**Manual Técnico**

GRUPO #4

Tema: Sistema de Configuración del protocolo de enrutamiento BGP para la

interconexión entre ISP.

Profesora:

* Msig. Adriana Collaguazo Jaramillo

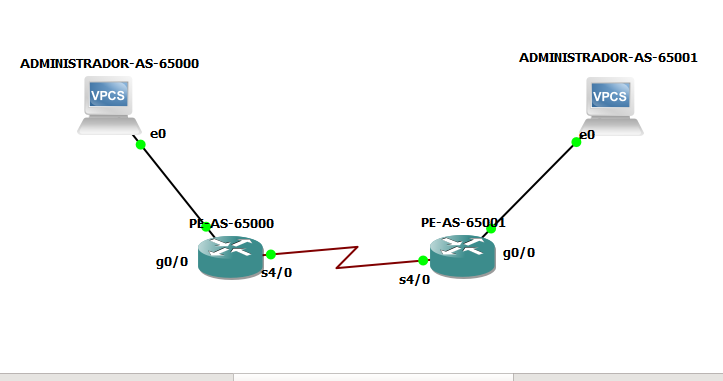
Integrantes:

* Nécker Iván Espinosa Feijoo
* Alex Antonio Mendoza Santos
* Sophia Denisse Gómez Quimí
* Jamil Andrés Nagua Campi
* Andrés Xavier Vizueta Aldaz

Diagrama de la red:

Dispositivo donde se ejecuta la aplicación

Dispositivo donde se ejecuta la aplicación



**Loopback**

**Loopback**

El diagrama de red es simple, pero a su vez es la mejor forma para poder explicar y ejecutar lo que requiere el proyecto.

El proyecto consiste en la conexión entre ISP´s, mediante el protocolo de enrutamiento BGP, siendo cada ISP cada uno de los routers, cada router tiene un numero autónomo.

Para la simulación en el software GNS3, en lugar de un servidor usamos la maquina física, siendo conectada a travez del icono “cloud”, el cual se interconecta mediante un adaptador de red virtual previamente creado y asignado una dirección IP.

Recursos de hardware/software:

* Hardware:
  + 2 enrutadores marca: CISCO modelo: 2811
  + Dos ordenadores portátiles con la aplicación previamente instalada.
* Software:
  + GNS3 version 2.1.2 para Windows (64-bit)
  + Python 2.7.12 con las librerías: Telnetlib, easygui, wxPython, wx
  + WxFormBuilder versión 3.5
  + Plataforma de desarrollo colaborativo GitHub
  + Aplicación web Asana

Código fuente:

* Configuración previa desde conexión serial al enrutador a configurar:

enable

conf t

username NOMBRE\_DE\_USUARIO\_ADMINISTRADOR password PASSWORD\_DE\_USUARIO\_ADMINISTRADOR

enable password PASSWORD\_DEL\_ADMINISTRADOR

line vty 0 4

login local

transport input all

int INTERFAZ\_DE\_LA\_LOOPBACK

ip address DIRECCION\_IP MASCARA\_DE\_LA\_RED

no shutdown

end

debug telnet

wr

* Código de la interfaz gráfica:

# -\*- coding: utf-8 -\*-

###########################################################################

## Python code generated with wxFormBuilder (version Jun 17 2015)

## http://www.wxformbuilder.org/

##

## PLEASE DO "NOT" EDIT THIS FILE!

###########################################################################

import wx

import wx.xrc

import easygui as eg

import funciones\_telnet

###########################################################################

## Class principal

###########################################################################

class principal(wx.Frame):

def \_\_init\_\_(self, parent):

wx.Frame.\_\_init\_\_(self, parent, id=wx.ID\_ANY, title=u"PROYECTO DE CONMUTACION Y ENRUTAMIENTO",

pos=wx.DefaultPosition, size=wx.Size(500, 500),

style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL)

self.SetSizeHintsSz(wx.DefaultSize, wx.DefaultSize)

gbSizer4 = wx.GridBagSizer(0, 0)

gbSizer4.SetFlexibleDirection(wx.BOTH)

gbSizer4.SetNonFlexibleGrowMode(wx.FLEX\_GROWMODE\_SPECIFIED)

sbSizer11 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, u"LOGIN"), wx.VERTICAL)

gbSizer8 = wx.GridBagSizer(0, 0)

gbSizer8.SetFlexibleDirection(wx.BOTH)

gbSizer8.SetNonFlexibleGrowMode(wx.FLEX\_GROWMODE\_SPECIFIED)

self.m\_staticText35 = wx.StaticText(sbSizer11.GetStaticBox(), wx.ID\_ANY, u"USUARIO",

wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText35.Wrap(-1)

gbSizer8.Add(self.m\_staticText35, wx.GBPosition(0, 0), wx.GBSpan(1, 1), wx.ALL, 5)

self.user = wx.TextCtrl(sbSizer11.GetStaticBox(), wx.ID\_ANY, u"user", wx.DefaultPosition, wx.DefaultSize, 0)

gbSizer8.Add(self.user, wx.GBPosition(0, 1), wx.GBSpan(1, 1), wx.ALL, 5)

self.password1234 = wx.StaticText(sbSizer11.GetStaticBox(), wx.ID\_ANY, u"PASSWORD", wx.DefaultPosition, wx.DefaultSize, 0)

self.password1234.Wrap(-1)

gbSizer8.Add(self.password1234, wx.GBPosition(1, 0), wx.GBSpan(1, 1), wx.ALL, 5)

self.password = wx.TextCtrl(sbSizer11.GetStaticBox(), wx.ID\_ANY, u"xxxxxxx", wx.DefaultPosition, wx.DefaultSize, wx.TE\_PASSWORD)

gbSizer8.Add(self.password, wx.GBPosition(1, 1), wx.GBSpan(1, 1), wx.ALL, 5)

sbSizer11.Add(gbSizer8, 1, wx.EXPAND, 5)

gbSizer4.Add(sbSizer11, wx.GBPosition(3, 2), wx.GBSpan(1, 1), wx.EXPAND, 5)

sbSizer12 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, wx.EmptyString), wx.VERTICAL)

self.m\_staticText34 = wx.StaticText(sbSizer12.GetStaticBox(), wx.ID\_ANY, u"SISTEMA DE AUTENTICACION", wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText34.Wrap(-1)

sbSizer12.Add(self.m\_staticText34, 0, wx.ALL, 5)

self.m\_bitmap3 = wx.StaticBitmap(sbSizer12.GetStaticBox(), wx.ID\_ANY,wx.Bitmap("fondo.jpg", wx.BITMAP\_TYPE\_ANY), wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer12.Add(self.m\_bitmap3, 0, wx.ALL, 5)

gbSizer4.Add(sbSizer12, wx.GBPosition(1, 3), wx.GBSpan(1, 1), 0, 5)

self.btn\_validar = wx.Button(self, wx.ID\_ANY, u"VALIDAR", wx.DefaultPosition, wx.DefaultSize, 0)

gbSizer4.Add(self.btn\_validar, wx.GBPosition(6, 2), wx.GBSpan(1, 1), wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALIGN\_CENTER\_VERTICAL, 5)

self.btn\_salir1 = wx.Button(self, wx.ID\_ANY, u"SALIR", wx.DefaultPosition, wx.DefaultSize, 0) gbSizer4.Add(self.btn\_salir1, wx.GBPosition(6, 3), wx.GBSpan(1, 1),wx.ALL | wx.ALIGN\_CENTER\_VERTICAL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.SetSizer(gbSizer4)

self.Layout()

self.Centre(wx.BOTH)

# Connect Events

self.btn\_validar.Bind(wx.EVT\_BUTTON, self.validar\_ingreso)

self.btn\_salir1.Bind(wx.EVT\_BUTTON, self.salir1)

def \_\_del\_\_(self):

pass

# Virtual event handlers, overide them in your derived class

def validar\_ingreso(self, event):

x = 0

f = open("base\_de\_datos.txt", 'r')

a = f.readlines()

for linea in a:

lista = linea.split(',')

usr = lista[0]

contra = lista[1]

contra2 = (contra[0:(len(contra) - 1)])

if (self.user.GetLineText(0) == usr and self.password.GetLineText(0) == contra2):

x = 1

if (x == 1):

f.close()

frame2.Show()

else:

msg = "usario o contraseña incorrecto";

titulo = "error"

choices = ["yes", "no"]

reply = eg.buttonbox(msg, title=titulo, choices=choices)

def salir1(self, event):

frame.Destroy()

Comentario: Este bloque de código se genera a través de la aplicación wxFormBuilder, que es un software que permite la realización de interfaz gráfica y genera el archivo de extensión .py. Esta muestra la ventana del login inicial para poder acceder al sistema.

###########################################################################

## Class opciones

###########################################################################

class opciones(wx.Frame):

def \_\_init\_\_(self, parent):

wx.Frame.\_\_init\_\_(self, parent, id=wx.ID\_ANY, title=u"BIENVENIDOS", pos=wx.DefaultPosition,

size=wx.Size(500, 300), style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL)

self.SetSizeHintsSz(wx.DefaultSize, wx.DefaultSize)

bSizer2 = wx.BoxSizer(wx.VERTICAL)

self.m\_staticText5 = wx.StaticText(self, wx.ID\_ANY, u"ELIJA UNA OPCION A REALIZAR", wx.DefaultPosition, wx.DefaultSize, wx.ALIGN\_CENTRE)

self.m\_staticText5.Wrap(-1)

bSizer2.Add(self.m\_staticText5, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL | wx.EXPAND, 5)

bSizer2.AddSpacer(5)

self.btn\_con\_basica = wx.Button(self, wx.ID\_ANY, u"CONFIGURACION BASICA", wx.DefaultPosition, wx.DefaultSize, 0)

bSizer2.Add(self.btn\_con\_basica, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

bSizer2.AddSpacer(5)

self.btn\_bgp = wx.Button(self, wx.ID\_ANY, u"CONFIGURAR BGP", wx.DefaultPosition, wx.DefaultSize, 0)

bSizer2.Add(self.btn\_bgp, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

bSizer2.AddSpacer(5)

self.btn\_shows = wx.Button(self, wx.ID\_ANY, u"REALIZAR SHOWS", wx.DefaultPosition, wx.DefaultSize, 0)

bSizer2.Add(self.btn\_shows, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

bSizer2.AddSpacer(5)

self.btn\_salir = wx.Button(self, wx.ID\_ANY, u"SALIR", wx.DefaultPosition, wx.DefaultSize, 0)

bSizer2.Add(self.btn\_salir, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.SetSizer(bSizer2)

self.Layout()

self.Centre(wx.BOTH)

# Connect Events

self.btn\_con\_basica.Bind(wx.EVT\_BUTTON, self.configuracion\_basica)

self.btn\_bgp.Bind(wx.EVT\_BUTTON, self.configurar\_bgp)

self.btn\_salir.Bind(wx.EVT\_BUTTON, self.salir2)

def \_\_del\_\_(self):

pass

# Virtual event handlers, overide them in your derived class

def configuracion\_basica(self, event):

frame3.Show()

def configurar\_bgp(self, event):

frame4.Show()

def salir2(self, event):

frame5.Destroy()

Comentario: Este bloque de código se genera a través de la aplicación wxFormBuilder, que es un software que permite la realización de interfaz gráfica y genera el archivo de extensión .py. Estas líneas de comando muestran el frame donde tenemos los botones de opciones disponibles dentro del sistema de configuración BGP.

###########################################################################

## Class Conf\_Basica

###########################################################################

class Conf\_Basica(wx.Frame):

def \_\_init\_\_(self, parent):

wx.Frame.\_\_init\_\_(self, parent, id=wx.ID\_ANY, title=u"CONFIGURACION BASICA", pos=wx.DefaultPosition, size=wx.Size(562, 356), style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL)

self.SetSizeHintsSz(wx.DefaultSize, wx.DefaultSize)

gbSizer3 = wx.GridBagSizer(0, 0)

gbSizer3.SetFlexibleDirection(wx.BOTH)

gbSizer3.SetNonFlexibleGrowMode(wx.FLEX\_GROWMODE\_SPECIFIED)

self.m\_staticText26 = wx.StaticText(self, wx.ID\_ANY, u"--\*-- UD VA A REALIZAR LA CONFIGURACION BASICA DEL ROUTER --\*--", wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText26.Wrap(-1)

gbSizer3.Add(self.m\_staticText26, wx.GBPosition(0, 3), wx.GBSpan(1, 1),.ALL | wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALIGN\_CENTER\_VERTICAL, 5)

sbSizer6 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, u"CONFIGURAR"), wx.VERTICAL)

self.m\_staticText27 = wx.StaticText(sbSizer6.GetStaticBox(), wx.ID\_ANY, u"INGRESE EL HOSTNAME", wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText27.Wrap(-1)

sbSizer6.Add(self.m\_staticText27, 0, wx.ALL, 5)

self.hostname = wx.TextCtrl(sbSizer6.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer6.Add(self.hostname, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.btn\_configuracion\_basica = wx.Button(sbSizer6.GetStaticBox(), wx.ID\_ANY, u"CONFIGURACION BASICA", wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer6.Add(self.btn\_configuracion\_basica, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

gbSizer3.Add(sbSizer6, wx.GBPosition(3, 3), wx.GBSpan(1, 1), wx.EXPAND | wx.SHAPED | wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALIGN\_CENTER\_VERTICAL, 5)

sbSizer8 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, wx.EmptyString), wx.VERTICAL)

self.btn\_salir1 = wx.Button(sbSizer8.GetStaticBox(), wx.ID\_ANY, u"SALIR", wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer8.Add(self.btn\_salir1, 0, wx.ALL | wx.EXPAND, 5)

sbSizer8.AddSpacer(5)

self.btn\_regresar = wx.Button(sbSizer8.GetStaticBox(), wx.ID\_ANY, u"REGRESAR", wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer8.Add(self.btn\_regresar, 0, wx.ALL | wx.BOTTOM | wx.EXPAND, 5)

sbSizer8.AddSpacer(5)

gbSizer3.Add(sbSizer8, wx.GBPosition(5, 3), wx.GBSpan(1, 1), wx.EXPAND | wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALIGN\_CENTER\_VERTICAL, 5)

self.SetSizer(gbSizer3)

self.Layout()

self.Centre(wx.BOTH)

# Connect Events

self.btn\_configuracion\_basica.Bind(wx.EVT\_BUTTON, self.realizar\_conf\_bas)

self.btn\_salir1.Bind(wx.EVT\_BUTTON, self.salir3)

self.btn\_regresar.Bind(wx.EVT\_BUTTON, self.returned1)

def \_\_del\_\_(self):

pass

# Virtual event handlers, overide them in your derived class

def realizar\_conf\_bas(self, event):

host = str(self.hostname.GetLineText(0))

if (host != ""):

funciones\_telnet.configuracionBasica(tn, host)

else:

print("error")

def salir3(self, event):

frame2.PageDown()

def returned1(self, event):

self.Close(True)

Comentario: Este bloque de código se genera a través de la aplicación wxFormBuilder, que es un software que permite la realización de interfaz gráfica y genera el archivo de extensión .py. Estas líneas de código ejecutan un frame, que forma parte de la opción de configuración básica dentro del sistema.

###########################################################################

## Class op2

###########################################################################

class op2(wx.Frame):

def \_\_init\_\_(self, parent):

wx.Frame.\_\_init\_\_(self, parent, id=wx.ID\_ANY, title=u"CONFIGURACION BGP", pos=wx.DefaultPosition, size=wx.Size(779, 598), style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL)

self.SetSizeHintsSz(wx.DefaultSize, wx.DefaultSize)

gbSizer2 = wx.GridBagSizer(0, 0)

gbSizer2.SetFlexibleDirection(wx.BOTH)

gbSizer2.SetNonFlexibleGrowMode(wx.FLEX\_GROWMODE\_SPECIFIED)

self.m\_staticText12 = wx.StaticText(self, wx.ID\_ANY,u--\*-- UD VA A CONFIGURAR EL PROTOCOLO DE

RUTAMIENTO BGP --\*-- ",wx.DefaultPosition, wx.DefaultSize, wx.ALIGN\_CENTRE)

self.m\_staticText12.Wrap(-1)

gbSizer2.Add(self.m\_staticText12, wx.GBPosition(2, 1), wx.GBSpan(1, 10), wx.ALL |

wx.ALIGN\_CENTER\_HORIZONTAL | wx.EXPAND, 5)

sbSizer2 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, u"NUMERO DE AS"), wx.VERTICAL)

self.m\_staticText14 = wx.StaticText(sbSizer2.GetStaticBox(), wx.ID\_ANY, u"INGRESE EL NUMERO DE AS LOCAL", wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText14.Wrap(-1)

sbSizer2.Add(self.m\_staticText14, 0, wx.ALL, 5)

self.numero\_as\_local = wx.TextCtrl(sbSizer2.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer2.Add(self.numero\_as\_local, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.m\_staticText15 = wx.StaticText(sbSizer2.GetStaticBox(), wx.ID\_ANY, u"INGRESE EL NUMERO DE AS REMOTO",wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText15.Wrap(-1)

sbSizer2.Add(self.m\_staticText15, 0, wx.ALL, 5)

self.numero\_as\_remoto = wx.TextCtrl(sbSizer2.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer2.Add(self.numero\_as\_remoto, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.btn\_guardar\_as = wx.Button(sbSizer2.GetStaticBox(), wx.ID\_ANY, u"GUARDAR AS", wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer2.Add(self.btn\_guardar\_as, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.mostrar = wx.TextCtrl(sbSizer2.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition,

wx.DefaultSize, wx.TE\_READONLY)

sbSizer2.Add(self.mostrar, 1, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

gbSizer2.Add(sbSizer2, wx.GBPosition(4, 2), wx.GBSpan(1, 1), wx.ALIGN\_CENTER | wx.EXPAND | wx.SHAPED, 5)

sbSizer3 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, u"CONFIGURAR VECINDAD"), wx.VERTICAL)

self.m\_staticText17 = wx.StaticText(sbSizer3.GetStaticBox(), wx.ID\_ANY, u"INGRESE LA DIRECCION IP DEL VECINO", wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText17.Wrap(-1)

sbSizer3.Add(self.m\_staticText17, 0, wx.ALL, 5)

self.ip\_vecino = wx.TextCtrl(sbSizer3.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer3.Add(self.ip\_vecino, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.btn\_configurar\_vecino = wx.Button(sbSizer3.GetStaticBox(), wx.ID\_ANY, u"CONFIGURAR VECINO",

wx.DefaultPosition, wx.DefaultSize, 0)

sbSizer3.Add(self.btn\_configurar\_vecino, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

gbSizer2.Add(sbSizer3, wx.GBPosition(4, 4), wx.GBSpan(1, 1), wx.ALIGN\_CENTER | wx.EXPAND | wx.SHAPED, 5)

self.btn\_regresar2 = wx.Button(self, wx.ID\_ANY, u"REGRESAR", wx.DefaultPosition, wx.DefaultSize, 0)

gbSizer2.Add(self.btn\_regresar2, wx.GBPosition(10, 4), wx.GBSpan(1, 1), wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.btn\_salir3 = wx.Button(self, wx.ID\_ANY, u"SALIR", wx.DefaultPosition, wx.DefaultSize, 0)

gbSizer2.Add(self.btn\_salir3, wx.GBPosition(10, 2), wx.GBSpan(1, 1), wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

sbSizer4 = wx.StaticBoxSizer(wx.StaticBox(self, wx.ID\_ANY, u"CONFIGURAR REDES"), wx.VERTICAL)

self.m\_staticText19 = wx.StaticText(sbSizer4.GetStaticBox(), wx.ID\_ANY, u"INGRESE LAS REDES A PUBLICAR",wx.DefaultPosition, wx.DefaultSize, 0)

self.m\_staticText19.Wrap(-1)

sbSizer4.Add(self.m\_staticText19, 0, wx.ALL, 5)

self.network = wx.TextCtrl(sbSizer4.GetStaticBox(), wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition,

wx.Size(-1, -1), 0)

bSizer4.Add(self.network, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

self.btn\_configurar\_redes = wx.Button(sbSizer4.GetStaticBox(), wx.ID\_ANY, u"CONFIGURAR REDES",

wx.DefaultPosition, wx.DefaultSize, 0)

sSizer4.Add(self.btn\_configurar\_redes, 0, wx.ALL | wx.ALIGN\_CENTER\_HORIZONTAL, 5)

gbSizer2.Add(sbSizer4, wx.GBPosition(8, 3), wx.GBSpan(1, 1), wx.EXPAND, 5)

self.SetSizer(gbSizer2)

self.Layout()

self.Centre(wx.BOTH)

# Connect Events

self.btn\_guardar\_as.Bind(wx.EVT\_BUTTON, self.guardarAS)

self.btn\_configurar\_vecino.Bind(wx.EVT\_BUTTON, self.vecinosGP)

self.btn\_regresar2.Bind(wx.EVT\_BUTTON, self.returned2)

self.btn\_salir3.Bind(wx.EVT\_BUTTON, self.salir4)

self.btn\_configurar\_redes.Bind(wx.EVT\_BUTTON, self.conf\_redes)

def \_\_del\_\_(self):

pass

# Virtual event handlers, overide them in your derived class

def guardarAS(self, event):

local = str(self.numero\_as\_local.GetLineText(0))

localas = local

remoto = str(self.numero\_as\_remoto.GetLineText(0))

remotoas = remoto

if (local != "") and (remoto != ""):

self.mostrar.SetValue("Local: " + local + " Remoto: " + remoto)

funciones\_telnet.SesionBGP(tn, localas)

print ("Se guardo el as")

else:

print("error")

def vecinosGP(self, event):

vecino = str(self.ip\_vecino.GetLineText(0))

if (vecino != ""):

funciones\_telnet.configurarVecino(vecino, localas, remotoas, tn)

print("se guardo correntamente")

else:

print("error")

def returned2(self, event):

event.Skip()

def salir4(self, event):

event.Skip()

def conf\_redes(self, event):

red = str(self.network.GetLineText(0))

if (red != ""):

funciones\_telnet.configurarNetwork(tn, red, localas)

print ("se guardo la conexion de red ")

# tn.read\_all()

else:

print("error")

# s=tn.read\_all()

# print(s)

Comentario: Este bloque de código se genera a través de la aplicación wxFormBuilder, que es un software que permite la realización de interfaz gráfica y genera el archivo de extensión .py. Estas líneas de código generan el frame de la opción de configuración BGP, en ella encontramos los tres requerimientos para poder establecer una sesión BGP entre dos sistemas autónomos, como son la configuración de los números AS, las direcciones IP de los enrutadores vecinos y las redes que tienen alcance dentro de la sesión BGP.

tn = funciones\_telnet.conexion\_telnet("192.168.137.2", "admin", "admin")

localas = ""

remotoas = ""

app = wx.App(False)

frame = principal(None)

frame2 = opciones(None)

frame3 = op1(None)

frame4 = op2(None)

frame5 = op3(None)

frame.Show(True)

app.MainLoop()

Comentario: En estas líneas de código se manda a ejecutar la interfaz grafica del frame principal, que es la del login. Adicional a eso se crean dos variables que capturan el valor del sistema autónomo local y el remoto, para poder luego establecer las configuración de los enrutadores vecinos y las redes dentro de la sesión BGP.

Código de las funciones:

import getpass

import sys

import telnetlib

import time

#Funcion para realizar la conexion telnet con el router al que se va a aplicar el enrutamiento

def conexion\_telnet(ip,user,password):

HOST = ip

tn = telnetlib.Telnet(HOST,port='23')

tn.read\_until("Username: ")

tn.write(user + "\n")

if password:

# tn.read\_until("Password: ")

tn.write(password + "\n")

print("Conexion telnet realizada con exito")

return tn

Comentario: esta función devuelve el enlace de la conexión telnet establecida entre el ordenador donde se ejecuta la aplicación y el enrutador a configurar.

#Funcion para aplicar la configuracion basica del router

def configuracionBasica(tn, hostname):

tn.write("enable \n")

tn.write("admin\n")

tn.write("conf t \n")

tn.write("hostname "+ hostname+" \n")

tn.write("username monitoreo privilege 5 secret monitoreo \n")

tn.write("banner motd # ACCESO SOLO A PERSONAL AUTORIZADO# \n")

tn.write("line vty 0 4 \n")

tn.write("transport input all \n")

tn.write("login local \n")

tn.write("exec-timeout 3 3 \n")

tn.write("logging synchronous \n")

tn.write("line console 0 \n")

tn.write("transport output all \n")

tn.write("login local \n")

tn.write("exec-timeout 3 3 \n")

tn.write("logging synchronous \n")

configurarInterfaces(tn)

tn.write("end \n")

tn.write("wr \n")

tn.write("\n")

print("configuracion realizada con exito!!!")

Comentario: Esta función recibe como parámetro de entrada la sesión telnet establecida entre el ordenador y el enrutador, para luego proceder a ejecutar los comandos que nos permiten realizar la configuración básica en el enrutador. Al final se emite un mensaje por consola que nos indica si hubo o no éxito en la configuración del dispositivo.

#Configura la ip y mascara de subred al leer un archivo donde se encuentra cada interfaz con su respectiva ip y mascara de red

def configurarInterfaces(tn):

f = open("NOMBRE\_DEL\_ARCHIVO.txt", 'r')

a = f.readlines()

for linea in a:

lista = linea.split(',')

print(lista)

interfaz = lista[0]

ip = lista[1]

mascara = lista[2]

mascara2 = (mascara[0:(len(mascara) - 1)])

tn.write("interface " + interfaz + "\n")

tn.write("ip address " + ip + " " + mascara2 + "\n")

tn.write("no shutdown \n")

tn.write("exit \n")

f.close()

Comentario: En esta función se hace lectura de un archivo de extensión .txt, el cual contiene información acerca de las interfaces que tiene habilitadas el enrutador donde está realizando la configuración del protocolo BGP.

#Funcion para habilitar el enrutamiento BGP en el router con su respectivo AS

def SesionBGP(tn, ASlocal):

tn.write("enable \n")

tn.write("admin\n")

tn.write("configure terminal\n")

tn.write("router BGP " + ASlocal+"\n")

tn.write("end \n")

tn.write("wr \n")

tn.write("\n")

Comentario: Esta función tiene como parámetros de entrada la sesión telnet establecida entre el ordenador donde se ejecuta la aplicación y el enrutador a configurar y el número del sistema autónomo al cual pertenece este enrutador. Se ejecutan los comandos dentro del enrutador y se establece el protocolo BGP.

#Función para configurar el router vecino con su AS

def configurarVecino(ipNeighbor, ASlocal, ASremoto,tn):

tn.write("conf terminal \n")

tn.write("\n")

tn.write("\n")

tn.write("ip route 0.0.0.0 0.0.0.0 lo0 name to\_core\_isp \n")

tn.write("\n")

tn.write("\n")

tn.write("router bgp "+ASlocal+"\n")

tn.write("\n")

tn.write("\n")

tn.write("neighbor " +ipNeighbor+ " remote-as " +ASremoto+"\n")

tn.write("\n")

tn.write("\n")

tn.write("end \n")

tn.write("\n")

tn.write("wr \n")

tn.write("\n")

Comentario: La función configurarVecino tiene como parámetros de entrada la dirección IP del enrutador el cual es su próximo salto hacia el sistema autónomo remoto, los números de los sistemas autónomos tanto local como remoto y la conexión telnet establecida entre el ordenador y el enrutador a configurar. Se agrega la dirección IP asociada al enrutador con el cual se establece conexión BGP a través de la líneas de comando en la configuración global del enrutador.

#Función para anunciar las redes que son ingresadas como parámetro de entrada

def configurarNetwork(tn, network, ASlocal):

tn.write("conf terminal \n")

tn.write("router bgp " + ASlocal + "\n")

if(network=="0.0.0.0"):

tn.write("network "+ network+" \n")

else:

tn.write("network " + network + " mask 255.255.255.0" +"\n")

tn.write("end \n")

tn.write("wr \n")

tn.write("\n")

Comentario: Esta función recibe como parámetros de entrada la sesión telnet establecida entre el enrutador y el ordenador donde se ejecuta la aplicación, el numero de la red que entra a la sesión BGP y el número del sistema autónomo local. Se emiten comandos en el enrutador se valida si la red es “0.0.0.0” en ese caso de emite de manera automática la máscara de red “0.0.0.0”.

def validarFormatoIP(ipAddress):

# Validar el ingreso de la dirección IP, que contenga los cuatro octetos y que cada uno esté dentro del rango(0-255)

flag=False

try:

if (len(ipAddress.split("."))) != 4:

print("ERROR: Ingrese una direcccion IP valida!!!")

elif (int(ipAddress.split(".")[0]) < 0 or int(ipAddress.split(".")[0]) > 255):

print("ERROR: Ingrese una direcccion IP valida!!!")

elif (int(ipAddress.split(".")[1]) < 0 or int(ipAddress.split(".")[1]) > 255):

print("ERROR: Ingrese una direcccion IP valida!!!")

elif (int(ipAddress.split(".")[2]) < 0 or int(ipAddress.split(".")[2]) > 255):

print("ERROR: Ingrese una direcccion IP valida!!!")

elif (int(ipAddress.split(".")[3]) < 0 or int(ipAddress.split(".")[3]) > 255):

print("Ingrese una direcccion IP (valida!!!")

else:

flag=True

except ValueError:

print("ERROR: La direccion IP debe ser numerica")

return(flag)

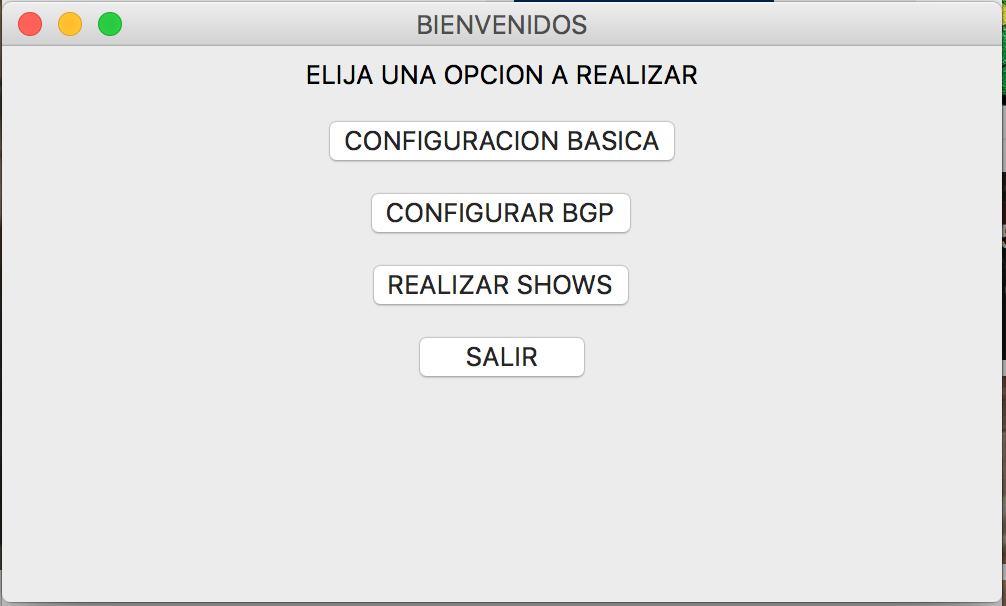
Comentario: La función validarFormatoIP nos sirve tanto para el ingreso de las variables de IP de los enrutadores vecinos dentro la sesión BGP, así como las redes a publicar dentro de la sesión BGP. Separamos los octetos para verificar que se encuentren dentro del rango de 0 a 255, una vez realizado esto se emite un mensaje de error indicando que el formato ingresado es incorrecto.

Interfaz gráfica:



Esta pantalla sirve para logear un usuario. Al darle click en el botón Validar, se revisa que los campos de usuario y password no estén vacíos.

Después de esto, se procede a comparar en una base de datos donde se encuentran los usuarios con sus respectivas contraseñas. Si coincide, el usuario ingresa al sistema para hacer la conexión BGP. En caso de que no coincida con ninguno, no podrá ingresar al sistema y mostrará un mensaje de error.

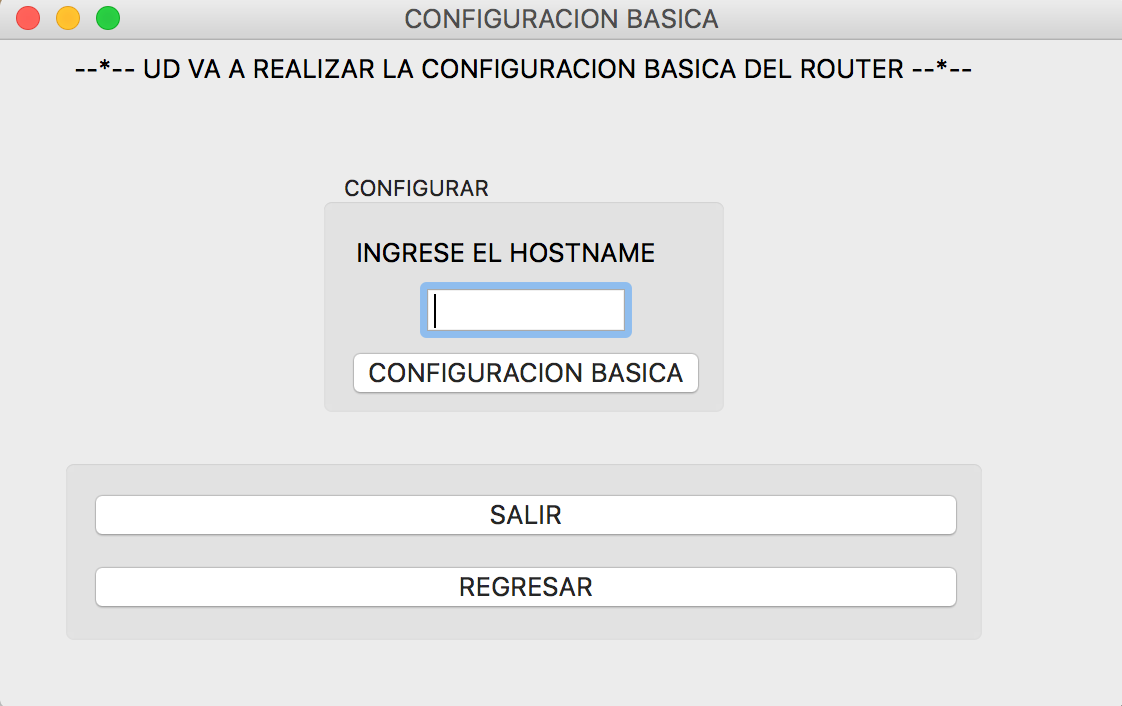


Al ingresar al sistema, muestra una pantalla de menú en la que, en caso de ser la primera vez que se conecta al router se deberá ir a configuración básica para configurar el router.

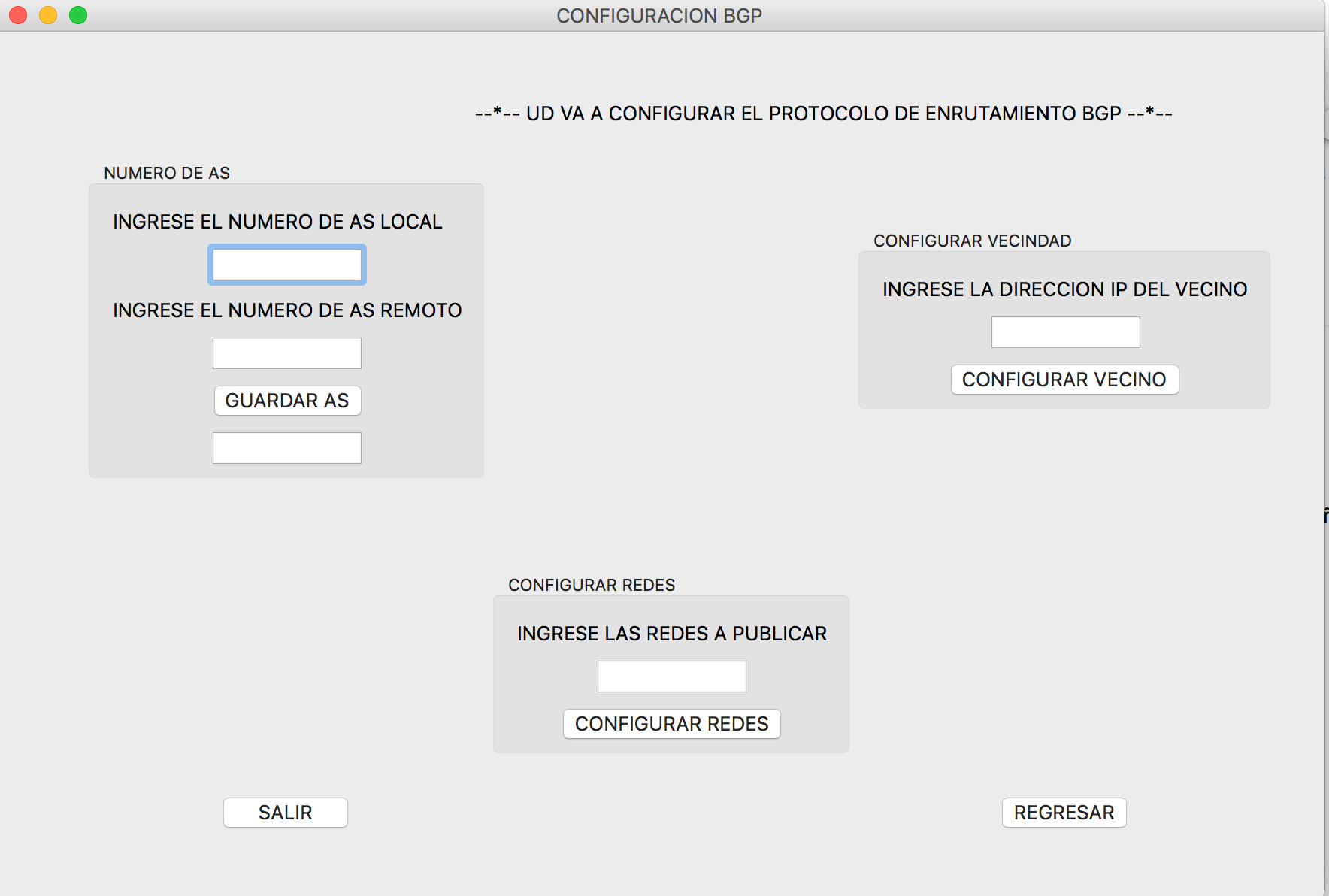
En caso de que el router este configurado, se procede a dar click en Configurar BGP para realizar la configuración del protocolo de enrutamiento.

Al dar click en el botón Realizar Shows aparecerá otra pantalla, en la que se podrá realizar diferentes shows relacionados a la sesión BGP.

Al dar click en Salir, se cierra la pantalla y el sistema completo.



Pantalla donde se configurará el router al dar click en Configuración Básica. Se procede a asignarle como hostname el nombre que se ingresa en el cuadro de texto. Además, se configura las interfaces con sus respectivas ip, esto lo hace al leer un archivo de texto con sus ips y mascara de subred respectivas.

Esta pantalla es la encargada de realizar la comunicación BGP. Primero se debe ingresar el número de AS local y remoto. En caso de que no se ingrese, mostrara un mensaje de error. Y ninguna de las otras opciones podrá ser ejecutada. Una vez que se hayan ingresado los sistemas autónomos, se procede a ingresar la ip del neighbor al dar click en Configurar Vecino.

Luego de esto, se debe ingresar las redes a publicar al dar click en configurar redes.

Las dos opciones anteriores están validadas para que no funcionen antes de ingresar los sistemas autónomos. Así mismo para que bote error si los campos se encuentran vacíos.

Una vez realizado todo esto, la configuración BGP en un router está completa. Pero para que haya comunicación entre los dos sistemas autónomos, el sistema debe ser ejecutado a su vez en el router vecino.