Rapport Projet : Interface de configurations des routeurs Cisco

Ce projet consiste à développer une interface graphique d'utilisateur (GUI) qui permettra la configuration des routeurs Cisco dans un réseau. A la fin de celui-ci, nous aurons une interface permettant à l'utilisateur de rentrer les adresse IP, les masques, les passerelles par défaut et l'affichage d'une table de routage à partir de ces différentes informations. Ce projet présente un intérêt important pour l'utilisateur, celui de ne pas connaître les commandes spécifiques des configurations sous Linux. Ce projet a été construit de sorte à répondre aux exigences du client.

Afin de mener à bien ce travail j'ai utilisé différentes technologies nécessaires telles que Paramiko, MVC, QtDesigner et PyQT.

Pour optimiser au mieux mon temps et le résultat de mon interface, j'ai divisé mon projet en plusieurs sous-étapes :

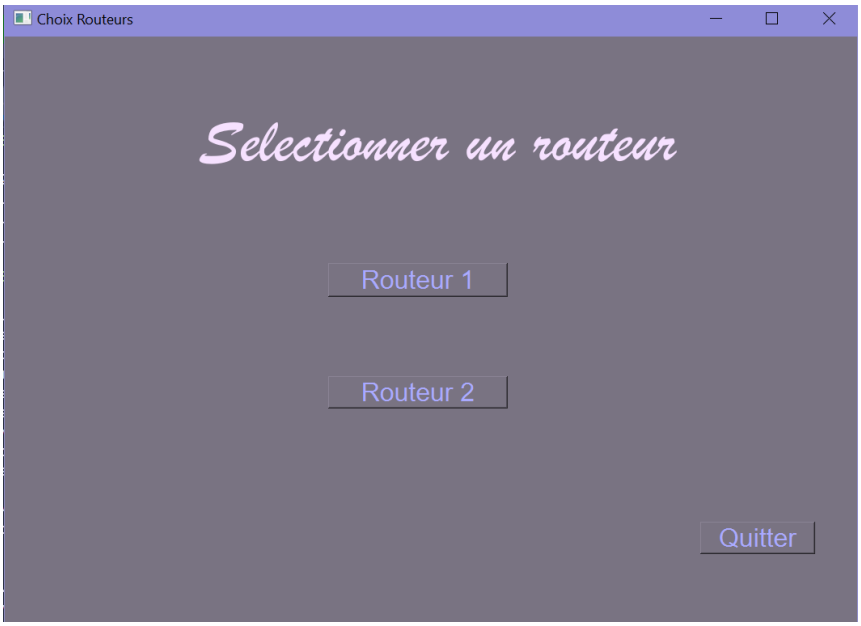
- Développement graphique de mon interface sous QtDesigner
- Connexion SSH
- Configuration du routeur 1 /2 (nom du routeur, des adresses IP, des passerelles par défaut)
- Affichage de la table de routage
- Visualisation du Ping

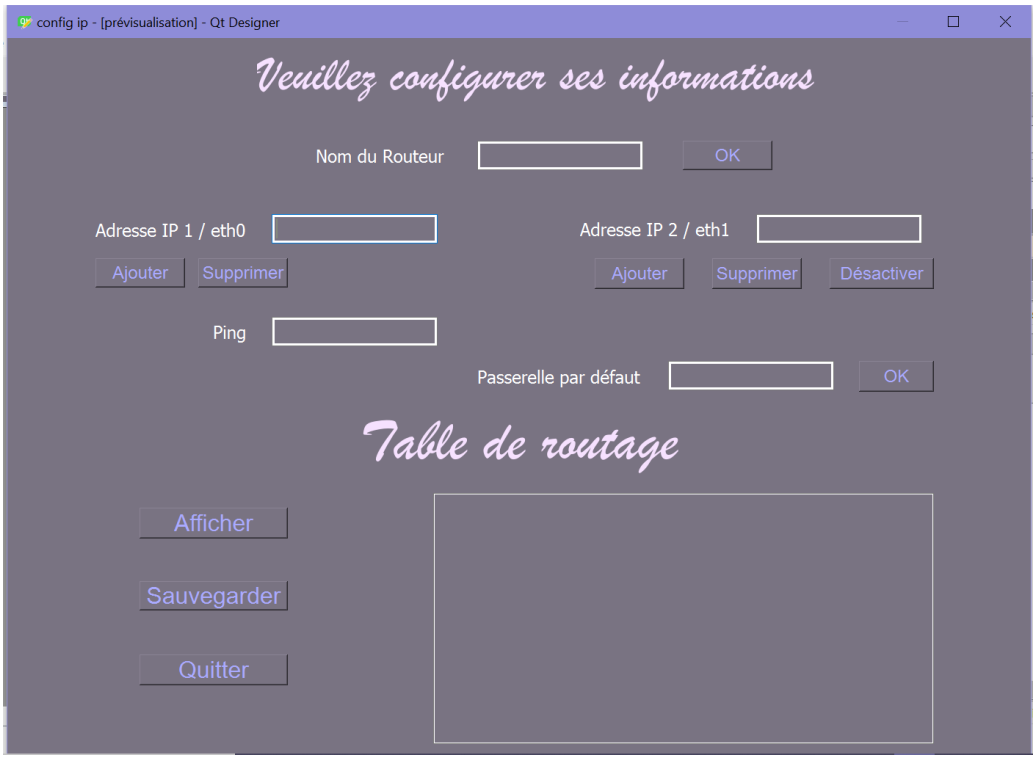
Développement graphique de l'interface sur QtDesigner

J'ai décidé de constituer mon interface en 2 fenêtres.

Tout au long du développement de ce projet, j'ai réalisé plusieurs ajustements sur celle-ci me rendant compte qu'il me manquait certaines fonctionnalités.

Mon interface finale





Pour passer de ces interfaces .UI en fichier .PY j'ai utilisé la commande 'python -m PyQt5.uic.pyuic -x fichier.ui -o fichier.py'.

Connexion SSH

SSH permet une connexion à distance. Il nous a permis de connecter notre interface à notre routeur via un fichier nommé SSH.

```python
class ssh :
    client =None

    def __init__(self, hostname, port, username, password):
        try:
            print("Connecting to server.")
            self.client=client.SSHClient() #Instance d'un client SSH
            self.client.set_missing_host_key_policy(client.AutoAddPolicy())
            self.client.connect(hostname, port=port, username=username, password=password)
            # établit la connexion avec le serveur SSH

        except:
            print("Exception raised !")

    def sendCommand(self, command):
        if(self.client):
            stdin,stdout,stderr= self.client.exec_command(command)
            #j'execute une commande sur le terminal distant
            #stdin: permet de passer des configurations en plus
            #stdout: contient le resultat du terminal
            #stderr: contient les erreurs du terminal
            # print (stdout.read().decode())
            return stdout.read().decode()
            #on affiche le contenu du fichier stdout
        else:
            print("Connection not opened.")
        self.client.close()
        # ferme la connexion
```

Plus précisément on a défini une fonction pour le routeur 1 ‘connexion_R1’ comprenant son hostname, son mot de passe, son port et son username. Cette fonction sera appelée dans chaque autre fonction qui a la possibilité de changer les configurations du routeur. Cela permet donc à l’utilisateur de modifier ses configurations à distance via l’interface.

```
def connexion_R1(self): #fonction pour établir la connexion
#entre le routeur 1 de la VM à nos interfaces
    hostname="192.168.43.2"
    print(hostname)
    username="etudiant"
    password="vitrygtr"
    port=22

    return hostname , port , username ,password
```

On fait la même chose pour le routeur 2.

```
def connexion_R2(self): #on connecte le routeur2 à notre interface
    hostname="192.168.43.4"
    print(hostname)
    username="etudiant"
    password="vitrygtr"
    port=22

    return hostname , port , username ,password
```

Développement des fonctionnalités

Fenêtre 1

Ma première fenêtre consiste a sélectionné un routeur pour ensuite rentrer ses caractéristiques. L’utilisateur aura juste a cliqué sur le routeur qu’il souhaite configurer.

Pour cela, j’ai configuré 2 fonctions nommées ‘suivant1’, ‘suivant2’ que j’ai connecté à un pushbutton.

```
def suivant1(self): #fonction permettant de passer de la fenêtre  1 à la 2
    self.MainWindow3 = QtWidgets.QMainWindow()
    self.View= F3bis.View()
    self.Model= F3bis.Model()
    self.Controller= F3bis.Controller(self.Model)
    self.View.setupUi(self.MainWindow3, self.Model, self.Controller, self.pushButton_R1)
    self.MainWindow3.show() #ouverture de la deuxième fenêtre
    MainWindow.hide() #fermeture de la fenêtre actuelle
```

```
self.pushButton_R1.clicked.connect(self.suivant1)
#connexion du boutton R1 à la fonction suivant1 pour passer à la fenêtre 2
```

Ici on a appliqué une fonction à un bouton. Ainsi lorsque l’utilisateur va cliquer sur le pushbutton du routeur 1 alors la fonction va être appelé pour passer de la fenêtre 1 à la 2 et emmener l’utilisateur à la page de configuration.

```
def suivant2(self): #fonction pour passer de la fenêtre 1 à la 2 à partir du routeur2
    self.MainWindow3 = QtWidgets.QMainWindow()
    self.View= F3bis.View()
    self.Model= F3bis.Model()
    self.Controller= F3bis.Controller(self.Model)
    self.View.setupUi(self.MainWindow3, self.Model, self.Controller, self.pushButton_2_R2)
    self.MainWindow3.show() #ouverture de la fenêtre 2
    MainWindow.hide() #fermeture de la fenêtre actuelle
```

```
self.pushButton_2_R2.clicked.connect(self.suivant2)
#appel de la fonction suivant2 via le pushbutton du routeur2 pour passer à la fenêtre 2
```

De la même manière que pour le bouton du routeur 1. Lorsque l'utilisateur clique sur le pushbutton du routeur 2 celui va appeler la fonction suivant2 qui va permettre de passer à la fenêtre suivante.

```
def quitter(self): #fonction pour fermer la fenêtre actuelle
    MainWindow.close()
```

```
self.pushButton_2_quitter.clicked.connect(self.quitter)
# on connecte le bouton quitter à la fonction quitter pour fermer la fenêtre actuelle
```

L'une des dernières options de cette fenêtre est la fonction quitter qui va permettre à l'utilisateur de fermer cette fenêtre à tout moment.

Fenêtre 2

Cette fenêtre comprend toutes les fonctionnalités que l'utilisateur peut modifier s'il le veut, à l'aide de pushbutton et de lineEdit. Pour cette partie, je dispose d'un fichier principal contenant une architecture MVC (« Moel-View-Controller). Cette méthode permet de séparer l'interface (View) de la partie gestion (Controller) et stockage (Model), ce qui nous facilite la tâche. Je dispose aussi d'un second fichier qui comporte toutes les fonctions qui font référence aux commandes Linux.

Dans cette interface, l'utilisateur dispose de différentes fonctionnalités. Il peut modifier le nom du routeur, ajouter ou supprimer des adresses IP, configurer une passerelle par défaut ainsi que désactiver la carte eth1. Sachant que mes routeurs ne disposent que de deux cartes réseaux (eth0 et eth1). A l'issue de toutes ses informations saisies, l'utilisateur pour afficher la table de routage ainsi que sauvegarder ses configurations entrées et quitter la page une fois finit.

- **Changement du nom du routeur**

Si le nom de routeur ne convient pas à l'utilisateur, il a la possibilité d'en changer grâce au champ et au bouton dédiés à celui-ci.

```
  def namerouteur(self, nomrouteur, lineEdit_4_routeur):
      #fonction callback quand on clique dans bouton
      #il accède au fichier fcom et il va utiliser la commande changement_NR1
      if nomrouteur.text() == "Routeur 1"  :  #cas du routeur 1
          s0= lineEdit_4_routeur.text()
          R1 = F2.Ui_MainWindow()
          hostname, port, username, password = R1.connexion_R1()
          fcom.changement_NR1(s0, hostname, port , username, password)
      elif nomrouteur.text() == "Routeur 2": #cas du routeur 2
          p0 =lineEdit_4_routeur.text()
          R2 = F2.Ui_MainWindow()
          hostname, port , username,password = R2.connexion_R2()
          fcom.changement_NR2(p0, hostname,port, username,password)

#fonction qui permet de changer les noms des routeurs sur la VM Routeur1
```

Dans la fonction 'name routeur', j'ai ajouté une condition « if et elif » qui permet d'exécuter le code selon le routeur que l'utilisateur choisit. Par exemple, si l'utilisateur choisit le routeur 1 alors une connexion, grâce à la fonction connexion_R1, va s'établir entre l'interface et le routeur via l'adresse IP eth0 pour que l'utilisateur puisse modifier le nom du routeur.

```
#changement nom de routeur
def changement_NR1(s0, hostname, port, username, password):
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo hostnamectl set-hostname " + s0)
    #changer le nom du routeur grâce à la commande linux

def changement_NR2(p0,hostname, port , username,password):
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo hostnamectl set-hostname " +p0)
    #changer le nom du routeur
```

En parallèle cette fonction va s'exécuter permettant de changer automatiquement le nom du routeur à travers la commande Linux suivante 'sudo hostnamectl set-hostname' suivit du nom saisi par l'utilisateur.

- **Ajout d'une adresse IP / Ping**

L'utilisateur peut s'il le souhaite ajouter une adresse IP (eth0 ou eth1) sur le routeur via l'interface grâce à un bouton ajouter.

```
    def config_eth0(self, nomrouteur, lineEdit_ip1,lineEditping):
        if nomrouteur.text() == "Routeur 1"  :
            s1 = lineEdit_ip1.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            # on se connecte sur le routeur 1 et apres appelle la fonction pour l'executer
            res= fcom.add_ip_eth0_R1(s1, hostname, port, username, password)
            if res != "erreur":
                lineEditping.setText("connecté")

            else:
                lineEditping.setText(res)

        elif nomrouteur.text() == "Routeur 2":
            p1 = lineEdit_ip1.text()
            R2 = F2.Ui_MainWindow()
            hostname, port , username,password = R2.connexion_R2()
            # on se connecte sur le routeur 2 et apres appelle la fonction pour l'executer
            fcom.add_ip_eth0_R2(p1,hostname, port, username, password)
            res= fcom.add_ip_eth0_R2(p1, hostname, port, username, password)
            if res != "erreur":
                  lineEditping.setText("connecté")

            else:
                  lineEditping.setText(res)
            #fonction qui permet d'ajouter une adresse ip sur la carte eth0 du routeur 1 ou 2
```

De la même manière que pour le nom du routeur j'ai ajouté une condition « if et elif » pour que selon le routeur choisi il exécute des commandes différentes.

```
#ajout d'une adresse IP sur eth0
def add_ip_eth0_R1(s1, hostname, port, username, password): # on configure  l'adresse ip1 de eth0
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip addr add "+ s1 + " dev eth0 ")
    res= connect.sendCommand("ping -c 1" + s1 + ">/dev/null/ || echo erreur")
    return res

def add_ip_eth0_R2(p1,hostname, port , username,password): #configurer @ip
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip addr add "+ p1 + " dev eth0 ")
    res= connect.sendCommand("ping -c 1" + p1 + ">/dev/null/ || echo erreur")
    return res
```

```
self.pushButton_3_ok_IP1.clicked.connect(self.config_eth0)
#boutton ajouter de l'IP1 connecter à la fonction configuration du routeur1
```

Par exemple si l'on se connecte sur le routeur 2, alors on va utiliser la condition « elif » puis aller voir dans le fichier où il y a les commandes Linux pour ajouter une adresse IP grâce à la commande « sudo ip addr add +dev eth0 ».

Une autre fonctionnalité est ajoutée ici lorsque l'adresse IP va être ajouté dans eth0 alors en même temps il va y avoir une autre requête qui va se passer pour vérifier si l'adresse IP ajouté peut être pinguer. Si c'est le cas il y aura marqué connecter.

Pour changer l'adresse IP sur la carte eth1 on effectue le même déroulement (voir annexe).

- **Suppression d'une adresse IP**

Cette fonctionnalité va permettre à l'utilisateur de supprimer une adresse IP s'il n'en veut plus grâce à un bouton supprimer.

```
def suppr_ip1(self, nomrouteur, lineEdit_ip1):
    if nomrouteur.text() == "Routeur 1"  :
        s3 = lineEdit_ip1.text()
        R1 = F2.Ui_MainWindow()
        hostname, port, username, password = R1.connexion_R1()
        # on se connecte sur le routeur 1 et apres appelle la fonction pour l'executer
        fcom.delete_ip_eth0_R1(s3, hostname, port, username, password)

    elif nomrouteur.text() == "Routeur 2":
        p3 = lineEdit_ip1.text()
        R2 = F2.Ui_MainWindow()
        hostname, port, username, password = R2.connexion_R2()
        # on se connecte sur le routeur 2 et apres appelle la fonction pour l'executer
        fcom.delete_ip_eth0_R2(p3, hostname, port, username, password)

#fonction qui permet de supprimer l'adresse ip1 de la carte eth0 routeur1 et 2
```

Pour cela, on dispose toujours d'une condition « if et elif ». Pour le cas du routeur 2, on va utiliser la condition « elif » puis aller dans le fichier commande récupérer la commande permettant de supprimer une adresse donc « sudo ip addr del + dev eth0 ».

```
#suppression d'une adresse IP sur eth0
def delete_ip_eth0_R1(s3, hostname, port, username, password):
    #on supprime l'adresse ip1 de eth0
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip addr del "+ s3 + " dev eth0 ")
def delete_ip_eth0_R2(p3, hostname, port, username, password):
    #on supprime l'adresse ip1 de eth0
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip addr del "+ p3 + " dev eth0 ")
```

La suppression de l'adresse IP2 sur eth1 s'effectue de la même manière.

- **Ajout de la passerelle par défaut**

La passerelle par défaut permet d'envoyer un message entre 2 réseaux. L'utilisateur a la possibilité d'en ajouter une grâce au champ et au bouton dédiés à cela.

```
def pdef_R1(self, lineEdit_2_pass1, nomrouteur):
    if nomrouteur.text() == "Routeur 1"  :
        s5 = lineEdit_2_pass1.text()
        R1 = F2.Ui_MainWindow()
        hostname, port, username, password = R1.connexion_R1()
        fcom.add_pdef_R1(s5, hostname, port, username, password)

    elif nomrouteur.text() == "Routeur 2":
        p5 = lineEdit_2_pass1.text()
        R2 = F2.Ui_MainWindow()
        hostname, port, username, password = R2.connexion_R2()
        fcom.add_pdef_R2(p5, hostname, port, username, password)
#fonction qui permet d'ajouter une passerelle par défaut au routeur1 et 2
```

Si l'utilisateur clique sur le routeur 1 alors il va effectuer ces commandes. Ensuite il va y avoir récupération de la commande « sudo ip route add » pour changer la passerelle par défaut.

```
#configuration d'une passerelle par défaut
def add_pdef_R1(s5, hostname, port, username, password): #on change la passerelle par défaut
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip route add default via "+ s5)

def add_pdef_R2(p5, hostname, port, username, password): #on change la passerelle par défaut
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand("sudo ip route add default via "+ p5)
```

Donc en ajoutant la passerelle par défaut cela va permettre à l'utilisateur que son paquet sorte du réseau pour aller vers un autre.

- **Désactivation de la carte eth1**

La fonctionnalité désactiver a été faite sur la carte eth1, je n'ai pas pu la réaliser sur eth0 car la connexion entre le routeur et l'interface s'établit avec l'adresse IP de l'eth0 sinon après on se retrouvera déconnecté.

Au niveau du remplissage du champ pour l'adresse IP 2 sur la carte eth1, l'utilisateur a la possibilité d'ajouter, de supprimer l'adresse IP ou de désactiver la carte eth1 grâce à des boutons que j'ai configurés.

S'il appuie sur désactiver voici la commande effectuée.

```
    def desact_eth1(self, nomrouteur):
        if nomrouteur.text() == "Routeur 1"  :
            R1 = F2.Ui_MainWindow()
            hostname, port , username,password = R1.connexion_R1()
            # on se connecte sur le routeur1 et apres appelle la fonction pour l'executer
            fcom.desact_eth1(hostname, port, username, password)
    # fonction pour désactiver eth1 cad pour désactiver l'adresse ip2 du routeur 1
        elif nomrouteur.text() == "Routeur 2":
            R2 = F2.Ui_MainWindow()
            hostname, port , username,password = R2.connexion_R2()
            # on se connecte sur le routeur2 et apres appelle la fonction pour l'executer
            fcom.desact_eth1(hostname, port, username, password)
    # fonction pour désactiver eth1 cad pour désactiver l'adresse ip2 du routeur 2
```

S'il choisit le routeur 1 alors il va réaliser la condition « if », il va donc aller dans le fichier commande et effectuer la commande « sudo ifdown eth1 » alors eth1 sera désactivé.

```
def desact_eth1(hostname, port, username, password): #on désactive eth1
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand(" sudo ifdown eth1")

def activ_eth1(hostname, port, username, password): #on désactive eth1
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    connect.sendCommand(" sudo ifup eth1")
```

```
self.pushButton_dsactiver.clicked.connect(self.desact_eth1)
```

Lorsqu'on appuie sur le pushbutton désactiver alors il retourne à la fonction desact_eth1.

Si l'on veut réactiver la carte eth1 il faut entrer une adresse IP puis cliquer sur ajouter, automatiquement ça va la réactiver.

- **Affichage table de routage**

L'utilisateur va pouvoir afficher sa table de routage à partir des données qu'il a rentré grâce au bouton afficher. La table routage est une structure composée de plusieurs informations telles que l'adresse de réseau et l'adresse IP de chaque carte ainsi que leur passerelle par défaut. Celle-ci est donc essentielle pour acheminer les paquets vers l'extérieur afin de prendre le bon chemin d'acheminement.

Si l'utilisateur choisit le routeur 1 alors les commandes suivantes vont s'effectuer.

```
def table_routage(self,nomrouteur, textEdit_tablederoutage):
    if nomrouteur.text() == "Routeur 1"  :
        R1 = F2.Ui_MainWindow()
        hostname, port, username, password = R1.connexion_R1()
        # on se connecte sur le routeur 1 et apres apelle la fonction pour l'executer
        resultat = fcom.table(hostname, port, username, password)
        # print(resultat)
        textEdit_tablederoutage.setText(resultat)

    elif nomrouteur.text() == "Routeur 2":
        R2 = F2.Ui_MainWindow()
        hostname, port, username, password = R2.connexion_R2()
        # on se connecter sur le routeur 2 et apres apelle la fonction pour l'executer
        resultat = fcom.table(hostname, port, username, password)
        #on retourne le resultat de la commande de sudo add ip route
        #dans une variable resultat qui sera appelé dans l'emplacement de l'affichage de la
        #table de routage
        # print (resultat)
        textEdit_tablederoutage.setText(resultat)
      #affiche table routage routeur 1 et 2
```

Les commandes précédentes font appellent au fichier commande pour savoir la commande pour afficher la table de routage qui est « sudo ip route ».

```
#affichage table de routage
def table(hostname, port, username, password):
    connect=ssh(hostname=hostname, port=port, username=username, password=password)
    res = connect.sendCommand("sudo ip route")
    return res
```

- **Fichier sauvegarde**

L'utilisateur à la possibilité grâce à un fichier texte de sauvegarder, de conserver les informations qu'il vient de rentrer, pour cela il lui suffit de cliquer sur le bouton sauvegarder.

```
def sauvegarder(self, nomrouteur, lineEdit_4_routeur,
                lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
    #1ere méthode
    if nomrouteur.text() == "Routeur 1"  :
        f = open('sauvegarde.txt', 'w')
        s0 = lineEdit_4_routeur.text()
        s1 = lineEdit_ip1.text()
        s2 = lineEdit_7_ip2.text()
        s5 = lineEdit_2_pass1.text()
        f.write('\nNom du routeur : ' + s0 +
                '\nAdresse IP1 eth0: ' + s1 +
                '\nAdresse IP2 eth1 : ' + s2 +
                '\nPasserelle par défaut : ' + s5)
        f.close()
    elif nomrouteur.text() == "Routeur 2":
        f = open('sauvegarde.txt', 'w')
        p0 = lineEdit_4_routeur.text()
        p1 = lineEdit_ip1.text()
        p2 = lineEdit_7_ip2.text()
        p5 = lineEdit_2_pass1.text()
        f.write('\nNom du routeur : ' + p0 +
                '\nAdresse IP1 eth0: ' + p1 +
                '\nAdresse IP2 eth1 : ' + p2 +
                '\nPasserelle par défaut : ' + p5)
        f.close()
  #fichier de sauvegarde utilise à l'utilisateur
```

Quel que soit le routeur utilisé, l'utilisateur à la possibilité grâce à ces commandes effectuées de sauvegarder. Dans le cas du routeur 2, après avoir rempli les configurations, il clique sur sauvegarder. A ce moment-là, le fichier sauvegarde.txt s'ouvre et à la place les nouvelles configurations se notent, puis le fichier se ferme.

Fonctionnalités non codées

Malgré tout il me reste quelques fonctionnalités non codées tels que la fonction lire. J'ai eu des difficultés à comprendre sa fonction et comment l'implémenter dans mon code. De plus, le ping n'est pas totalement fonctionnel.

Conclusion

A l'issu de ce projet, nous obtenons une interface fonctionnelle et utilisable par l'utilisateur. Nous avons une connexion entre mes routeurs et les configurations effectuées via le biais de mon interface. Ce travail m'a permis de développer les bases apprises au cours de cette première année tel que la programmation sous Python. Il m'a aussi permis d'utiliser les connaissances étudiées en réseau ainsi qu'en Linux.

ANNEXE

Code SSH

```
@author: Alicia
"""
#fichier ssh pour connecter mon interface et le routeur
# contient uniquement le code ssh
from paramiko import client
#paramiko est une bibliothèque de python
# import getpass, sys


class ssh :
    client =None

    def __init__(self, hostname, port, username, password):
        try:
            print("Connecting to server.")
            self.client=client.SSHClient() #Instance d'un client SSH
            self.client.set_missing_host_key_policy(client.AutoAddPolicy())
            self.client.connect(hostname, port=port, username=username, password=password)
            # établit la connexion avec le serveur SSH

        except:
            print("Exception raised !")

    def sendCommand(self, command):
        if(self.client):
            stdin,stdout,stderr= self.client.exec_command(command)
            #j'execute une commande sur le terminal distant
            #stdin: permet de passer des configurations en plus
```

```
            stdin,stdout,stderr= self.client.exec_command(command)
            #j'execute une commande sur le terminal distant
            #stdin: permet de passer des configurations en plus
            #stdout: contient le resultat du terminal
            #stderr: contient les erreurs du terminal
            # print (stdout.read().decode())
            return stdout.read().decode()
            #on affiche le contenu du fichier stdout
        else:
            print("Connection not opened.")
        self.client.close()
        # ferme la connexion

# hostname="192.168.43.2" etape pour vérifier que mon routeur se connecte bien avec mon interface
# username="etudiant"
# password="vitrygtr"
# port=22

# hostname = sys.argv
# port = sys.argv
# username = sys.argv
# password = getpass.getpass("Mot de passe : ")


if __name__ == "__main__":
    user = ssh(hostname,port,username,password)
    # user.sendCommand("ip a")
    resultat = user.sendCommand("ip route")
    print(resultat)
```

Code première fenêtre et ses fonctionnalités

```
from PyQt5 import QtCore, QtGui, QtWidgets
import F3bis
import SSH


#code design fenêtre 2
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(527, 364)
        MainWindow.setStyleSheet("background-color: rgb(121, 115, 130);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label_titref2 = QtWidgets.QLabel(self.centralwidget)
        self.label_titref2.setGeometry(QtCore.QRect(120, 40, 391, 51))
        self.label_titref2.setStyleSheet("color: rgb(246, 225, 255);\n"
"font: italic 40pt \"Brush Script MT\";")
        self.label_titref2.setObjectName("label_titref2")
        self.pushButton_R1 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_R1.setGeometry(QtCore.QRect(200, 140, 111, 21))
        self.pushButton_R1.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
        self.pushButton_R1.setObjectName("pushButton_R1")
        self.pushButton_2_R2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2_R2.setGeometry(QtCore.QRect(200, 210, 111, 20))
        self.pushButton_2_R2.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
        self.pushButton_2_R2.setObjectName("pushButton_2_R2")
        self.pushButton_2_quitter = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2_quitter.setGeometry(QtCore.QRect(430, 300, 71, 20))
        self.pushButton_2_quitter.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
```

```
        self.pushButton_2_quitter.setObjectName("pushButton_2_quitter")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 527, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Choix Routeurs"))
        self.label_titref2.setText(_translate("MainWindow", "Selectionner un routeur "))
        self.pushButton_R1.setText(_translate("MainWindow", "Routeur 1"))
        self.pushButton_2_R2.setText(_translate("MainWindow", "Routeur 2"))
        self.pushButton_2_quitter.setText(_translate("MainWindow", "Quitter"))

        self.pushButton_R1.clicked.connect(self.suivant1)
        #connexion du boutton R1 à la fonction suivant1 pour passer à la fenêtre 2
        self.pushButton_2_R2.clicked.connect(self.suivant2)
        #appel de la fonction suivant2 via le pushbutton du routeur2 pour passer à la fenêtre 2
        self.pushButton_2_quitter.clicked.connect(self.quitter)
        # on connecte le bouton quitter à la fonction quitter pour fermer la fenêtre actuelle

    def suivant1(self): #fonction permettant de passer de la fenêtre  1 à la 2
        self.MainWindow3 = QtWidgets.QMainWindow()
        self.View= F3bis.View()
        self.Model= F3bis.Model()
        self.Controller= F3bis.Controller(self.Model)
        self.View.setupUi(self.MainWindow3, self.Model, self.Controller, self.pushButton_R1)
        self.MainWindow3.show() #ouverture de la deuxième fenêtre
        MainWindow.hide() #fermeture de la fenêtre actuelle


    def suivant2(self): #fonction pour passer de la fenêtre 1 à la 2 à partir du routeur2
```

```
    def suivant2(self): #fonction pour passer de la fenêtre 1 à la 2 à partir du routeur2
        self.MainWindow3 = QtWidgets.QMainWindow()
        self.View= F3bis.View()
        self.Model= F3bis.Model()
        self.Controller= F3bis.Controller(self.Model)
        self.View.setupUi(self.MainWindow3, self.Model, self.Controller, self.pushButton_2_R2)
        self.MainWindow3.show() #ouverture de la fenêtre 2
        MainWindow.hide() #fermeture de la fenêtre actuelle

    def connexion_R1(self): #fonction pour établir la connexion
    #entre le routeur 1 de la VM à nos interfaces
        hostname="192.168.43.2"
        print(hostname)
        username="etudiant"
        password="vitrygtr"
        port=22

        return hostname , port , username ,password

    def connexion_R2(self): #on connecte le routeur2 à notre interface
        hostname="192.168.43.4"
        print(hostname)
        username="etudiant"
        password="vitrygtr"
        port=22

        return hostname , port , username ,password

    def quitter(self): #fonction pour fermer la fenêtre actuelle
        MainWindow.close()

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Code deuxième fenêtre

```
from PyQt5 import QtCore, QtGui, QtWidgets
import F2
import fcom
class View(object):
    def setupUi(self, MainWindow, model, ctrl, nomrouteur):
        self.nomrouteur= nomrouteur
        self.myModel = model
        self.myCtrl = ctrl
        self.MainWindow= MainWindow
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(697, 526)
        MainWindow.setStyleSheet("background-color: rgb(121, 115, 130);")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label_ip1 = QtWidgets.QLabel(self.centralwidget)
        self.label_ip1.setGeometry(QtCore.QRect(60, 120, 111, 21))
        self.label_ip1.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.label_ip1.setObjectName("label_ip1")
        self.label_2_ip2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2_ip2.setGeometry(QtCore.QRect(390, 120, 101, 20))
        self.label_2_ip2.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.label_2_ip2.setObjectName("label_2_ip2")
        self.lineEdit_ip1 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_ip1.setGeometry(QtCore.QRect(180, 120, 113, 20))
        self.lineEdit_ip1.setStyleSheet("font: 75 12pt \"Arial\";\n"
"color: rgb(255, 255, 255);")
        self.lineEdit_ip1.setObjectName("lineEdit_ip1")
        self.lineEdit_2_pass1 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_2_pass1.setGeometry(QtCore.QRect(450, 220, 113, 20))
        self.lineEdit_2_pass1.setStyleSheet("font: 75 12pt \"Arial\";\n"
"color: rgb(255, 255, 255);")
        self.lineEdit_2_pass1.setObjectName("lineEdit_2_pass1")
        self.label_3_routeur = QtWidgets.QLabel(self.centralwidget)
        self.label_3_routeur.setGeometry(QtCore.QRect(210, 70, 91, 20))
        self.label_3_routeur.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.label_3_routeur.setObjectName("label_3_routeur")
        self.lineEdit_4_routeur = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_4_routeur.setGeometry(QtCore.QRect(320, 70, 113, 20))
        self.lineEdit_4_routeur.setStyleSheet("font: 75 12pt \"Arial\";\n"
"color: rgb(255, 255, 255);")
        self.lineEdit_4_routeur.setObjectName("lineEdit_4_routeur")
```

```
        self.lineEdit_4_routeur.setObjectName("lineEdit_4_routeur")
        self.label_6_pass1 = QtWidgets.QLabel(self.centralwidget)
        self.label_6_pass1.setGeometry(QtCore.QRect(320, 220, 111, 21))
        self.label_6_pass1.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.label_6_pass1.setObjectName("label_6_pass1")
        self.label_8_titre_f3 = QtWidgets.QLabel(self.centralwidget)
        self.label_8_titre_f3.setGeometry(QtCore.QRect(170, 0, 411, 51))
        self.label_8_titre_f3.setStyleSheet("color: rgb(246, 225, 255);\n"
"font: italic 34pt \"Brush Script MT\";")
        self.label_8_titre_f3.setObjectName("label_8_titre_f3")
        self.lineEdit_7_ip2 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_7_ip2.setGeometry(QtCore.QRect(510, 120, 113, 20))
        self.lineEdit_7_ip2.setStyleSheet("font: 75 12pt \"Arial\";\n"
"color: rgb(255, 255, 255);")
        self.lineEdit_7_ip2.setObjectName("lineEdit_7_ip2")
        self.pushButton_2_quitter = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2_quitter.setGeometry(QtCore.QRect(40, 410, 71, 20))
        self.pushButton_2_quitter.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
        self.pushButton_2_quitter.setObjectName("pushButton_2_quitter")
        self.pushButton_supp_IP1 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_supp_IP1.setGeometry(QtCore.QRect(130, 150, 61, 20))
        self.pushButton_supp_IP1.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_supp_IP1.setObjectName("pushButton_supp_IP1")
        self.pushButton_2_ok_pass1 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2_ok_pass1.setGeometry(QtCore.QRect(580, 220, 51, 21))
        self.pushButton_2_ok_pass1.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_2_ok_pass1.setObjectName("pushButton_2_ok_pass1")
        self.pushButton_3_ok_IP1 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3_ok_IP1.setGeometry(QtCore.QRect(60, 150, 61, 20))
        self.pushButton_3_ok_IP1.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_3_ok_IP1.setObjectName("pushButton_3_ok_IP1")
```

```
        self.pushButton_3_ok_IP1.setObjectName("pushButton_3_ok_IP1")
        self.pushButton_4_supp_AIP2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_4_supp_AIP2.setGeometry(QtCore.QRect(480, 150, 61, 21))
        self.pushButton_4_supp_AIP2.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_4_supp_AIP2.setObjectName("pushButton_4_supp_AIP2")
        self.pushButton_5_ok_AIP2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_5_ok_AIP2.setGeometry(QtCore.QRect(400, 150, 61, 21))
        self.pushButton_5_ok_AIP2.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_5_ok_AIP2.setObjectName("pushButton_5_ok_AIP2")
        self.pushButton_7_ok_nrouteur = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_7_ok_nrouteur.setGeometry(QtCore.QRect(460, 70, 61, 20))
        self.pushButton_7_ok_nrouteur.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_7_ok_nrouteur.setObjectName("pushButton_7_ok_nrouteur")
        self.pushButton_sauvegarde = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_sauvegarde.setGeometry(QtCore.QRect(150, 410, 101, 20))
        self.pushButton_sauvegarde.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
        self.pushButton_sauvegarde.setObjectName("pushButton_sauvegarde")
#         self.pushButton_lire = QtWidgets.QPushButton(self.centralwidget)
#         self.pushButton_lire.setGeometry(QtCore.QRect(40, 340, 71, 20))
#         self.pushButton_lire.setStyleSheet("color: rgb(170, 170, 255);\n"
# "border-top-color: rgb(255, 255, 255);\n"
# "border-bottom-color: rgb(255, 255, 255);\n"
# "font: 75 16pt \"Arial\";")
#         self.pushButton_lire.setObjectName("pushButton_lire")
        self.lineEditping = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEditping.setEnabled(True)
        self.lineEditping.setGeometry(QtCore.QRect(180, 190, 113, 20))
        self.lineEditping.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.lineEditping.setReadOnly(True)
        self.lineEditping.setObjectName("lineEditping")
        self.label_titre_f4 = QtWidgets.QLabel(self.centralwidget)
```

```
        self.lineEditping = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEditping.setEnabled(True)
        self.lineEditping.setGeometry(QtCore.QRect(180, 190, 113, 20))
        self.lineEditping.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.lineEditping.setReadOnly(True)
        self.lineEditping.setObjectName("lineEditping")
        self.label_titre_f4 = QtWidgets.QLabel(self.centralwidget)
        self.label_titre_f4.setGeometry(QtCore.QRect(240, 250, 231, 51))
        self.label_titre_f4.setStyleSheet("color: rgb(246, 225, 255);\n"
"font: italic 40pt \"Brush Script MT\";")
        self.label_titre_f4.setObjectName("label_titre_f4")
        self.textEdit_tablederoutage = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit_tablederoutage.setGeometry(QtCore.QRect(290, 310, 341, 171))
        self.textEdit_tablederoutage.setStyleSheet("font: 75 14pt \"Arial\";\n"
"color: rgb(255, 255, 255);")
        self.textEdit_tablederoutage.setObjectName("textEdit_tablederoutage")
        self.pushButton_3_afficher = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3_afficher.setGeometry(QtCore.QRect(160, 340, 81, 21))
        self.pushButton_3_afficher.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 16pt \"Arial\";")
        self.pushButton_3_afficher.setObjectName("pushButton_3_afficher")
        self.pushButton_dsactiver = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_dsactiver.setGeometry(QtCore.QRect(560, 150, 71, 21))
        self.pushButton_dsactiver.setStyleSheet("color: rgb(170, 170, 255);\n"
"border-top-color: rgb(255, 255, 255);\n"
"border-bottom-color: rgb(255, 255, 255);\n"
"font: 75 12pt \"Arial\";")
        self.pushButton_dsactiver.setObjectName("pushButton_dsactiver")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(140, 190, 31, 20))
        self.label.setStyleSheet("color: rgb(255, 255, 255);\n"
"font: 75 12pt \"MS Shell Dlg 2\";")
        self.label.setObjectName("label")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 697, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
```

```
    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)
    self.lineEdit_4_routeur.setText(self.nomrouteur.text())
    #permet d'afficher le nom du routeur dans l'interface!

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "config ip"))
    self.label_ip1.setText(_translate("MainWindow", "Adresse IP 1 / eth0"))
    self.label_2_ip2.setText(_translate("MainWindow", "Adresse IP 2 / eth1"))
    self.label_3_routeur.setText(_translate("MainWindow", "Nom du Routeur "))
    self.label_6_pass1.setText(_translate("MainWindow", "Passerelle par défaut"))
    self.label_8_titre_f3.setText(_translate("MainWindow", "Veuillez configurer ses informations"))
    self.pushButton_2_quitter.setText(_translate("MainWindow", "Quitter"))
    self.pushButton_supp_IP1.setText(_translate("MainWindow", "Supprimer"))
    self.pushButton_2_ok_pass1.setText(_translate("MainWindow", "OK"))
    self.pushButton_3_ok_IP1.setText(_translate("MainWindow", "Ajouter"))
    self.pushButton_4_supp_AIP2.setText(_translate("MainWindow", "Supprimer"))
    self.pushButton_5_ok_AIP2.setText(_translate("MainWindow", "Ajouter"))
    self.pushButton_7_ok_nrouteur.setText(_translate("MainWindow", "OK"))
    self.pushButton_sauvegarde.setText(_translate("MainWindow", "Sauvegarder"))
    # self.pushButton_lire.setText(_translate("MainWindow", "Lire"))
    self.label_titre_f4.setText(_translate("MainWindow", "Table de routage "))
    self.pushButton_3_afficher.setText(_translate("MainWindow", "Afficher"))
    self.pushButton_dsactiver.setText(_translate("MainWindow", "Désactiver"))
    self.label.setText(_translate("MainWindow", "Ping"))

    self.pushButton_7_ok_nrouteur.clicked.connect(self.namerouteur)
    #boutton ok du routeur connecter à la fonction namerouteur
    self.pushButton_3_ok_IP1.clicked.connect(self.config_eth0)
    #boutton ajouter de l'IP1 connecter à la fonction configuration du routeur1
    self.pushButton_5_ok_AIP2.clicked.connect(self.config_eth1)
    #boutton ajouter de l'IP2 connecter à la fonction configuration du routeur2
    self.pushButton_supp_IP1.clicked.connect(self.suppr_ip1)
    #boutton supprimer IP1 connecter la fonction supprimer IP1
    self.pushButton_2_quitter.clicked.connect(self.quitter)
    #boutton supprimer IP2 connecter la fonction supprimer IP2
    self.pushButton_sauvegarde.clicked.connect(self.sauvegarder)
    #boutton sauvegarder relié à la fonction sauvegarder
    self.pushButton_4_supp_AIP2.clicked.connect(self.suppr_ip2)
    #boutton supprimer IP2 connecté à la fonction suppr IP2
    self.pushButton_2_ok_pass1.clicked.connect(self.pdef_R1)
    #boutton ok de la passerelle par défaut connecté à la fonction pdef du routeur 1
```

```
        # #configuration du routeur2
        self.pushButton_3_afficher.clicked.connect(self.table_routage)

        self.pushButton_dsactiver.clicked.connect(self.desact_eth1)

        # self.pushButton_lire.clicked.connect(self.lire)



    def namerouteur(self, lineEdit_4_routeur):
        self.myCtrl.namerouteur(self.nomrouteur, self.lineEdit_4_routeur)
    def config_eth0(self, lineEdit_ip1):
        self.myCtrl.config_eth0(self.nomrouteur, self.lineEdit_ip1,self.lineEditping)
    def config_eth1(self, lineEdit_7_ip2):
        self.myCtrl.config_eth1(self.lineEdit_7_ip2, self.nomrouteur)
    def suppr_ip1(self, lineEdit_ip1):
        self.myCtrl.suppr_ip1(self.nomrouteur, self.lineEdit_ip1)
    def suppr_ip2(self, lineEdit_7_ip2):
        self.myCtrl.suppr_ip2(self.lineEdit_7_ip2, self.nomrouteur)
    def pdef_R1(self, lineEdit_2_pass1):
        self.myCtrl.pdef_R1(self.lineEdit_2_pass1, self.nomrouteur)
    def table_routage(self, textEdit_tablederoutage):
        self.myCtrl.table_routage(self.nomrouteur, self.textEdit_tablederoutage)
    def desact_eth1(self):
        self.myCtrl.desact_eth1(self.nomrouteur)
    def quitter(self):
        self.MainWindow.close()
    def sauvegarder(self):
        self.myCtrl.sauvegarder(self.nomrouteur, self.lineEdit_4_routeur, self.lineEdit_ip1, self.lineEdit_7_ip2, self.lineEdit_2_pass1)

    def lire(self):
        self.myCtrl.lire(self.nomrouteur, self.lineEdit_4_routeur, self.lineEdit_ip1, self.lineEdit_7_ip2, self.lineEdit_2_pass1)
class Controller:
    def __init__(self, model):
        self.myModel = model

    def namerouteur(self, nomrouteur, lineEdit_4_routeur):
       self.myModel.namerouteur(nomrouteur, lineEdit_4_routeur)
    def config_eth0(self, nomrouteur, lineEdit_ip1,lineEditping):
        self.myModel.config_eth0(nomrouteur, lineEdit_ip1,lineEditping)
    def config_eth1(self, lineEdit_7_ip2, nomrouteur):
        self.myModel.config_eth1(lineEdit_7_ip2, nomrouteur)
    def suppr_ip1(self, nomrouteur, lineEdit_ip1):
        self.myModel.suppr_ip1(nomrouteur, lineEdit_ip1)
```

```
    def suppr_ip2(self, lineEdit_7_ip2, nomrouteur):
        self.myModel.suppr_ip2(lineEdit_7_ip2, nomrouteur)
    def pdef_R1(self, lineEdit_2_pass1, nomrouteur):
        self.myModel.pdef_R1(lineEdit_2_pass1, nomrouteur)
    def table_routage(self, nomrouteur, textEdit_tablederoutage):
        self.myModel.table_routage(nomrouteur, textEdit_tablederoutage)
    def desact_eth1(self, nomrouteur):
        self.myModel.desact_eth1(nomrouteur)
    def quitter(self):
        self.myModel.quitter()
    def sauvegarder(self, nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
        self.myModel.sauvegarder(nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1)
    def lire(self, nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
        self.myModel.lire(nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1)
class Model:

    def namerouteur(self, nomrouteur, lineEdit_4_routeur):
        #fonction callback quand on clique dans bouton
        #il accède au fichier fcom et il va utiliser la commande changement_NR1
        if nomrouteur.text() == "Routeur 1"  :  #cas du routeur 1
            s0= lineEdit_4_routeur.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            fcom.changement_NR1(s0, hostname, port , username, password)
        elif nomrouteur.text() == "Routeur 2": #cas du routeur 2
            p0 =lineEdit_4_routeur.text()
            R2 = F2.Ui_MainWindow()
            hostname, port , username,password = R2.connexion_R2()
            fcom.changement_NR2(p0, hostname,port, username,password)

  #fonction qui permet de changer les noms des routeurs sur la VM Routeur1


    def config_eth0(self, nomrouteur, lineEdit_ip1,lineEditping):
        if nomrouteur.text() == "Routeur 1"  :
            s1 = lineEdit_ip1.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            # on se connecte sur le routeur 1 et apres appelle la fonction pour l'executer
            res= fcom.add_ip_eth0_R1(s1, hostname, port, username, password)
            if res != "erreur":
                lineEditping.setText("connecté")
```

```
            else:
                lineEditping.setText(res)

        elif nomrouteur.text() == "Routeur 2":
            p1 = lineEdit_ip1.text()
            R2 = F2.Ui_MainWindow()
            hostname, port , username,password = R2.connexion_R2()
            # on se connecte sur le routeur 2 et apres appelle la fonction pour l'executer
            fcom.add_ip_eth0_R2(p1,hostname, port, username, password)
            res= fcom.add_ip_eth0_R2(p1, hostname, port, username, password)
            if res != "erreur":
                  lineEditping.setText("connecté")

            else:
                  lineEditping.setText(res)
            #fonction qui permet d'ajouter une adresse ip sur la carte eth0 du routeur 1 ou 2

    def config_eth1(self, lineEdit_7_ip2, nomrouteur):
        if nomrouteur.text() == "Routeur 1"  :
            s2 = lineEdit_7_ip2.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            # on se connecte sur le routeur 1 et apres apelle la fonction pour l'executer
            fcom.activ_eth1(hostname, port, username, password)
            fcom.add_ip_eth1_R1(s2, hostname, port, username, password)


        elif nomrouteur.text() == "Routeur 2":
            p2= lineEdit_7_ip2.text()
            R2 = F2.Ui_MainWindow()
            hostname, port, username, password = R2.connexion_R2()
            # on se connecte sur le routeur 2 et apres apelle la fonction pour l'executer
            fcom.activ_eth1(hostname, port, username, password)
            fcom.add_ip_eth1_R2(p2, hostname, port , username, password)

    #fonction qui permet d'ajouter une adresse ip sur la carte eth1 du routeur 1 et 2

    def suppr_ip1(self, nomrouteur, lineEdit_ip1):
        if nomrouteur.text() == "Routeur 1"  :
            s3 = lineEdit_ip1.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            # on se connecte sur le routeur 1 et apres appelle la fonction pour l'executer
```

```
            fcom.delete_ip_eth0_R1(s3, hostname, port, username, password)

        elif nomrouteur.text() == "Routeur 2":
            p3 = lineEdit_ip1.text()
            R2 = F2.Ui_MainWindow()
            hostname, port, username, password = R2.connexion_R2()
            # on se connecte sur le routeur 2 et apres appelle la fonction pour l'executer
            fcom.delete_ip_eth0_R2(p3, hostname, port, username, password)

    #fonction qui permet de supprimer l'adresse ip1 de la carte eth0 routeur1 et 2
    def suppr_ip2(self, lineEdit_7_ip2, nomrouteur):
        if nomrouteur.text() == "Routeur 1"  :
            s4 = lineEdit_7_ip2.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()   # connecter sur le VM et apres apelle la fonction pour l'executer
            fcom.delete_ip_eth1_R1(s4, hostname, port, username, password)
     #fonction qui permet de supprimer l'adresse ip2 de la carte eth1 du routeur1
        elif nomrouteur.text() == "Routeur 2":
            p4 = lineEdit_7_ip2.text()
            R2 = F2.Ui_MainWindow()
            hostname, port, username, password = R2.connexion_R2()   # connecter sur le VM et apres apelle la fonction pour l'executer
            fcom.delete_ip_eth1_R2(p4, hostname, port, username, password)


    def pdef_R1(self, lineEdit_2_pass1, nomrouteur):
        if nomrouteur.text() == "Routeur 1"  :
            s5 = lineEdit_2_pass1.text()
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            fcom.add_pdef_R1(s5, hostname, port, username, password)

        elif nomrouteur.text() == "Routeur 2":
            p5 = lineEdit_2_pass1.text()
            R2 = F2.Ui_MainWindow()
            hostname, port, username, password = R2.connexion_R2()
            fcom.add_pdef_R2(p5, hostname, port, username, password)
    #fonction qui permet d'ajouter une passerelle par défaut au routeur1 et 2


    def table_routage(self,nomrouteur, textEdit_tablederoutage):
        if nomrouteur.text() == "Routeur 1"  :
            R1 = F2.Ui_MainWindow()
            hostname, port, username, password = R1.connexion_R1()
            # on se connecte sur le routeur 1 et apres apelle la fonction pour l'executer
```

```
            resultat = fcom.table( hostname, port, username, password)
            # print(resultat)
            textEdit_tablederoutage.setText(resultat)
            # a = print(resultat)


        elif nomrouteur.text() == "Routeur 2":
            R2 = F2.Ui_MainWindow()
            hostname, port, username, password = R2.connexion_R2()
            # on se connecter sur le routeur 2 et apres apelle la fonction pour l'executer
            resultat = fcom.table(hostname, port, username, password)
            #on retourne le resultat de la commande de sudo add ip route
            #dans une variable resultat qui sera appelé dans l'emplacement de l'affichage de la
            #table de routage
            # print (resultat)
            textEdit_tablederoutage.setText(resultat)
            # b = print(resultat)
          #affiche table routage routeur 1 et 2
    def desact_eth1(self, nomrouteur):
        if nomrouteur.text() == "Routeur 1"  :
            R1 = F2.Ui_MainWindow()
            hostname, port , username,password = R1.connexion_R1()
            # on se connecte sur le routeur1 et apres appelle la fonction pour l'executer
            fcom.desact_eth1(hostname, port, username, password)
    # fonction pour désactiver eth1 cad pour désactiver l'adresse ip2 du routeur 1
        elif nomrouteur.text() == "Routeur 2":
            R2 = F2.Ui_MainWindow()
            hostname, port , username,password = R2.connexion_R2()
            # on se connecte sur le routeur2 et apres appelle la fonction pour l'executer
            fcom.desact_eth1(hostname, port, username, password)
    # fonction pour désactiver eth1 cad pour désactiver l'adresse ip2 du routeur 2

    def sauvegarder(self, nomrouteur, lineEdit_4_routeur,
                    lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
        #1ere méthode
        if nomrouteur.text() == "Routeur 1"  :
            f = open('sauvegarde.txt', 'w')
            s0 = lineEdit_4_routeur.text()
            s1 = lineEdit_ip1.text()
            s2 = lineEdit_7_ip2.text()
            s5 = lineEdit_2_pass1.text()
            # a= print(resultat)
            f.write('\nNom du routeur : ' + s0 +
                    '\nAdresse IP1 eth0: ' + s1 +
```

```
                '\nAdresse IP2 eth1 : ' + s2 +
                '\nPasserelle par défaut : ' + s5)
            f.close()
        elif nomrouteur.text() == "Routeur 2":
            f = open('sauvegarde.txt', 'w')
            p0 = lineEdit_4_routeur.text()
            p1 = lineEdit_ip1.text()
            p2 = lineEdit_7_ip2.text()
            p5 = lineEdit_2_pass1.text()
            # b= print(resultat)
            f.write('\nNom du routeur : ' + p0 +
                    '\nAdresse IP1 eth0: ' + p1 +
                    '\nAdresse IP2 eth1 : ' + p2 +
                    '\nPasserelle par défaut : ' + p5)
            f.close()
      #fichier de sauvegarde utilise à l'utilisateur
        #2eme méthode
        # with open('sauvegarde.txt','w') as f:
        #    if nomrouteur.text() == "Routeur 1"  :
        #        s0 = lineEdit_4_routeur.text()
        #        s1 = lineEdit_ip1.text()
        #        s2 = lineEdit_7_ip2.text()
        #        s5 = lineEdit_2_pass1.text()
        #        f.write('\nNom du routeur : ' + s0 + '\nAdresse IP eth0: ' + s1 + '\nAdresse IP eth1 : ' + s2 + '\nPasserelle par défaut : ' + s5)
        #  # on sauvegarde dans un fichier txt les adresses ip, le nom de routeur et la passerelle par défaut que l'utilisateur à entrer
        #    elif nomrouteur.text() == "Routeur 2":
        #        p0 = lineEdit_4_routeur.text()
        #        p1 = lineEdit_ip1.text()
        #        p2 = lineEdit_7_ip2.text()
        #        p5 = lineEdit_2_pass1.text()
        #        f.write('\nNom du routeur : ' + p0 + '\nAdresse IP eth0: ' + p1 + '\nAdresse IP eth1 : ' + p2 + '\nPasserelle par défaut : ' + p5)


    # def lire(self, nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
    #     #1ere méthode
    #     if nomrouteur.text() == "Routeur 1"  :
    #         f = open('lire.txt', 'r')
    #         s0 = lineEdit_4_routeur.text()
    #         s1 = lineEdit_ip1.text()
    #         s2 = lineEdit_7_ip2.text()
    #         s5 = lineEdit_2_pass1.text()
    #         f.read
    #         f.write (lineEdit_4_routeur.text(),lineEdit_ip1.text(),  lineEdit_7_ip2.text(), lineEdit_2_pass1.text())
    #     elif nomrouteur.text() == "Routeur 2":
```

```
    # def lire(self, nomrouteur, lineEdit_4_routeur, lineEdit_ip1, lineEdit_7_ip2, lineEdit_2_pass1):
    #     #1ere méthode
    #     if nomrouteur.text() == "Routeur 1"  :
    #         f = open('lire.txt', 'r')
    #         s0 = lineEdit_4_routeur.text()
    #         s1 = lineEdit_ip1.text()
    #         s2 = lineEdit_7_ip2.text()
    #         s5 = lineEdit_2_pass1.text()
    #         f.read
    #         f.write (lineEdit_4_routeur.text(),lineEdit_ip1.text(),  lineEdit_7_ip2.text(), lineEdit_2_pass1.text())
    #     elif nomrouteur.text() == "Routeur 2":
    #         f = open('lire.txt', 'r')
    #         # p0 = lineEdit_4_routeur.text()
    #         # p1 = lineEdit_ip1.text()
    #         # p2 = lineEdit_7_ip2.text()
    #         # p5 = lineEdit_2_pass1.text()
    #         f.read
    #         f.read(lineEdit_4_routeur.text(),lineEdit_ip1.text(),  lineEdit_7_ip2.text(), lineEdit_2_pass1.text())




if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow3 = QtWidgets.QMainWindow()
    View= View()
    Model= Model()
    Controller= Controller(Model)
    pushButton_3_afficher = QtWidgets.QPushButton()
    View.setupUi(MainWindow3, Model, Controller,pushButton_3_afficher)
    MainWindow3.show()
    sys.exit(app.exec_())
```