parseInt(jTextField1.getText()) >= 1) && (Integer.parseInt(jTextField1.getText()) <= MAXVLAN)) {

**switch** (existeVlanCiudad()) {

**case** 0:

JOptionPane.showMessageDialog(**null**, "no hay vlan disponibles en la ciudad" + jComboBox5.getSelectedItem().toString());

**break**;

**case** -1:

**if** (!poseeVLANempresa()) {

registrarvlanBD\_R(Integer.parseInt(jTextField1.getText()), jComboBox5.getSelectedItem().toString(), jComboBox1.getSelectedItem().toString(), -1);

} **else** {

int closeSelected = JOptionPane.showConfirmDialog(**null**, "¿desea que se cree otra vlan para la empresa?", "Alerta", JOptionPane.YES\_NO\_OPTION);

**if** (closeSelected == JOptionPane.YES\_OPTION) {

registrarvlanBD\_R(Integer.parseInt(jTextField1.getText()), jComboBox5.getSelectedItem().toString(), jComboBox1.getSelectedItem().toString(), -1);

} **else** {

System.out.println("NO SE CREO VLAN");

}

}

**break**;

**case** 1:*//existe vlan dentro de la ciudad*

System.out.println("vlan existe se asignara otra");

JOptionPane.showMessageDialog(**null**, "VLAN " + jTextField1.getText() + " existe");

**if** (!poseeVLANempresa()) {

**for** (int i = 1; i <= MAXVLAN; i++) {

**for** (int v : nvlan) {

**if** (i != v) {

registrarvlanBD\_R(i, jComboBox5.getSelectedItem().toString(), jComboBox1.getSelectedItem().toString(), 1);

i = (MAXVLAN + 1);

**break**;

}

}

}

} **else** {

System.out.println("la vlan existe y la empresa tiene una vlan quiere otra");

**for** (int i = 1; i <= MAXVLAN; i++) {

**for** (int v : nvlan) {

**if** (i != v) {

int closeSelected = JOptionPane.showConfirmDialog(**null**, "¿desea que se cree otra vlan para la empresa?", "Alerta", JOptionPane.YES\_NO\_OPTION);

**if** (closeSelected == JOptionPane.YES\_OPTION) {

registrarvlanBD\_R(i, jComboBox5.getSelectedItem().toString(), jComboBox1.getSelectedItem().toString(), 1);

System.out.println("se creo vlan");

} **else** {

System.out.println("NO SE CREO VLAN");

}

i = (MAXVLAN + 1);

**break**;

}

}

}

}

**break**;

**default**:

**break**;

}

} **else** {

JOptionPane.showMessageDialog(**null**, "El número de VLAN debe de ser entre 1 a 4096 vuelva a intentarlo.");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "El número de VLAN debe de ser entre 1 a 4096 vuelva a intentarlo.");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Campos incompletos, vuelva a intentar.");

}

} **else** **if** (jRadioButton2.isSelected()) {

**if** (!(jComboBox2.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox3.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox4.getSelectedItem().toString().equals("Seleccione..."))) {

String nombreVRF = obtenerVRF(jComboBox9, jComboBox2);

**if** (nombreVRF.equals("no\_existe")) {

registrarVrfRoutBD(Integer.parseInt(jComboBox3.getSelectedItem().toString()), jComboBox2.getSelectedItem().toString(), jComboBox4.getSelectedItem().toString());

} **else** {

JOptionPane.showMessageDialog(**null**, "existe VRF" + nombreVRF + "PARA la empresa " + jComboBox2.getSelectedItem().toString() + " seleccionada");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Falta información, inténtelo de nuevo");

}

} **else** **if** (jRadioButton3.isSelected()) {

**if** (!(jComboBox6.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox7.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox8.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox9.getSelectedItem().toString().equals("Seleccione...")) && !(jComboBox10.getSelectedItem().toString().equals("Seleccione...")) && !(jTextField2.getText().equals(""))) {

String empresa = jComboBox6.getSelectedItem().toString();

String ciudad = jComboBox7.getSelectedItem().toString();

int vlan = Integer.parseInt(jComboBox8.getSelectedItem().toString());

String vrf = jComboBox9.getSelectedItem().toString();

String router = jComboBox10.getSelectedItem().toString();

int noEnlace = 0;

String ipSecundaria = "";

**try** {

String enlace = asignarEnlace();

System.out.println(enlace);

**if** (enlace.equals("")) {

JOptionPane.showMessageDialog(**null**, "No se asigno direccionamiento");

}**else** **if**(enlace.equals("exedeEnlaces")){

JOptionPane.showMessageDialog(**null**, "Exede el numero de Enlaces disponible para la Vlan se enviara un correo al Administrador");

}

**else** {

String[] a = enlace.split(",");

String ip = "";

noEnlace = Integer.parseInt(a[2]);

char[] w = a[0].toCharArray();

int cont = 0;

String ultimobit = "";

**for** (char x : w) {

**if** (cont == 3) {

ultimobit = ultimobit + x;

} **else** {

ip = ip + x;

}

**if** (x == '.') {

cont++;

}

}

**if** (a.length < 4) {

ip = ip + Integer.toString((Integer.parseInt(ultimobit) + 1));

} **else** {

**if** (a[3].equals("cambiodemascara")) {

ipSecundaria = ip + Integer.toString(Integer.parseInt(ultimobit) + Integer.parseInt(a[2]) + 1);

}

}

System.out.println(ip);

PE PE\_SSH = **null**;

sshConnector = **new** SSHConnector();

**for** (PE pe : Inicio.pes) {

**if** (ciudad.equals(pe.getCiudad())) {

PE\_SSH = **new** PE(pe.getNombre(), pe.getCiudad(), pe.getDireccionIP(), pe.getInt\_vlan());

**break**;

}

}

String result = "";

sshConnector = **new** SSHConnector();

sshConnector.connect(user, PE\_SSH, Inicio.PUERTOSSH);

**if** (ipSecundaria.equals("")) {

result = sshConnector.executeCommand("config t\ninterface " + PE\_SSH.getInt\_vlan() + "." + vlan + "\nip address " + ip + " " + a[1] + "\n");

} **else** {

result = sshConnector.executeCommand("config t\ninterface " + PE\_SSH.getInt\_vlan() + "." + vlan + "\nip address " + ipSecundaria + " " + a[1] + " secondary\n");

}

jTextArea1.setText(result);

sshConnector.disconnect();

Connection reg = conDB.getConnection();

Registro r = **new** Registro();

**if** (a.length < 4) {

JOptionPane.showMessageDialog(**null**, "Se asigna exitosamente la subred: "+a[0]+"/"+a[1]+" de acuerdo a "+a[2]+" enlaces");

r.registrarDireccionamiento(reg, a[0], a[1], empresa, ciudad, vlan, vrf, PE\_SSH.getNombre(), noEnlace);

} **else** {

**if** (a[3].equals("agregoensubred")) {

JOptionPane.showMessageDialog(**null**, "la subinterface "+PE\_SSH.getInt\_vlan()+"."+vlan+" no se asignara subred pero se actualiza la cantidad de enlaces");

r.modificartabla(reg, "direccionamiento", "dir\_red", a[0], "enlaces", a[2]);

} **else** **if** (a[3].equals("cambiodemascara")) {

JOptionPane.showMessageDialog(**null**, "Se creo una nueva subred");

r.modificartabla(reg, "direccionamiento", "dir\_red", a[0], "enlaces", a[2]);

r.modificartabla(reg, "direccionamiento", "dir\_red", a[0], "submask\_red", a[1]);

}

}

}

} **catch** (JSchException ex) {

JOptionPane.showMessageDialog(**null**, "Falla en la conexión con el dispositivo PE, inténtelo más tarde");

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (IllegalAccessException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (IOException ex) {

Logger.getLogger(Administracion.class.getName()).log(Level.SEVERE, **null**, ex);

} **catch** (Exception e) {

System.out.println("ERROR");

}

} **else** {

JOptionPane.showMessageDialog(**null**, "Falta información, inténtelo de nuevo");

}

} **else** {

System.out.println("DEBE SELECCIONAR ALGO");

}

**for** (int z : nvlan) {

System.out.println(z);

}

nvlan.clear();

}