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 :

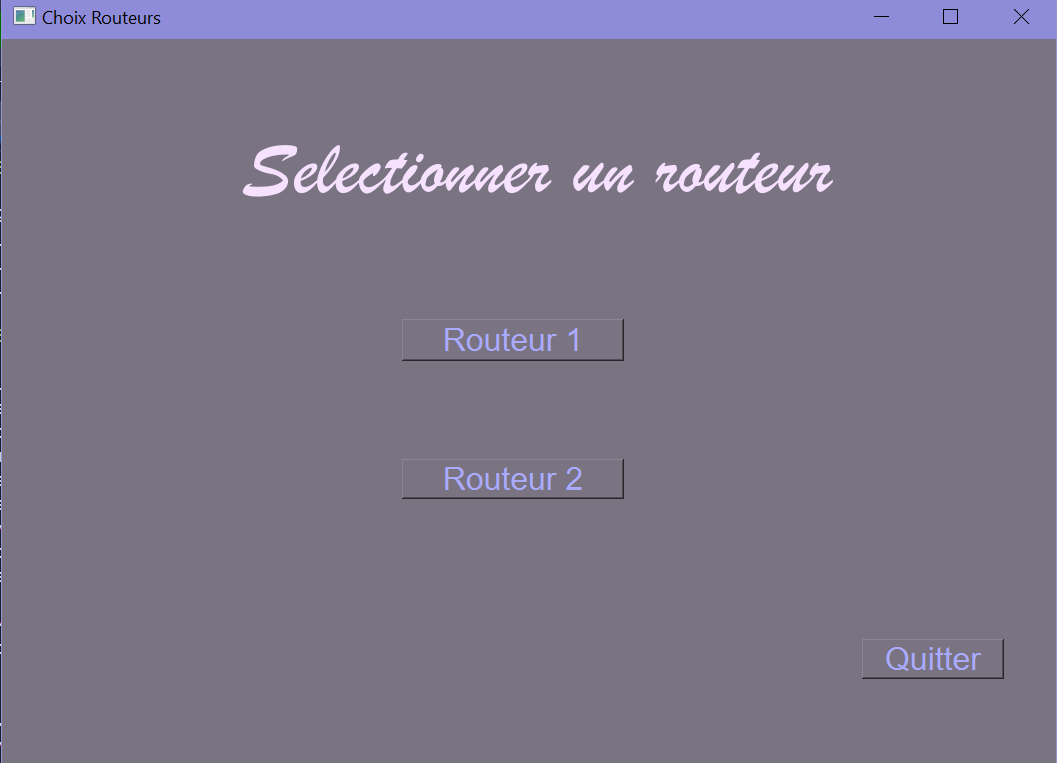
* Développement graphique de mon interface sous QtDesigner
* Connexion SSH
* Configuration du routeur 1 /2 (nom du routeur, des adresses IP, des passerelles par défaut)
* Affichage de la table de routage
* Visualisation du Ping

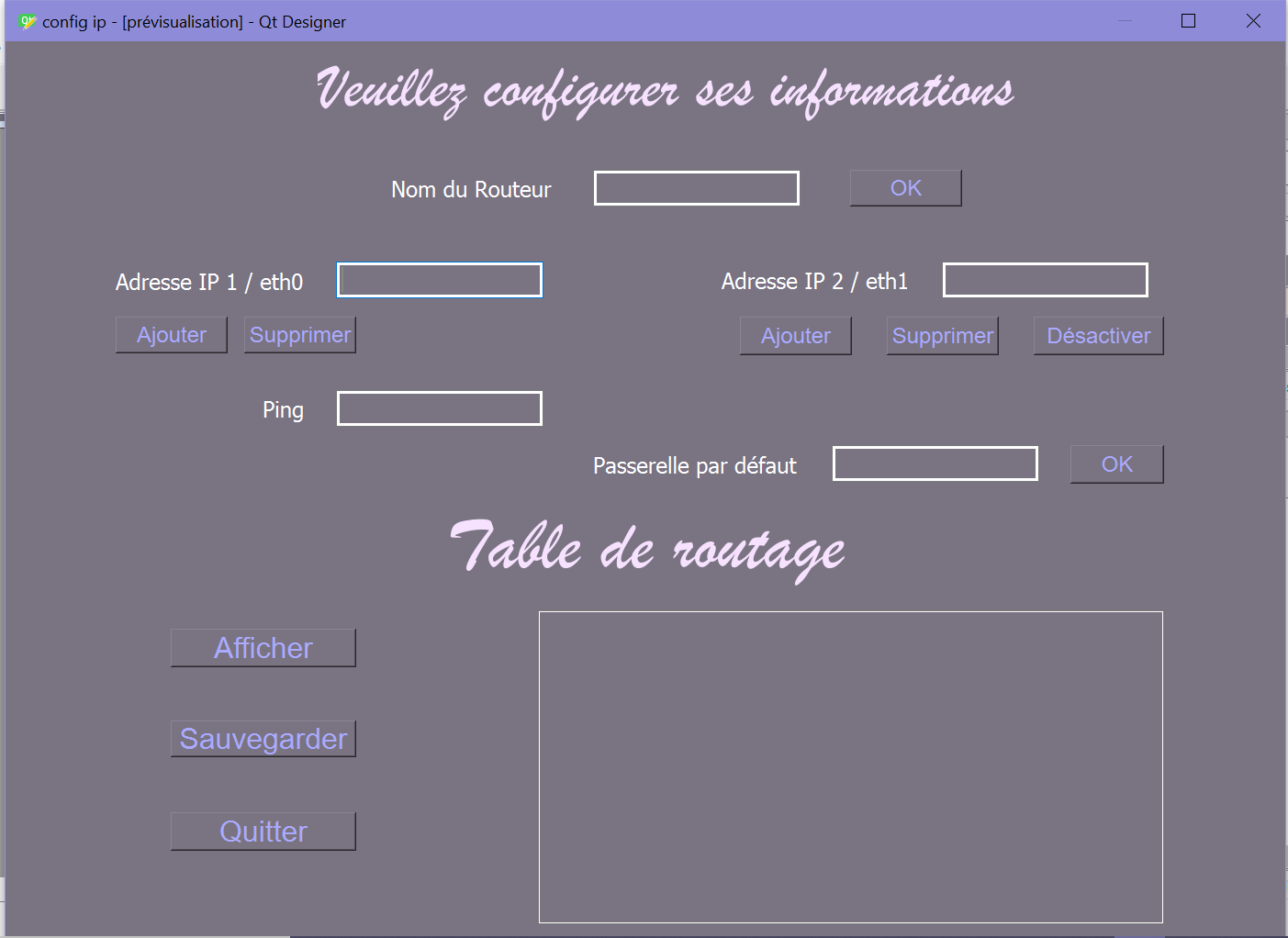
Développement graphique de l’interface sur QtDesigner

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**

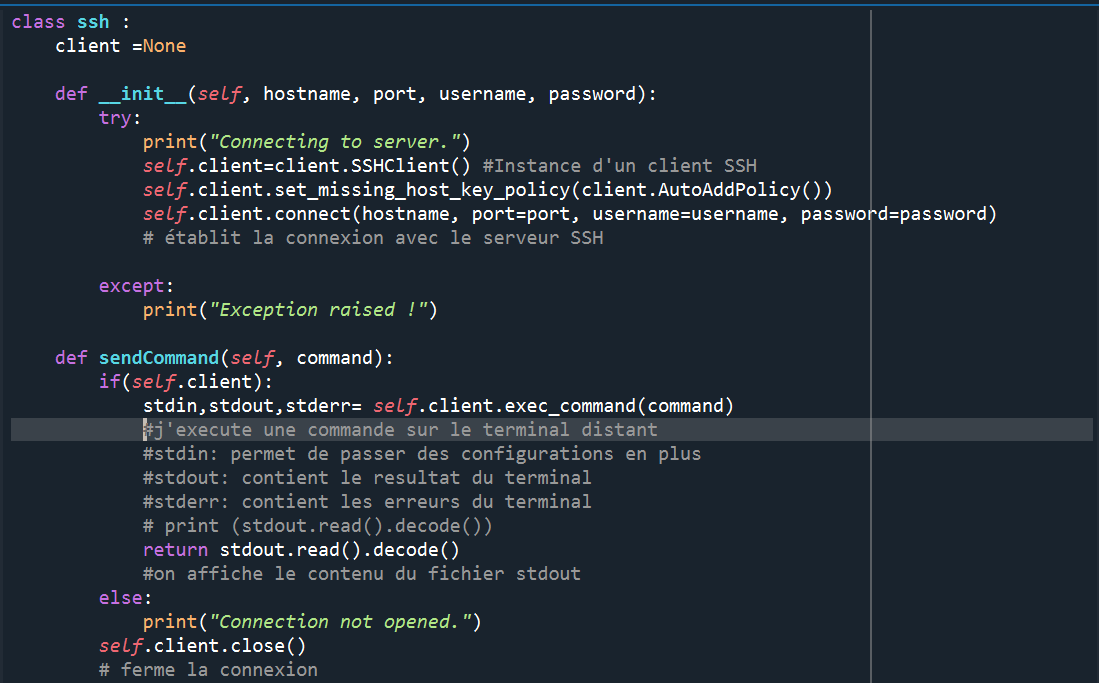




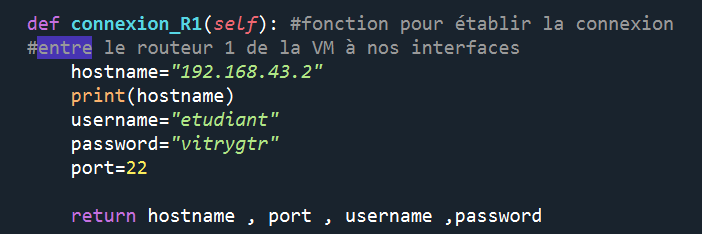
Pour passer de ces interfaces .UI en fichier .PY j’ai utilisé la commande ‘python -m PyQt5.uic.pyuic -x fichier.ui -o fichier.py’.

Connexion SSH

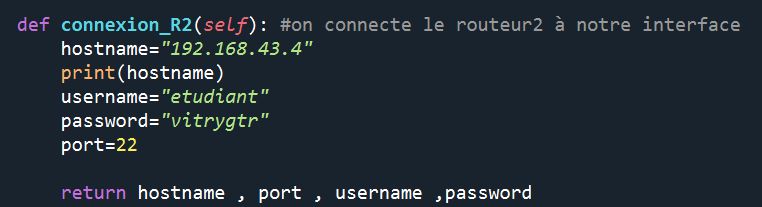
SSH permet une connexion à distance. Il nous a permis de connecter notre interface à notre routeur via un fichier nommé SSH.



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.



On fait la même chose pour le routeur 2.

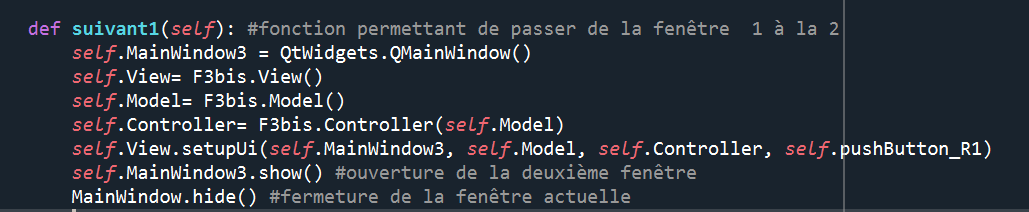


Développement des fonctionnalités

**Fenêtre 1**

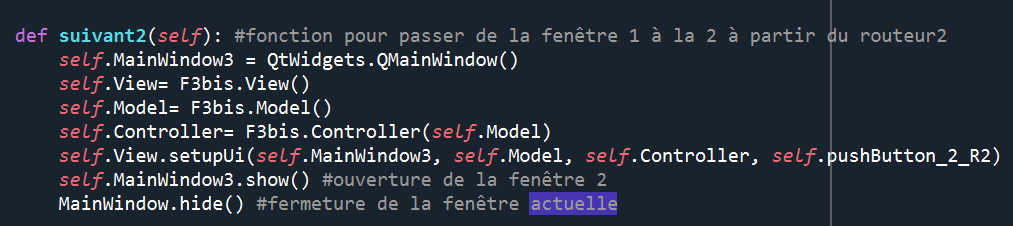
Ma première fenêtre consiste a sélectionné un routeur pour ensuite rentrer ses caractéristiques. L’utilisateur aura juste a cliqué sur le routeur qu’il souhaite configurer.

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



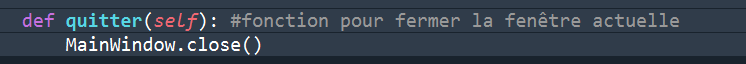


Ici on a appliqué une fonction à un bouton. Ainsi lorsque l’utilisateur va cliquer sur le pushbutton du routeur 1 alors la fonction va être appelé pour passer de la fenêtre 1 à la 2 et emmener l’utilisateur à la page de configuration.





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.





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.

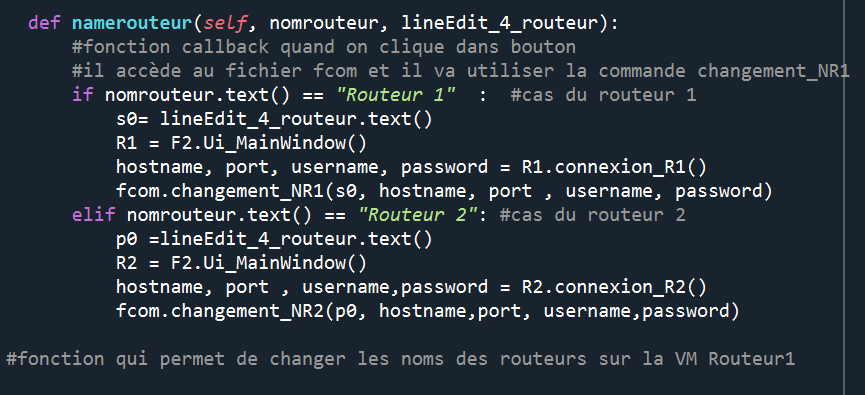
**Fenêtre 2**

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.

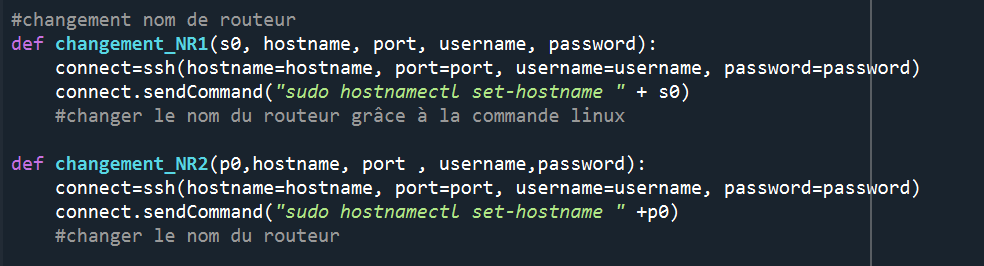
Dans cette interface, l’utilisateur dispose de différentes fonctionnalités. Il peut modifier le nom du routeur, ajouter ou supprimer des adresses IP, configurer une passerelle par défaut ainsi que désactiver la carte eth1. Sachant que mes routeurs ne disposent que de deux cartes réseaux (eth0 et eth1). 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.



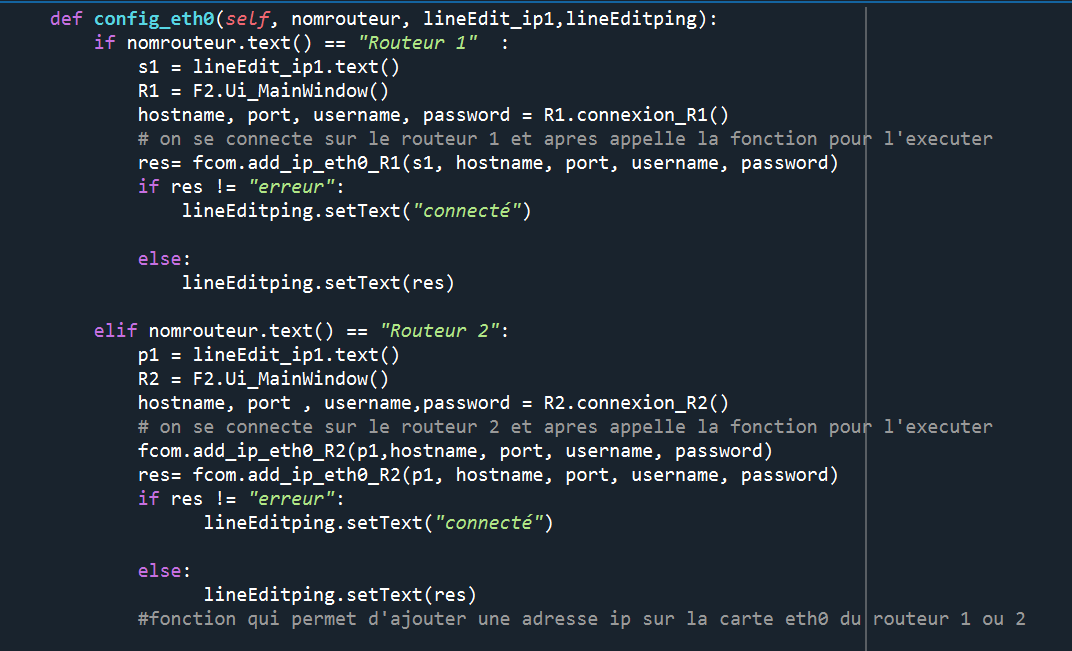
Dans la fonction ‘name routeur’, j’ai ajouté une condition « if et elif » qui permet d’exécuter le code selon le routeur que l’utilisateur choisit. Par exemple, si l’utilisateur choisit le routeur 1 alors une connexion, grâce à la fonction connexion\_R1, va s’établir entre l’interface et le routeur via l’adresse IP eth0 pour que l’utilisateur puisse modifier le nom du routeur.



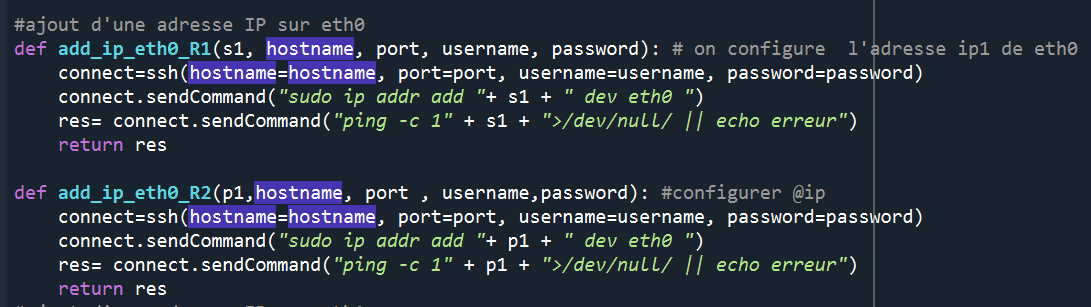
En parallèle cette fonction va s’exécuter permettant de changer automatiquement le nom du routeur à travers la commande Linux suivante ‘sudo hostnamectl set-hostname’ suivit du nom saisi par l’utilisateur.

* **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.



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.





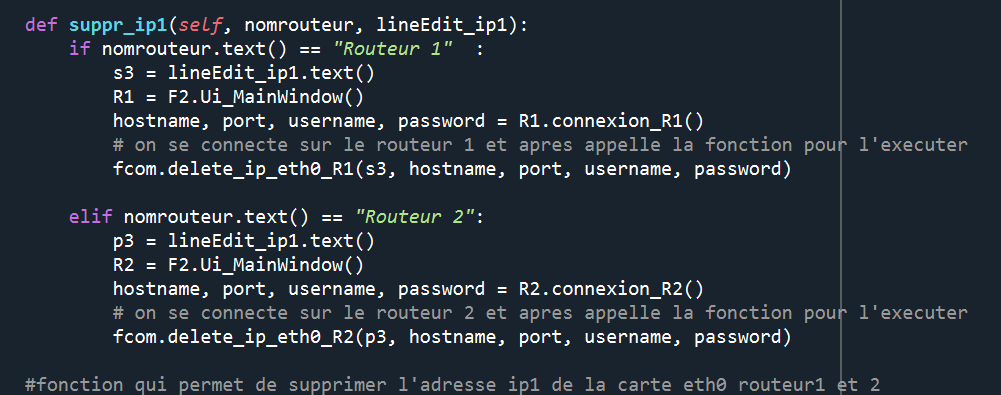
Par exemple si l’on se connecte sur le routeur 2, alors on va utiliser la condition « elif » puis aller voir dans le fichier où il y a les commandes Linux pour ajouter une adresse IP grâce à la commande « sudo ip addr add +dev eth0 ».

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.

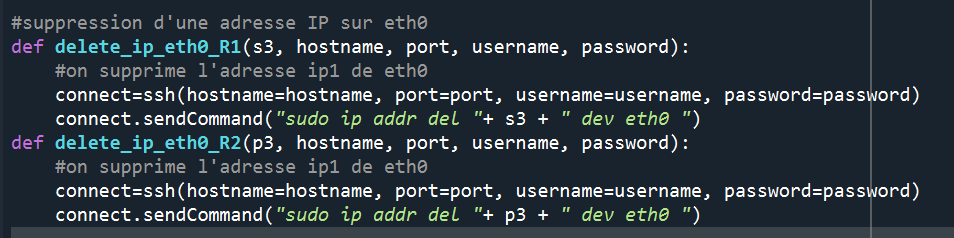
Pour changer l’adresse IP sur la carte eth1 on effectue le même déroulement (voir annexe).

* **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.



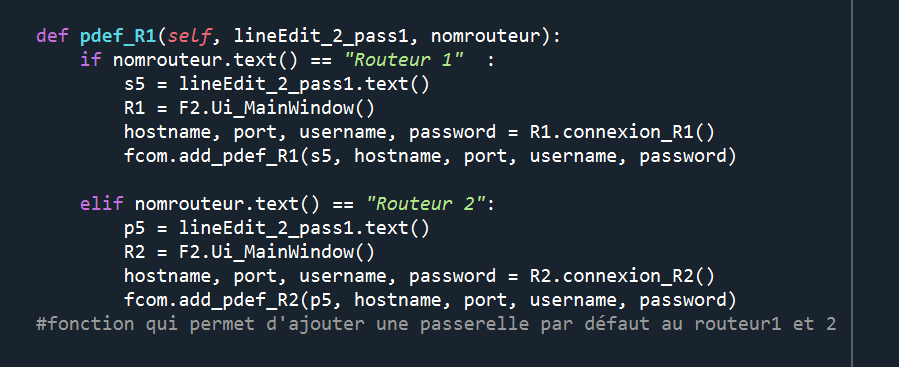
Pour cela, on dispose toujours d’une condition « if et elif ». Pour le cas du routeur 2, on va utiliser la condition « elif » puis aller dans le fichier commande récupérer la commande permettant de supprimer une adresse donc « sudo ip addr del + dev eth0 ».



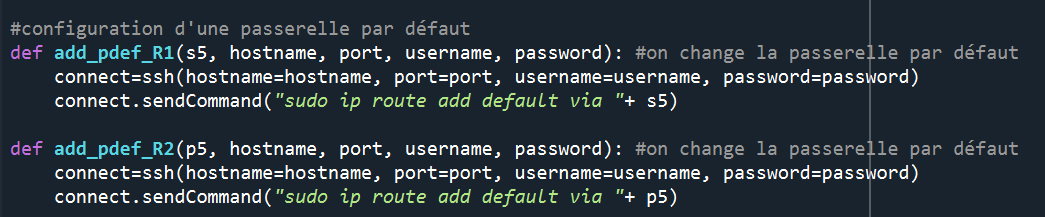
La suppression de l’adresse IP2 sur eth1 s’effectue de la même manière.

* **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.



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.



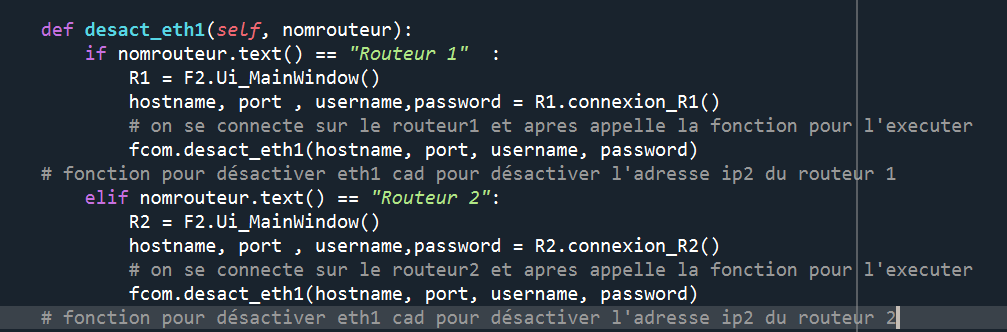
Donc en ajoutant la passerelle par défaut cela va permettre à l’utilisateur que son paquet sorte du réseau pour aller vers un autre.

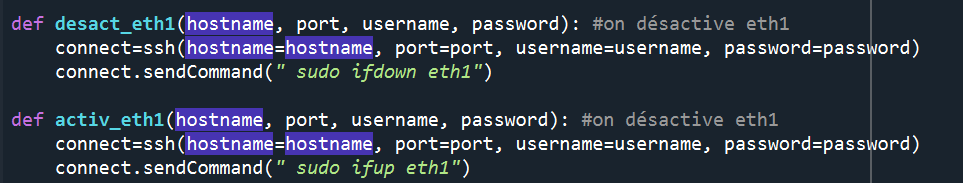
* **Désactivation de la carte eth1**

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.



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.



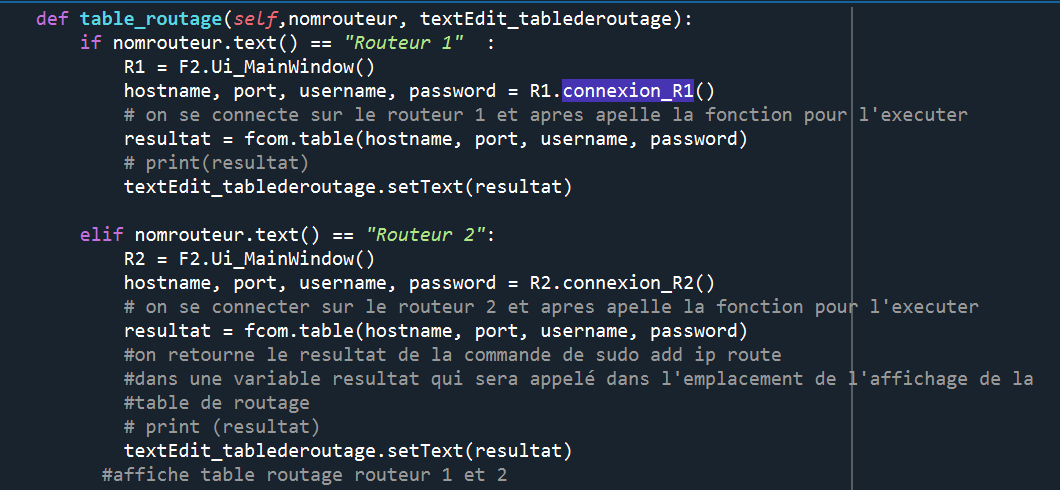
Lorsqu’on appuie sur le pushbutton désactiver alors il retourne à la fonction desact\_eth1.

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

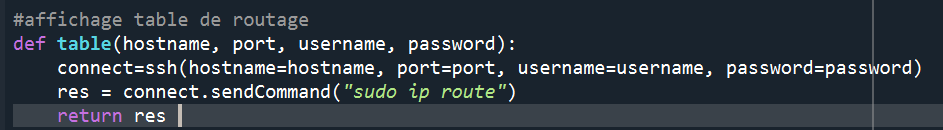
* **Affichage table de routage**

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.

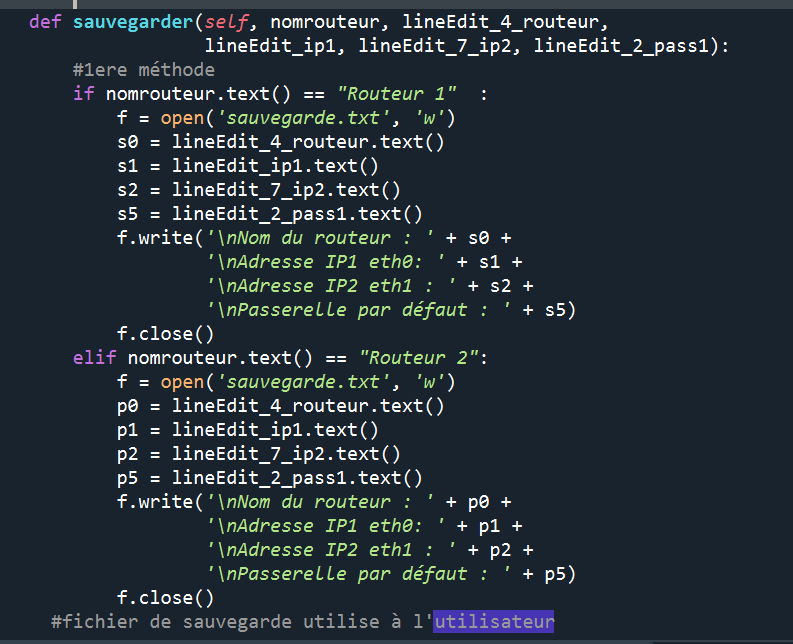


Les commandes précédentes font appellent au fichier commande pour savoir la commande pour afficher la table de routage qui est « sudo ip route ».



* **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.



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**

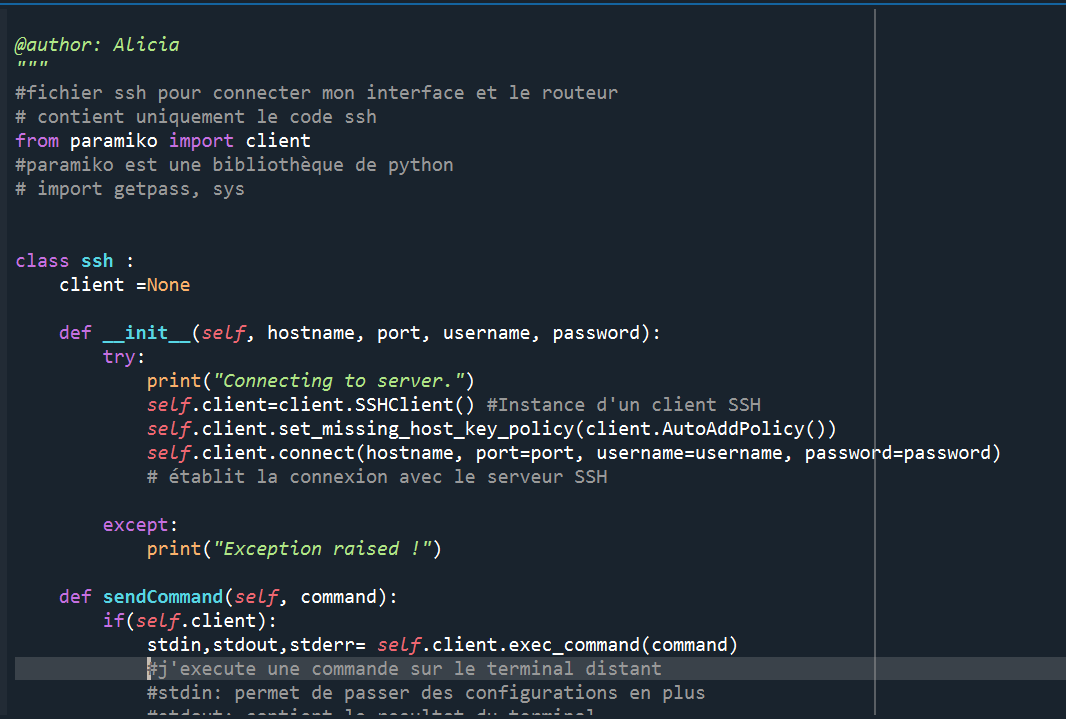
Malgré tout il me reste quelques fonctionnalités non codées tels que la fonction lire. J’ai eu des difficultés à comprendre sa fonction et comment l’implémenter dans mon code. De plus, le ping n’est pas totalement fonctionnel.

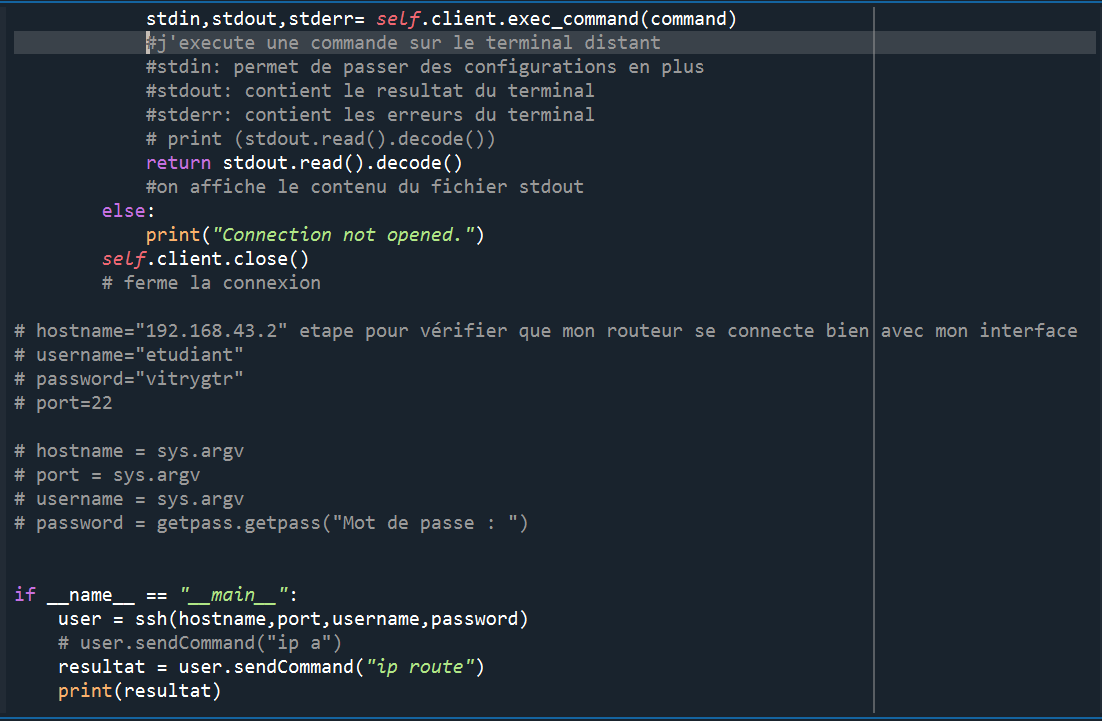
Conclusion

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.

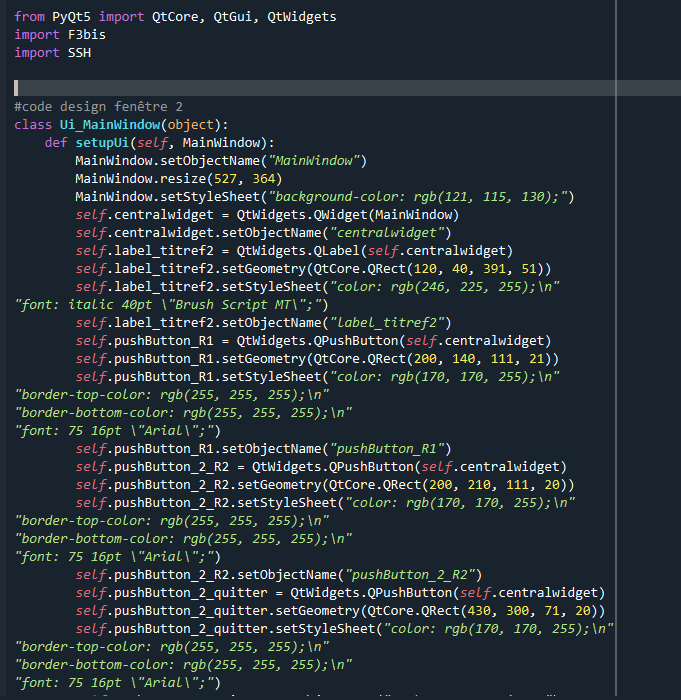
ANNEXE

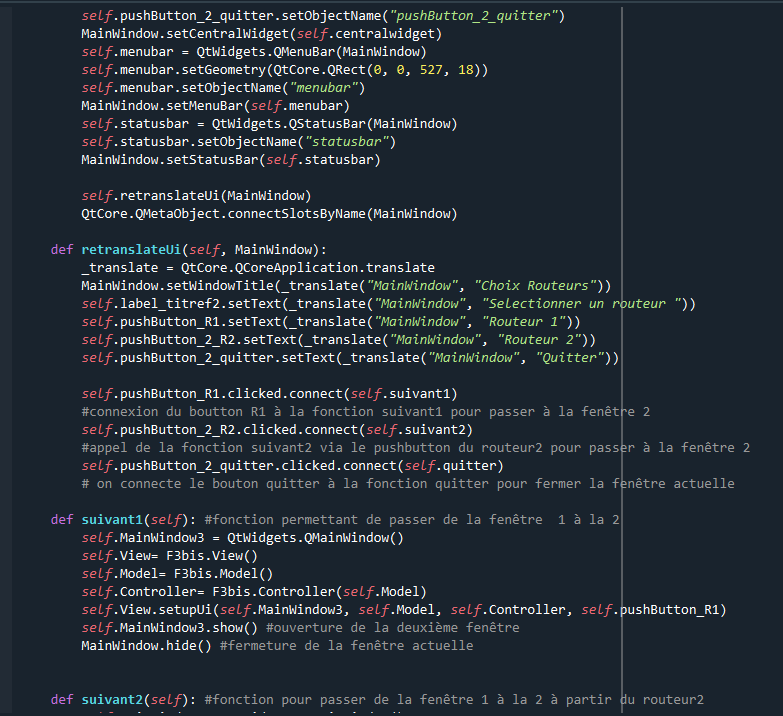
**Code SSH**

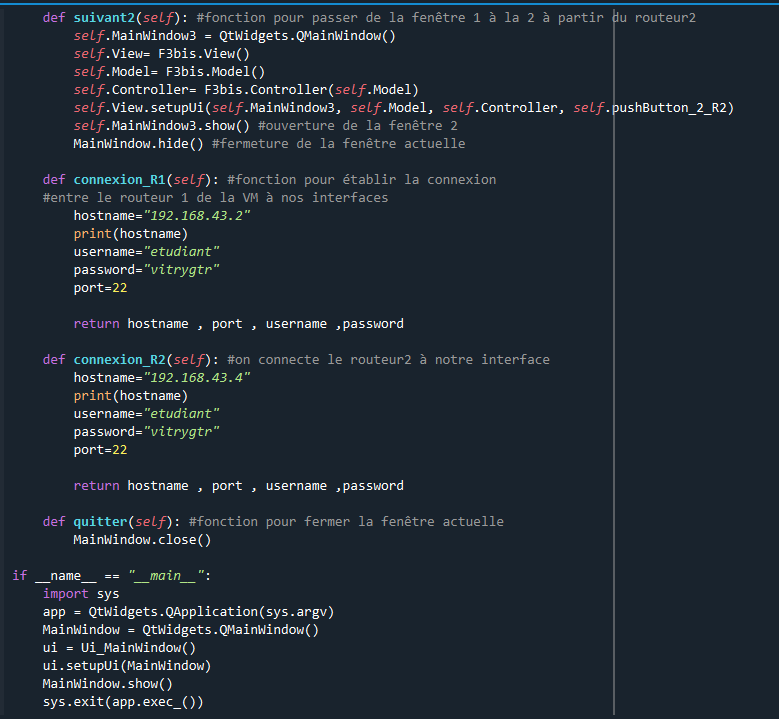




**Code première fenêtre et ses fonctionnalités**







**Code deuxième fenêtre**

