Escuela Superior Politécnica del Litoral

Conmutación y Enrutamiento | 2T 2018

MANUAL TÉCNICO



Contenido

[Overview 3](#_Toc536375365)

[Introducción 3](#_Toc536375366)

[Objetivo y alcances del sistema 3](#_Toc536375367)

[Normas, políticas y procedimientos 3](#_Toc536375368)

[Recursos de software 3](#_Toc536375369)

[Python 3](#_Toc536375370)

[Paramiko 3](#_Toc536375371)

[MySQL 4](#_Toc536375372)

[GNS3 4](#_Toc536375373)

[Recursos de Simulación 4](#_Toc536375374)

[Recursos de Hardware 4](#_Toc536375375)

[Switch Catalyst 2960 4](#_Toc536375376)

[Enrutador cisco 2811 4](#_Toc536375377)

[Caso de estudio 5](#_Toc536375378)

[Diagrama y Direccionamiento de la Red 5](#_Toc536375379)

[Código Fuente 6](#_Toc536375380)

[Login.py 6](#_Toc536375381)

[Remote.py 7](#_Toc536375382)

[SelectorLocal.py 8](#_Toc536375383)

[ConfiguracionBasica.py 9](#_Toc536375384)

[DireccionamientoLocal.py 10](#_Toc536375385)

[SelectorCredenciales.py 11](#_Toc536375386)

[AnadirUsuario.py 12](#_Toc536375387)

[Plantilla.py 12](#_Toc536375388)

[SelectorConfiguracionMPLS.py 13](#_Toc536375389)

[ConfigurarCE.py 14](#_Toc536375390)

[ConfigurarPE2.py 15](#_Toc536375391)

[Funciones2.py 17](#_Toc536375392)

[Database.py 27](#_Toc536375393)

[Soporte 27](#_Toc536375394)

# Overview

## **Introducción**

MPLStudio es una aplicación desarrollado para Windows que permite realizar conexiones remotas por medio del protocolo SSH. Con esta aplicación se puede automatizar la configuración de dispositivos intermedios de una red MPLS-VPN en capa 3 para un ISP. Dentro de esta configuración se incluyen aspectos como enrutamiento dinámico, redistribución de rutas, etiquetado de paquetes bajo LDP, configuración de rutas estáticas y creación de VRFs para clientes del ISP.

## **Objetivo y alcances del sistema**

* Automatizar la configuración de dispositivos intermediarios en una red de datos.
* Ingresar a un dispositivo P, PE, CE que se encuentra activo en la red para establecer una sesión remota.
* Configurar los enrutadores dependiendo de su funcionalidad en la red MPLS.
* Facilitar el trabajo del administrador de redes mediante ingreso de parámetros de la red por teclado.

## **Normas, políticas y procedimientos**

El ingreso a la aplicación está controlado por un login inicial el cual se valida con las credenciales de usuarios previamente autorizados y registrados en la base de datos.

Se espera que en todo momento la conexión hacia el dispositivo a configurar sea estable, por lo cual se recomienda cerrar cualquier otro tipo de aplicación que realice conexiones de este tipo durante el uso del aplicativo.

Todos los campos de ingreso de datos dentro de la aplicación están debidamente validados.

# Recursos de software

Dentro de esta sección se describen las librerías y frameworks de terceros usados durante la implementación de MPLStudio.

## **Python**

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

## **Paramiko**

Paramiko es una implementación de Python (2.7, 3.4+) del protocolo SSHv2, que proporciona funcionalidad de cliente y servidor. Mientras que aprovecha una extensión de Python C para la criptografía de bajo nivel (criptografía), Paramiko en sí es una interfaz pura de Python alrededor de los conceptos de redes SSH.

## **SQLITE**

MySQLITE es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

## **GNS3**

GNS3 es un software utilizado por cientos de miles de ingenieros de redes a nivel mundial para emular, configurar, probar y solucionar problemas de redes virtuales y reales. GNS3 le permite ejecutar una pequeña topología que consta de solo unos pocos dispositivos en su computadora portátil, a aquellos que tienen muchos dispositivos alojados en múltiples servidores o incluso alojados en la nube.

# Recursos de Simulación

Las pruebas realizadas sobre MPLStudio en su etapa de desarrollo fueron llevadas a cabo bajo el programa GNS3, el cual permite la simulación de sistemas de redes creando interfaces virtuales lo cual facilita al programador a testear configuraciones reales en un ambiente controlado.

El IOS usado para simular los enrutadores corresponde a c7200-adventerprisek9-mz.153-3.XB12 el cual representa a los enrutadores 7200 de Cisco. Es importante resaltar que en esta versión de IOS los enrutadores no cuentan con Modo Usuario por lo cual para es necesario tomar correctivos en el código al momento de usar otra versión del IOS.

# Recursos de Hardware

## **Switch Catalyst 2960**

Soportan voz, video, datos y acceso seguro. Disponen de 24 puertos 10/100 más dos puertos SFP. Capacidad de configurar LAN virtuales, función Power over Ethernet que permite implementar fácilmente nuevas funciones como comunicaciones por voz e inalámbricas sin necesidad de realizar nuevas conexiones. Seguridad integrada.

## **Enrutador cisco 2811**

Memoria RAM de 256 MB instalados, soportando hasta 768 MB, memoria FLASH de 64 MB instalados soportando hasta 256 MB. Utiliza IPSec como protocolo de transporte, protección firewall, cifrado del hardware, soporte de MPLS, diseño modular, criptografía de 128 bits, asistencia técnica VPN, filtrado de URL, cifrado de 256 bits y cumple con las normas de la IEEE 802.3af.

# Caso de estudio

## **Diagrama y Direccionamiento de la Red**

|  |
| --- |
|  |
| **Ilustración 1:** Diagrama de RED |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dispositivo | Dirección red/CIDR | Interfaz | Dirección IP | Mascara de Subred | Puerta de Enlace predeterminado |
| P\_Matriz | 20.255.255.0/30 | F1/0 | 20.255.255.1 | 255.255.255.252 | N/A |
| 1.1.1.1/32 | Lo0 | 1.1.1.1 | 255.255.255.255 | N/A |
| PE\_GYE | 20.255.255.0/30 | F1/0 | 20.255.255.2 | 255.255.255.252 | N/A |
| 20.20.20.0/30 | F0/0 | 20.20.20.1 | 255.255.255.252 | N/A |
| 2.2.2.2/32 | Lo0 | 2.2.2.2 | 255.255.255.255 | N/A |
| CE\_NORTE | 20.20.20.0/30 | F0/0 | 20.20.20.2 | 255.255.255.252 | N/A |
| 192.168.2.1/24 | F1/0 | 192.168.2.2 | 255.255.255.252 | N/A |
| CLIENTE\_GYE | 192.168.2.1/24 | NIC | 192.168.2.1 | 255.255.255.0 | 192.168.2.2 |

**Tabla 1:** Tabla de Direccionamiento

# Código Fuente

En esta sección se describirán las distintas funciones y su implementación dentro del proyecto.

### **Login.py**

|  |
| --- |
| **Nombre de archivo:** login.py  **Nombre de función:** showRemoteWindow()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** ingresar()  **def** showRemoteWindow(self):  self.remoteWindow = QtWidgets.QDialog()    self.ui = Ui\_Remote()  self.ui.setupUi(self.remoteWindow)  self.remoteWindow.show()  **Nombre de archivo:** login.py  **Nombre de función:** showRemoteWindow()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla. Actualmente este código no se ejecuta ya que la conexión por puerto COM esta deshabilitado en esta versión de MPLStudio.  **Método de llamado:** ingresar()  **def** showlocalWindow(self):  self.localWindow = QtWidgets.QDialog()    self.ui = Ui\_local()  self.ui.setupUi(self.localWindow)  self.localWindow.show()  **Nombre de archivo:** login.py  **Nombre de función:** ingresar()  **Descripción:** Esta función verifica los datos ingresados en el cuadro de texto de login para la aplicación.Realiza conexión con la base de datos bajo SQL y una vez que los datos estén validados muestra la pantalla correspondiente al tipo de conexión. Por defecto en esta versión del aplicativo automáticamente entra a showRemoteWindow()  **Método de llamado:** self.btn\_ingresar.clicked()  **def** ingresar(self):  username = self.txt\_usuario.text()  password = self.txt\_contrasena.text()    connection = sqlite3.connect(**"login.db"**)  result = connection.execute(**"SELECT \* FROM USERS WHERE** **USERNAME = ? AND CONTRASENA = ?"**, (username, password))  **if** (len(result.fetchall()) >0):  print(**"Usuario encontrado ! "**)  loginWindow.close()  **if** self.chk\_remote.checkState():  self.showRemoteWindow()  **else**:  self.showlocalWindow()  **else**:  print(**"Usuario no encontrado !"**) ctypes.windll.user32.MessageBoxW(0, **"Credenciales**  **invalidas"**,  **"Error"**,0)  connection.close()  **Nombre de archivo:** login.py  **Nombre de función:** MAIN  **Descripción:** Este es el llamado a la acción MAIN que se ejecuta al momento de abrir la aplicación. Carga laventana de login  **Método de llamado: Automático.**  **if** \_\_name\_\_ == **"\_\_main\_\_"**:  **import** sys  app = QtWidgets.QApplication(sys.argv)  loginWindow = QtWidgets.QDialog()    ui = Ui\_Dialog()  ui.setupUi(loginWindow)  loginWindow.show()  sys.exit(app.exec\_()) |
|  |

### **Remote.py**

|  |
| --- |
| **Nombre de archivo:** remote.py  **Nombre de función:** showSelectorLocal()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** verificarIP()  **def** showSelectorLocal(self,remote\_conn):  self.selectorLocal = QtWidgets.QDialog()    self.ui = selectorLocal.Ui\_SelectorLocal()  self.ui.setupUi(self.selectorLocal,remote\_conn)  self.selectorLocal.show()  **Nombre de archivo:** remote.py  **Nombre de función:** verificarIP()  **Descripción:** Obtiene los datos de los campos de texto de ip ingresados por el cliente, verifica que soncampos con datos numéricos y que los mismos sean campos coherentes con el formato de dirección ipv4. Si todos los campos están correctos intenta realizar la conexión por medio de ssh. Si ocurre algún problema muestra mensaje popup informando en que campo en específico existe una inconsistencia.  **Método de llamado:** self.btn\_conectar.clicked()  def verificarIP(self, Form):  if self.txt\_1.text().isnumeric() and (int(self.txt\_1.text()) < 256):  if self.txt\_2.text().isnumeric() and (int(self.txt\_2.text()) < 256):  if self.txt\_3.text().isnumeric() and (int(self.txt\_3.text()) < 256):  if self.txt\_4.text().isnumeric() and (int(self.txt\_4.text()) < 256):  print("estableciendo conexion...")  ip = self.txt\_1.text() + "." + self.txt\_2.text() + "." + self.txt\_3.text() + "." + self.txt\_4.text()  print(  self.txt\_1.text() + "." + self.txt\_2.text() + "." + self.txt\_3.text() + "." + self.txt\_4.text()+" "+self.txt\_usuario.text()+" "+self.txt\_contrasena.text())  """try:  remote\_conn\_pre, remote\_conn = login\_ssh(ip, self.txt\_usuario.text(),self.txt\_contrasena.text())  print("1")  disable\_paging(remote\_conn)  print("2")  back\_home(remote\_conn)  print("3")  self.showSelectorLocal(remote\_conn)    print("4")  Form.close()  except Exception :  evento=4  arch = open("temp.txt", "r")  nombre=""  time=""  for linea in arch:  linea = linea.split(",")  nombre = str(linea[0])  time = str(linea[1])  arch.close()  file = open("historial.txt", "a")  file.write(nombre +","+ time +","+ str(evento)+"\n")  ctypes.windll.user32.MessageBoxW(0, "No se pudo establecer la conexion, verificar ip o credenciales.", "Error", 0)"""  try:  remote\_conn\_pre, remote\_conn = login\_ssh(ip, self.txt\_usuario.text(),self.txt\_contrasena.text())  print("verificando ip")  print("1")  disable\_paging(remote\_conn)  print("2")  back\_home(remote\_conn)  print("3")  self.showSelectorLocal(remote\_conn)  print("4")  Form.close()  except Exception :  ctypes.windll.user32.MessageBoxW(0, "No se pudo establecer la conexion, verificar ip o credenciales.", "Error", 0)  else:  ctypes.windll.user32.MessageBoxW(0, "Error en el cuarto octeto", "Error", 1)  else:  ctypes.windll.user32.MessageBoxW(0, "Error en el tercer octeto", "Error", 1)  else:  ctypes.windll.user32.MessageBoxW(0, "Error en el segundo octeto", "Error", 1)  else:  ctypes.windll.user32.MessageBoxW(0, "Error en el primer octeto", "Error", 1) |
|  |

### **SelectorLocal.py**

|  |
| --- |
| **Nombre de archivo:** selectorLocal.py  **Nombre de función:** showConfiguracionBasicaWindow()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana y mostrándola en pantalla.  **Método de llamado:** btn\_configuracionBasica.clicked ()  **def**  showConfiguracionBasicaWindow(self,SelectorLocal,remote\_conn):  self.configuracionBasicaWindow = QtWidgets.QDialog()    self.ui = configuracionBasica.Ui\_ConfiguracionBasica()  self.ui.setupUi(self.configuracionBasicaWindow,remote\_conn)  self.configuracionBasicaWindow.show()  SelectorLocal.close()  **Nombre de archivo:** remote.py  **Nombre de función:** showSelectorMPLS()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_configuracionMPLS.clicked ()  **def** showSelectorMPLS(self,SelectorLocaL, remote\_conn): self.selectorConfiguracionMPLS = QtWidgets.QDialog() self.ui =    selectorConfiguracionMPLS.Ui\_SelectorConfiguracionMPLS()  self.ui.setupUi(self.selectorConfiguracionMPLS, remote\_conn)  self.selectorConfiguracionMPLS.show()  SelectorLocaL.close() |

### **ConfiguracionBasica.py**

|  |
| --- |
| **Nombre de archivo:** configuracionBasica.py  **Nombre de función:** showPlantilla ()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_plantilla.clicked()  **def** showPlantilla(self,Form,remote\_conn):  self.plantilla = QtWidgets.QDialog()  self.ui = Ui\_Plantilla()    self.ui.setupUi(self.plantilla,remote\_conn)  self.plantilla.show()  Form.close()  **Nombre de archivo:** configuracionBasica.py  **Nombre de función:** showSelectorCredenciales ()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_credenciales.clicked()  **def**    showSelectorCredenciales(self,Form,remote\_conn):  self.selectorCredenciales = QtWidgets.QDialog()  self.ui = selectorCredenciales.Ui\_SelectorCredenciales()  self.ui.setupUi(self.selectorCredenciales, remote\_conn)  self.selectorCredenciales.show()  Form.close()  **Nombre de archivo:** configuracionBasica.py  **Nombre de función:** showDireccionamientoLocal ()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_direccionamiento.clicked()  **def** showDireccionamientoLocal(self,Form, remote\_conn):  self.direccionamientoLocal = QtWidgets.QDialog()  self.ui = direccionamientoLocal.Ui\_DireccionamientoLocal()    self.ui.setupUi(self.direccionamientoLocal, remote\_conn)  self.direccionamientoLocal.show()  Form.close()  **Nombre de archivo:** configuracionBasica.py  **Nombre de función:** showSelectorLocal ()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_atras.clicked()  **def** showSelectorLocal(self,Form,remote\_conn):  self.selectorLocal = QtWidgets.QDialog()  self.ui = selectorLocal.Ui\_SelectorLocal()    self.ui.setupUi(self.selectorLocal,remote\_conn)  self.selectorLocal.show()  Form.close() |

### **DireccionamientoLocal.py**

|  |
| --- |
| **Nombre de archivo:** DireccionamientoLocal.py  **Nombre de función:** habilitadorInt# () \*Reemplazar # por numero de interfaz  **Descripción:** Cuando esta función es llamada habilita y pone visibles los campos de texto para ingresardirecciones ipv4 por el usuario.  **Método de llamado:** habilitadorCampos()  **####Ejemplo con Interfaz 1**  **def** habilitadorInt1(self):  self.txt\_ip1\_1.setEnabled(**True**)  self.txt\_ip1\_2.setEnabled(**True**)    self.txt\_ip1\_3.setEnabled(**True**)  self.txt\_ip1\_4.setEnabled(**True**)  self.txt\_mask1\_1.setEnabled(**True**)  self.txt\_mask1\_2.setEnabled(**True**)  self.txt\_mask1\_3.setEnabled(**True**)  self.txt\_mask1\_4.setEnabled(**True**)  **Nombre de archivo:** DireccionamientoLocal.py  **Nombre de función:** habilitadorCampos(interfaces)  **Descripción:** Esta función recibe como parámetro una lista de interfaces disponibles del router paraconfigurar, de esta forma obtiene el numero de interfaces (máximo 5), les asigna un ID y las habilita por medio de la función habilitadorInt#()  **Método de llamado:** automatico  **def** habilitadorCampos(self, interfaces):   1. = len(interfaces) **if** n ==0:   ctypes.windll.user32.MessageBoxW(0, **"No existe ninguna** **interfaz disponible para configurar"**, **"Error"**,1)  **elif** n ==1:  self.lbl\_int1.setText(interfaces[0][0])  self.lbl\_int1.setEnabled(**True**)  self.habilitadorInt1()  **elif** n ==2:  self.lbl\_int1.setText(interfaces[0][0])  self.lbl\_int1.setEnabled(**True**)  self.lbl\_int2.setText(interfaces[1][0])  self.lbl\_int2.setEnabled(**True**)  self.habilitadorInt2()  **elif** n ==3:  self.lbl\_int1.setText(interfaces[0][0])  self.lbl\_int1.setEnabled(**True**)  self.lbl\_int2.setText(interfaces[1][0])  self.lbl\_int2.setEnabled(**True**)  self.lbl\_int3.setText(interfaces[2][0])  self.lbl\_int3.setEnabled(**True**)    self.habilitadorInt3()  **elif** n ==4:  self.lbl\_int1.setText(interfaces[0][0])  self.lbl\_int1.setEnabled(**True**)  self.lbl\_int2.setText(interfaces[1][0])  self.lbl\_int2.setEnabled(**True**)  self.lbl\_int3.setText(interfaces[2][0])  self.lbl\_int3.setEnabled(**True**)  self.lbl\_int4.setText(interfaces[3][0])  self.lbl\_int4.setEnabled(**True**)  self.habilitadorInt4()  **elif** n >=5:  self.lbl\_int1.setText(interfaces[0][0])  self.lbl\_int1.setEnabled(**True**)  self.lbl\_int2.setText(interfaces[1][0])  self.lbl\_int2.setEnabled(**True**)  self.lbl\_int3.setText(interfaces[2][0])  self.lbl\_int3.setEnabled(**True**)  self.lbl\_int4.setText(interfaces[3][0])  self.lbl\_int4.setEnabled(**True**)  self.lbl\_int5.setText(interfaces[4][0])  self.lbl\_int5.setEnabled(**True**)  self.habilitadorInt5() |

### **SelectorCredenciales.py**

|  |
| --- |
| **Nombre de archivo:** selectorCredenciales.py  **Nombre de función:** verUsuarios()  **Descripción:** Hace el llamado a un método que devuelve los usuarios existentes en el router y los muestraen pantalla  **Método de llamado:** btn\_verUsuarios.clicked()  **def** verUsuarios(self,remote\_conn): funciones2.back\_home(remote\_conn) lista=funciones2.sh\_usernames(remote\_conn) time.sleep(2)    print(lista)  ctypes.windll.user32.MessageBoxW(0,lista ,  **"Información usuarios"**,0)  **Nombre de archivo:** selectorCredenciales.py  **Nombre de función:** showAnadirUsuarios()  **Descripción:** Muestra la ventana para poder añadir usuarios al enrutador  **Método de llamado:** btn\_anadirUsuario.clicked()  **def** showAnadirUsuario(self,SelectorCredenciales,remote\_conn):  print(**"here 1"**)  self.AnadirUsuario = QtWidgets.QDialog()    self.ui = anadirUsuario.Ui\_AnadirUsuario()  self.ui.setupUi(self.AnadirUsuario,remote\_conn)  self.AnadirUsuario.show()  SelectorCredenciales.close() |

### **AnadirUsuario.py**

|  |
| --- |
| **Nombre de archivo:** AnadirUsuario.py  **Nombre de función:** addUsuario()  **Descripción:** Valida el tipo de dato de los campos de texto y si todo es correcto añade un usuario alenrutador mediante otra función.  **Método de llamado:** btn\_anadir.clicked()  **def** addUsuario(self,AnadirUsuario,remote\_conn):  print(**"########################CODIGO ANADIR USUARIOS"**)  print(type(remote\_conn))  **if** (self.txt\_privilegio.text().isnumeric() **and**  ((int(self.txt\_privilegio.text()))<=15) **and**  self.txt\_usuario.text()!=**""** **and** self.txt\_contrasena.text()!=**""**):  funciones2.conf\_credencial(remote\_conn,    self.txt\_usuario.text(), self.txt\_contrasena.text(),  self.txt\_privilegio.text())  ctypes.windll.user32.MessageBoxW(0, **"Configuracion** **realizada con éxito"**, **"éxito!"**,0)  self.atras(AnadirUsuario, remote\_conn)  **else**:  ctypes.windll.user32.MessageBoxW(0, **"Verifica los campos,** **asegúrese del valor de privilegio"**,**"Error!!"**,0) |

### **Plantilla.py**

|  |
| --- |
| **Nombre de archivo:** plantilla.py  **Nombre de función:** showConfiguracionBasica()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** atras()  **def** showConfiguracionBasica(self,Form,remote\_conn):  self.configuracionBasica = QtWidgets.QDialog()  self.ui = configuracionBasica.Ui\_ConfiguracionBasica()    self.ui.setupUi(self.configuracionBasica,remote\_conn)  self.configuracionBasica.show()  Form.close()  **Nombre de archivo:** plantilla.py  **Nombre de función:** configurar()  **Descripción:** Primero valido todos los campos de ingreso de texto. Si los campos necesarios (hostname ydomain name) no están completos o no validados la función lanza un mensaje de alerta.  Los campos opcionales (dns1,dns2) se validan y se incluyen en la configuración siempre y cuando se detecte contenido en su campo de texto, caso contrario se omiten de la configuración.  **Método de llamado:** btn\_configurar.clicked()  **def** configurar(self,Form,remote\_conn):  hostname = self.txt\_hostname.text() domainName = self.txt\_domainName.text() **if**    (self.verificarIP(self.txt\_dns1\_1,self.txt\_dns1\_2,self.txt\_dns1\_3 ,self.txt\_dns1\_4)):  dns1 =  self.txt\_dns1\_1.text()+**"."**+self.txt\_dns1\_2.text()+**"."**+self.txt\_dn  s1\_3.text()+**"."**+self.txt\_dns1\_4.text()  **else**:  dns1=**""**  ctypes.windll.user32.MessageBoxW(0, **"No se configurará** **DNS1"**, **"Alerta"**,0)  **if**  (self.verificarIP(self.txt\_dns2\_1,self.txt\_dns2\_2,self.txt\_dns2\_3 ,self.txt\_dns2\_4)):  dns2 =  self.txt\_dns2\_1.text()+**"."**+self.txt\_dns2\_2.text()+**"."**+self.txt\_dn  s2\_3.text()+**"."**+self.txt\_dns2\_4.text()  **else**:  dns2=**""**  ctypes.windll.user32.MessageBoxW(0, **"No se configurará** **DNS2"**, **"Alerta"**,0)  tipo\_ssh = **"paramiko"**  *#tipo\_ssh = "<paramiko.Channel 0 (open) window=1004 ->*  *<paramiko.Transport at 0x7546c70 (cipher aes128-cbc, 128 bits)*  *(active; 1 open channel(s))>>"*  print(type(remote\_conn) **is** paramiko.channel.Channel)  **if** type(remote\_conn) **is** paramiko.channel.Channel:  **if** len(hostname)>0 **and** len(domainName)>0:  funciones2.conf\_plantilla(remote\_conn, hostname,  domainName, dns1, dns2)  ctypes.windll.user32.MessageBoxW(0, **"Configuración**  **realizada con éxito"**,  **"Done"**,0)  **else**:  ctypes.windll.user32.MessageBoxW(0, **"Llene los campos** **necesarios para la configuración (\*)"**,  **"Error"**,0)  print(**"Entro por REMOTE"**)  **else**:  print(**"Entro por LOCAL"**)  funciones\_com.conf\_plantilla(remote\_conn, hostname, domainName, dns1, dns2) |

### **SelectorConfiguracionMPLS.py**

|  |
| --- |
| **Nombre de archivo:** selectorConfiguracionMPLS.py  **Nombre de función:** showConfigurarPE()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_configurarPE.clicked()  **def** showConfigurarPE(self,Form, remote\_conn):  self.configurarPE2 = QtWidgets.QDialog()  self.ui = Ui\_ConfigurarPE2()    self.ui.setupUi(self.configurarPE2, remote\_conn)  self.configurarPE2.show()  Form.close()  **Nombre de archivo:** selectorConfiguracionMPLS.py  **Nombre de función:** showConfigurarCE()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_configurarCE.clicked()  **def** showConfigurarCE(self,Form, remote\_conn):  self.configurarCE = QtWidgets.QDialog()  self.ui = Ui\_ConfigurarCE()  self.ui.setupUi(self.configurarCE, remote\_conn)  self.configurarCE.show()  Form.close()  **Nombre de archivo:** selectorConfiguracionMPLS.py  **Nombre de función:** showSelectorLocal()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_atras.clicked()  **def** showSelectorLocal(self, Form, remote\_conn):  self.selectorLocal = QtWidgets.QDialog()  self.ui = selectorLocal.Ui\_SelectorLocal()    self.ui.setupUi(self.selectorLocal, remote\_conn)  self.selectorLocal.show()  Form.close()  **Nombre de archivo:** selectorConfiguracionMPLS.py  **Nombre de función:** configurarP()  **Descripción:** Envía la configuración al enrutador obteniendo los datos de las funciones del archivo*funciones2.py* (ver funciones2.py)  **Método de llamado:** btn\_configurarP.clicked()  **def** configurarP(self, remote\_conn):  **if** (type(remote\_conn) **is** paramiko.channel.Channel):funciones2.config\_OSPF(remote\_conn) funciones2.save\_ID(remote\_conn)  **else**:  funciones\_com.config\_OSPF(remote\_conn)    print(**"1"**)  funciones2.config\_cef\_mpls\_ldp(remote\_conn)  print(**"2"**)  ctypes.windll.user32.MessageBoxW(0, **"Configuración realizada** **con éxito"**,  **"Done"**,0) |

### **ConfigurarCE.py**

|  |
| --- |
| **Nombre de archivo:** configurarCE.py  **Nombre de función:** showSelectorMPLS()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_atras.clicked()  **def** showSelectorMPLS(self, Form, remote\_conn): self.selectorMpls = QtWidgets.QDialog() self.ui =  selectorConfiguracionMPLS.Ui\_SelectorConfiguracionMPLS()  self.ui.setupUi(self.selectorMpls, remote\_conn)  self.selectorMpls.show()  Form.close()  **Nombre de archivo:** configurarCE.py  **Nombre de función:** configurar()  **Descripción:** Envía la configuración al enrutador obteniendo los datos de las funciones del archivo*funciones2.py* (ver funciones2.py)  **Método de llamado:** btn\_configurar.clicked()  **def** configurar(self,Form, remote\_conn):  **if** self.verificarIP():  print(**"ENTROOOOOOOOOOOOOOO"**)  funciones2.conf\_route\_CE(remote\_conn, self.txt\_1.text() +     * + self.txt\_2.text() + **"."** + self.txt\_3.text() + **"."** + self.txt\_4.text())   ctypes.windll.user32.MessageBoxW(0, **"Configuración** **realizada con éxito"**,  **"Done"**,0) |

### **ConfigurarPE2.py**

|  |
| --- |
| **Nombre de archivo:** configurarPE2.py  **Nombre de función:** showSelectorMPLS()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_atras.clicked()  **def** showSelectorMPLS(self, Form, remote\_conn):  self.selectorMpls = QtWidgets.QDialog()    self.ui = selectorConfiguracionMPLS.Ui\_SelectorConfiguracionMPLS()  self.ui.setupUi(self.selectorMpls, remote\_conn)  self.selectorMpls.show()  Form.close()  **Nombre de archivo:** configurarPE2.py  **Nombre de función:** showagregarCliente()  **Descripción:** Esta función permite la navegación entre ventanas, creando la instancia de la ventana ymostrándola en pantalla.  **Método de llamado:** btn\_agregarCliente.clicked()  **def** showAgregarCliente(self,Form,remote\_conn):  self.agregarCliente = QtWidgets.QDialog()    self.ui = agregarCliente.Ui\_AgregarCliente()  self.ui.setupUi(self.agregarCliente, remote\_conn)  self.agregarCliente.show()  Form.close()  **Nombre de archivo:** configurarPE2.py  **Nombre de función:** confPE()  **Descripción:** Permite la configuración de un dispositivo PE, validando que los campos ingresados paradirecciones IP sean correctos. Luego, se procede a configurar los protocolos iBGP, MP-BGP, CEF, LDP, MPLS, creación de VRF y enrutamiento entre dispositivos PE y CE.  **Método de llamado:** btn\_anadir.clicked()  **def** ConfPE(self, remote\_conn):  **if**  self.verificarIP(self.txt\_ip1\_1,self.txt\_ip1\_2,self.txt\_ip1\_3,self.txt \_ip1\_4):  **if** self.verificarIP(self.txt\_ip2\_1,self.txt\_ip2\_2,self.txt\_ip2\_3, self.txt\_ip2\_4):  **if** self.verificarIP(self.txt\_mask1\_1,self.txt\_mask1\_2,self.txt\_mask1\_3, self.txt\_mask1\_4):  **if** self.verificarIP(self.txt\_mask2\_1,self.txt\_mask2\_2, self.txt\_mask2\_3, self.txt\_mask2\_4):  **if** self.verificarIP(self.txt\_mask3\_1,self.txt\_mask3\_2, self.txt\_mask3\_3, self.txt\_mask3\_4):  **if** self.txt\_vlan.text().isnumeric() **and**  ((int(self.txt\_AS.text()))<=65535) **and** self.txt\_AS.text().isnumeric()  **and** self.txt\_vrf.text()!=**"" and** self.txt\_interfaz.text()!=**""** :  **if** (type(remote\_conn) **is**  paramiko.channel.Channel):  funciones2.back\_home(remote\_conn)  print(**"1"**)  funciones2.config\_iBGP(remote\_conn)  print(**"2"**)    time.sleep(1)  funciones2.config\_MP\_BGP(remote\_conn)  print(**"3"**)  time.sleep(1)  funciones2.config\_cef\_mpls\_ldp(remote\_conn)  print(**"4"**)  time.sleep(1)  funciones2.config\_vrf(remote\_conn,  self.txt\_vrf.text(), self.txt\_AS.text(), self.txt\_vlan.text())  print(**"5"**)  time.sleep(1)  redCliente = self.txt\_ip2\_1.text() +   * + self.txt\_ip2\_2.text() + **"."** + self.txt\_ip2\_3.text() + **"."** + self.txt\_ip2\_4.text()   maskCliente = self.txt\_mask2\_1.text()   * **"."** +self.txt\_mask2\_2.text() + **"."** +self.txt\_mask2\_3.text() + **"."** * self.txt\_mask2\_4.text()   dirCliente = self.txt\_ip1\_1.text() +   * + self.txt\_ip1\_2.text() + **"."** + self.txt\_ip1\_3.text() + **"."** + self.txt\_ip1\_4.text()   maskDirCliente =  self.txt\_mask1\_1.text() + **"."** + self.txt\_mask1\_2.text() + **"."** + self.txt\_mask1\_3.text() + **"."** + self.txt\_mask1\_4.text()  gatewayCliente =  self.txt\_mask3\_1.text()+**"."**+self.txt\_mask3\_2.text()+**"."**+self.txt\_mask3  \_3.text()+**"."**+self.txt\_mask3\_4.text()  print(**"6"**)  time.sleep(1)  funciones2.config\_add\_interfaz\_vrf(remote\_conn, self.txt\_vrf.text(), self.txt\_vlan.text(),  self.txt\_interfaz.text(), dirCliente, maskDirCliente)  print(**"7"**)  time.sleep(1)  funciones2.config\_route\_PE\_CE(remote\_conn, self.txt\_vrf.text(), redCliente, maskCliente,  gatewayCliente)  print(**"8"**)  time.sleep(1)  funciones2.redistribute\_vrf(remote\_conn, self.txt\_vrf.text())  print(**"9"**)  time.sleep(4)  remote\_conn.recv(MAX\_BUFFER)  self.txt\_Clientes.setText(funciones2.show\_res(remote\_conn)) ctypes.windll.user32.MessageBoxW(0,  **"Configuración realizada con éxito"**,  **"Done"**,0)  **else**:  funciones\_com.config\_OSPF(remote\_conn)  print(**"1"**)  funciones\_com.config\_iBGP(remote\_conn)  print(**"2"**)  funciones\_com.config\_MP\_BGP(remote\_conn)  print(**"3"**)  funciones\_com.config\_cef\_mpls\_ldp(remote\_conn)  print(**"4"**)  funciones\_com.config\_vrf(remote\_conn,  self.txt\_vrf.text(), self.txt\_AS.text(), self.txt\_vlan.text())  print(**"5"**)  redCliente = self.txt\_ip2\_1.text() +   * + self.txt\_ip2\_2.text() + **"."** + self.txt\_ip2\_3.text() + **"."** + self.txt\_ip2\_4.text()   maskCliente = self.txt\_mask2\_1.text()   * **"."** +self.txt\_mask2\_2.text() + **"."** +self.txt\_mask2\_3.text() + **"."** * self.txt\_mask2\_4.text()   dirCliente = self.txt\_ip1\_1.text() +   * + self.txt\_ip1\_2.text() + **"."** + self.txt\_ip1\_3.text() + **"."** + self.txt\_ip1\_4.text()   maskDirCliente =  self.txt\_mask1\_1.text() + **"."** + self.txt\_mask1\_2.text() + **"."** + self.txt\_mask1\_3.text() + **"."** + self.txt\_mask1\_4.text()  print(**"6"**)  funciones\_com.config\_add\_interfaz\_vrf(remote\_conn, self.txt\_vrf.text(), self.txt\_vlan.text(),  self.txt\_interfaz.text(), dirCliente, maskDirCliente)  print(**"7"**)  funciones\_com.config\_route\_PE\_CE(remote\_conn, self.txt\_vrf.text(), redCliente, maskCliente,  self.txt\_interfaz.text())  print(**"8"**)  funciones\_com.redistribute\_vrf(remote\_conn, self.txt\_vrf.text())  print(**"9"**)  self.txt\_Clientes.setText(funciones\_com.show\_res(remote\_conn))  **else**:  self.txt\_Clientes.setText(**"Error al**  **configurar el dispositivo"**)  **else**:  self.txt\_Clientes.setText(**"Error al configurar**  **el dispositivo"**)  **else**:  self.txt\_Clientes.setText(**"Error al configurar el**  **dispositivo"**)  **else**:  self.txt\_Clientes.setText(**"Error al configurar el**  **dispositivo"**)  **else**:  self.txt\_Clientes.setText(**"Error al configurar el** **dispositivo"**)  **else**:  self.txt\_Clientes.setText(**"Error al configurar el** **dispositivo"**) |

### **Funciones2.py**

|  |
| --- |
| **Nombre de archivo:** funciones2.py  **Nombre de función:** obtener\_FechayHora()  **Descripción:** Permite obtener la hora y fecha del sistema en cadena de texto.  **Método de llamado:** varios.  **def** obtener\_FechaYHora():  info=datetime.datetime.now()  fecha=str(info.day)+**"-"**+str(info.month)+**"-"**+str(info.year)  **if** info.minute<10:  minutos=**"0"**+str(info.minute)    **else**:  minutos=str(info.minute)  **if** info.second<10:  segundos=**"0"**+str(info.second)  **else**:  segundos=str(info.second)  **if** info.hour<10:  horas=**"0"**+str(info.hour)  **else**:  horas=str(info.hour)  hora=horas+**"H"**+minutos+**"M"**+segundos+**"S"**  **return** fecha,hora  **Nombre de archivo:** funciones2.py  **Nombre de función:** login\_ssh()  **Descripción:** Permite inicializar una conexión por SSH utilizando la librería paramiko, dado por parámetrouna dirección IP de alguna de las interfaces del dispositivo, un usuario y contraseña correspondiente a una credencial de acceso para este tipo de conexión del dispositivo objetivo. Se retorna dos objetos llamados remote\_conn\_pre y remote\_conn, este último es el que permite la interacción con la CLI de los dispositivos y es el que se envía por parámetro en todas las funciones que interactúen con la CLI del dispositivo que se este configurando.  **Método de llamado:** varios.  **def** login\_ssh(IP, username, password):   * *Crea instancia de un cliente SSH.* remote\_conn\_pre = paramiko.SSHClient() * *Agrega automáticamente hosts no confiables(asegúrese de que esté bien para la política de seguridad en su entorno).*     remote\_conn\_pre.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())  *# Inicia conexión SSH.*  remote\_conn\_pre.connect(IP, username=username, password=password, look\_for\_keys=**False**, allow\_agent=**False**)   * *Permite una conexión interactiva con el host.* remote\_conn = remote\_conn\_pre.invoke\_shell() * *Leer 1000 bytes de la salida actual del host.* output = remote\_conn.recv(MAX\_BUFFER)   **return** remote\_conn\_pre, remote\_conn  **Nombre de archivo:** funciones2.py  **Nombre de función:** disable\_paging()  **Descripción:** Permite eliminar cualquier texto adicional que se encuentre en la CLI del dispositvo en el quese encuentre haciendo conexión SSH, de forma que se pueda leer la última parte del texto de la CLI con la que se interactúe.  **Método de llamado:** varios.  **def** disable\_paging(remote\_conn):  *'''Disable paging on a Cisco router'''* remote\_conn.send(**"terminal length 0\r\n"**) **while**(**not** remote\_conn.recv\_ready()):    time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  **Nombre de archivo:** funciones2.py  **Nombre de función:** back\_home()  **Descripción:** Permite volver al modo de configuración privilegiado, con lo que se evita posibles malasconfiguraciones por encontrarse en el modo de configuración incorrecto.  **Método de llamado:** varios.  **def** back\_home(remote\_conn):  remote\_conn.send(**"\n"**)    **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  print(output)  **if**(**'config' in** output):  remote\_conn.send(**"end\n"**)  remote\_conn.send(**"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  **Nombre de archivo:** funciones2.py  **Nombre de función:** sh\_usernames()  **Descripción:** Permite leer todas las credenciales de acceso para el protocolo SSH del router objetivo,devolviendo una cadena de texto con la información del nombre de usuario de dichas credenciales.  **Método de llamado:** varios.  **def** sh\_usernames(remote\_conn):  list\_users = **""**  remote\_conn.send(**"show start | include username\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()    output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**)  print(output)  **for** i **in** range(1,len(output) -1):  line = output[i].split(**" "**)  **if** (**"username" in** line[0]):  list\_users += (line[1] + **"\n"**)  print(**"\n"** + list\_users)  **return** list\_users  **Nombre de archivo:** funciones2.py  **Nombre de función:** add\_import()  **Descripción:** Permite agregar la importación de rutas de una nueva VRF, de forma que se permita lacomunicación entre dispositivos de distintas VRF.  **Método de llamado:** varios.  **def** add\_import(remote\_conn, name\_vrf, vlan):  **if**(vlan.isnumeric()):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"ip vrf "** + name\_vrf + **"\n"**)  remote\_conn.send(**"route-target import "** + vlan + **":"** + vlan +  **"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  output = output.replace(**'\r\n'**, **'\n'**)    print(**"/////"**)  print(output)  **if "please configure" in** output:  ctypes.windll.user32.MessageBoxW(0, **"La VRF ingresada no**  **tiene un route distinguisher asignado"**,  **"Error"**,0)  **else**:  ctypes.windll.user32.MessageBoxW(0, **"Configuración** **realizada con éxito"**,  **"Done"**,0)  back\_home(remote\_conn)  **else**:  ctypes.windll.user32.MessageBoxW(0, **"VLAN incorrecta"**, **"Error"**,0)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_dir()  **Descripción:** Permite configurar el direccionamiento en una interfaz, dado la interfaz, la dirección IP y la máscara de subred por parámetro para realizar esta configuración.  **Método de llamado:** varios.  **def** config\_dir(remote\_conn,interfaces, IPs, masks):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"int "**+interfaces+ **"\n"**)  remote\_conn.send(**"ip add "**+IPs+**" "**+masks+ **"\n"**)  remote\_conn.send(**"no sh\n"**)  remote\_conn.send(**"do wr\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** conf\_plantilla()  **Descripción:** Permite configurar la plantilla básica de un enrutador, así como su nombre de dominio,servidores dns, hostname y mensaje de bienvenido al conectarse al dispositivo.  **Método de llamado:** varios.  **def** conf\_plantilla(remote\_conn, hostname, domain\_name, dns1, dns2):  remote\_conn.send(**"conf t\n"**)  **if**(len(dns1)>0):  remote\_conn.send(**"ip name-server "**+dns1+**"\n"**)  **if** (len(dns2)>0):  remote\_conn.send(**"ip name-server "** + dns2+**"\n"**) remote\_conn.send(**"host "**+hostname+**"\n"**+**"ip domain-name**  **"**+domain\_name+**"\n"**+  **"banner motd # ACCESO SOLO A PERSONAL AUTORIZADO#"**+**"\n"**+  **"line vty 0 4"**+**"\n"**+  **"transport input all"**+**"\n"**+    **"login local"**+**"\n"**+  **"exec-timeout 3 3"**+**"\n"**+  **"logging synchronous"**+**"\n"**+  **"line console 0"**+**"\n"**+  **"transport output all"**+**"\n"**+  **"login local"**+**"\n"**+  **"exec-timeout 3 3"**+**"\n"**+  **"logging synchronous"**+**"\n"**+  **"do wr"**+**"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** conf\_credencial()  **Descripción:** Permite agregar una nueva credencial de acceso SSH en un enrutador, dado un nombre deusuario, contraseña y privilegio de dicha credencial.  **Método de llamado:** varios.  **def** conf\_credencial(remote\_conn,user,password,privilegio):  **try**:  remote\_conn.send(**"conf t"** + **"\n"**+    **"username "**+user+**" privilege "**+privilegio+**"**  **secret "**+password+**'\n'**+  **"do wr"** + **'\n'**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **except** Exception:  ctypes.windll.user32.MessageBoxW(0, **"Ocurrio un error"**,  **"Error"**,1)  **Nombre de archivo:** funciones2.py  **Nombre de función:** get\_dirs\_red()  **Descripción:** Permite obtener todas las direcciones de red conectadas di rectamente a un enrutador, deforma que estas redes puedan ser anunciadas por OSPF.  **Método de llamado:** varios.  **def** get\_dirs\_red(remote\_conn):  list\_dirs\_red = []  remote\_conn.send(**"sh ip route connected | exclude /32\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  time.sleep(3)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**)  print(output)    **for** i **in** range(8,len(output) -1):  **if** (**"connected" in** output[i] **and "/" in** output[i] **and not**(**"ip"**  **in** output[i])):  line = output[i].split(**" "**)  print(**" OSPF"**)  **for** j **in** range(0,len(line)):  **if** (**"/" in** line[j]):  list\_dirs\_red.append(line[j])  **break**  back\_home(remote\_conn)  **return** list\_dirs\_red  **Nombre de archivo:** funciones2.py  **Nombre de función:** get\_wildcard()  **Descripción:** Permite obtener las wildcards de una lista de redes dadas por parámetro.  **Método de llamado:** varios.  **def** get\_wildcard(list\_dirs\_red):  list\_wildcards = []  **for** k **in** range(0,len(list\_dirs\_red)):  line = list\_dirs\_red[k].split(**'/'**)  mask = 32 - int(line[1])  wildcard = [0, 0, 0, 0]  octeto = 3    **while** (mask >8):  wildcard[octeto] = 2 \*\* 8 - 1  octeto -= 1  mask -= 8  wildcard[octeto] = 2 \*\* mask - 1  list\_wildcards.append(  str(wildcard[0]) + **'.'** + str(wildcard[1]) + **'.'** + str(wildcard[2]) + **'.'** + str(wildcard[3]))  **return** list\_wildcards  **Nombre de archivo:** funciones2.py  **Nombre de función:** get\_ip\_interfaz()  **Descripción:** Permite obtener una lista con todas las interfaces que tiene un enrutador, esta informacióncomprende el nombre de las interfaces, su estado (up/down) y la dirección IP asignada a la misma.  **Método de llamado:** varios.  **def** get\_ip\_interfaz(remote\_conn):  print(**"int"**)  back\_home(remote\_conn)  info\_interfaz=[]  remote\_conn.send(**"sh ip int b\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode() output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**) **for** i **in** range(1,len(output)-1):    line = output[i].split()  print(line)  **if not**(**"#" in** line[0]) **and "/" in** line[0]:  **if**(len(line[0].split(**"."**))==1):  interfaz = [line[0], line[4], line[1]]  **if** line[4]!=**"up"**:  interfaz[1]=**"down"**  info\_interfaz.append(interfaz)  back\_home(remote\_conn)  **return** info\_interfaz  **Nombre de archivo:** funciones2.py  **Nombre de función:** get\_ospf\_neig()  **Descripción:** Permite obtener los vecinos de OSPF, lo cual permite configurar el protocolo iBGP y MP-BGPentre los dispositivos PE.  **Método de llamado:** varios.  **def** get\_ospf\_neig(remote\_conn):  loopback\_rd=**""**  back\_home(remote\_conn)  remote\_conn.send(**"show ip interface brief | include Loopback\n"**)  time.sleep(3)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode() output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**) **for** i **in** range(0,len(output)):  line = output[i].split()  **if "Loopback" in** line[0]:  print(line[0])  loopback\_rd = (line[1])  neighbors=[]    list\_neig=[]  print(**"---"** + loopback\_rd)  remote\_conn.send(**"sh ip ospf database network | include Attached** **Router\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  output = (remote\_conn.recv(MAX\_BUFFER)).decode() output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**) **for** i **in** range(1,len(output)):  print(output[i])  **if**(**'Attached Router' in** output[i]):  list\_neig.append((output[i].split(**":"**))[1])  **for** j **in** range(0,len(list\_neig)):  dir = list\_neig[j][1:]  **if not**(dir **in** neighbors) **and** dir!=loopback\_rd:  neighbors.append(dir)  print(neighbors)  back\_home(remote\_conn)  **return** neighbors  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_OSPF()  **Descripción:** Permite configurar el protocolo OSPF en el AS 1 y en el área de backbone 0. Para esto seobtienen todas las direcciones de redes conectadas directamente de un enrutador, posteriormente la wildcard de cada una de estas redes y son anunciadas por el protocolo ya mencionado.  **Método de llamado:** varios.  **def** config\_OSPF(remote\_conn):  list\_dirs\_red = get\_dirs\_red(remote\_conn)  list\_wildcards = get\_wildcard(list\_dirs\_red)  print(list\_dirs\_red)  print(list\_wildcards)    remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"router ospf 1\n"**)  **for** i **in** range(0,len(list\_dirs\_red)):  remote\_conn.send(**"network "**+list\_dirs\_red[i].split(**'/'**)[0]+**"**  **"**+list\_wildcards[i]+**" area 0"**+**"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** conf\_mpls\_interfaces()  **Descripción:** Permite habilitar mpls en todas las interfaces habilitadas (estado up) y que no sean lasinterfaces Loopback. Para esto se obtiene primero todas las interfaces disponibles del enrutador con la función get\_ip\_interfaz.  **Método de llamado:** varios.  **def** conf\_mpls\_interfaces(remote\_conn):  back\_home(remote\_conn)  interfaces = get\_ip\_interfaz(remote\_conn)  print(interfaces)  remote\_conn.send(**"conf t\n"**)  **for** i **in** range(0,len(interfaces)):    **if** (interfaces[i][1]==**"up" and not**(**"Loopback" in** interfaces[i][0])):  print(i)  remote\_conn.send(**"int "**+interfaces[i][0]+**"\n"**)  remote\_conn.send(**"mpls ip\n"**)  remote\_conn.send(**"exit\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  time.sleep(5)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_cef\_mpls\_ldp()  **Descripción:** Se configura el protocolo CEF, LDP y MPLS.  **Método de llamado:** varios.  **def** config\_cef\_mpls\_ldp(remote\_conn):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"ip cef\n"**)  remote\_conn.send(**"mpls label protocol ldp\n"**)  remote\_conn.send(**"mpls ldp router-id loopback0\n"**)  remote\_conn.send(**"mpls ip\n"**)  print(**"2.0"**)  conf\_mpls\_interfaces(remote\_conn)  back\_home(remote\_conn  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_vrf()  **Descripción:** Se crea una VRF con su respectiva router distinguisher, exportación e importación de susrutas.  **Método de llamado:** varios.  **def** config\_vrf(remote\_conn, name\_vrf, AS, vlan):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"ip vrf "**+name\_vrf+**"\n"**)    remote\_conn.send(**"rd "**+AS+**":"**+vlan+**"\n"**)  remote\_conn.send(**"route-target export "**+vlan+**":"**+vlan+**"\n"**) remote\_conn.send(**"route-target import "** + vlan + **":"** + vlan +  **"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_add\_interfaz\_vrf()  **Descripción:** Se agrega una interfaz a una VRF, para lo cual se requiere el nombre de la VRF, la VLAN delcliente, la interfaz por la que se conectará el cliente y el direccionamiento de la interfaz por donde se conectará el cliente. Esto se configura en un dispositivo tipo PE.  **Método de llamado:** varios.  **def** config\_add\_interfaz\_vrf(remote\_conn, name\_vrf, vlan, interfaz, ip,mascara):  remote\_conn.send(**"interface "**+interfaz+**"\n"**)  remote\_conn.send(**"no sh\n"**)  remote\_conn.send(**"no mpls ip\n"**)  remote\_conn.send(**"interface "**+interfaz+**"."**+vlan+**"\n"**)    remote\_conn.send(**"encapsulation dot1Q "**+ vlan + **"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  remote\_conn.send(**"ip vrf forwarding "** + name\_vrf + **"\n"**)  remote\_conn.send(**"ip add "**+ip+**" "**+mascara+ **"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_route\_PE\_CE()  **Descripción:** Permite configurar una ruta que le permite comunicarse por su respectiva cliente a uncliente, recibiendo como parámetro el nombre de la VRF y la red de la LAN del cliente.  **Método de llamado:** varios.  **def** config\_route\_PE\_CE(remote\_conn, name\_vrf, red\_CE, mascara\_CE,int\_salida):    remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"ip route vrf "**+name\_vrf+**" "**+red\_CE+**"**  **"**+mascara\_CE+**" "**+int\_salida+**"\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_iBGP()  **Descripción:** Permite configurar el protocolo iBGP entre los dispositivos PE. Para esto, se lee las ip deloopback de los dispositivos P con la función read\_file de un archivo de texto, ya que se agregará como vecinos iBGP a los vecinos OSPF, siendo los dispositivos P vecinos OSPF pero no pueden ser vecinos iBGP. Por lo que de esta manera se evita establecer a un dispositivo P como vecino iBGP.  **Método de llamado:** varios.  **def** config\_iBGP(remote\_conn):  bgp\_neigh = get\_ospf\_neig(remote\_conn)  rd\_p = read\_file()  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"router bgp 1\n"**)  **for** i **in** range(0,len(bgp\_neigh)):  **if not**(bgp\_neigh[i] **in** rd\_p):  remote\_conn.send(**"neighbor "**+bgp\_neigh[i]+**" remote-as**    **1\n"**)  remote\_conn.send(**"neighbor "** + bgp\_neigh[i] + **" update-source loopback0\n"**)  remote\_conn.send(**"neighbor "** + bgp\_neigh[i] + **" next-hop-**  **self\n"**)  remote\_conn.send(**"no auto-summary\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** config\_MP\_BGP()  **Descripción:** Permite configurar el protocolo MP-BGP entre los dispositivos PE. Para esto, se lee las ip deloopback de los dispositivos P con la función read\_file de un archivo de texto, ya que se agregará como vecinos MP-BGP a los vecinos OSPF, siendo los dispositivos P vecinos OSPF, pero no pueden ser vecinos MP-BGP. Por lo que de esta manera se evita establecer a un dispositivo P como vecino MP-BGP.  **Método de llamado:** varios.  **def** config\_MP\_BGP(remote\_conn):  bgp\_neigh = get\_ospf\_neig(remote\_conn)  rd\_p = read\_file()  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"router bgp 1\n"**)  remote\_conn.send(**"address-family vpnv4\n"**)  print(**"en BGP "**)  print(bgp\_neigh)  **for** i **in** range(0,len(bgp\_neigh)):  **if not** (bgp\_neigh[i] **in** rd\_p):  remote\_conn.send(**"neighbor "** + bgp\_neigh[i] + **"**    **activate\n"**)  remote\_conn.send(**"neighbor "** + bgp\_neigh[i] + **" send-**  **community extended\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  remote\_conn.send(**"exit-address-family\n"**)  remote\_conn.send(**"end\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** show\_res()  **Descripción:** Permite obtener la salida del comando show vrf, el cual es mostrado si se realiza con éxitotoda la configuración correspondiente a un enrutador tipo PE.  **Método de llamado:** varios.  **def** show\_res(remote\_conn):  copy = **False**  salida = **""**  remote\_conn.recv(1000)  remote\_conn.send(**"show vrf\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)    time.sleep(1)  output = (remote\_conn.recv(MAX\_BUFFER)).decode()  output = output.split(**'\r\n'**)  **for** i **in** range(0,len(output)):  **if**(**"show vrf" in** output[i]):  copy = **True**  **if** (copy == **True**):  salida+=(output[i]+**"\r\n"**)  print(output)  **return** salida  **Nombre de archivo:** funciones2.py  **Nombre de función:** save\_ID()  **Descripción:** Permite guardar en un archivo de texto la dirección ip de la interfaz loopback de unenrutador, esto se hace en los P y es considerado en la configuración de los protocolos iBGP y MP-BGP como ya se explicó en sus respectivas funciones de configuración.  **Método de llamado:** varios.  **def** save\_ID(remote\_conn):  back\_home(remote\_conn)  remote\_conn.send(**"show ip interface brief | include Loopback\n"**)  **while** (**not** remote\_conn.recv\_ready()):  time.sleep(0.5)    output = (remote\_conn.recv(MAX\_BUFFER)).decode() output = output.replace(**'\r\n'**, **'\n'**).split(**'\n'**) **for** i **in** range(0,len(output)):  line = output[i].split()  **if "Loopback" in** line[0]:  write\_file(line[1])  back\_home(remote\_conn)  **Nombre de archivo:** funciones2.py  **Nombre de función:** write\_file()  **Descripción:** Permite escribir en el archivo de texto rd\_p.txt la información dada por parámetro, la cualcorresponde a la dirección IP de loopback de un enrutador tipo P.  **Método de llamado:** varios.  **def** write\_file(dir\_id):    archivo = open(**'rd\_P.txt'**, **'a'**)  archivo.write(dir\_id+**'\n'**)  archivo.close() |

### **Database.py**

|  |
| --- |
| **Nombre de archivo:** database.py  **Descripción:** Script individual, al ser corrido se actualiza la tabla de usuarios y contraseñas del aplicativo.Reemplazar en los campos ‘USUARIO’ ‘PRIVILEGIO’ ‘CONTRASENA’ por las credenciales de la persona que desea agregar a la base de datos.  **import** sqlite3  **def** createTable():  connection = sqlite3.connect(**'login.db'**)  connection.execute(**"INSERT INTO USERS** **VALUES(?,?,?)"**,(**'USUARIO'**,**'PRIVILEGIO'**,**'CONTRASENA'**))    connection.commit()  result = connection.execute(**"SELECT \* FROM USERS"**)  **for** data **in** result:  print(**"Username : "**,data[0])  print(**"Privilegio : "**,data[1])  print(**"Password :"**,data[2])  connection.close()  createTable() |

# Soporte

Para más información o algún problema comunicarse con el equipo de soporte del software.

* [luiachav@espol.edu.ec](mailto:luiachav@espol.edu.ec)
* [genmaenc@espol.edu.ec](mailto:genmaenc@espol.edu.ec)
* [dtgarzon@espol.edu.ec](mailto:dtgarzon@espol.edu.ec)
* [msegarra@espol.edu.ec](mailto:msegarra@espol.edu.ec)