**MANUAL TÉCNICO**

Contenido

[REQUISITOS 2](#_Toc505282877)

[RED DEL PROVEEDOR DE INTERNET 2](#_Toc505282878)

[Package basicgraph 3](#_Toc505282879)

[INTERFAZ.JAVA 3](#_Toc505282880)

[CLASES Y LIBRERIAS 4](#_Toc505282881)

[CLASE INTERFAZ 4](#_Toc505282882)

[CREAR VRF 4](#_Toc505282883)

[**private void CrearVrf(){** 4](#_Toc505282884)

[**private void EditarVrf(CE obj, int operacion){** 6](#_Toc505282885)

[ELIMINAR VRF 7](#_Toc505282886)

[**private void EliminarVrf(){** 7](#_Toc505282887)

[**private String ip(String origen){** 9](#_Toc505282888)

[CONECTIVIDAD 9](#_Toc505282889)

[**private void ejecutarActionPerformed(java.awt.event.ActionEvent evt) {** 9](#_Toc505282890)

[OBSERVAR RED 11](#_Toc505282891)

[**private void redActionPerformed(java.awt.event.ActionEvent evt** 11](#_Toc505282892)

[CLASES 13](#_Toc505282893)

[CLASE CE 13](#_Toc505282894)

[**public class CE implements Comparable {** 13](#_Toc505282895)

[CLASE LEERVRF 14](#_Toc505282896)

[**public class LeerVrf {** 14](#_Toc505282897)

[CLASE PE 16](#_Toc505282898)

[**public class PE implements Comparable {** 16](#_Toc505282899)

[CLASE PROVEDOR 17](#_Toc505282900)

[**public class Provedor implements Comparable {** 17](#_Toc505282901)

[CLASE SSHCONECTOR 18](#_Toc505282902)

[**public String executeCommand(String ip,int actividad, CE vrf)** 18](#_Toc505282903)

[CLASE GRAPH 20](#_Toc505282904)

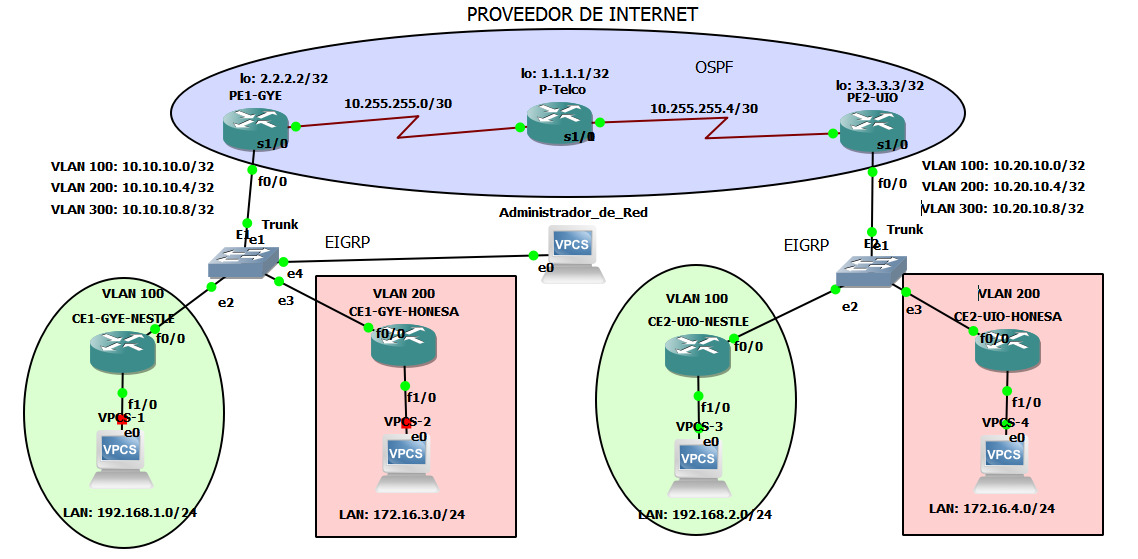
[**public void saveToHtml(String fileName, String title) throws IOException, URISyntaxException {** 20](#_Toc505282905)

[RECOMENDACIONES 22](#_Toc505282906)

# REQUISITOS

* GNS3 V1.2 o superiores (Opcional para simulaciones de red)
* NetBeans IDE 8.1
* jdk 1.8 de java
* Windows 10 o 7

# RED DEL PROVEEDOR DE INTERNET



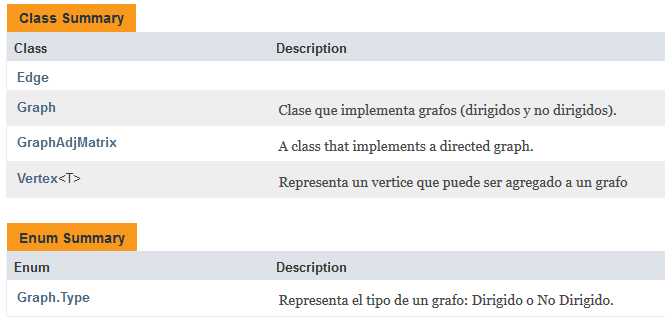
Para la administración de la red el administrador debe conectarse en cualquiera de los dos switchs que están conectados a PE1-GYE Y el que esta PE2-UIO la vlan de administración exclusivamente al puerto f0/4 que está configurado para la administración

DESARROLLO DE LA INTERFAZ

Nuestra aplicación consta de dos paquetes, el primero es para crear el grafo y el segundo es para programar nuestra aplicación



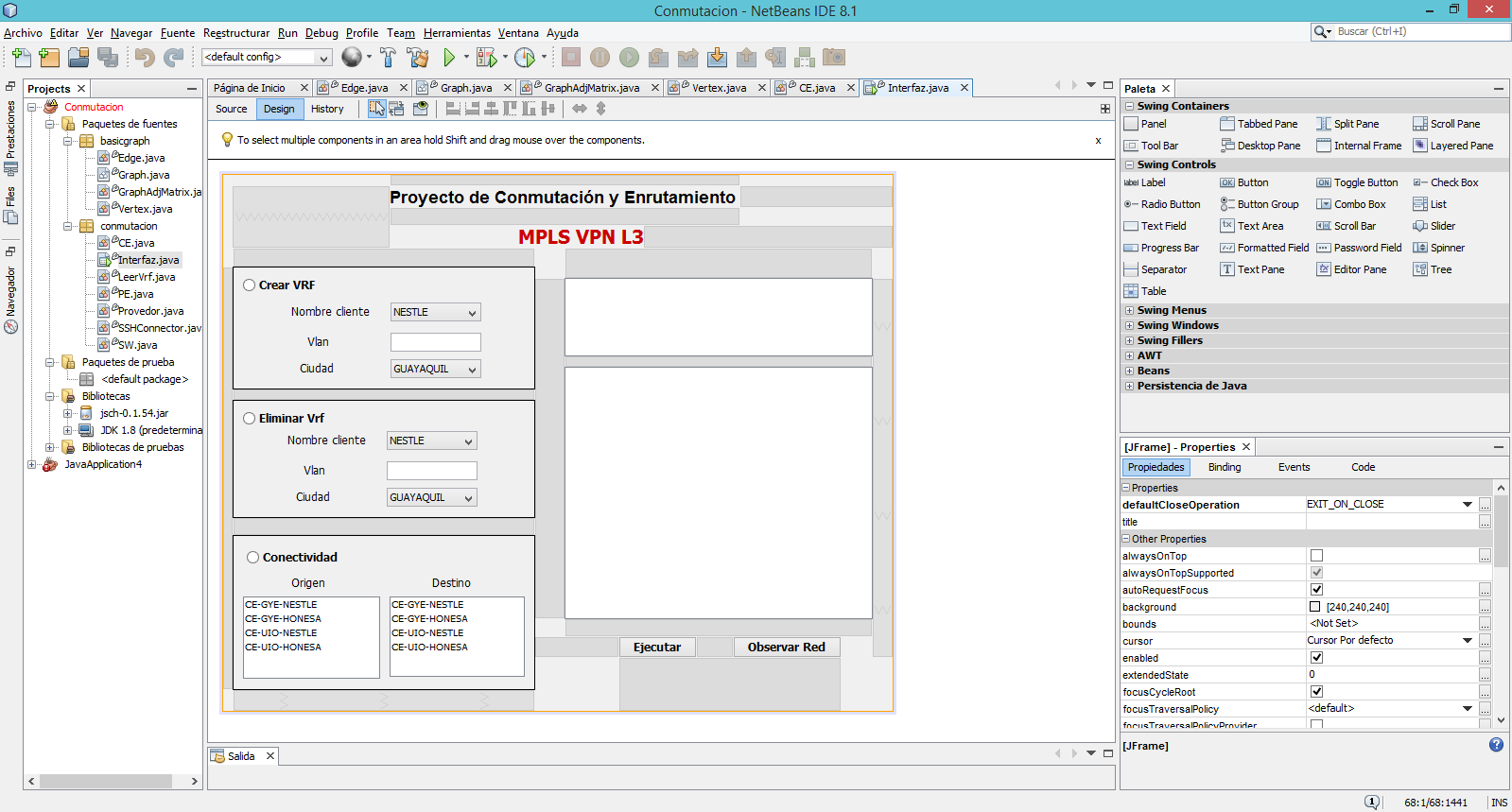
## Package basicgraph



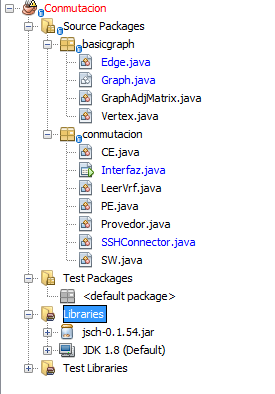
## INTERFAZ.JAVA

En la clase interfaz.java ubicamos los objetos necesarios para poder ubicar los campos y así crear la interfaz de nuestro aplicativo. Dicha interfaz contara con tres partes que son:

* Crear vrf: crea la vlans para la vrf
* Eliminar vrf: permite la eliminación de una vlans según su vrf
* Conectividad: permite hacer un ping entre los router de los clientes
* Observar la red: Es un botón que nos permitirá revisar en una página html la red implementada, sucursales activas, etc.



# CLASES Y LIBRERIAS



Para poder ejecutar nuestro servicio es necesario de la Librería jsch la cual es usada para el acceso a la red mediante ssh

# CLASE INTERFAZ

## CREAR VRF

En la parte de crear una Vrf debemos tener en consideración los usuarios en una matriz en la ciudad de Quito y Guayaquil por eso es necesario considerar las siguientes validaciones

//Este método es para poder crear Vrf validando todos los parámetros

### **private void CrearVrf(){**

texto.setText("");

String tex="Se creara la VRF en la VLAN ";

tex=tex+vlan.getText();

tex=tex+" al cliente "+(String) cliente.getSelectedItem()+" en la ciudad de "+ (String) ciudad.getSelectedItem();

if(vlan.getText().isEmpty()){

texto.setText("Error: Ingrese todos los parametros ");

}else {

try {

int vlan1=Integer.parseInt(vlan.getText());

int acum=0;

String text="";

LeerVrf vrf=new LeerVrf();

LinkedList<CE> listaVrf=vrf.cargarVrf();

for (CE obj : listaVrf) {

acum=acum+1;

System.out.println(obj);

**//validaciones**

boolean igualCiudad=ciudad.getSelectedItem().equals(obj.ciudad);

boolean igualCliente=cliente.getSelectedItem().equals(obj.cliente);

boolean igualVlan=vlan1==obj.vlan;

if(igualCiudad && igualCliente ){

texto.setText(texto.getText()+"En la ciudad "+ obj.ciudad+ " ya existe vrf para el cliente "+ obj.cliente);

} else {

if(igualCiudad && igualVlan ){

texto.setText(texto.getText()+"En la ciudad "+ obj.ciudad+ "ya existe la Vlan"+ obj.vlan);

}else{

if(acum==listaVrf.size()){

if (texto.getText().isEmpty()){

CE crear=new CE();

crear.setCiudad((String) ciudad.getSelectedItem());

crear.setCliente((String) cliente.getSelectedItem());

crear.setVlan(vlan1);

texto.setText(texto.getText()+"Se Creara con exito la Vrf en la "+ ciudad.getSelectedItem()+ " para el cliente "+ cliente.getSelectedItem() +" la Vlan " +vlan1);

vrf.guardarVrf(listaVrf,crear,"crear");

EditarVrf(crear,1);

}

}

}

}

}

} catch (NumberFormatException nfe){

texto.setText("Error: Ingrese una correcta Vlan ");

}

}

}

// si cumple todas las validaciones se logra editar la Vrf

### **private void EditarVrf(CE obj, int operacion){**

String ip = "",ip1="";

String origen=ipPe(obj.ciudad);

System.out.println(origen);

SSHConnector sshConnector = new SSHConnector(); //Eliminar Vrf

try {

sshConnector.connect(USERNAME, PASSWORD, origen, PORT);

String result = sshConnector.executeCommand(ip1,operacion,obj);

texto1.setText(result);

sshConnector.disconnect();

System.out.println(result);

} catch (JSchException ex) {

System.out.println("No se pudo ejecutar los comandos");

} catch (IllegalAccessException ex) {

System.out.println("No se pudo conectar");

} catch (IOException ex) {

System.out.println("Error de comando ingresado");

} catch (InterruptedException ex) {

System.out.println("La sesion fue interrumpida");

}

}



## ELIMINAR VRF

// este método es para eliminar la Vrf

### **private void EliminarVrf(){**

String tex="Se Eliminara la VRF en la VLAN ";

tex=tex+vlan1.getText();

tex=tex+" al cliente "+(String) cliente1.getSelectedItem()+" en la ciudad de "+ (String) ciudad1.getSelectedItem();

if(vlan1.getText().isEmpty()){

texto.setText("Error: Ingrese todos los parametros ");

}else {

try {

texto.setText("");

int vlan=Integer.parseInt(vlan1.getText());

int acum=0;

LeerVrf vrf=new LeerVrf();

LinkedList<CE> listaVrf=vrf.cargarVrf();

for (CE obj : listaVrf) {

acum=acum+1;

System.out.println(obj);

System.out.println("");

**//validaciones**

boolean igualCiudad=ciudad1.getSelectedItem().equals(obj.ciudad);

boolean igualCliente=cliente1.getSelectedItem().equals(obj.cliente);

boolean igualVlan=vlan==obj.vlan;

if(igualCiudad && igualCliente && igualVlan ){

texto.setText(texto.getText()+"Se eliminara con exito la Vrf en la de "+ obj.ciudad+ " para el cliente "+ obj.cliente +" la Vlan " +obj.vlan);

vrf.guardarVrf(listaVrf,obj,"eliminar");

EditarVrf(obj,2);

} else {

if(acum==listaVrf.size()){

if (texto.getText().isEmpty()){

texto.setText("No existe la Vrf para ese cliente");

}

}

}

}

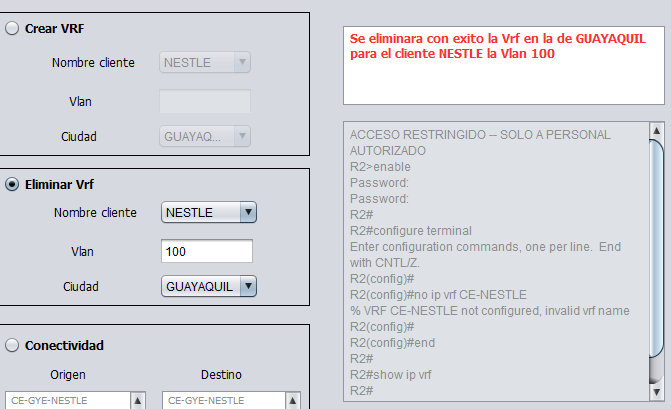
acum=0;

} catch (NumberFormatException nfe){

texto.setText("Error: Ingrese una correcta Vlan ");

}

Si todos los parámetros cumplen con las validaciones la vlan se eliminara con éxito, y nos mostrara la configuración realizada en el router



// este método valida el origen y destino, para poder ingresar al Router mediante una ip asignada en los puertos

### **private String ip(String origen){**

String ip = "";

if(origen=="CE-GYE-NESTLE"){

ip="10.10.10.2";

}else if(origen=="CE-UIO-NESTLE"){

ip="10.20.10.2";

}else if(origen=="CE-GYE-HONESA"){

ip="10.10.10.6";

}else if(origen=="CE-UIO-HONESA"){

ip="10.20.10.6";

}

return ip;

}

## CONECTIVIDAD

// Este método es para validar la opción elegida por el usuario

### **private void ejecutarActionPerformed(java.awt.event.ActionEvent evt) {**

**//Crear Vrf**

if(crearVrf.isSelected()){

CrearVrf();

}else if(eliminarVrf.isSelected()){

EliminarVrf();

**//Conectividad**

}else if(conectividad.isSelected()){

String ip = "",ip1="";

String origen=this.origen.getSelectedValue();

String destino=this.destino.getSelectedValue();

ip=ip(origen);

System.out.println(ip);

ip1=ip(destino);

System.out.println(ip1);

try {

SSHConnector sshConnector = new SSHConnector();

sshConnector.connect(USERNAME, PASSWORD, ip, PORT);

texto.setText("Se realiza ping entre la "+ip + " del cliente "+ this.origen.getSelectedValue() +" a la ip " +ip1 + " del cliente "+ this.destino.getSelectedValue());

String result = sshConnector.executeCommand(ip1,3,null);

texto1.setText(result);

sshConnector.disconnect();

} catch (JSchException ex) {

System.out.println("No se pudo ejecutar los comandos");

} catch (IllegalAccessException ex) {

System.out.println("No se pudo conectar");

} catch (IOException ex) {

System.out.println("Error de comando ingresado");

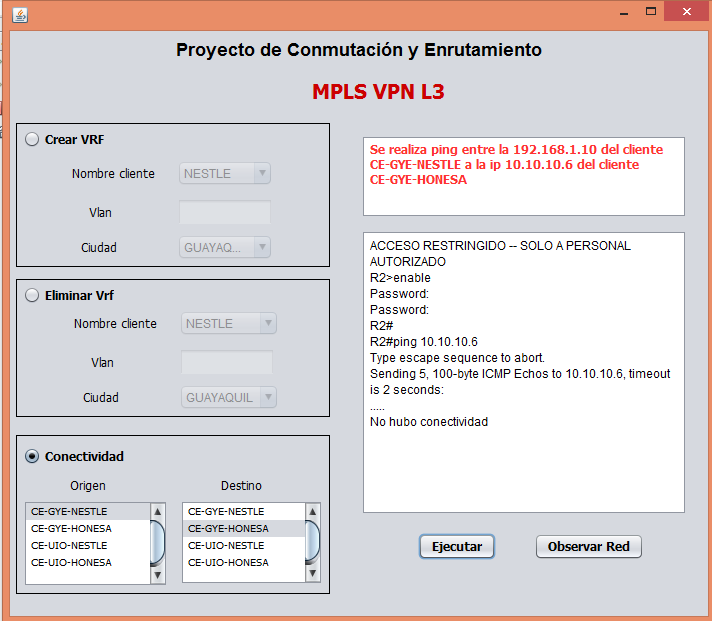
} catch (InterruptedException ex) {

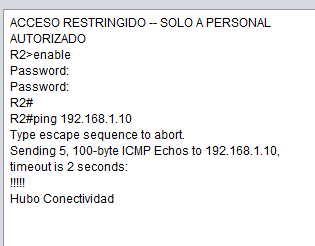
System.out.println("La sesion fue interrumpida");

}

}

Probamos conectividad entre dos PC clientes





## OBSERVAR RED

//Este método es para generar el grafo de la red, asignando vértices

**private void redActionPerformed(java.awt.event.ActionEvent evt) {**

Graph mpls = new Graph(Graph.Type.DIRECTED) {};

LeerVrf vrf=new LeerVrf();

LinkedList<CE> listaCE=vrf.cargarVrf();

**//crear objetos**

CE r1 = new CE();

Provedor p = new Provedor();

PE p1 = new PE();

PE p2 = new PE();

SW sw = new SW();

SW sw1 = new SW();

sw.setCiudad("QUITO");

sw1.setCiudad("GUAYAQUIL");

p1.setCiudad("QUITO");

p2.setCiudad("GUAYAQUIL");

**//crear Vertices de la red**

Vertex isp = new Vertex(p);

Vertex pe1 = new Vertex(p1);

Vertex pe2 = new Vertex(p2);

Vertex swv1 = new Vertex(sw);

Vertex swv2 = new Vertex(sw1);

**//agregar Vertices a la red**

mpls.addVertex(isp);

mpls.addVertex(pe1);

mpls.addVertex(pe2);

mpls.addVertex(swv1);

mpls.addVertex(swv2);

**//adyacencias**

mpls.addEdge(isp, pe1);

mpls.addEdge(isp, pe2);

mpls.addEdge(pe1, swv1);

mpls.addEdge(pe2, swv2);

DefaultListModel modelo = new DefaultListModel();

Iterator<CE> iterado = listaCE.iterator();

while(iterado.hasNext()){

CE x = iterado.next();

Vertex ve = new Vertex(x);

mpls.addVertex(ve);

if(x.ciudad.equals("QUITO")){

mpls.addEdge(swv1, ve);

}else{

mpls.addEdge(swv2, ve);

}

modelo.addElement(x.getCiudad());

}

**//validaciones**

try {

mpls.saveToHtml("Grafo.html", "Grafo de Relacion");

} catch (IOException ex) {

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

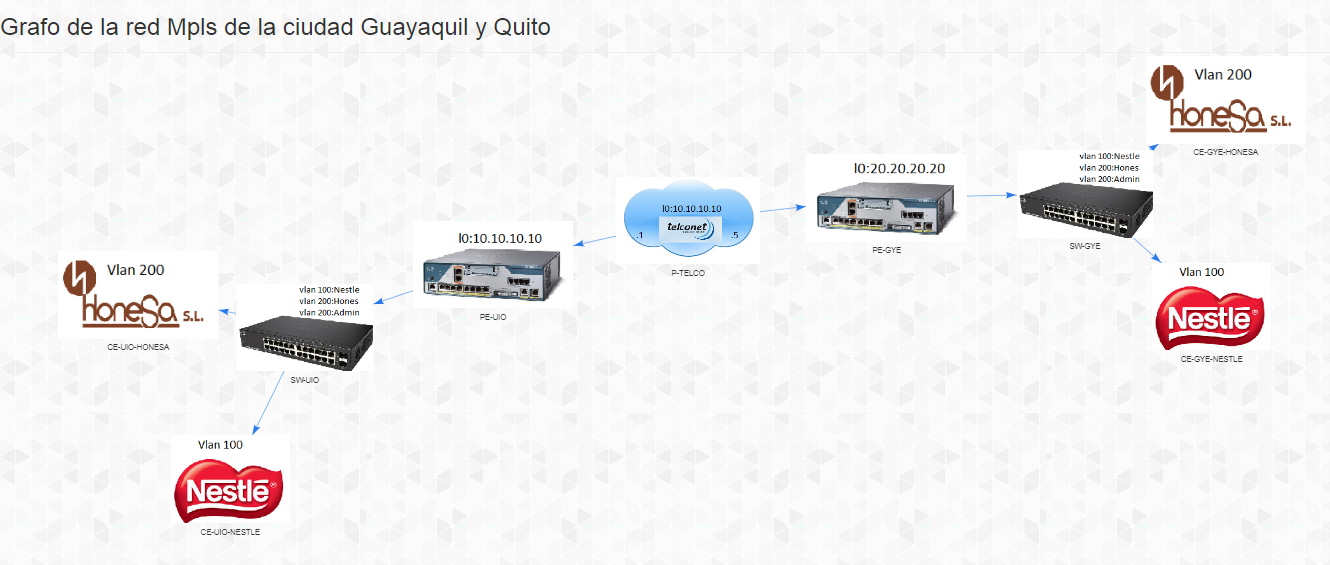
} catch (URISyntaxException ex) {

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

}

}

Al presionar el botón presionar red, nos muestra la topología de la red actual de nuestra estructura mpls en una pagina web



# CLASES

## CLASE CE

//Este clase es para designar los parámetros que tiene todo Router Cliente

### **public class CE implements Comparable {**

String cliente;

int vlan;

String ciudad;

public CE() {

}

public CE(String cliente, int vlan, String ciudad) {

this.cliente = cliente;

this.vlan = vlan;

this.ciudad = ciudad;

}

public String getCliente() {

return cliente;

}

public void setCliente(String cliente) {

this.cliente = cliente;

}

public int getVlan() {

return vlan;

}

public void setVlan(int vlan) {

this.vlan = vlan;

}

public String getCiudad() {

return ciudad;

}

public void setCiudad(String ciudad) {

this.ciudad = ciudad;

}

@Override

public String toString() {

String city="";

if(ciudad.equals("QUITO")){

city="UIO";

}else{

city="GYE";

}

return "CE-"+city+"-" + cliente ;

}

@Override

public int compareTo(Object t) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

## CLASE LEERVRF

En esta clase es para asignada para poder cargar y editar las vrf de la base de datos, esta base de datos esta alojada dentro de archivos del programa

### **public class LeerVrf {**

private boolean band=true;

/\*\*

\* @return LinkedList<Vrf> retornamos una lista de Vrf c

\*/

#### **public LinkedList<CE> cargarVrf() {**

FileReader fr;

BufferedReader br = null;

String linea;

String valores[];

CE c1;

LinkedList<CE> vrf=new LinkedList<>();

try {

fr = new FileReader("vrf.txt");

br = new BufferedReader(fr);

**//abrimos el archivo y recorremos linea por linea**

while ((linea = br.readLine()) != null) {

valores = linea.split(",", 5);

c1 = new CE();

**//ingresamos todos los valores a un objeto Cuenta c1**

c1.ciudad=valores[0];

c1.cliente=valores[2];

c1.vlan=Integer.parseInt(valores[1]);

vrf.add(c1);

}

**//manegamos excepciones**

} catch (FileNotFoundException ex) {

System.out.println("El archivo no existe");

} catch (IOException ex) {

System.out.println("Error de Lectura");

}

return vrf;

}

#### **public void guardarVrf(LinkedList<CE> lista,CE vrf,String tipo) {**

FileWriter fw;

BufferedWriter bw;

PrintWriter pw;

try {

fw = new FileWriter("vrf.txt");

bw = new BufferedWriter(fw);

pw = new PrintWriter(bw);

int acum=0;

for (CE obj : lista) {

if(tipo=="crear"){

if(acum==0){

pw.write(obj.ciudad+",");

pw.write(obj.vlan+",");

pw.write(obj.cliente);

pw.write("\n");

pw.write(vrf.ciudad+",");

pw.write(vrf.vlan+",");

pw.write(vrf.cliente);

pw.write("\n");

acum=acum+1;

}else{

pw.write(obj.ciudad+",");

pw.write(obj.vlan+",");

pw.write(obj.cliente);

pw.write("\n");

}

}else{

if(obj!=vrf){

pw.write(obj.ciudad+",");

pw.write(obj.vlan+",");

pw.write(obj.cliente);

pw.write("\n");

}

}

}

pw.close();

bw.close();

System.out.println(vrf);

} catch (IOException ex) {

System.out.println("ERROR: No se puedo acceder al archivo");

}

}

## CLASE PE

//Este clase es para designar los parámetros que tiene todo Router Provider Edge

### **public class PE implements Comparable {**

String ciudad;

public PE() {

}

public PE(String ciudad) {

this.ciudad = ciudad;

}

public String getCiudad() {

return ciudad;

}

public void setCiudad(String ciudad) {

this.ciudad = ciudad;

}

@Override

public String toString() {

String city="";

if(ciudad.equals("QUITO")){

city="UIO";

}else{

city="GYE";

}

return "PE-"+city;

}

@Override

public int compareTo(Object t) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

## CLASE PROVEDOR

//Este clase es para designar los parámetros que tiene todo Router Provedor

### **public class Provedor implements Comparable {**

String provedor;

public String getProvedor() {

return provedor;

}

public void setProvedor(String provedor) {

this.provedor = provedor;

}

@Override

public String toString() {

return "P-TELCO";

}

@Override

public int compareTo(Object t) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

## CLASE SSHCONECTOR

En esta clase validamos los commandos que vamos a ingresar al router mediante ssh

### **public String executeCommand(String ip,int actividad, CE vrf)**

throws IllegalAccessException, JSchException, IOException, InterruptedException {

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

// 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);

synchronized(commander){

channelExec.setOutputStream(System.out, true);

channelExec.connect();

//Crear Vrf

if(actividad==1){

commander.println("enable");

commander.println("cisco");

commander.println("configure terminal");

commander.println("ip vrf CE-"+vrf.cliente);

commander.println("rd 1:"+vrf.vlan);

commander.println("route-target export 1:"+vrf.vlan);

commander.println("route-target import 1:"+vrf.vlan);

commander.println("exit");

commander.println("interface f0/0."+vrf.vlan);

commander.println("encapsulation dot1Q "+vrf.vlan);

commander.println("ip vrf forwarding CE-"+vrf.cliente);

String ipvrf="10.";

if(vrf.cliente.equals("NESTLE")){

if(vrf.ciudad.equals("GUAYAQUIL")){

ipvrf=ipvrf+"10.10.1";

}else{

ipvrf=ipvrf+"20.10.1";

}

}else{

if(vrf.ciudad.equals("GUAYAQUIL")){

ipvrf=ipvrf+"10.10.5";

}else{

ipvrf=ipvrf+"20.10.5";

}

}

commander.println("ip address "+ipvrf+" 255.255.255.0");

commander.println("end");

commander.println("configure terminal");

commander.println("router eigrp 1");

commander.println("address-family ipv4 vrf CE-"+vrf.cliente);

commander.println("network 10.10.10.0");

commander.println("no auto-summary");

commander.println("redistribute bgp 1");

commander.println("default-metric 100000 1 255 1 1500");

commander.println("autonomous-system 10");

commander.println("exit-address-family");

commander.println("end");

commander.println("configure terminal");

commander.println("router bgp 1");

commander.println("address-family ipv4 vrf CE-"+vrf.cliente);

commander.println("redistribute eigrp 10");

commander.println("exit-address-family");

commander.println("end");

commander.println("show ip vrf");

commander.println("end");

//Eliminar Vrf

}else if (actividad==2){

commander.println("enable");

commander.println("cisco");

commander.println("configure terminal");

commander.println("no ip vrf CE-"+vrf.cliente);

commander.println("end");

commander.println("show ip vrf");

//conectividad

}else {

commander.println("enable");

commander.println("cisco");

commander.println("ping " +ip);

commander.wait(400);

// commander.notify();

}

// channelExec.wait();

}

## CLASE GRAPH

En esta clase generamos el Html para observer la red

### **public void saveToHtml(String fileName, String title) throws IOException, URISyntaxException {**

Iterator<Vertex> vertexIterator;

Iterator<Edge> edgeIterator;

Vertex vertex;

Edge edge;

boolean flag;

StringBuilder sb = new StringBuilder();

FileWriter fw;

BufferedWriter bw;

String s = "";

int i;

fw = new FileWriter(fileName);

bw = new BufferedWriter(fw);

sb.append("<!DOCTYPE html>\n\n<html lang=\"en\">\n <head>\n <meta charset=\"utf-8\" />\n");

sb.append(String.format(" <title>%s</title>\n", title));

sb.append(" <script type=\"text/javascript\" src=\"https://cdnjs.cloudflare.com/ajax/libs/vis/4.13.0/vis.min.js\"></script>\n");

sb.append("<meta charset=\"utf-8\"> <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

sb.append("<link rel=\"stylesheet\" href=\"http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css\">");

sb.append("<script src=\"https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js\"></script>");

sb.append("<script src=\"http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js\"></script>");

sb.append(" <link href=\"vis/vis.css\" rel=\"stylesheet\" type=\"text/css\"/>\n");

sb.append(" <script type=\"text/javascript\">\n" +

" function draw() {\n" +

" // create some nodes\n");

sb.append("var nodes = [");

vertexIterator = adjList.keySet().iterator();

while ( vertexIterator.hasNext() ) {

vertex = vertexIterator.next();

i = vertexIndices.get(vertex);

// sb.append(String.format("{id: %d, \"label\": \"hola\", \"group\": 1}", i));

sb.append(String.format("{id: %d, shape: 'image', image: 'image/%s.png', label:\"%s\"}", i,vertex.getContent().toString(), vertex.getContent().toString()));

sb.append(vertexIterator.hasNext() ? ",\n" : "");

}

sb.append("];\n");

// create some edges

sb.append("var edges = [\n");

vertexIterator = adjList.keySet().iterator();

while (vertexIterator.hasNext()) {

flag = false;

vertex = vertexIterator.next();

edgeIterator = adjList.get(vertex).iterator();

i = vertexIndices.get(vertex);

while ( edgeIterator.hasNext() ) {

flag = true;

edge = edgeIterator.next();

sb.append(String.format("{\"from\": %d, \"to\": %d}",i, vertexIndices.get(edge.getDestination())));

sb.append(edgeIterator.hasNext() ? ",\n" : "");

}

if(flag)

sb.append(vertexIterator.hasNext() ? ",\n" : "");

}

sb.append("];\n");

sb.append(" // create a network\n" +

" var container = document.getElementById('grafo');\n" +

" var data = {\n" +

" nodes: nodes,\n" +

" edges: edges\n" +

" };\n" +

" var options = {\n" +

" nodes: {\n" +

" shape: 'dot',\n" +

" size: 80\n" +

" },\n" +

" physics: {\n" +

" forceAtlas2Based: {\n" +

" gravitationalConstant: -100,\n" +

" centralGravity: 0.0001,\n" +

" springLength:250,\n" +

" springConstant: 0.18\n" +

" },\n" +

" maxVelocity: 146,\n" +

" solver: 'forceAtlas2Based',\n" +

" timestep: 0.35,\n" +

" stabilization: {iterations: 150}\n" +

" },edges: {arrows:{to:{enable:true}}}\n" +

" };\n" +

" var network = new vis.Network(container, data, options);\n" +

"\n" +

" }\n" +

" </script>\n" +

" <script src=\"../../googleAnalytics.js\"></script>");

fw.write(sb.toString());

fw.write(" </head>\n <body onload=\"draw()\" background=\"wall.png\">");

// fw.write(String.format("<h1>%s</h1>", this.type == Graph.Type.DIRECTED ? "Grafo Dirigido" : "Grafo No Dirigido"));

fw.write("<div class=\"page-header\">\n" +

" <h1>Grafo de la red Mpls de la ciudad Guayaquil y Quito</h1>\n" +

"</div>");

// fw.write(String.format("<h2> Vertices: %d Arcos: %d </h2>", getNumVertices(), getNumEdges()));

fw.write(" <div id=\"grafo\" style=\"position: absolute; top: 60px; bottom: 20px; width: 100%;\"></div>\n </body>\n</html>");

fw.close();

Desktop.getDesktop().browse(new URI(fileName));

}

# RECOMENDACIONES

Leer detalladamente el manual del usuario y observar cómo está compuesta nuestra red