

**Escuela Superior Politécnica del Litoral**

Conmutación y Enrutamiento



***2T 2017***

MANUAL TÉCNICO

MPLStudio

MANUAL TÉCNICO

Contenido

[Overview 4](#_Toc505269453)

[Introducción 4](#_Toc505269454)

[Objetivo y alcances del sistema 4](#_Toc505269455)

[Normas, políticas y procedimientos 4](#_Toc505269456)

[Recursos de software 5](#_Toc505269457)

[Python 5](#_Toc505269458)

[Paramiko 5](#_Toc505269459)

[PyQT 5](#_Toc505269460)

[GNS3 5](#_Toc505269461)

[Recursos de Simulación 6](#_Toc505269462)

[Recursos de Hardware 6](#_Toc505269463)

[Caso de estudio 7](#_Toc505269464)

[Diagrama y Direccionamiento de la Red 7](#_Toc505269465)

[Configuraciones propuestas 8](#_Toc505269466)

[Configuración de direccionamiento entre dispositivos PE 8](#_Toc505269467)

[Configuración de un IGP (interior Gateway Protocol) en los PE1 y PE2 9](#_Toc505269468)

[Comunicación entre PE’s 9](#_Toc505269469)

[Intercambio de clientes entre PE1 y PE2 10](#_Toc505269470)

[Disminución de carga de procesamiento utilizando el protocolo CEF 11](#_Toc505269471)

[Creación de VRFs y asociación de sub-interfaces en los PE 11](#_Toc505269472)

[Configuración de la comunicación entre PE y CE 12](#_Toc505269473)

[Redistribución de rutas del aprendidas por el CE 13](#_Toc505269474)

[Configuración de direccionamiento en Provider 13](#_Toc505269475)

[Configuración de un IGP (Interior Gateway Protocol) para la comunicación entre PE1 y PE2 13](#_Toc505269476)

[Disminución de carga de procesamiento utilizando el protocolo CEF 14](#_Toc505269477)

[Configuración de los Customer Edge (Gye\_CFR y Cue\_AlmacenesJR) 14](#_Toc505269478)

[Código Fuente 15](#_Toc505269479)

[Login.py 16](#_Toc505269480)

[Remote.py 17](#_Toc505269481)

[SelectorLocal.py 19](#_Toc505269482)

[ConfiguracionBasica.py 20](#_Toc505269483)

[DireccionamientoLocal.py 21](#_Toc505269484)

[SelectorCredenciales.py 23](#_Toc505269485)

[AnadirUsuario.py 23](#_Toc505269486)

[Plantilla.py 24](#_Toc505269487)

[ConfigurarCE.py 26](#_Toc505269488)

[ConfigurarPE2.py 27](#_Toc505269489)

[Funciones2.py 30](#_Toc505269490)

[Database.py 40](#_Toc505269491)

[Soporte 41](#_Toc505269492)

# Overview

## Introducción

MPLStudio es un aplicativo de escritorio desarrollado para Windows que permite realizar conexiones remotas por medio del protocolo *ssh.* Con este aplicativo se puede automatizar la configuración de dispositivos intermedios del backbone de una red MPLS-VPN-L3 de 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 *VRF´s* para clientes del ISP.

Además, se incluyen un apartado que permite la configuración de direccionamiento de las interfaces disponibles en el dispositivo (enrutador), y la opción de configurar parámetros básicos del equipo como nombres de dominio, dns, hostname y credenciales de usuarios.

## Objetivo y alcances del sistema

* Automatizar la configuración de dispositivos intermediarios en una red de datos.
* Mejorar la interacción hombre-máquina mediante interfaces gráficas amigables.
* 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. Para más información ver ……

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, sin embargo, MPLStudio no verifica que la información que el usuario ingrese es correcta, es decir las fallas humanas de tipo diseño no están validadas.

# 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 programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

## 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.

## PyQT

PyQt es uno de los dos productos de Python más populares para el Qt cross-platform GUI/XML/SQL C++. PyQt desarrollado por Riverbank Computing Limited. PyQt contiene más de 620 clases que cubren interfaces gráficas de usuario, manejo de XML, comunicación de red, bases de datos SQL, navegación web y otras tecnologías disponibles en Qt.

Esta librería junto con su Framework *PyQt Designer* fueron usadas para la creación y diseño de ventanas e interfaces graficas del aplicativo.

## GNS3

GNS3 es un simulador gráfico de red que te permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellos.

Para permitir completar simulaciones, GNS3 está estrechamente vinculada con:

* Dynamips, un emulador de IOS que permite a los usuarios ejecutar imágenes binarias IOS de Cisco System.
* Dynagen, un front-end basado en texto para Dynamips
* Qemu y VirtualBox, para permitir utilizar máquinas virtuales como un firewall PIX.
* VPCS, un emulador de PC con funciones básicas de networking
* IOU (IOS on Unix), compilaciones especiales de IOS provistas por Cisco para correr directamente en sistemas UNIX y derivados.

GNS3 es una excelente herramienta complementaria a los verdaderos laboratorios para los administradores de redes de Cisco o las personas que quieren pasar sus CCNA, CCNP, CCIE DAC o certificaciones.

# 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

# Caso de estudio

## Diagrama y Direccionamiento de la Red

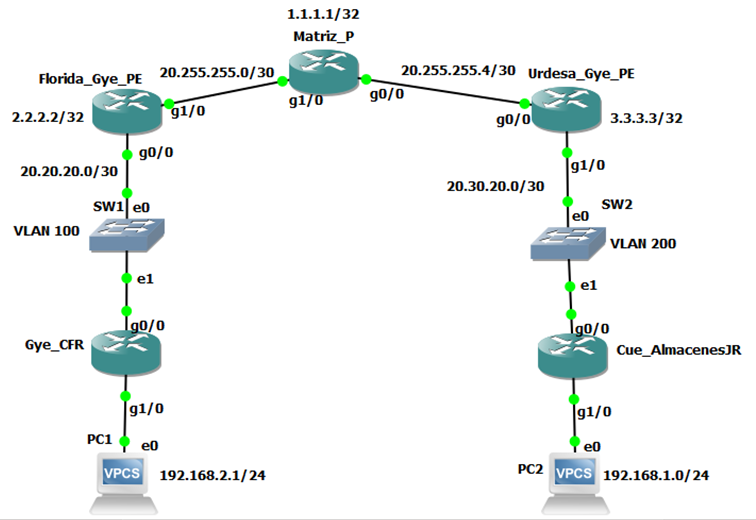


Ilustración 1

Bajo el diagrama de la red propuesto en la *Ilustración 1* se realizaron las configuraciones en dispositivos simulados para probar el correcto funcionamiento de la aplicación, obteniendo resultados favorables y replicables en cualquier otro escenario. En la *Ilustración 2* se incluye el direccionamiento en ipv4 proporcionado a cada dispositivo.

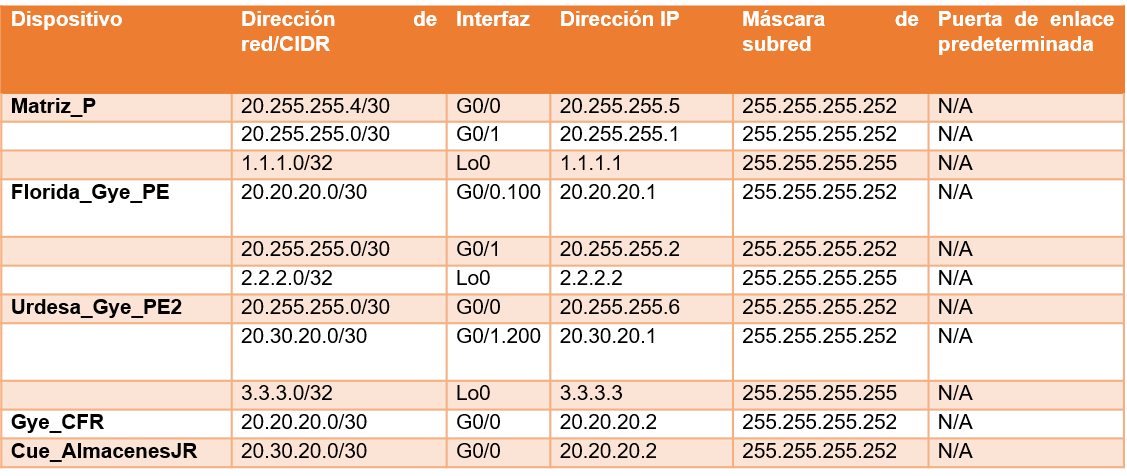


Ilustración 2

## Configuraciones propuestas

Este apartado muestra las configuraciones propuestas para los dispositivos que toman parte en la red MPLS-VPN L3.

### Configuración de direccionamiento entre dispositivos PE

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE(config)# interface Loopback0  Florida\_GYE\_PE (config-if)# ip address 2.2.2.2 255.255.255.255  Florida\_GYE\_PE (config-if)# interface GigabitEthernet1/0  Florida\_GYE\_PE (config-if)# description HACIA-P  Florida\_GYE\_PE (config-if)# ip address 20.255.255.2 255.255.255.252  Florida\_GYE\_PE (config-if)# duplex full  Florida\_GYE\_PE (config-if)# mpls ip  Florida\_GYE\_PE (config-if)# exit |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE(config)# interface Loopback0  Urdesa\_GYE\_PE (config-if)# ip address 3.3.3.3 255.255.255.255  Urdesa\_GYE\_PE (config-if)# interface GigabitEthernet0/0  Urdesa\_GYE\_PE (config-if)# description HACIA-P  Urdesa\_GYE\_PE (config-if)# ip address 20.255.255.6 255.255.255.252  Urdesa\_GYE\_PE (config-if)# mpls ip  Urdesa\_GYE\_PE (config-if)# exit |

### Configuración de un IGP (interior Gateway Protocol) en los PE1 y PE2

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# router ospf 1  Florida\_GYE\_PE (config)# network 2.2.2.2 0.0.0.0 area 0  Florida\_GYE\_PE (config)# network 20.255.255.0 0.0.0.3 area 0 |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# router ospf 1  Urdesa\_GYE\_PE (config)# network 3.3.3.3 0.0.0.0 area 0  Urdesa\_GYE\_PE (config)# network 20.255.255.0 0.0.0.3 area 0 |

### Comunicación entre PE’s

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# router bgp 1  Florida\_GYE\_PE (config-router)# neighbor 3.3.3.3 remote-as 1  Florida\_GYE\_PE (config-router)# neighbor 3.3.3.3 update-source Loopback0  Florida\_GYE\_PE (config-router)# neighbor 3.3.3.3 next-hop-self  Florida\_GYE\_PE (config-router)# no auto-summary  Florida\_GYE\_PE (config-router)# exit |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# router bgp 1  Urdesa\_GYE\_PE (config-router)# neighbor 2.2.2.2 remote-as 1  Urdesa\_GYE\_PE (config-router)# neighbor 2.2.2.2 update-source Loopback0  Urdesa\_GYE\_PE (config-router)# neighbor 2.2.2.2 next-hop-self  Urdesa\_GYE\_PE (config-router)# no auto-summary  Urdesa\_GYE\_PE (config-router)# exit |

### Intercambio de clientes entre PE1 y PE2

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# router bgp 1  Florida\_GYE\_PE (config-router)# address-family vpnv4  Florida\_GYE\_PE (config-router-af)# neighbor 3.3.3.3 activate  Florida\_GYE\_PE (config-router-af)# neighbor 3.3.3.3 send-community extended  Florida\_GYE\_PE (config-router-af)# exit-address-family |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# router bgp 1  Urdesa\_GYE\_PE (config-router)# address-family vpnv4  Urdesa\_GYE\_PE (config-router-af)# neighbor 2.2.2.2 activate  Urdesa\_GYE\_PE (config-router-af)# neighbor 2.2.2.2 send-community extended  Urdesa\_GYE\_PE (config-router-af)# exit-address-family |

### Disminución de carga de procesamiento utilizando el protocolo CEF

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# ip cef  Florida\_GYE\_PE (config)# mpls label protocol ldp  Florida\_GYE\_PE (config)# mpls ldp router-id Loopback0  Florida\_GYE\_PE (config)# mpls ip |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# ip cef  Urdesa\_GYE\_PE (config)# mpls label protocol ldp  Urdesa\_GYE\_PE (config)# mpls ldp router-id Loopback0  Urdesa\_GYE\_PE (config)# mpls ip |

### Creación de VRFs y asociación de sub-interfaces en los PE

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# ip vrf CE\_CRF  Florida\_GYE\_PE (config-vrf)# rd 65535:100  Florida\_GYE\_PE (config-vrf)# route-target export 100:100  Florida\_GYE\_PE (config-vrf)# route-target import 100:100  Florida\_GYE\_PE (config-vrf)# route-target import 200:200  Florida\_GYE\_PE (config-if)# interface GigabitEthernet0/0  Florida\_GYE\_PE (config-if)# no shutdown  Florida\_GYE\_PE (config-if)# interface GigabitEthernet0/0.100  Florida\_GYE\_PE (config-if)# description HACIA-SW1-CE  Florida\_GYE\_PE (config-if)# encapsulation dot1Q 100  Florida\_GYE\_PE (config-if)# ip vrf forwarding CE\_CRF  Florida\_GYE\_PE (config-if)# ip address 20.20.20.1 255.255.255.252  Florida\_GYE\_PE (config-if)# exit |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# ip vrf CE\_AlmacenesJr  Urdesa\_GYE\_PE (config-vrf)# rd 65535:100  Urdesa\_GYE\_PE (config-vrf)# route-target export 200:200  Urdesa\_GYE\_PE (config-vrf)# route-target import 200:200  Urdesa\_GYE\_PE (config-vrf)# route-target import 100:100  Urdesa\_GYE\_PE (config-if)# interface GigabitEthernet1/0  Urdesa\_GYE\_PE (config-if)# no shutdown  Urdesa\_GYE\_PE (config-if)# interface GigabitEthernet0/1.200  Urdesa\_GYE\_PE (config-if)# description HACIA-SW2-CE  Urdesa\_GYE\_PE (config-if)# encapsulation dot1Q 100  Urdesa\_GYE\_PE (config-if)# ip vrf forwarding CE\_AlmacenesJr  Urdesa\_GYE\_PE (config-if)# ip address 20.30.20.1 255.255.255.252  Urdesa\_GYE\_PE (config-if)# exit |

### Configuración de la comunicación entre PE y CE

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# ip route vrf CE\_CRF 192.168.2.1 255.255.255.0 20.20.20.2 |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# ip route vrf CE\_AlmacenesJr 192.168.1.1 255.255.255.0 20.30.20.1 |

### Redistribución de rutas del aprendidas por el CE

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Florida\_GYE\_PE  **Descripción:** | |
|  | Florida\_GYE\_PE (config)# router bgp 1  Florida\_GYE\_PE (config-router)# address-family ipv4 vrf CEA  Florida\_GYE\_PE (config-router-af)# redistribute static |
| **Nombre de dispositivo:** Urdesa\_GYE\_PE  **Descripción:** | |
|  | Urdesa\_GYE\_PE (config)# router bgp 1  Urdesa\_GYE\_PE (config-router-af)# address-family ipv4 vrf CEA  Urdesa\_GYE\_PE (config-router-af)# redistribute static  Urdesa\_GYE\_PE (config-router-af)# exit-address-family |

### Configuración de direccionamiento en Provider

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Matriz\_P  **Descripción:** | |
|  | Matriz\_P(config)# interface Loopback0  Matriz\_P(config-if)# ip address 1.1.1.1 255.255.255.255  Matriz\_P(config-if)# interface GigabitEthernet0/0  Matriz\_P(config-if)# description HACIA-Florida\_GYE\_PE  Matriz\_P(config-if)# ip address 20.255.255.5 255.255.255.252  Matriz\_P(config-if)# mpls ip  Matriz\_P(config-if)# interface GigabitEthernet0/1  Matriz\_P(config-if)# description HACIA-Urdesa\_GYE\_PE  Matriz\_P(config-if)# ip address 20.255.255.1 255.255.255.252  Matriz\_P(config-if)# mpls ip |

### Configuración de un IGP (Interior Gateway Protocol) para la comunicación entre PE1 y PE2

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Matriz\_P  **Descripción:** | |
|  | Matriz\_P(config)# router ospf 1  Matriz\_P(config-router)# network 1.1.1.1 0.0.0.0 area 0  Matriz\_P(config-router)# network 20.255.255.0 0.0.0.7 area 0  Matriz\_P(config-router)# exit |

### Disminución de carga de procesamiento utilizando el protocolo CEF

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Matriz\_P  **Descripción:** | |
|  | Matriz\_P(config)# ip cef  Matriz\_P(config)# mpls label protocol ldp  Matriz\_P(config)# mpls ldp router-id Loopback0  Matriz\_P(config)# mpls ip |

### Configuración de los Customer Edge (Gye\_CFR y Cue\_AlmacenesJR)

|  |  |
| --- | --- |
| **Nombre de dispositivo:** Gye\_CFR  **Descripción:** | |
|  | Gye\_CFR (config)# interface GigabitEthernet0/0  Gye\_CFR (config-if)# description HACIA-SW1-Florida\_Gye\_PE  Gye\_CFR (config-if)# ip address 20.20.20.2 255.255.255.252  Gye\_CFR (config-if)# no shutdown  Gye\_CFR (config-if)# interface GigabitEthernet0/1  Gye\_CFR (config-if)# description LAN  Gye\_CFR (config-if)# ip address 192.168.2.2 255.255.255.0  Gye\_CFR (config-if)# no shutdown  Gye\_CFR (config-if)# exit  Gye\_CFR (config)# ip route 0.0.0.0 0.0.0.0 20.20.20.2 |
| **Nombre de dispositivo:** Cue\_AlmacenesJR  **Descripción:** | |
|  | Cue\_AlmacenesJR (config)# interface GigabitEthernet0/0  Cue\_AlmacenesJR (config-if)# description HACIA-SW2-Urdesa\_GYE\_PE  Cue\_AlmacenesJR (config-if)# ip address 20.30.20.2 255.255.255.252  Cue\_AlmacenesJR (config-if)# no shutdown  Cue\_AlmacenesJR (config-if)# interface GigabitEthernet0/1  Cue\_AlmacenesJR (config-if)# description LAN  Cue\_AlmacenesJR (config-if)# ip address 192.168.1.2 255.255.255.0  Cue\_AlmacenesJR (config-if)# no shutdown  Cue\_AlmacenesJR (config-if)# exit  Cue\_AlmacenesJR (config)# ip route 0.0.0.0 0.0.0.0 20.30.20.2 |

# Código Fuente

En esta sección se describirán las distintas funciones y su implementación dentro del proyecto. Se excluyen de esta explicación los métodos creados por el framework PyQt Designer.

## 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 y mostrándola 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 y mostrándola 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 la ventana 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 y mostrándola 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 son campos 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()) *# !!!!!!!!!!!!!AQUI!!!!!!!!!!!!!!!!!!!!!!!* 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 y mostrándola 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 y mostrándola 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 y mostrándola 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 y mostrándola 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 y mostrándola 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 ingresar direcciones 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 para configurar, 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):  n = 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() |
| **Nombre de archivo:** DireccionamientoLocal.py  **Nombre de función:** set#IP () \*Reemplazar # por numero ordinal ej: Primera,Segunda,Tercera,etc.\*  **Descripción:** Valida si el usuario ha ingresado datos para determinado campo de ipv4, de ser correcta la validación procede a la configuración en el enrutador, caso contrario muestra mensaje de error.  **Método de llamado:** setDireccionamiento() | |
|  | **####Ejemplo con PrimeraIP**  **def** setPrimeraIp(self, int, remote\_conn):  m1 = self.txt\_mask1\_1.text()  m2 = self.txt\_mask1\_2.text()  m3 = self.txt\_mask1\_3.text()  m4 = self.txt\_mask1\_4.text()  ip1 = self.txt\_ip1\_1.text()  ip2 = self.txt\_ip1\_2.text()  ip3 = self.txt\_ip1\_3.text()  ip4 = self.txt\_ip1\_4.text()  valIp = ip1 + ip2 + ip3 + ip4  valMask = m1 + m2 + m3 + m4  **if** (valIp != **"" and** valMask != **""**):  ip = ip1 + **"."** + ip2 + **"."** + ip3 + **"."** + ip4  mask = m1 + **"."** + m2 + **"."** + m3 + **"."** + m4  **if** (type(remote\_conn) **is** paramiko.channel.Channel):  funciones2.config\_dir(remote\_conn, int, ip, mask)  ctypes.windll.user32.MessageBoxW(0, **"Se configuro la interfaz"** + int,  **"Exito"**, 0)  **else**:  funciones\_com.config\_dir(remote\_conn, int, ip, mask) |

## 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 muestra en 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 al enrutador 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 y mostrándola 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 y domain 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\_dns1\_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\_dns2\_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 y mostrándola 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 y mostrándola 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 y mostrándola 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 y mostrándola 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 y mostrándola 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 y mostrándola 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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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\_pagin()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** redistribute\_vrf()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **Método de llamado:** varios. | |
|  | **def** redistribute\_vrf(remote\_conn, name\_vrf):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"router bgp 1\n"**)  remote\_conn.send(**"address-family ipv4 vrf "**+name\_vrf+**"\n"**)  remote\_conn.send(**"redistribute static\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\_route\_CE()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **Método de llamado:** varios. | |
|  | **def** conf\_route\_CE(remote\_conn, ip\_salida):  remote\_conn.send(**"conf t\n"**)  remote\_conn.send(**"no router ospf 1\n"**)  remote\_conn.send(**"ip route 0.0.0.0 0.0.0.0 "**+ip\_salida+**" \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:** config\_iBGP()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** sabe\_ID()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **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:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **Método de llamado:** varios. | |
|  | **def** write\_file(dir\_id):  archivo = open(**'rd\_P.txt'**, **'a'**)  archivo.write(dir\_id+**'\n'**)  archivo.close() |
| **Nombre de archivo:** funciones2.py  **Nombre de función:** read\_file()  **Descripción:** ¡!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  **Método de llamado:** varios. | |
|  | **def** read\_file():  rd\_p = []  archivo = open(**'rd\_P.txt'**, **'r'**)  lines = archivo.readlines()  **for** i **in** range (0, len(lines)):  rd\_p.append(lines[i][:-1])  print(rd\_p)  **return** rd\_p |

## Database.py

|  |  |
| --- | --- |
| **Nombre de archivo:** database.py.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.

* [jpcadena@espol.edu.ec](mailto:jpcadena@espol.edu.ec)
* [ilcedeno@espol.edu.ec](mailto:ilcedeno@espol.edu.ec)
* [caencede@espol.edu.ec](mailto:caencede@espol.edu.ec)
* [yofiguer@espol.edu.ec](mailto:yofiguer@espol.edu.ec)
* [aaorella@espol.edu.ec](mailto:aaorella@espol.edu.ec)
* [lvrivas@espol.edu.ec](mailto:lvrivas@espol.edu.ec)