

ECOLE PRATIQUE DES HAUTES ETUDES COMMERCIALES

# EPHEC

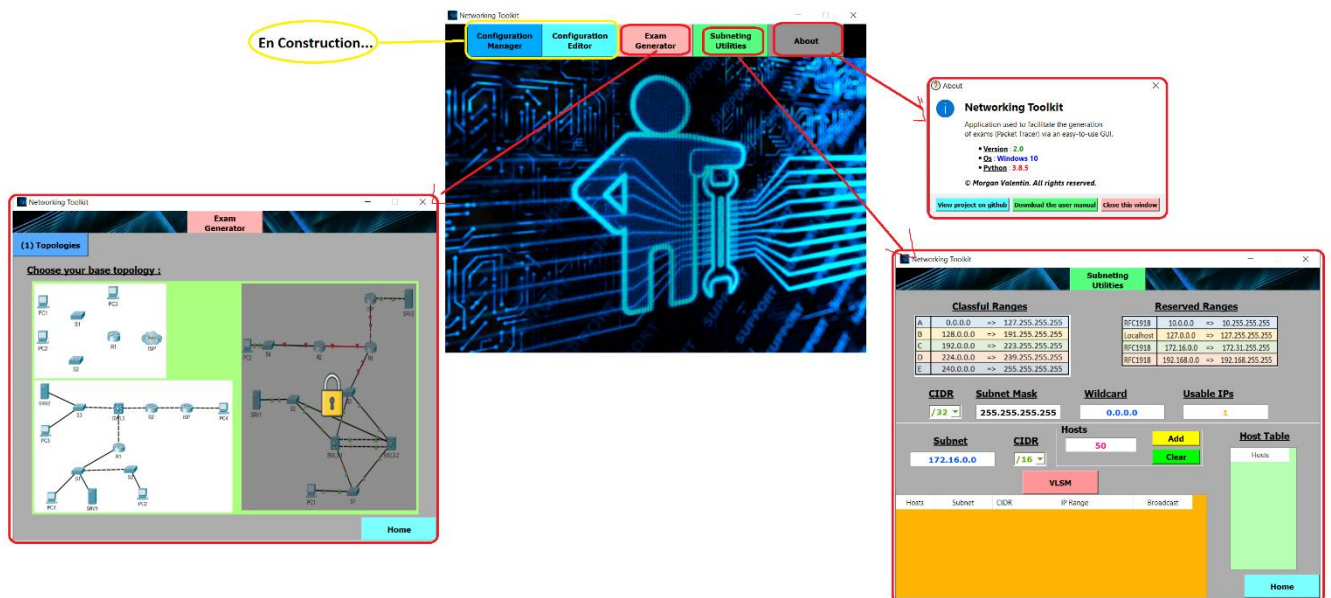
Avenue du Ciseau, 15

1348 Louvain-la-Neuve

## Networking Toolkit : Boite à outil permettant de générer facilement des examens pour Packet Tracer.

Travail de fin d'études présenté en vue de l'obtention du diplôme de bachelier en Informatique et Systèmes orientation Technologie de l'Informatique

**Morgan VALENTIN**



Rapporteur : Claude MASSON

Année Académique 2020 – 2021



## Avant-propos

Tout d'abord, je tiens à remercier **Madame Masson**, qui était toujours présentes lorsque j'en avais besoin, que ce soit pour de simples échanges de mails ou bien pour obtenir un rendez-vous via Microsoft Teams afin de discuter d'éventuelles difficultés que je rencontrais. Je la remercie aussi pour avoir testé mon programme et de m'avoir fait un feedback.

Je tiens également à remercier **Madame Vroman** pour son temps qu'elle m'a consacré étant donné qu'elle n'est pas mon rapporteur TFE. Je la remercie également pour avoir testé mon programme et de m'avoir fait un feedback sur celui-ci.

Je remercie surtout ces deux professeurs pour m'avoir soutenue dans ce projet de TFE.

# Table des matières

<b>1. Introduction</b>	<b>1</b>
1.1. Contexte	2
1.2. Objectifs	2
1.2.1. Clients	2
1.3. Besoins et contraintes	3
<b>2. Analyse</b>	<b>4</b>
2.1. Outil Packet Tracer	4
2.1.1. « Wizard » permettant de configurer un examen	5
2.2. Fonctionnalités à implémentées	6
2.2.1. Fonctionnalités des examens de 1 <sup>ère</sup>	6
2.2.2. Fonctionnalités des examens de 2 <sup>ème</sup>	6
2.2.3. Autres fonctionnalités aussi importantes	7
<b>3. Méthodologie</b>	<b>10</b>
3.1. Choix des technologies	10
3.2. Outils	11
<b>4. Développement</b>	<b>12</b>
4.1. Les différentes phases :	12
4.1.1. Phase n°1 : Fonctionnalités du périphérique	12
4.1.2. Phase n°2 : Structures de données	13
4.1.3. Phase n°3 : Design de la page (GUI)	14
4.1.4. Phase n°4 : Partie « logique » de la page	16
4.1.5. Phase n°5 : Tests	17
4.1.6. Phase n°6 : Génération des fichiers	18
4.2. Difficultés rencontrées	19
<b>5. Sécurité</b>	<b>20</b>
5.1. Sécurité au niveau de l'obtention du programme	20
5.2. Sécurité au niveau du programme	22
<b>6. Déploiement</b>	<b>23</b>
6.1. Automatisation du processus de déploiement	23
6.1.1. Code du script d'automatisation	24
<b>7. Suite</b>	<b>25</b>
7.1. Fonctionnalités prévues	25
<b>8. Conclusion</b>	<b>27</b>
<b>9. Glossaire</b>	<b>28</b>
<b>10. Bibliographie</b>	<b>31</b>



# 1. Introduction

Dans le cadre du TFE, il est demandé de répondre à un réel besoin venant d'un client. C'est-à-dire de partir d'un problème, passer par un chemin de réflexion, de proposer une solution et de la mettre en place.

Etant donné que je préfère le domaine du réseau, ce fut un peu plus compliqué de trouver des besoins réalisables.

Ma première idée de TFE tournait autour du fait que la configuration d'un réseau « réaliste » peut prendre beaucoup de temps car on doit passer du temps à lire les documentations des services réseau (par exemple pour mettre en place un serveur DHCP<sup>1</sup> ou DNS<sup>2</sup>), réfléchir au plan d'adressage IP, les appareils en jeu, etc...

Donc mon idée consistait à faire un programme où il faudrait simplement mettre les données qu'on souhaite et il se débrouillera pour générer le « code » à copier dans les divers appareils afin de les configurer.

Après en avoir parlé avec les professeurs de réseau de l'ephec, ils m'ont dit qu'il y avait quelque chose de similaire qui leur serait d'une grande utilité : ***permettre de générer des examens de manière plus rapide.***

L'idée proposée par les professeurs de l'Ephec, semble différente mais une grande partie du principe reste la même : avoir une interface simple où il suffit de mettre les données souhaitées et le programme se débrouillera pour générer ce qu'il faut pour les examens.

## 1.1. Contexte

Certains professeurs de l'Ephec, plus particulièrement les professeurs de réseau, peuvent prendre jusqu'à 3 jours pour mettre en place un examen (par exemple de 2<sup>ème</sup> année) correcte via un outil de la société Cisco<sup>3</sup>, « Packet Tracer<sup>4</sup> ».

D'un côté la création du réseau (le schéma réseau, l'adressage IP, les périphériques en jeu, les services réseau, ...) prends un temps considérable ne permettant pas toujours d'en générer plusieurs afin de faire face à la tricherie.

Et d'un autre côté, le temps de rechercher chaque case dans le « wizard<sup>5</sup> » de l'outil Cisco, d'attribuer les points correspondants et surtout de vérifier que tous ce qui doit être côté a bien été pris en compte par le logiciel.

## 1.2. Objectifs

L'objectif principale est de réduire le temps de création, de vérification et de mise en place d'examen réseau via l'outil Packet Tracer, permettant ainsi aux professeurs de gagner du temps.

### 1.2.1. Clients

Dans le cadre de mon TFE, j'ai eu l'opportunité d'échanger avec 2 clientes :

- Madame **Masson** – *Rapporteur TFE*
  - Professeur qui donne les examens de réseau de 1<sup>ère</sup> année.
- Madame **Vroman**
  - Professeur qui donne les examens de réseau de 2<sup>ème</sup> année.

Elles m'ont expliqué le problème vis-à-vis de la génération des examens réseau, et j'ai décidé de relever ce défi même si au départ, je ne savais absolument pas comment m'y prendre.

J'ai discuté avec mes clientes afin de déterminer les problèmes, voir comment elles font pour générer un examen et discuter d'une solution envisageable.

### 1.3. Besoins et contraintes

#### Respect de la demande du client :

- Il est demandé de mettre en place une solution permettant de générer des examens compatibles avec l'outil « Packet Tracer ».
- L'outil doit être capable de générer un fichier texte (ou autre) ayant une forte ressemblance à la structure d'encodage du programme « Packet Tracer » afin de faciliter l'encodage dans celui-ci.

#### Facilité d'utilisation :

- L'outil doit pouvoir s'exécuter sans passer par du code (ou par la ligne de commande) car les clients ne sont pas tous des programmeurs.
- L'outils doit être le plus « User-Friendly » possible afin d'éviter un maximum d'erreurs venant de l'utilisateur.

#### Utilisation du programme :

- Le programme doit être disponible hors ligne afin de ne pas dépendre d'une connexion internet pour travailler.
- La taille de la fenêtre du programme doit être au maximum la moitié de celle d'un écran d'ordinateur portable afin d'avoir d'un côté le programme « Packet Tracer » et de l'autre ce générateur (remplaçant ainsi le « bloc-note »).



## 2. Analyse

Avant de pouvoir me lancer dans des explications plus techniques, je dois vous expliquer le programme utilisé par les professeurs et les étudiants lors des cours de réseau (Packet Tracer), son fonctionnement (globalement) et comment fonctionne la configuration d'un examen via celui-ci.

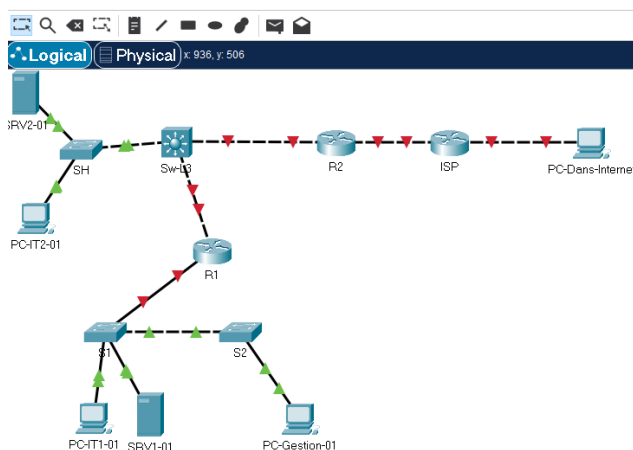
Je vais ensuite parcourir les fonctionnalités les plus importantes à mettre en place pour ce projet de TFE.

### 2.1. Outil Packet Tracer

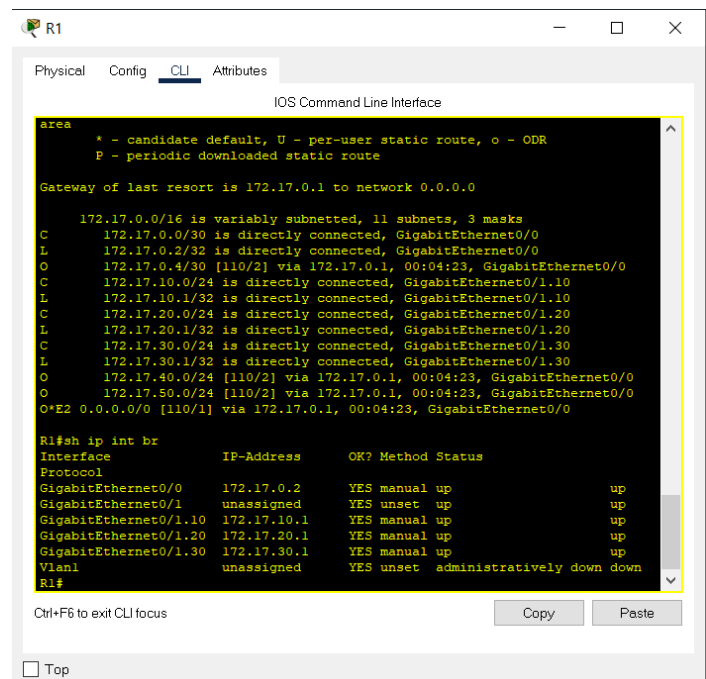
Packet Tracer est un outil propriétaire Cisco, permettant de simuler des réseaux. Lors des cours de réseaux (1<sup>ère</sup> et 2<sup>ème</sup> années), on est amené à utiliser ce logiciel pour s'entraîner, mettre en pratique les notions de théories vues au cours mais également pour passer des examens de réseaux.

Il y a d'un côté, la partie connexion des périphériques entre-eux, et d'un autre côté, la partie 'code' qui correspond à la configuration des périphériques.

Mon programme de TFE doit être capable de générer ce qui correspond à la configuration des appareils via une interface graphique simple. Pour ce qui concerne la partie connexion, elle est généralement configurée via un schéma légendé et une série de menu déroulant.



*Schéma réseau fait sur Packet Tracer*



*Capture d'écran du "CLI" (ligne de commande) de Packet Tracer*

### 2.1.1. « Wizard » permettant de configurer un examen

Pour générer un examen, il faut tout d'abord créer un réseau et configurer chaque appareil. Une fois cette étape faite, on peut cliquer sur une icône ressemblant à un « chapeau de fête » correspondant au 'Wizard' de Packet Tracer.

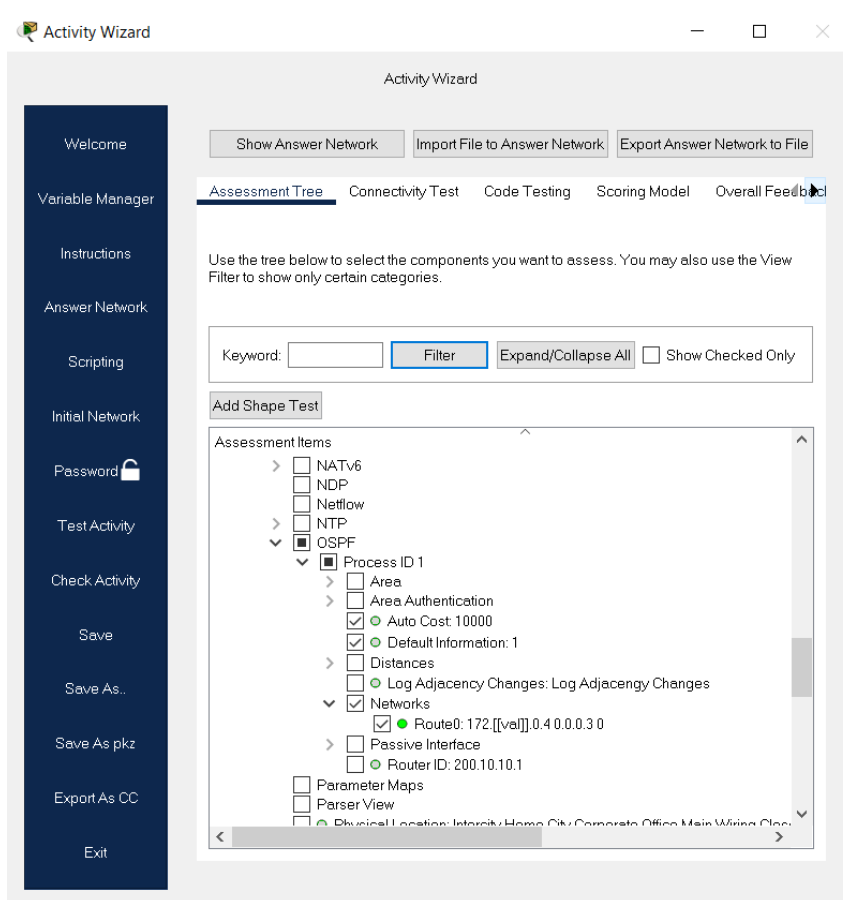
Celui-ci va considérer le réseau actuel comme réseau résolu et va afficher chaque commandes tapées (légèrement différent en réalité) sous forme d'arbre avec une case contenant soit une valeur ou contenant une autre case (donc un menu supplémentaire).

Par exemple, imaginons que dans le routage dynamique, on choisisse OSPF<sup>6</sup>.

Le 'code' pour OSPF est à gauche et la correspondance au niveau du 'wizard' est à droite.

Router ospf **1**  
Auto-cost reference bandwidth **10000**  
Network **192.168.10.0** **0.0.0.255** **0**  
Network **192.168.20.0** **0.0.0.255** **0**  
(...)

OSPF  
-Process ID **1**  
-Auto Cost : **10000**  
-Networks :  
-Route0 : **192.168.10.0** **0.0.0.255** **0**  
-Route1 : **192.168.20.0** **0.0.0.255** **0**



Capture d'écran du "wizard" de Packet Tracer

L'idée ici, c'est de vous montrer à quoi ressemble une partie de la configuration d'un périphérique et également comment elle est prise en compte dans le 'wizard' de Packet Tracer.

## 2.2. Fonctionnalités à implémentées

Les fonctionnalités à mettre en place sont réparties en principalement 2 grandes catégories, les fonctionnalités concernant les examens de 1<sup>ère</sup> et les ceux concernant les examens de 2<sup>ème</sup>. Il y a néanmoins, d'autres fonctionnalités qui sont-elles également assez importantes.

### 2.2.1. Fonctionnalités des examens de 1<sup>ère</sup>

- Système de VLSM<sup>7</sup>.
  - C'est-à-dire de permettre d'avoir plusieurs sous-réseaux basés sur un réseau.
- Gestion des règles d'attribution d'adresse IP basé sur celle du professeur.
- Sécurité des mots de passes, système d'accès distant SSH<sup>8</sup>.

### 2.2.2. Fonctionnalités des examens de 2<sup>ème</sup>

- Gestion des VLANs<sup>9</sup>.
- Configurations de serveurs<sup>10</sup>, routeurs<sup>11</sup> et de commutateur couche 3<sup>12</sup>.
- Service DNS
- Service DHCP
- Routage dynamique (OSPF – RIP<sup>13</sup>)
- NAT<sup>14</sup>
  - Redirection de port
  - Liste d'accès NAT
- SSH pour certains routeurs ainsi que des listes d'autorisations d'accès SSH.

### 2.2.3. Autres fonctionnalités aussi importantes

La liste des fonctionnalités décrite plus haut concernent les fonctionnalités principales et celles qui sont nécessaires pour avoir un examen correct.

Il y a cependant d'autres fonctionnalités qui sont en quelque sorte considéré comme bonus, mais toutes ont leur importance.

- Système de table comprenant ce que l'utilisateur a mis dans le programme.
  - Permet d'avoir un visuel sur ce que le programme va s'apprêter à encoder.
  - Equiper d'un bouton 'Clear' et 'Add' permettant d'ajouter dans la table ou bien de tout nettoyer au cas de bêtises.

Networking Toolkit

Exam Generator

(1) Topologies (2) VLAN

**VLAN**

VLAN n° : 20 Name : PROD

Subnet : 192.168.20.0 /24

Gateway : 192.168.20.254

DHCP IP ? :

Clear Add

SRV2 S3 R1 R2 ISP PC4  
PC3 S1 SRV1 S2 PC2

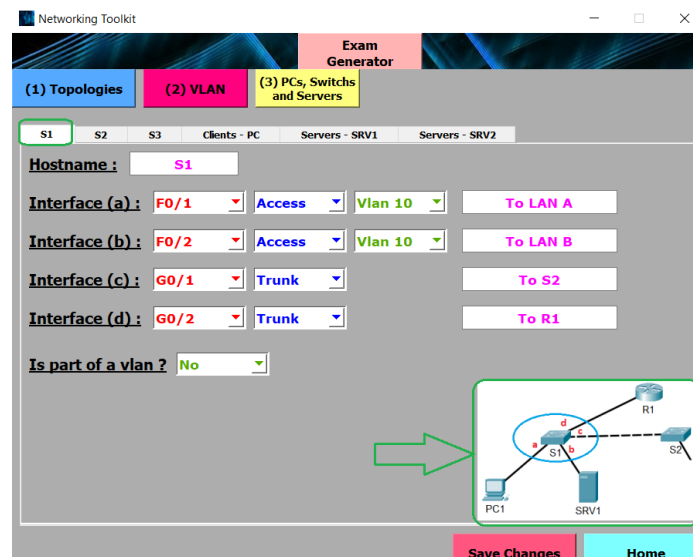
Vlan n°	Vlan name	Subnet	Mask	Gateway	DHCP ?	Is native ?
10	IT	192.168.10.0 (/24)	255.255.255.0	192.168.10.254	192.168.20.11	No
20	PROD	192.168.20.0 (/24)	255.255.255.0	192.168.20.254	No	Yes

Save Changes Home

*Capture d'écran d'une partie de la fenêtre "Exam Generator" du programme Networking Toolkit*

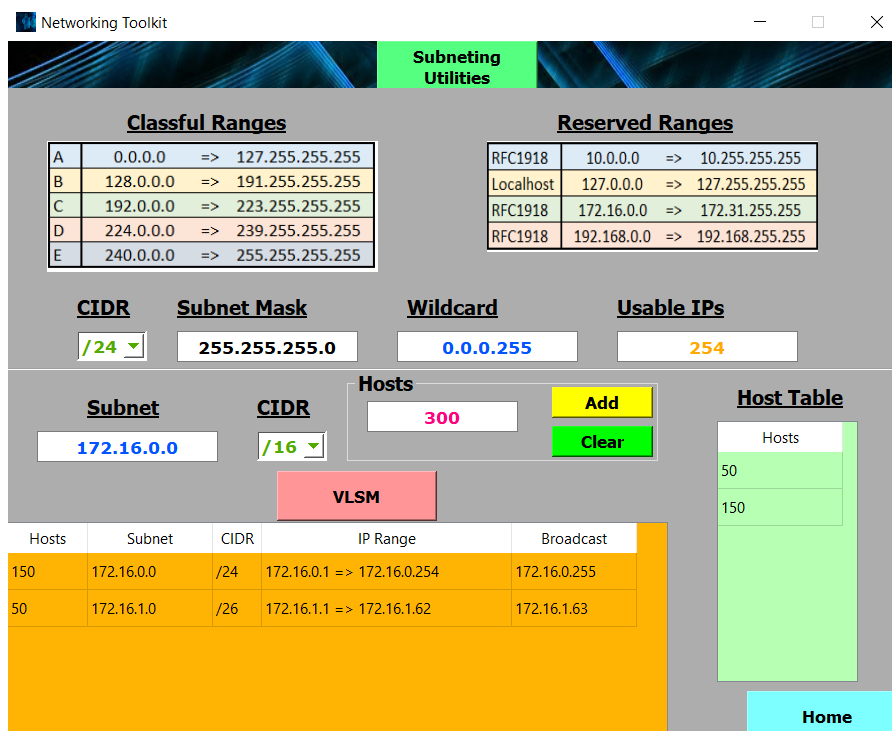
- Mise en place de texte devant un champ d'entrée afin d'expliquer ce qu'il faut y mettre.

- Schéma du réseau que l'utilisateur est en train de configurer ainsi que l'appareil en question entouré et légendé (si on est en train de le configurer)



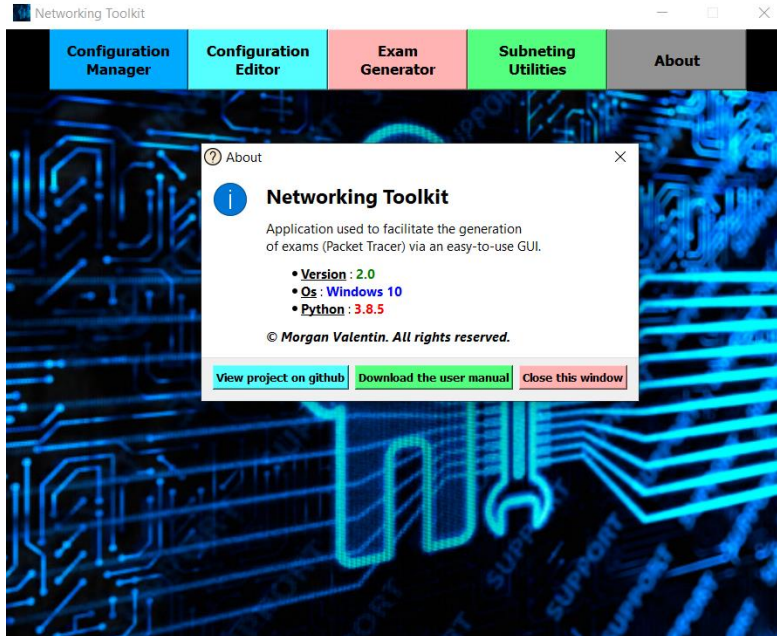
Capture d'écran d'une partie de la fenêtre "Exam Generator" du programme Networking Toolkit

- Page permettant d'effectuer des calculs de sous-réseau ; une sorte de calculatrice pour les professeurs de réseau.



Capture d'écran de la fenêtre "Subnetting Utilities" du programme Networking Toolkit

- Page 'About' permettant de spécifier la version du programme.
  - Permet de voir sur la page GitHub les bugs connus pour cette version.
  - Contient également un bouton permettant de télécharger le manuel d'utilisation du programme.



*Capture d'écran de la fenêtre "About" du programme Networking Toolkit*

### 3. Méthodologie

Dès que j'avais fini mes examens de janvier, je me suis directement lancé dans l'apprentissage du langage de programmation `python`<sup>15</sup>. J'ai fait de même avec la librairie `PyQt5`<sup>16</sup> afin de ne pas perdre trop de temps.

J'avais mis en place un `Trello`<sup>17</sup> avec les différentes fonctionnalités prévues, mais avec le stage qui me prenait beaucoup de temps, ce n'était plus possible pour moi de garder un horaire avec des dates limites pour effectuer certaines tâches.

La méthode de travail adapté au stage : dès que j'ai du temps de libre, je travaille le TFE. Le seul point négatif avec cette stratégie-là, c'est que je n'ai pas pu faire une véritable liste avec toutes les fonctionnalités implémentées.

#### 3.1. Choix des technologies

J'ai décidé d'opter pour le langage de programmation **python**. C'est l'opportunité pour moi d'apprendre un nouveau langage de programmation et en plus python est un outil très utile pour le domaine du réseau.

J'utilise la librairie « **PyQt5** », me permettant ainsi de créer des interfaces graphiques supérieur à celle par défaut avec « `Tkinter`<sup>18</sup> ».

##### Récapitulatif :

Langage	Version	Librairie ?
Python	3.8.5	PyQt5 (Permet de faire des interfaces graphiques moderne)

### 3.2. Outils

J'ai été amené à utiliser plusieurs outils lors de ce TFE dont certains que je ne connaissais pas auparavant.

- **Git et Github**
  - Outil de versionning très populaire, me permettant ainsi d'avoir une copie de mon code dans le cloud et de pouvoir revenir en arrière si nécessaire.
- **PyCharm**
  - IDE Python, j'ai utilisé la version pro avec la licence ephéc.
- **PyInstaller**
  - Module python incroyable permettant de compiler toutes les librairies / modules python en un seul exécutable Windows.
- **www.flaticon.com**
  - Site où j'ai pris quelques icones pour faire les différentes boites de dialogues et les manuels d'utilisation.
- **Packet Tracer (version 7.3)**
  - L'outil utilisé en cours de réseau.
- **Typora**
  - Une sorte d'IDE pour Markdown (le langage utilisé par Github pour le README.md, par exemple)
  - Il m'a permis de générer un pdf directement via du code Markdown.



## 4. Développement

Le développement de la solution « Networking toolkit » n'était pas facile à mettre en place dû au passage par de nombreuses phases avant de proprement « programmer ».

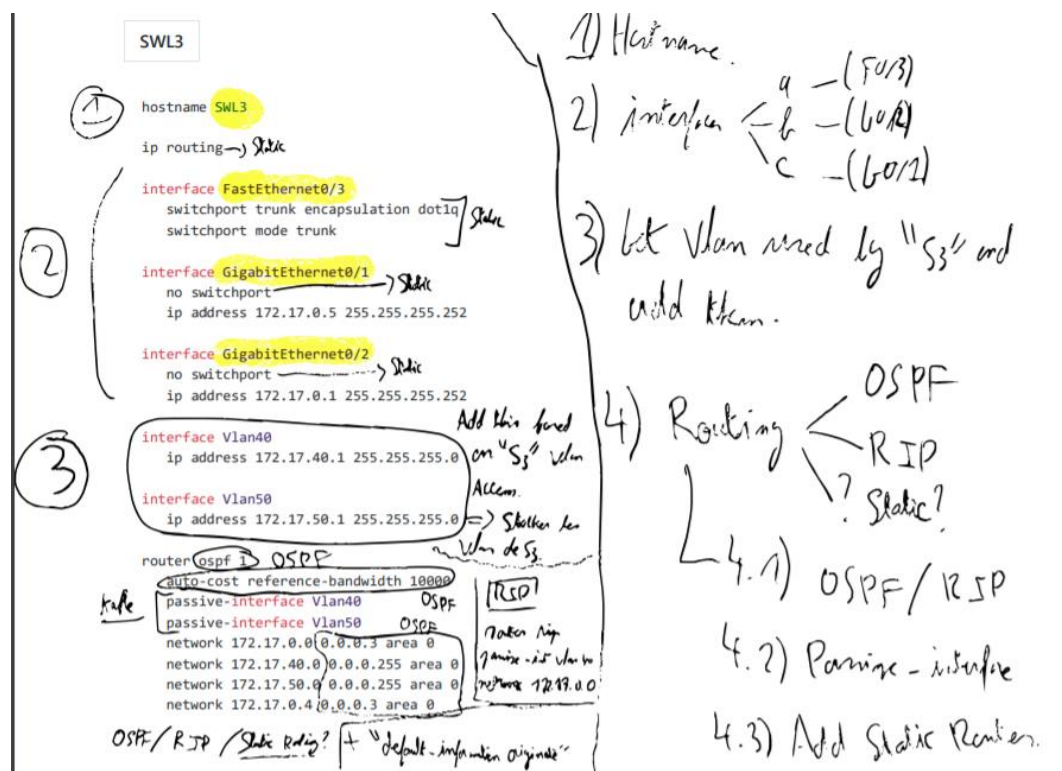
Je vais expliquer de mon mieux chaque phase par laquelle je suis passé.

### 4.1. Les différentes phases :

#### 4.1.1. Phase n°1 : Fonctionnalités du périphérique

La 1<sup>ère</sup> phase consiste à déterminer le type de l'appareil à configurer (assez simple), et d'analyser les fonctionnalités que celui-ci prend en charge et ce qui est demandé au niveau de l'examen (par exemple, de la NAT pour un routeur ou bien un service DHCP pour un serveur).

C'est à la fin de cette phase que je sais exactement quelles fonctionnalités de dois proposer pour l'appareil en question.



Analyse des fonctionnalités du commutateur de couche 3

#### 4.1.2. Phase n°2 : Structures de données

Cette phase porte sur comment faire pour que le programme retienne les informations saisies par l'utilisateur, plus précisément, quelle structure de données serait la plus utile pour enregistrer les données mais aussi pour permettre la manipulation de ceux-ci de manière optimale.

Par exemple, pour tous ce qui concerne le routage, j'utilise ce qu'on appelle un « dictionnaire » en python. C'est un système de « clé-valeur », donc associé à une clé une valeur correspondante. La valeur peut être un tableau, tout comme un autre dictionnaire par exemple.

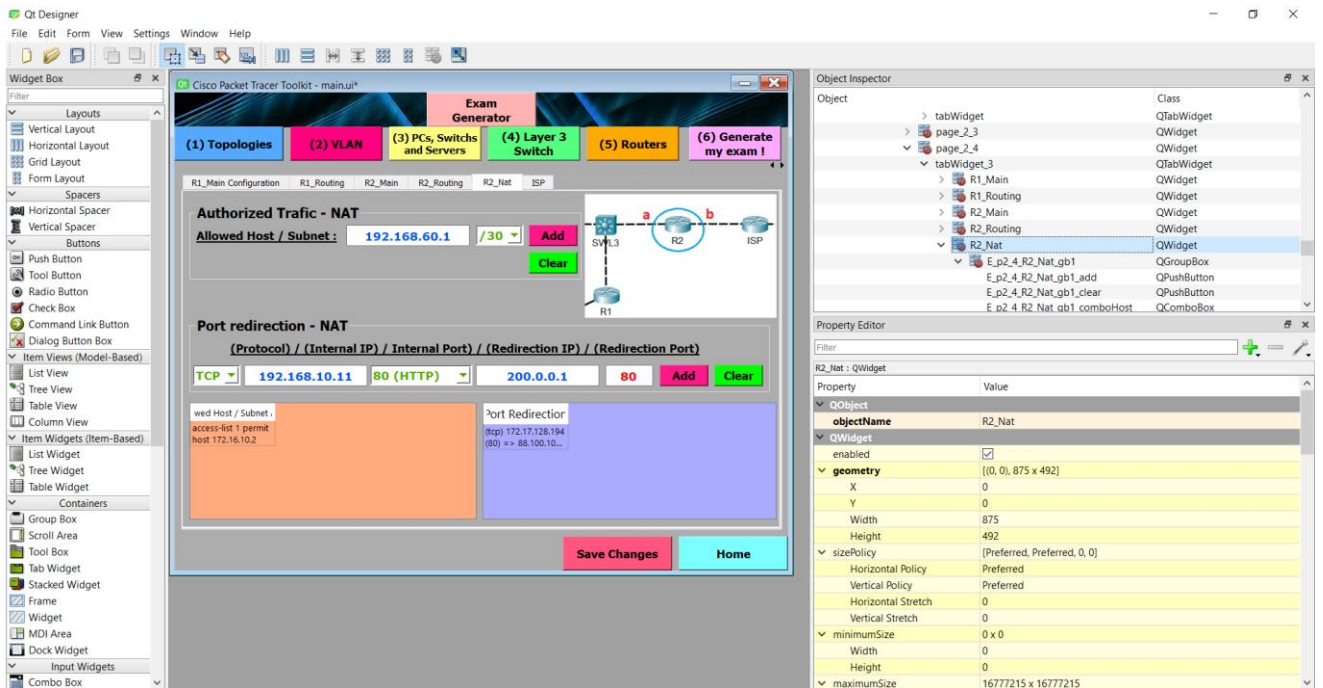
Et d'un autre côté, pour enregistrer les listes d'accès, j'emploi un tableau contenant ces informations sous forme de liste. Etant donné que dans Packet Tracer, il s'agit de la même ligne de texte pour configurer une liste d'accès que pour l'encoder au niveau de l'examen, alors je n'ai pas besoin d'une structure complexe tel qu'un dictionnaire.

### 4.1.3. Phase n°3 : Design de la page (GUI)

Cette phase est la plus « simple » du côté où elle ne nécessite pas autant de réflexion que les autres phases.

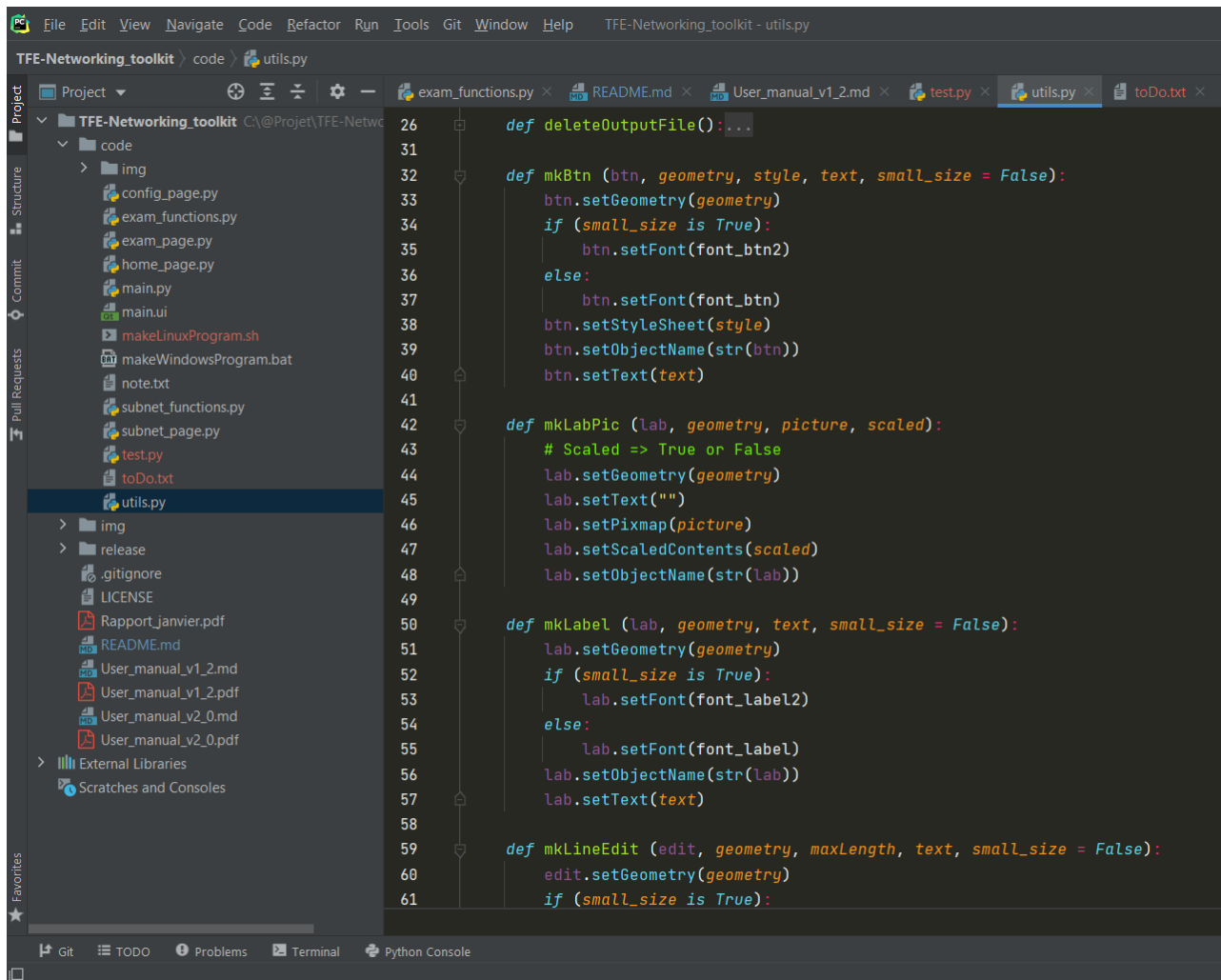
Cependant, le petit défi, c'est d'être capable de mettre toutes les informations nécessaires dans une fenêtre de taille limitée (pour rappel, je dois avoir un programme dont la taille de la fenêtre est égale à la moitié de la taille de l'écran).

Pour cette phase, j'utilise un outil (Qt Designer), qui me permet de faire le design du programme.



Capture d'écran de l'éditeur d'interface graphique "Qt Designer"

Même s'il est possible de générer le code directement via cet outil, le code généré demande beaucoup de temps pour l'adapter à mes besoins. Donc j'utilise cet outil pour voir à quoi va ressembler mon interface, et j'utilise des fonctions utilitaires (fonctions que j'ai créées dans le but de « construire » l'interface graphique de manière plus simple) pour mettre en place les composants graphique sur la fenêtre du programme.



```
26 def deleteOutputFile():...
31
32 def mkBtn(btn, geometry, style, text, small_size = False):
33     btn.setGeometry(geometry)
34     if (small_size is True):
35         btn.setFont(font_btn2)
36     else:
37         btn.setFont(font_btn)
38     btn.setStyleSheet(style)
39     btn.setObjectName(str(btn))
40     btn.setText(text)
41
42 def mkLabPic (lab, geometry, picture, scaled):
43     # Scaled => True or False
44     lab.setGeometry(geometry)
45     lab.setText("")
46     lab.setPixmap(picture)
47     lab.setScaledContents(scaled)
48     lab.setObjectName(str(lab))
49
50 def mkLabel (lab, geometry, text, small_size = False):
51     lab.setGeometry(geometry)
52     if (small_size is True):
53         lab.setFont(font_label2)
54     else:
55         lab.setFont(font_label)
56     lab.setObjectName(str(lab))
57     lab.setText(text)
58
59 def mkLineEdit (edit, geometry, maxLength, text, small_size = False):
60     edit.setGeometry(geometry)
61     if (small_size is True):
```

Extrait du fichier "utils.py" du repo GitHub

#### 4.1.4. Phase n°4 : Partie « logique » de la page

C'est dans cette phase où je réfléchis comment va se remplir un menu déroulant, ou quelles informations vont être affichées dans la table par exemple.

Après "Add" (clicked) → Route dimmed  
Après "Add" (clicked) → Enable (again)

GB1 →  
GB2 →

OSPF Process Id : 10  
Reference Bandwidth : 1000  
Network : 10.10.99.1 /30  
Area n° : 0

Static Routing  
Subnet : 0.0.0.0 /30  
Next hop / output interface : G0/2

Save Changes Home

1) Add interface a, b and c  
2) Add vlan used by S3!  
Ex: G0/1, G0/2, F0/26, Vlan up, Vlan S3

Exclude interface "A"

1) Add interface X to end C  
2) Add possible IP of router between Sub3-R1, Sub3-R2 and remove IP used by Sw23 from interface a and c

if OSPF: GB1 → Show, GB2 → Hide, GB3 → Hide  
if RIP: GB1 → Hide, GB2 → Show, GB3 → Show

192.168.0.0/24  
192.168.0.1/24  
192.168.0.2/24  
192.168.0.3/24  
192.168.0.4/24  
192.168.0.5/24  
192.168.0.6/24  
192.168.0.7/24  
192.168.0.8/24  
192.168.0.9/24  
192.168.0.10/24  
192.168.0.11/24  
192.168.0.12/24  
192.168.0.13/24  
192.168.0.14/24  
192.168.0.15/24  
192.168.0.16/24  
192.168.0.17/24  
192.168.0.18/24  
192.168.0.19/24  
192.168.0.20/24  
192.168.0.21/24  
192.168.0.22/24  
192.168.0.23/24  
192.168.0.24/24  
192.168.0.25/24  
192.168.0.26/24  
192.168.0.27/24  
192.168.0.28/24  
192.168.0.29/24  
192.168.0.30/24  
192.168.0.31/24

Réflexion sur la partie logique concernant le routage pour le commutateur de couche 3

#### 4.1.5. Phase n°5 : Tests

Cette phase me permet de savoir si j'ai fini ma page, ou si j'ai oublié une fonctionnalité, ou que quelque chose ne fonctionne pas comme prévu.

Pour tester mon programme, je simule le fait d'être un utilisateur du programme. Même après avoir publié une version du programme, je me rends compte par la suite qu'il y a parfois certains bugs. Tous ces bugs sont renseignés sur la page « readme.md » du repo GitHub du projet.

Version	Où se situe le problème	Description du problème	Solution
1.2	Page " (3) Connectivity "	Les valeurs par défauts des descriptions des interfaces sont toujours les mêmes.	<del>Sera corrigé dans la prochaine version.</del> Corrigé ! (v1.4)
1.2	Fichier de sortie : " packet-tracer.yaml "	Au niveau de la route statique, il manque "-0-1" à la fin pour que Packet Tracer la prenne en compte.	<del>Sera corrigé dans la prochaine version.</del> Corrigé ! (v1.4)
1.2	Page " (4) Addons "	Après avoir sauvegarder les données de la page 3, le bouton " (5) Generate my exam ! " est déjà visible.	Il faudra d'abord passer par la page " (4) Addons " et sauvegarder avant de pouvoir générer l'examen. <del>Sera corrigé dans la prochaine version.</del> Corrigé ! (v1.4)
1.4	Fichier de sortie : " packet-tracer.yaml "	Manque les descriptions des interfaces des switches.	<del>Sera corrigé prochainement.</del> Corrigé ! (v1.6)
1.4	Page " About "	Le numéro de version n'est pas à jour (toujours sur 1.2).	Un bouton permettant de télécharger directement le manuel d'utilisation sera ajouté également. <del>Sera corrigé prochainement.</del> Corrigé ! (v1.6)
2.0	(Examen de niveau 2) Page " (3) Pcs, Switchs and Servers ", onglet Servers.	Il manque un champ "DNS" au niveau des pool DHCP. Dans le pool, il devrait être possible d'indiquer le serveur DNS que les clients prendront en compte.	Il faudra ajouté un champ DNS au niveau des 2 serveurs + ajouter le résultat dans les 2 fichiers de sorties (solution et Packet Tracer). <del>Sera corrigé prochainement.</del>
2.0	(Examen de niveau 2) Fichier de sortie : " packet-tracer.yaml "	Les différentes zones OSPF ne sont pas prises en compte (champs " Area ") dans le fichier de sortie Packet Tracer.	<del>Sera corrigé prochainement.</del>

Extrait de la section "problèmes connus" du readme du repo GitHub/

#### 4.1.6. Phase n°6 : Génération des fichiers

Une fois que l'utilisateur à cliquer sur le bouton « Generate my exam ! », il y a deux fichiers qui sont créer :

- solution.txt (ou solution\_v2.txt - si c'est un examen de 2<sup>ème</sup>)
  - Fichier reprenant le « code » à insérer dans les appareils, permettant ainsi de les configurer.
- packet\_tracer.yaml (ou packet\_tracer\_v2.yaml – si c'est un examen de 2<sup>ème</sup>)
  - Fichier ayant une structure ressemblante très fortement à celle du « wizard » de l'examen de Packet Tracer. (Fichier le plus important pour le client)

C'est là que les clients peuvent voir si le programme leur est véritablement utile. Car si ce que le client à demander ne se trouve pas dans le fichier, alors ça ne va pas.

Juste à titre informatif, le 2<sup>ème</sup> fichier est celui dont les clients sont le plus intéressé. C'est également le plus compliqué étant donné qu'il doit y avoir un tri des clés, des valeurs qui « disparaissent » en fonction d'une fonctionnalité utilisé ou non, etc ...

## 4.2. Difficultés rencontrées

La chose la plus compliqué dans ce projet, c'est le « démarrage » de celui-ci. Une fois que j'ai pris le temps d'analyser les fonctionnalités nécessaires à l'examen pour chaque appareil, c'était déjà plus simple.

Au niveau des phases, la plus complexe pour moi, c'est la phase 2 (structures de données). Je la juge comme la plus compliqué, car c'est dans cette phase où je choisis une structure de données, puis c'est vraiment à la fin (phase 6 - génération de fichiers) où je me rends compte qu'en fait, ce n'est plus possible de juste travailler avec un tableau et qu'il me faudrait un dictionnaire par exemple.

Une autre difficulté que j'ai rencontrée, c'est le temps. Le stage me fatiguait de plus en plus, ce qui avait comme impact que j'avais moins de temps à consacrer au TFE et que sous la fatigue, je suis moins productif. La conséquence, c'est que le Trello que j'utilisais au début du TFE, n'a plus été mis à jour par manque de temps.



## 5. Sécurité

Dans le cadre de mon TFE, pour que le client puisse utiliser le programme, il doit télécharger une archive zip venant du repo GitHub, l'extraire et enfin exécuter le programme.

On peut donc dire qu'il faut vérifier l'aspect sécurité lors du téléchargement du programme, c'est-à-dire s'assurer que le fichier téléchargé n'a pas été modifié en cours de route (Intégrité du fichier) et lors de l'exécution de celui-ci.

### 5.1. Sécurité au niveau de l'obtention du programme

Afin de prouver l'intégralité du fichier téléchargé, donc dire que celui-ci n'a pas été modifié lors du téléchargement par exemple, on utilise un mécanisme de hash (empreinte en français).

Explications :

- Un hash est **unique** (donc si vous modifiez un fichier, alors le hash sera lui aussi différent)
- Une fois un hash généré, **impossible de faire l'opération inverse**.
  - Par exemple, j'ai un mot de passe qui est « toto », si je génère le hash de celui-ci, il n'est pas possible que sur base du hash généré je puisse retrouver mon mot de passe original.
- Utilité ?
  - Si je génère le hash de mon fichier et j'obtiens par exemple « azerty ».
  - Je vous donne le fichier et je vous dis comment j'ai fait pour générer mon hash (type de fonction de hachage).
  - Vous générer le hash et si le hash généré correspond au mien, alors le fichier n'a pas été altéré.
    - Car si le fichier aurait été modifié, alors le hash aurait également changé (vu qu'un hash est unique).

Dans le fichier « README.md » du GitHub, j'ai mis un tableau avec le hash SHA-1<sup>19</sup> correspondant à chaque version du programme que j'ai mis à disposition.

[Lien vers la section dans le readme.](#)

**Vérification de l'intégrité du fichier :**

Nom du fichier	hash sha1 associé
release_windows_v1.0.zip	2f7677315ed1412290d8318b9d17e308f580d369
release_windows_v1.2.zip	3b8b4821e52105fd6737129626b17d7b59a05e40
release_windows_v1.4.zip	8c0a6e78f2746a66930f85804e6217500d8fd533
release_windows_v1.6.zip	b627149beb60631cb93b5a2aa7dc63bf48a6badd
release_windows_v2.0.zip	0774c2af9874d10ab4dbe8aeca835166786aaf35

**Comment générer un hash sha1 sous windows ?**

1. Télécharger le fichier sur votre machine.
2. Ouvrir l'explorateur de fichier windows et aller dans le dossier où se situe le fichier téléchargé (généralement "Téléchargement")
3. Maintenir la touche "Maj" ("shift" en anglais) + faire un click droit avec la souris et sélectionner "Ouvrir powershell ici"
4. Taper la commande " `certutil -hashfile nom_du_fichier` "
5. Vérifier si le hash correspond à celui mis dans le tableau ci-dessus.

**PS : Si le hash est différent (donc les données ont été altérées = possible virus), ne l'ouvrez surtout pas.**

*Capture d'écran de la section concernant l'intégralité du fichier du repo GitHub.*

*(J'ai mis les explications à la suite de ce tableau afin de faciliter la chose.)*

Du coup, pour vérifier que notre fichier téléchargé n'a pas été modifié en cours de route, il suffit d'ouvrir une fenêtre PowerShell<sup>20</sup> (là où se situe le fichier téléchargé) et de taper la commande suivante : « **certutil -hashfile** » suivi du nom du fichier téléchargé.

Vous allez obtenir une valeur. Comparez la valeur obtenue avec celle mise dans le tableau. Si les valeurs sont identiques, alors c'est parfait.

**Vérification de l'intégrité du fichier :**

Nom du fichier	hash sha1 associé
release_windows_v1.0.zip	2f7677315ed1412290d8318b9d17e308f580d369
release_windows_v1.2.zip	3b8b4821e52105fd6737129626b17d7b59a05e40
release_windows_v1.4.zip	8c0a6e78f2746a66930f85804e6217500d8fd533
release_windows_v1.6.zip	b627149beb60631cb93b5a2aa7dc63bf48a6badd
release_windows_v2.0.zip	0774c2af9874d10ab4dbe8aeca835166786aaf35

**Comment générer un hash :**

Select Windows PowerShell

```
PS C:\Users\Jervis2-3\Downloads> certutil -hashfile release_windows_v2.0.zip
SHA1 hash of release_windows_v2.0.zip:
0774c2af9874d10ab4dbe8aeca835166786aaf35
CertUtil: -hashfile command completed successfully.
PS C:\Users\Jervis2-3\Downloads>
```

1. Télécharger le fichier sur
2. Ouvrir l'explorateur de f

*Exemple de génération d'un hash SHA1 via PowerShell sous Windows.*

## 5.2. Sécurité au niveau du programme

Au niveau du programme, il faut savoir que celui-ci va créer 2 fichiers ('solution.txt' ou 'solution\_v2.txt' et 'packet\_tracer.yaml' ou 'packet\_tracer\_v2.yaml') directement sur le bureau après avoir cliqué sur le bouton « Generate my exam ! ».

Le programme va également supprimer tous fichiers ayant les mêmes noms que ceux cité plus haut (car c'est le programme qui est censé les créer, donc il va les supprimer afin d'en générer d'autres par la suite.

## 6. Déploiement

Pour ce qui concerne le déploiement du programme de TFE, il faut prendre en compte que les clients ne veulent pas devoir exécuter du code via un interpréteur python, ou lancer une commande via le PowerShell de Windows.

Les clients attendent de moi un outil où ils leurs suffiront de cliquer sur un exécutable Windows (comme pour d'autres applications) et que celui-ci se débrouille pour fonctionner.

Cela à pour contrainte que je dois trouver un moyen de rassembler tout le code python, toutes les librairies et tous les modules utilisés afin d'en faire qu'un exécutable Windows.

Pour cela, il y a un module incroyable qui s'appelle PyInstaller qui permet justement de faire cela. Le seul petit problème, c'est que celui-ci n'est pas capable de mettre également les images dans cet exécutable (c'est pour cela qu'il a le dossier « img » qui est copié également sur le bureau).

### 6.1. Automatisation du processus de déploiement

Comme dit plus haut, pour déployer l'application, je dois passer par le module « PyInstaller ». De base, il ouvre une fenêtre permettant de configurer ce que l'on souhaite via une interface graphique, puis génère le programme.

L'ennui de procéder de cette manière, c'est que je dois à chaque fois ouvrir le programme PyInstaller, choisir les fichiers de codes, et faire des manipulations qui sont répétitives. Après avoir fait quelques essais et lu la documentation, j'ai mis en place un script batch (s'exécute dans la ligne de commande Windows) permettant d'automatiser tout ce processus me permettant ainsi de juste exécuter ce script et d'attendre qu'il soit terminé.

**Lien vers le code du script :** [https://github.com/momo007Dev/TFE-Networking\\_toolkit/blob/main/code/makeWindowsProgram.bat](https://github.com/momo007Dev/TFE-Networking_toolkit/blob/main/code/makeWindowsProgram.bat)

### 6.1.1. Code du script d'automatisation

```
1 @echo off
2 title makeWindowsProgram
3 mode con cols=80 lines=16
4 color 02
5
6 :: Get the User's Desktop path
7 SETLOCAL
8 FOR /F "usebackq" %%f IN (`PowerShell -NoProfile -Command "Write-Host([Environment]::GetFolderPath('Desktop'))") DO (
9     SET "DESKTOP_FOLDER=%%f"
10 )
11
12 echo [*] This script will attempt to build all python dependencies into one windows executable
13 echo (*) The output will be available on the desktop (exe + img folder)
14
15 echo Creating one file executable...
16
17 :: Removes the current executable
18 if exist "%DESKTOP_FOLDER%\Networking Toolkit.exe" (
19     DEL /S/Q "%DESKTOP_FOLDER%\Networking Toolkit.exe"
20 )
21 :: Removes the current img folder
22 if exist "%DESKTOP_FOLDER%\img\" (
23     rmdir /S/Q "%DESKTOP_FOLDER%\img\"
24 )
25
26 XCOPY %CD%\img %DESKTOP_FOLDER%\img /Q /I
27 python -m PyInstaller --noconfirm --onefile --windowed --distpath "%DESKTOP_FOLDER%" --icon "C:/@Projet/TFE-Networking_toolkit/code/img/logo.ico"
28 --name "Networking Toolkit" --add-data "C:/@Projet/TFE-Networking_toolkit/code/config_page.py;" --add-data "C:/@Projet/TFE-Networking_toolkit/code/exam_functions.py;"
29 --add-data "C:/@Projet/TFE-Networking_toolkit/code/exam_page.py;" --add-data "C:/@Projet/TFE-Networking_toolkit/code/home_page.py;"
30 --add-data "C:/@Projet/TFE-Networking_toolkit/code/subnet_functions.py;" --add-data "C:/@Projet/TFE-Networking_toolkit/code/subnet_page.py;"
31 --add-data "C:/@Projet/TFE-Networking_toolkit/code/utils.py;" "C:/@Projet/TFE-Networking_toolkit/code/main.py"
32 rmdir /S/Q %cd%\build
33 rmdir /S/Q %cd%\__pycache__
34 DEL /S/Q "%CD%\Networking Toolkit.spec"
35
36 echo Finished !
37 color 06
```

*Script d'automatisation de déploiement*

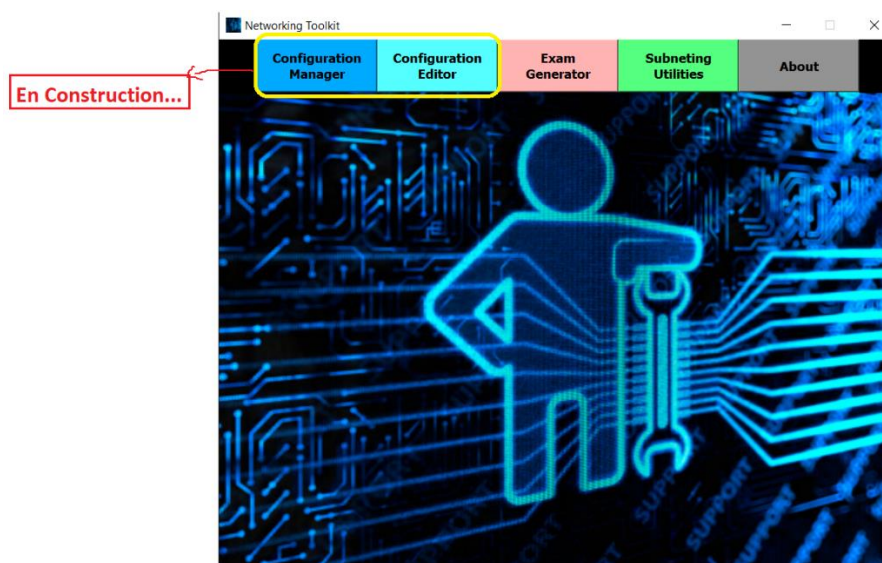
## 7. Suite

J'ai mis en place un système de générateur d'examen pour Packet Tracer pour les examens de 1<sup>ère</sup> année et de 2<sup>ème</sup> année (1<sup>er</sup> Quadrimestre).

Je compte par la suite mettre en place l'examen de 2<sup>ème</sup> année (2<sup>ème</sup> Quadrimestre) qui contient toute la matière vue au cours de d'infrastructure réseau.

Ce qui est vraiment chouette avec ce projet, c'est que les possibilités d'ajout de fonctionnalités sont énormes.

### 7.1. Fonctionnalités prévues



*Capture d'écran de la fenêtre principale du programme Networking Toolkit*

Comme vous pouvez le voir dans l'image ci-dessus, il y a deux boutons qui pour l'instant affiche une boîte de dialogue (lors d'un click sur l'un d'entre eux) indiquant que ces deux pages sont en construction.

- **Le 1<sup>er</sup> bouton – « Configuration Manager » :**
  - L'idée, c'est qu'ici, on choisit un périphérique et le programme demande ce que l'utilisateur souhaite configurer sur celui-ci. Le principe réside sur le fait que l'utilisateur aura le « code » permettant de configurer le périphérique en question par le biais d'une interface simple et de quelques clics.
  - **Utilité ?**
    - La configuration des périphériques dans la section concernant la configuration des examens du programme, est « contraint » aux possibilités de l'examen. C'est-à-dire, que je ne vais pas proposer des fonctionnalités avancées pour un examen de 1<sup>ère</sup> année par exemple mais également le fait que la topologie<sup>21</sup> ne permet pas de mettre en place tel ou tel fonctionnalité.
    - Ce configurateur proposera également de configurer des services linux comme un serveur DHCP ou DNS en se basant sur de vraies configurations (donc plus directement lié à Packet Tracer)
- **Le 2<sup>ème</sup> bouton – « Configuration Editor »**
  - C'est tout simplement une sorte d'IDE où l'utilisateur pourrait :
    - Visualiser les fichiers générés par le programme avec des couleurs dans un premier temps et les éditer.
    - Être capable de vérifier « l'orthographe du code » afin d'aider l'utilisateur s'il souhaite éditer l'un des fichiers pour ajouter du code. Autrement dit, être capable de détecter des erreurs.
    - Être capable de proposer des mots à l'utilisateur qui seront bien évidemment en respect avec la ligne de commande Packet Tracer.

## 8. Conclusion

Ce TFE était pour moi l'opportunité de mettre en place un projet concret de « A à Z », c'est-à-dire avoir une demande réelle de clients, des retours avec les clients, devoir me débrouiller tout seul pour proposer une solution et le fait d'avoir pu délivrer cette solution au client.

Ce projet m'a permis aussi d'apprendre un langage de programmation : le python. J'ai également manipulé une librairie me permettant de faire des interfaces graphiques moderne et eu l'occasion de voir comment passer du code au programme final (exécutable Windows) où il suffit de « cliquer dessus ».

Le plus belle de chose de ce projet selon moi, c'est de voir que les clients sont véritablement satisfaits de ma solution. Je ne vais pas le nier, j'ai dû surmonter une multitude d'obstacles mais malgré tout cela, j'ai quand même fini par réussir à mettre en place la solution du client.



## 9. Glossaire

### 1) DHCP

- Dynamic Host Configuration Protocol, est un protocole permettant de configurer de manière dynamique les clients. Celui-ci se base sur un serveur spécifique qui a pour but d'attribuer des adresses IP aux hôtes qui en font la demande. **(Reformulation tiré du livre Réseau 4<sup>ème</sup> édition de Andrew Tanenbaum).**

### 2) DNS

- Domain Name System, est un protocole qui s'occupe de la résolution des noms (dire quel nom correspond à quel adresse IP). Il associe une adresse logique (un nom de domaine) à une adresse physique (adresse IP).

### 3) Cisco

- « Cisco Systems est une entreprise informatique américaine spécialisée, à l'origine, dans le matériel réseau (routeurs et commutateurs Ethernet), et depuis 2009 dans les serveurs. » **Wikipédia.**

### 4) Packet-Tracer

- Outil permettant de simuler du matériel réseau Cisco (créé par la société Cisco) dont le but est de permettre aux utilisateurs d'approfondir leurs connaissances en réseau, plus spécifiquement dans la technologie Cisco.

### 5) Wizard

- C'est une sorte d'assistant faisant partie d'un programme qui a pour but de nous guider à travers certaines étapes.
- (Dans le cas du wizard de Packet Tracer, celui-ci a pour but de guider l'utilisateur dans la manière de générer un examen réseau).

### 6) OSPF

- Open Shortest Path First, est un protocole de routage, de type « à état de liens ». Ce protocole échange uniquement des informations sur l'état de ces liaisons ce qui rend la convergence (changement/adaptation du réseau) bien plus rapide. **(Reformulation tiré du livre « Technologie des ordinateurs et des réseaux » de Pierre-Alain Goupille).**

### 7) VLSM

- Variable Length Subnetting Mask, comme son nom l'indique, c'est un masque de sous-réseau de longueur variable. Le VLSM permet le découpage en sous-réseaux afin d'optimiser le processus d'attribution d'adresses.

## 8) SSH

- Secure Shell, permet d'administrer une machine à distance de manière sécurisé via un terminal (shell) linux.

## 9) VLAN

- Virtual Local Area Network, permet de regrouper des machines sans avoir à tenir compte de leur emplacement dans le réseau. La division du réseau est donc logique et non physique.
- Les messages envoyés à un VLAN ne sont reçus que par les « membres » de ce VLAN. **(Reformulation tiré du livre « Technologie des ordinateurs et des réseaux » de Pierre-Alain Goupille).**

## 10) Serveurs

- Machine offrant divers services tel que des pages web (serveur web), services mails, serveur de fichiers, etc.

## 11) Routeurs

- Machine qui relie des stations (clients par exemple) situées sur des réseaux ou sous-réseaux différents (ou non), quel que soit le protocole utilisé. Il a pour mission d'assurer l'acheminement des paquets, filtrer et contrôler le trafic entre les réseaux. **(Reformulation tiré du livre « Technologie des ordinateurs et des réseaux » de Pierre-Alain Goupille).**

## 12) Commutateur de couche 3

- « Un commutateur réseau (en anglais switch), est un équipement qui relie plusieurs segments (câbles ou fibres) dans un réseau informatique et de télécommunication et qui permet de créer des circuits virtuels. » **Wikipédia.**
- Couche 3, veut simplement dire que c'est un commutateur de couche 2 (classique) avec des capacités de routage (comme les routeurs).

## 13) RIP

- Routing Information Protocol, est un protocole de routage de type « distance-vecteur ». Ce protocole se contente de communiquer avec son voisin (routeur) le plus proche en échangeant une partie de sa table de routage. Si le nombre de routeur (voisins) traversés est supérieur à 15, RIP considère que la connexion n'est pas stable. **(Reformulation tiré du livre « Technologie des ordinateurs et des réseaux » de Pierre-Alain Goupille).**

## 14) NAT

- Network Adresse Translation, est une technique de traduction d'adresses IP.
- Généralement utilisé pour traduire une adresse IP interne (c'est-à-dire une adresse IP qui ne peut **pas** communiquer sur internet) vers une adresse IP externe (qui peut communiquer sur internet).

### 15) Python

- Langage de programmation « open source » interpréter et multiplateforme dont l'un de ces avantages permet aux programmeurs de se concentrer sur ce qu'ils font au lieu de comment ils le font.

### 16) PyQt5

- Module libre permettant de mettre en place des interfaces graphiques moderne via le langage python.

### 17) Trello

- Site web permettant la gestion de projet via des « cartes ». Très utile pour avoir une bonne organisation lors de travail de groupe.

### 18) Tkinter

- C'est la bibliothèque standard de python permettant de faire des interfaces graphique « basique ».

### 19) Hash SHA-1

- SHA-1 (Secure Hash Algorithm) est le type de la fonction de hachage utilisé pour générer le hash (ou condensat).
- SHA-1 possède la caractéristique d'avoir longueur de 40 caractères.

### 20) PowerShell

- « *PowerShell est une solution multiplateforme d'automatisation des tâches, composée d'un interpréteur de commandes (shell), d'un langage de script et d'un framework de gestion de la configuration.* » **Microsoft**.

### 21) Topologie

- Schéma représentant l'architecture du réseau.

## 10. Bibliographie

- 1) [Comment générer un hash de type SHA-1 / MD5 sous Windows.](#)
- 2) [Documentation des composants graphiques de la librairie « PyQt5 ».](#)
- 3) [Outil utilisé pour faire le design du programme.](#)
- 4) [PyCharm, l'IDE Python utilisé pour développer ce projet.](#)
- 5) [« Cheat-sheet » utilisé pour certains calculs de sous-réseaux.](#)
- 6) [Typora, un IDE Markdown, utilisé pour faire le manuel d'utilisation du programme.](#)
- 7) [Définition pour « Cisco ».](#)
- 8) [Définition pour « Commutateur ».](#)
- 9) [Définition pour « PowerShell ».](#)
- 10) Livre « Technologie des ordinateurs et des réseaux », de Pierre-Alain Goupille (9<sup>ème</sup> édition).
- 11) Livre « Réseaux » de Andrew Tanenbaum (4<sup>ème</sup> édition).