h e g

Haute école de gestion
Genève

Création d'un algorithme d'aide à l'évaluation des modules de réseaux

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

Par :

Aurélie Sauge

Conseiller au travail de Bachelor :

André Seydoux

Genève, 21 août 2022

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre Bachelor HES en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : https://www.urkund.com .

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seule le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Versoix, le 21 août 2022

Aurélie Sauge

Remerciements

Je remercie M. André Seydoux de m'avoir encadré et aidé dans la réalisation de ce projet. Son soutien et ses conseils ont été précieux pour la réussite de mon projet.

Je remercie aussi Christiane Sauge pour la relecture de mon dossier.

Je remercie aussi Coralie Chevalley et Steven Vaglio pour avoir testé la beta de l'application, ainsi que pour leurs judicieuses remarques.

Résumé

Ce projet consiste à aider les professeurs lors de l'évaluation du module de réseaux. En effet, la correction des contrôles continus ou examens pratiques est fastidieuse et longue. Pour chaque élève, il faut récupérer le fichier .txt et l'analyser en le comparant au corrigé type, fait par le professeur. Un fichier .txt fait par un étudiant peut comporter plusieurs milliers de lignes à analyser. Ce temps de travail va donc être largement réduit grâce à l'application mise en place dans ce travail de bachelor.

L'application ResCorTxt permet d'évaluer le travail d'un étudiant en quelques minutes. Le professeur saisit le nom et le prénom de l'élève dont le travail est à corriger, insère le fichier .txt de l'élève et le fichier .txt corrigé dans l'application. Il sélectionne ensuite les différents critères d'évaluation et peut changer, s'il le souhaite, les différents coefficients pour chaque critère. Suite à cela, le professeur lance la correction et l'algorithme va analyser les deux fichiers pour évaluer et donner une note sur 6, et retourner un fichier Excel. Le fichier Excel va montrer la grille d'évaluation avec les différents points accordés ou non, ainsi qu'une synthèse des différents points mal compris ou faits faux par l'étudiant. Le fichier Excel automatisé pourra donc être revu par le professeur. Si ce dernier souhaite revérifier le travail d'analyse de l'algorithme, ou simplement modifier des coefficients, la note sera donc ajustée automatiquement.

Table des matières

Liste des tableaux

Aucune entrée de table d'illustration n'a été trouvée.

Liste des figures

1. Introduction

J'ai décidé de choisir ce sujet, car le domaine de réseau me parle énormément et me plaît. De plus, c'est un travail pratique, bien plus intéressant à mon goût qu'un travail théorique. Lorsque mon professeur, M. Seydoux m'en a parlé, j'ai immédiatement aimé l'idée, et ai directement commencé à réfléchir comment mettre en place l'algorithme de calcul.

ResCorTxt est une application destinée aux professeurs de réseau de la Haute École de Gestion et la correction des différents examens de réseau est un processus long et fastidieux. Chaque examen ou contrôle continu est corrigé par deux professeurs à tour de rôle pour obtenir une note finale. Chaque fichier .txt de chaque élève fait plus d'environ 1000 lignes par appareil, soit une quantité de lignes de commande importantes à analyser à l'œil nu. De plus, des erreurs peuvent être commises et c'est pour cela que deux professeurs corrigent chaque élève, le temps passé pour chaque élève étant d'autant plus important.

Cette application est une idée de mon professeur de mémoire, pour permettre un gain de temps lors des corrections. En effet, si un algorithme permet de lire le fichier et de ressortir uniquement les informations importantes en notant la valeur de ces informations, les professeurs n'auraient plus qu'à relire pour vérifier le travail fait par l'algorithme. C'est le but de ResCorTxt.

ResCorTxt permet de lire un fichier .txt élève et un fichier .txt corrigé et de les comparer. Il permet aussi de fixer des coefficients de correction pour différents critères d'évaluation. Avec toutes ces informations, l'algorithme programmé derrière l'application va analyser le fichier de l'élève et en ressortir uniquement les lignes de commandes importantes. Il va analyser les lignes et attribuer une note par ligne de commande. À la suite de ce fonctionnement, l'algorithme sera en capacité de calculer la note finale de l'élève sur 6.

Pour une utilisation encore plus pratique de cette application on peut, à la fin du calcul quand nous avons obtenu le résultat, télécharger un fichier Excel qui récapitule l'intégralité de la note. Toutes les commandes utilisées pour noter y seront rapportées et le calcul final de la note, dans la partie barème en fin de feuille, est entièrement automatisé sur Excel pour permettre une modification simple et un calcul en cascade.

2. Présentation des recherches post codage

2.1 Recherche des différents outils utiliser

Dans un premier temps, il a fallu faire des recherches dans l'optique de savoir quel langage utiliser durant le projet, et quels logiciels sont les plus optimisés pour permettre le bon déroulement du codage.

J'ai dû faire des recherches pour plusieurs parties de code. Nous avons la partie interface avec le design visuel, la partie algorithme avec le calcul de la note et la partie retour en fichier Excel.

Plusieurs choix s'offraient à moi. Je pouvais coder l'interface en C#, ayant déjà fait un cours avec ce langage de programmation. Pour l'algorithme, je pensais partir sur le langage Java, ayant le plus de connaissances dans ce langage et le maîtrisant correctement. Il fallait aussi trouver un langage de programmation qui me permette de remplir un fichier Excel automatisé.

Après plusieurs recherches dans ce sens je me suis rendue compte que mélanger le C# et le Java sur un logiciel serait bien plus compliqué que de coder en un unique langage. Le design est fait en unique partie en C#, et m'étant rendue compte que le codage de l'algorithme en java serait très similaire au codage en C#, j'ai décidé de faire l'intégralité de mon travail en C#.

Pour la recherche du logiciel, cela a été plus facile. L'école proposant une clé d'activation pour le logiciel Visual Studio, j'ai donc décidé de l'utiliser. Ce logiciel est très précis et très complet dans la multitude de fonctionnalités qu'il propose. J'ai donc crée l'intégralité de l'interface visuelle sur celui-ci. Lors du codage de l'algorithme de fond, je me suis rendue compte que j'utilisais souvent le débogueur et les Console.WriteLine. L'application Visual Studio ne proposant pas de pouvoir déclencher une unique méthode mais l'ensemble du projet, je ne pouvais pas accéder à mes Console.WriteLine. J'ai donc fait une seconde recherche sur un logiciel similaire, qui serait potentiellement moins performant niveau design mais plus pratique niveau lecture de méthode. Après plusieurs recherches, j'ai trouvé le logiciel Rider proposé par JetBrain. Forte de l'expérience des logiciels JetBrain que je trouve tous très intuitifs et faciles à manipuler, j'ai décidé d'utiliser celui-ci. Rider était donc plus facile

d'utilisation pour moi, car c'est une interface très similaire à IntelliJ (logiciel pour le langage Java), très utilisée durant l'intégralité du bachelor.

J'ai aussi dû faire des recherches dans l'utilisation d'un Excel en C#. En effet, je souhaitais pouvoir remplir et mettre en forme un Excel tout automatisé à la suite du calcul de la note. L'utilisateur n'aura plus qu'à télécharger le fichier et aura le détail de la note avec un descriptif des fautes commises par l'élève noté. Après plusieurs recherches, j'ai trouvé l'utilisation d'une librairie « Office.Interop.Excel ». Directement dédiée à C# pour la création d'un Excel, cette librairie permet le remplissage simple des cellules. Je me suis vite rendue compte que cette librairie ne serait pas suffisante, car il fallait pouvoir mettre en forme l'Excel et faire des remplissages avec formule dans certaines cellules. J'ai donc dû trouver une librairie plus puissante. « E-Iceblue »est une api qui permet de faire quasiment tout ce qui est possible de faire sur l'interface directe d'Excel en codage C#. L'utilisation de cet api n'a pas été des plus simples à prendre en main. Et après plusieurs essais infructueux, j'ai décidé de ne pas utiliser cette librairie.

2.2 Recherche des différentes erreurs à vérifier

Avant de pouvoir coder, il m'a fallu faire plusieurs recherches et m'interroger sur les différentes erreurs que peuvent faire les élèves évalués.

D'une manière générale, il fallait voir quelles erreurs pouvaient être faites lors de la création et du rendu du fichier .txt. J'ai trouvé trois erreurs que l'application va devoir prendre en charge :

- L'élève peut mettre 4 fois la même configuration. C'est-à-dire qu'il peut écrire dans le .txt 4 fois la même configuration pour le même appareil sans s'en rendre compte.
- L'élève peut aussi ne pas respecter l'ordre demandé pour l'enregistrement des appareils. On va demander un ordre d'enregistrements (R1, R2, R3, ...). Mais l'élève peut ne pas respecter cet ordre. Malgré tout, le travail a été fait et l'élève ne peut pas être totalement pénalisé pour ça.
- Il faut aussi gérer le manque d'appareils. Un élève peut ne pas avoir eu le temps de configurer un appareil, il faut donc pouvoir lui mettre les différentes commandes manquantes dans la grille d'évaluation et dans la synthèse.

D’une manière plus spécifique lors de la vérification des critères d’évaluation.

Configuration de base :

- L’élève peut ne pas mettre le bon hostname. À partir de cette erreur, il ne faut pas que l’algorithme interprète un appareil pour un autre. En effet, si un élève se trompe de hostname en appelant un routeur par « Switch1 », l’algorithme devra enlever les points du hostname, mais faire une vérification routeur sur l’appareil et ne pas le mettre dans la liste des switchs.

Interface de gestion :

- On vérifie qu’une interface utilisée, donc qui possède une adresse ip, n’est pas en mode « shutdow ».
- On va aussi vérifier que la passerelle par défaut des switchs correspond bien à l’adresse qui a été mise dans le routeur connecté, et non pas dans un autre routeur ou une autre adresse ip.

Câblage :

- On va vérifier que l’élève n’a pas mis 2 câblages sur la même interface. Et donc que l’interface ait plusieurs configurations. Par exemple, qu’elle soit prise pour une sous-interface (donc router-on-a-stick), et qu’elle soit connectée en direct au réseau cantonal de façon simple.
- On va aussi vérifier que les liaisons sont correctes. Que chaque interface dans chaque appareil est bien connectée à la bonne interface dans l’appareil voisin.

Routage :

- On va vérifier qu’il ne manque pas de réseau par rapport au corrigé.
- On vérifie que les bonnes interfaces ont été passivées et non pas une interface utilisée.
- On vérifie que toutes les lignes de commande en fonction de l’énoncé sont notées. Par exemple, oublier un redistribute alors que l’on a un routage dynamique et statique en même temps.
- On va vérifier qu’il ne manque pas l’ip route par défaut dans le routage statique.

Trunk :

- On vérifie que toutes les commandes sont présentes.
- On va aussi vérifier que la trunk a été faite sur la bonne interface.
- On va aussi vérifier que ce sont les bons vlan qui sont attribués.

Port access :

- On vérifie que toutes les commandes sont présentes.

Sécurité des ports actifs :

- On vérifie que toutes les commandes sont présentes.

Sécurité des ports down :

- On vérifie que dans le switch toutes les interfaces non utilisées sont « shutdown ».

Vlan :

- On vérifie que tous les vlan ont été créés et qu'ils sont bien attribués aux bonnes interfaces.

SSH :

- On va vérifier le bon nom de domaine.
- Il faut vérifier qu'il y a les clés cryptées.
- On vérifie que la configuration a été faite avec la version 2.
- On vérifie aussi que les lines ont été modifiées.

Access-list :

- On va vérifier que les réseaux sont justes.
- On vérifie aussi la spécificité des protocoles si c'est demandé.

Utilisateurs (PC) :

- On va simplement vérifier qu'on lui a attribué une adresse ip. Et que le test a été fait.

3. Développement de l'application

3.1 Création de l'interface

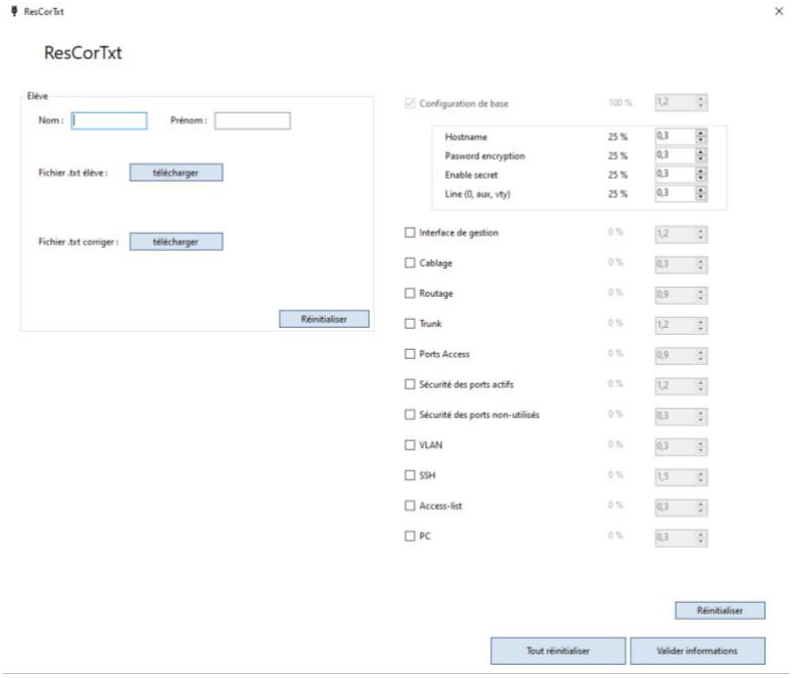
J'ai créé l'interface directement sur Visual Studio. J'ai opté pour un design assez simple et fonctionnel. L'interface principale comporte deux parties.

La première partie concerne les informations à entrer par l'utilisateur pour pouvoir faire le calcul. On va demander le nom et prénom de l'élève qui sera noté, ainsi que le fichier de l'élève et le fichier corrigé. Le nom est important , car il sera utilisé pour enregistrer le fichier Excel.

La deuxième partie concerne les différents critères d'évaluation. On peut y voir les différents critères d'évaluation qu'il faut cocher ou non en fonction de ce que l'on veut évaluer. La configuration de base est toujours évaluée, mais on peut décider de ne pas la compter dans la note finale en réduisant son coefficient à 0. Tous les autres critères d'évaluations doivent être cochés. Un coefficient de base est noté dans l'application par mes soins, mais chaque utilisateur pourra modifier le coefficient pour qu'il représente le pourcentage de la note qu'il souhaite.

La première partie a été directement créée sur l'interface design de Visual Studio. En revanche, la deuxième partie possède un emplacement, mais celle-ci sera créée lors du lancement de l'application. Cette méthode de création des critères d'évaluation permet un automatisme de création et l'ajout facile d'un critère supplémentaire.

Figure 1 : Interface ResCorTxt à l'ouverture de l'application



Une deuxième page a dû être créée : la page d'affichage du résultat. Cette page permet de transmettre directement les informations de la note au professeur. On pourra y voir la note générale obtenue, ainsi que le détail de chaque note obtenue pour chaque critère d'évaluation spécifique. La particularité est aussi de pouvoir informer le professeur qu'il a coché un critère d'évaluation à vérifier, mais que ce critère n'est pas présent dans le corrigé. Cela signifie qu'il ne sera pas vérifié. Cela peut être important pour donner une information directe au professeur et permettre de modifier le corrigé avant de faire tourner l'application pour chaque élève pour éviter de s'en rendre compte du problème à la fin du travail.

3.2 Création du fichier Excel

3.2.1 Création fichier

Pour ce projet, il était essentiel de devoir donner un récapitulatif du calcul au professeur. Avec celui-ci, le professeur aurait l'opportunité de vérifier le calcul, ou de modifier, s'il le souhaite, la note de l'élève. J'ai donc décidé de récupérer la grille d'évaluation actuellement utilisée par les professeurs. Et d'en recréer une directement remplie par le logiciel.

Pour pouvoir créer un fichier Excel directement dans le logiciel, le remplir, et pouvoir le récupérer, j'ai utilisé la librairie Office.Interop. Cette librairie permet de créer n'importe quel fichier office (Excel, Word, etc.) et de lui apporter des modifications. Il a fallu donc que je me renseigne sur internet pour savoir comment utiliser cette librairie.

Dans un premier temps, il a fallu créer une « Application.Office.Interop.Excel », qui me permet de créer un fichier Excel. Il faut ensuite créer nos différentes feuilles de calcul. Il y en a deux dans notre cas, la grille d'évaluation et la synthèse. Ensuite, il a fallu faire un remplissage des feuilles. J'ai décidé de créer des méthodes automatisées, expliquées dans la partie juste après. Et enfin de sauvegarder et fermer le fichier Excel.

3.2.2 Remplissage et automatisation

Pour pouvoir remplir le fichier Excel directement, il m'a fallu créer un moyen de transmettre les informations à afficher de ma classe « Calcul »à ma classe « CreateExcel ». J'ai utilisé le biais des List couplée avec des Dictionnary. Il existe une List pour chaque type d'appareils. Chaque List sera composée d'un Dictionnary avec le nom de l'appareil en clé et d'un autre Dictionnary en valeur. Le Dictionnary en valeur comportera un autre Dictionnary avec le nom de la configuration actuelle (CB = Configuration de base, IG = Interface de gestion, etc.) comme clé et d'un autre Dictionnary en valeur. Qui sera lui-même composé d'un Dictionnary avec comme clé le string de la commande vérifiée et en valeur un boolean si la commande est juste ou fausse. Ces listes seront donc récupérées dans notre classe « CreateExcel » pour pouvoir remplir les différentes informations de correction. Pour chaque liste, on va boucler sur celle-ci et afficher la ligne de commande vérifiée avec sa valeur. La valeur boolean va directement être remplacée par 1 si elle est égale à « True » et 0 si elle est égale à « False ». Ce sera donc l'insertion pour attribuer les différents points.

Après avoir bouclé sur l'ensemble des listes et affiché l'ensemble des commandes vérifiées, on va avoir une partie récapitulative pour vérifier la note obtenue. On aura donc un récapitulatif en fonction de chaque critère d'évaluation. Chaque critère aura le nombre de points que l'élève a obtenu, ce calcul est directement calculé par Excel grâce à une formule « somme » insérée dans le logiciel. Le nombre de points maximum pour ce critère d'évaluation sera aussi calculé avec la formule « somme » dans l'Excel. Un pourcentage de réussite est établi pour ce critère. Ce calcul est un produit en croix à partir des deux informations calculées juste avant directement repris

dans Excel. Les points obtenus par l'élève sont déterminés en fonction du coefficient donné. C'est aussi un calcul Excel fait en fonction des cellules du pourcentage et du coefficient, affiché dans la colonne juste après ce calcul. Finalement, le coefficient donné pour ce critère d'évaluation est simplement reporté depuis l'interface. Le dernier calcul automatisé est le calcul de la note. On va récupérer la somme des coefficients et la somme des notes par critères obtenus et les mettre sur 5. En effet, la somme des coefficients n'étant pas forcément égale à 5, il faut remettre le tout sur 5 pour obtenir notre note finale sur 6. On ajoute le 1 officiel et notre note est posée.

L'importance de ce fichier Excel est qu'il permettra au professeur de ne pas avoir à relire l'entièreté des lignes du fichier .txt de l'élève, vu que chaque ligne sera notée. Le professeur pourra simplement vérifier les lignes sur l'Excel une par une et vérifier si l'algorithme n'a pas été trop sévère et n'a pas attribué un point qui aurait pu être accepté. La modification du fichier par le professeur est très simple, car l'automatisation du barème final permet la simple modification des points (1 ou 0) par lignes de commande, pour modifier en cascade tous les calculs.

En ce qui concerne la feuille de synthèse, lors du remplissage de la grille d'évaluation, on va vérifier si la commande est juste ou non pour faire l'affichage correct. Mais on va aussi faire une vérification si l'élève a réussi ou non. Si la commande est juste, on va simplement l'écrire dans la grille d'évaluation, mais si la commande est fausse son affichage sera dans la grille d'évaluation, mais aussi dans la synthèse. En effet, cette synthèse permet au professeur et à l'élève de voir d'une manière plus simple l'ensemble des lignes de commande fausses et donc de voir l'ensemble des erreurs ou des informations incomprises par l'élève.

3.2.3 Mise en forme

J'ai voulu faire une mise en forme claire et simple pour une relecture plutôt rapide. Avec les différentes complexités des listes qui permettent le remplissage du Excel, il est possible de détailler de façon simple les lignes de commandes. On pourra donc afficher dans un premier temps les différents routeurs. Ils seront affichés un par un, avec les critères d'évaluations les uns après les autres. Cela va permettre une lecture par type d'appareils, par appareils et enfin par critères d'évaluation. Le barème est donc affiché en fin de fichier pour permettre un récapitulatif et un remplissage avec des formules automatisées.

C’est la même méthode pour le remplissage de la synthèse. On aura un affichage par type d’appareils, puis par appareils et enfin par critères d’évaluation.

3.3 Création de l’algorithme de calcul

La principale tâche de ce travail était de développer un algorithme qui permette le calcul d’une note en la simple analyse d’un fichier .txt. Cet algorithme a été développé en plusieurs parties. Chaque critère d’évaluation a donc son propre calcul, pour avoir une note finale détaillée, tout comme chaque critère peut avoir plusieurs sous-calculs pour la recherche de différents points d’évaluation. Toute cette infrastructure mène à un algorithme très complexe et très fastidieux, néanmoins très performant.

3.3.1 Lecture des fichiers .txt

Pour faire la vérification et pouvoir lancer l’algorithme, il faut des informations importantes. On va avoir besoin de deux fichiers .txt : le fichier à corriger, c’est-à-dire le fichier de l’élève, et le fichier corrigé, c’est-à-dire le fichier du professeur, le fichier « tout juste » ou exemple.

Pour pouvoir accéder à ces fichiers depuis l’application, l’utilisateur va devoir aller chercher les fichiers dans son explorateur de fichiers sur l’ordinateur qu’il utilise. Lorsqu’il va cliquer sur le bouton de téléchargement, une fenêtre de recherche dans l’explorateur va s’ouvrir. Il n’aura donc plus qu’à rechercher et sélectionner le fichier qu’il veut vérifier (le fichier élève) ou le modèle (corrigé). Pour que cette fonctionnalité apparaisse, on va utiliser la méthode « openFile » qui est directement implémentée dans Visual Studio. Cette fonctionnalité permet d’accéder à l’explorateur de fichiers de l’ordinateur en question et va retourner un string avec le chemin du fichier.

Une fois que le chemin des fichiers a été saisi et enregistré, on va passer par un lecteur de fichiers. On va passer par « readFile », directement implémenté en C#. Cette méthode permet de récupérer un fichier à un chemin précis et de le lire ligne par ligne. Cela va nous retourner une liste avec chaque ligne du fichier dans chaque cellule du tableau.

Pour une meilleure vision du fichier et un processus plus rapide de l’algorithme, on va directement travailler sur cette liste pour supprimer les lignes en trop. En effet, lorsque que l’on regarde un fichier .txt on remarque qu’il y a un saut de ligne entre chaque

commande et que beaucoup de lignes, surtout dans le show running-config, comportent uniquement des « ! ». Toutes ces lignes sont donc inutiles. On va donc faire un premier process en supprimant toutes les lignes vides et les lignes qui ne contiendraient que des « ! ». Cela va donc supprimer plus de la moitié des lignes du fichier, ce qui permettra de supprimer de grandes capacité de ressources de l'application lors des multiples boucles de recherche.

3.3.2 Séparation du matériel en type (Routeur, Switch, PC)

Pour une meilleure évaluation des différents critères d'évaluation, je me suis rendue compte qu'il serait préférable de séparer les différents appareils. En effet, certains critères d'évaluation ne concernent que les routeurs ou les switchs, il serait donc inutile de travailler sur l'ensemble des appareils. J'ai donc décidé de créer trois différentes listes pour séparer les routeurs, les switchs et les PCs. Chacune des listes comportera une ou plusieurs listes de lignes de code. Si, par exemple, on a trois routeurs, on disposera de la liste des routeurs (liste qui comprend l'entièreté des routeurs), qui comportera donc trois cellules. Et dans chacune des cellules, on aura les différentes lignes de code pour le routeur en question. Chacune des trois listes est formatée de la même façon.

Pour créer ces listes, on va dans un premier temps devoir faire une recherche sur la liste créée à la suite de la lecture du fichier. J'ai considéré qu'à chaque fois que l'on avait la commande « show running-config » on entrait dans un nouvel appareil (routeur ou switch) et que lorsque l'on avait la commande « windows ip config » on entrait dans la configuration d'un nouveau PC.

Suite à cette division de la liste initiale, il a fallu savoir si le nouvel appareil était un switch ou un routeur. J'ai, dans un premier temps, comparé le type au hostname (R1 = routeur, S1 = switch). Mais je me suis ensuite rappelé que l'erreur de nomination des appareils est courante. En effet, un élève peut se tromper en renommant un routeur avec S1. L'algorithme considérerait cet appareil comme un switch alors que c'est un routeur. Il y aurait donc de multiples erreurs de calcul lors de l'analyse. J'ai donc préféré reconnaître les différents appareils à leurs caractéristiques particulières, en faisant la vérification du nombre d'interfaces. Si l'appareil en question contient « F0/20 », donc à plus de 20 interfaces, cela ne peut être qu'un switch, sinon c'est un routeur. On aura donc trois listes complétées avec les différents appareils. Ces trois listes seront donc développées pour le corrigé, mais aussi pour l'élève. On aura au

final six listes. Ces listes seront les bases de la recherche dans les appareils pour la vérification des lignes de commandes.

Une autre vérification a dû être faite. En effet, lors du rendu, l'élève peut ne pas mettre les appareils dans l'ordre demandé sur l'énoncé du CC. Par manque de temps, ou mauvaise organisation, il peut mélanger ces appareils. On va donc faire une réorganisation des différentes listes remplies auparavant pour que les listes corrigées et les listes des élèves soient entrées dans les mêmes appareils dans le même ordre. Cela sera important pour toutes les vérifications futures.

On va donc vérifier les différents hostname des appareils. S'il correspond à un hostname du corrigé, il prendra la même place dans la liste de l'élève. En revanche, si l'élève n'a pas respecté la nomination de ces appareils, il n'y a pas de moyens pour s'assurer d'avoir les mêmes appareils en comparaison. C'est pourquoi cet algorithme est sensible à l'ordre demandé dans le corrigé et à la nomination des appareils. La note pourrait donc ne pas refléter la totalité du travail de l'élève.

En revanche, si le professeur remarque cette erreur, il peut faire la correction de place ou le changement de nomination de l'appareil dans un txt dupliqué pour lancer l'algorithme et faire un calcul de note plus correct.

3.3.3 Codage de la configuration de base

Le codage de la configuration de base avec les quatre critères attitrés : « hostname », « service password-encryption », « enable sercet » et les différentes configurations des lines, a été la première étape de l'algorithme. En effet, cette configuration m'a permis de faire des tests sur le fonctionnement prévu de l'algorithme et de modifier en amont ce qui devait l'être. Le codage de la configuration de base, surtout le calcul du hostname, a dû être refait plusieurs fois. Je me suis rendue compte au fur et à mesure des potentiels problèmes que pouvaient amener mes versions de l'algorithme. J'ai notamment constaté que mes listes, pour faire l'écriture dans le fichier Excel, n'étaient pas bonnes.

Le hostname est le critère de la configuration de base qui a demandé le plus de travail. Pour le corrigé, j'ai simplement fait une recherche du mot « hostname » et ai récupéré les caractères qui suivaient ce mot. La partie la plus complexe a été de devoir modifier ces caractères pour coller à celui de l'élève et permettre de noter. J'ai donc supprimé le nom tel que « yyy » ou « xxx », pour le remplacer par le nom de l'élève noté dans le

formulaire de l'application que l'utilisateur a entré. Après cette manipulation, j'ai pu comparer les hostname du corrigé modifié et les hostname de l'élève. Pour que le point soit validé, on va comparer une exactitude entre les deux.

Pour les deux autres critères, il fallait seulement vérifier qu'ils soient présents dans les différents appareils. S'ils sont présents, le point est accordé sinon, c'est une faute.

Pour la vérification des lines (console, aux, vty) j'ai dû faire une boucle imbriquée pour regarder que les commandes « password xxx » et « login » sont bien présentes dans tous les appareils. Dans cette partie, on ne va pas comparer au corrigé.

Une fois les quatre critères développés, un résultat de l'affichage du Excel et une note juste, j'ai pu commencer à coder les différents critères spécifiques.

3.3.4 Codage des différents critères d'évaluation spécifique

Pour faire la vérification des différents critères spécifiques, j'ai développé plusieurs méthodes qui seront appelées des « méthodes génériques » et qui sont utilisées pour plusieurs vérifications. Comme chaque critère d'évaluation peut avoir plusieurs vérifications différentes et qu'ils sont tous recherchés indépendamment, certaines méthodes, ou bout de code, auraient dû être dupliquées.

Pour pouvoir vérifier tous les différents critères d'évaluation de manières optimisées et éviter la redondance de code, j'ai créé des méthodes génériques réutilisées dans plusieurs vérifications. Il existe dix méthodes génériques.

On va avoir les méthodes qui concernent les routeurs :

- testParametreInterfaceRouteur
- testParametreRouteur
- testParametreRoutage
- testParametreRouterStick

Les méthodes qui concernent les switchs :

- testParametreInterfaceSwitch
- testParametreLineSwitch
- testParametreSwitch
- testParametreSwitchVlan

Et les méthodes utilisées peu importe le type d'appareils :

- testInterfaceCablage
- testCDP
- insertionListAffichageExcel

Les méthodes testParametreInterfaceRouteur et testParametreInterfaceSwitch sont identiques, mais ne se vérifient pas sur le même appareil. Ces méthodes permettent de vérifier toute les commandes qui seront inscrites dans une interface. C'est-à-dire que l'on va dans un premier temps chercher les configurations des interfaces puis vérifier les lignes qui suivront. On va donc pouvoir vérifier les adressages des interfaces, la sécurité des ports actifs ou down, etc.

Les méthodes testParametreRouteur et testParametreSwitch seront aussi des méthodes identiques qui sont spécifiques pour leur type d'appareils. Ces méthodes permettent de vérifier toutes les commandes qui seront directement en configuration terminale (c'est-à-dire à la racine de la configuration). On va donc simplement rechercher les commandes dans notre appareil. Ces méthodes permettent de vérifier les access-list, etc.

La méthode testParametreRoutage, permet de tester les différents routages possibles dans un routeur. On pourra simplement donner en paramètre de méthode le nom du routage à vérifier. Pour le moment le routage ne teste que l'EIGRP et l'OSPF.

La méthode testParametreRouterStick va vérifier toutes les sous-interfaces créées et vérifier leurs configurations.

Pour le switch, la méthode testParametreLineSwitch permet de vérifier les modifications dans les configurations des lines. Par exemple lors du SSH, on doit vérifier que le « transport input all » est passé en « transport input ssh ».

La méthode testParametreSwitchVlan va vérifier la bonne création des vlan et que chaque interface est connectée au bon vlan.

Les méthodes génériques générales testinterfaceCablage et testCDP, permettent de tester les liaisons entre les différents appareils et que les câblages sont respectés. Pour ces méthodes, il n'y a pas de vérification avec le corrigé. On va surtout vérifier que l'élève n'a pas fait d'erreurs dans le montage de son schéma et n'a pas mélangé les interfaces. Une partie des points seront attribués s'il a correctement câblé son

schéma, mais que ce dernier ne correspond pas au corrigé. L'élève aura quand même compris une partie du travail.

La méthode insertionListAffichageExcel permet simplement d'ajouter dans notre liste pour l'affichage dans l'Excel, les informations que l'on va lui donner en paramètres. Il faudra aussi en paramètres, mettre le type d'appareils pour enregistrer dans la bonne liste, et la valeur à enregistrer pour la commande actuelle.

3.3.4.1 Interface de gestion

Dans l'interface de gestion, j'ai fait la vérification des interfaces des switchs et des routeurs.

Pour chaque routeur, l'algorithme va vérifier les interfaces. Il va vérifier que chaque interface contient les informations de base pour fonctionner. On vérifie, grâce au « show run » qu'il y a une adresse ip, que celle-ci soit identique au corrigé ou qu'elle rentre dans les critères d'adressage ip de la HEG. On va aussi vérifier que l'interface est bien en mode no shutdown, donc que la commande « shutdown » n'est pas présente dans l'interface.

D'une manière plus générale, on va vérifier que les interfaces qui sont utilisées pour le corrigé sont aussi utilisées par l'élève, et que les interfaces « shutdown » dans le corrigé le sont aussi dans le travail de l'élève.

Pour la vérification des switchs, on va procéder de la même manière, mais plus précisément. On va vérifier que l'interface de gestion, VLAN 1, correspond au corrigé ou si l'élève a mis une adresse ip correcte au plan d'adressage. On va vérifier toutes les autres interfaces de la même manière que les interfaces du routeur (adressage ip correct et commande « no shutdown »).

Pour chaque switch, il y aura une vérification supplémentaire. On doit vérifier la passerelle par défaut inscrite. Dans un premier temps, l'algorithme vérifie si celle-ci est identique au corrigé. Si c'est le cas, les points sont attribués, sinon on va vérifier si l'adresse ip proposée est enregistrée dans un routeur programmé par l'élève. Si celle-ci correspond avec la programmation de l'élève les points sont attribués.

3.3.4.2 Câblage

La vérification du câblage du schéma va se faire sur les commandes « show ip interfaces brief » et « show cdp neighbors ».

Pour la vérification du « show ip interface brief », on va vérifier que les interfaces utilisées dans le schéma ont un statut « up » et un protocole « up ».

Pour la commande « show cdp neighbors » on va créer une liste avec les différents voisins et les informations de connexion (quelle interface en entrée et en sortie). Par la suite une vérification sera faite sur cette liste, que chaque matériel est connecté par la bonne interface en entrée et en sortie. S'il y a un branchement en plus qui n'est pas répertorié, l'information sera reportée dans la grille d'évaluation comme une faute et sera aussi intégrée dans la synthèse.

3.3.4.3 Routage

Il y a trois différentes possibilités pour le routage. Le routage statique, le routage dynamique et le router-on-a-stick.

Pour le routage statique, on va vérifier que les différents ip route demandés sont bien présents et ne soient pas des erreurs, que la route par défaut soit inscrite, et qu'il y ait la présence de la ligne de commande « redistribute static » dans la déclaration du routage dynamique, si les deux options (dynamique et static) sont utilisées ensemble. Cette spécificité va être validée par l'utilisation d'une méthode générique sur la recherche des lignes de commande générale dans un routeur, et à la comparaison avec le corrigé.

Le routage dynamique va être calculé grâce à une méthode générique qui boucle sur l'ensemble des lignes du routeur pour rechercher la partie routage. On va lancer le processus de recherche pour chaque type de routage, eigrp, ospf, etc et on va le comparer au corrigé.

Le routage on-a-stick est uniquement possible sur les interfaces d'un routeur. On va donc, grâce à la méthode générique testParametreRouterStick rechercher une interface qui ferait office de sous-interface, et la comparer au corrigé.

La vérification du routage est donc en grande partie comparée au corrigé pour voir si l'élève a juste ou faux.

3.3.4.4 Trunk

La vérification du trunk se fait par la présence des cinq lignes de code, « switchport mode trunk », « switchport trunk native », « switchport trunk allowed vlan ... », « switchport nonegotiate », « switchport mode access ».

Toutes les lignes de code du trunk sauf « switchport trunk allowed vlan ... », sont vérifiées avec une méthode générique créée pour évaluer les interfaces du switch. Cette méthode est utilisée dans plusieurs vérifications.

Le contrôle du « switchport trunk allowed vlan ... », va se faire en comparaison du corrigé pour valider quel vlan doit être inscrit dans la commande et vérifier si la commande est 100 % juste ou non.

3.3.4.5 Port Access

La vérification des Access port se fait uniquement sur les switchs. On aura trois critères de validation, le mode access, la vérification des vlan, et la vérification des ports.

Pour chacune de ces vérifications, on va comparer au corrigé. On va donc contrôler que les interfaces doivent être connectées au PC et mises en mode access avec toutes les commandes qui correspondent. Que les différents VLAN qui sont demandés pour le CC soient créés de la bonne façon. Et enfin que les différents ports du routeur soient bien attribués au bon VLAN.

Même si chacun de ces critères est comparé au corrigé et que chaque critère est sur les interfaces, on va faire trois fois le processus de recherche sur chaque interface, pour avoir une note plus détaillée et un résultat plus juste au final.

3.3.4.6 Sécurité des ports

Lors de la vérification de la sécurité des ports, on va vérifier quatre commandes :

- « switchport port-security »
- « switchport port-security violation »
- « switchport port-security maximum »
- « switchport sticky ou statique »

On va donc regarder dans les différentes interfaces des switchs, avec la méthode générique, que les quatre commandes précédentes sont bien présentes. Pour la dernière commande, on va simplement regarder qu'elle est présente, mais pas qu'elle soit identique au corrigé. En effet les adresses mac sont forcément différentes d'un appareil à l'autre.

3.3.4.7 Sécurité des ports non utilisés

Pour cette vérification, on ne va pas utiliser le corrigé. En effet, on va simplement boucler sur l'ensemble des interfaces des switchs. Si elles ne sont pas utilisées, donc pas d'adresse ip déclarée, on considère qu'elles doivent être shutdown. On va donc vérifier que la ligne « shutdown » soit activée si la ligne « adresse ip xxx » n'est pas inscrite, et inversement pour chaque interface.

3.3.4.8 VLAN

Lors de la vérification des VLAN, on va simplement contrôler que les différents VLAN créés dans le corrigé sont bien présents dans les switchs de l'élève. On va donc utiliser une des méthodes génériques déjà développées auparavant.

3.3.4.9 SSH

Dans la vérification du SSH, on va avoir plusieurs lignes de commandes à vérifier. On aura les cinq lignes de commandes pour programmer un SSH :

- username
- domaine name
- version
- login local
- transport input ssh

Pour les trois premières commandes, on va effectuer un contrôle sur l'ensemble des switchs, sur l'ensemble des lignes, car ce sont des commandes en « conf t »(donc le premier niveau). Par contre pour les deux dernières vérifications, on va aller vérifier directement dans les lignes du switch. On va donc utiliser deux méthodes génériques différentes. Une qui boucle sur l'ensemble des lignes et l'autre qui analyse uniquement les lignes.

Pour vérifier le ssh on va aussi devoir vérifier la création des crypto key. Pour ça, on va utiliser la méthode générique qui boucle sur l'ensemble des lignes du switch et vérifier qu'elles sont présentes.

3.3.4.10 Access-list

Pour les Access-list, on va faire la vérification qu'elles sont bien présentes. On va aussi vérifier si elles sont correctes, c'est-à-dire qu'elles soient, soit identiques au corrigé.

3.3.4.11 Utilisateur (PC)

Il a aussi fallu faire la vérification de la configuration des PCs. Cette configuration est très complexe, car chaque ordinateur est différent et que la plupart des adresses ip sont attribuées en DHCP. J'ai donc convenu avec mon professeur de mémoire de faire une vérification sur l'adressage ip, si le test est fait et si le test est réussi.

Pour la vérification de l'adressage ip j'ai réutilisé la classe de testIp pour vérifier que l'adresse enregistrée fait bien partie du plan d'adressage de la HEG et que l'adresse correspond à un utilisateur. Dans un deuxième temps, j'ai vérifié que le test du pinging a bien été fait. On va simplement vérifier que la ligne de commande pour lancer un ping est présente dans le fichier .txt. Et enfin, la dernière étape est de vérifier si le ping est fonctionnel, donc si la configuration du pc est correcte sur le schéma proposé. Suite à cela, on va vérifier que le ping retourne bien un « Packets : sent = 4, Received = 4, lost = 0 », c'est-à-dire que le transfert est correct et que l'information soit passée.

4. Difficultés rencontrées

Lors de ce projet, j'ai rencontré de multiples problèmes. J'ai dû trouver une solution pour y remédier et réussir à faire ce que j'avais prévu de réaliser.

4.1 Debugging

Le premier problème rencontré était le debugging. En effet, en utilisant Visual Studio, il n'est possible de lancer l'application que dans son entièreté. L'application lancée de cette façon ne fait donc pas apparaître les lignes d'aide mises dans le code, les « Console.log ». J'ai dû donc trouver une solution pour ce problème important. Sans le debugging il m'était impossible d'avancer dans l'écriture de l'algorithme de façon rapide. Chaque recherche d'erreur se fait à tâtons. J'ai d'abord essayé d'utiliser le système de break point et de debugging de Visual Studio. Le problème qui s'est ensuite posé, est que Visual Studio est un logiciel très complexe avec beaucoup de fonctionnalités différentes. Si on ne connaît que peu l'interface fonctionnelle de Visual Studio et que l'on lance le debugging, il est assez complexe de réussir à trouver ce que l'on veut et à ne pas perdre de temps supplémentaire. Après avoir essayé cette solution qui ne me convenait pas, j'ai décidé de trouver un autre logiciel pour pouvoir travailler. Après mûre réflexion et plusieurs recherches, j'ai trouvé le logiciel « Rider ». C'est un logiciel qui utilise le langage C# et qui est développé par JetBrains. Ayant déjà utilisé un logiciel développé par JetBrains, et connaissant leur méthode d'interface, c'était plus simple pour moi de partir sur cette solution. J'ai donc utilisé Rider lors de mon développement de code pour le calcul de l'algorithme. Toute la partie design de l'interface et création de l'application en soi, a été faite à partir de Visual Studio. Tout le reste du travail a été fait sur Rider.

4.2 Compatibilité des systèmes d'exploitation

Le second problème qui s'est posé est la compatibilité des systèmes d'exploitation. En effet, le projet a dû être développé en Windows Forms pour être compatible sur les ordinateurs fixes et portables des professeurs de réseau. L'ensemble du système informatique de la HEG étant en Windows, il n'y avait donc pas l'opportunité de développer en IOS. Le problème rencontré concerne l'infrastructure d'ordinateurs personnels. En effet, le pc portable que j'utilise le plus est un MacBook, sur IOS donc.

Visual Studio code existe, mais il ne donne pas l'opportunité de travailler sur des Windows Forms. Après plusieurs essais à lancer une machine virtuelle Windows et à lancer l'application Visual Studio, j'ai renoncé à utiliser mon pc portable. En effet, le pc n'arrivait pas à ouvrir Visual Studio correctement avec la faible capacité de ma virtual machine. J'ai travaillé l'ensemble du code sur un ordinateur fixe. Il m'était donc impossible de pouvoir transporter le projet et de pouvoir le montrer ou demander de l'aide à des personnes extérieures. C'est pourquoi toutes les réunions faites avec mon professeur encadrant, ont été faites par Teams pour permettre la visualisation du projet.

4.3 Mise en forme du fichier Excel

Un autre problème rencontré a été la mise en forme du fichier Excel. Avec mes recherches faites sur Office.Interop, j'avais remarqué qu'il ne me faudrait que cette librairie pour faire l'ensemble de la création, du remplissage, de l'automatisation et de la mise en page. Pourtant, quand j'ai commencé à travailler sur la création de mon fichier, ça n'a pas été aussi simple. La création du fichier, de mes différentes feuilles Excel, et de la sauvegarde a très bien fonctionné. Mais je me suis rendue compte que je ne pouvais pas faire de mise en page design. C'est-à-dire avoir la possibilité d'enlever les traits des tableaux, mettre les cellules en plus grand, etc. Toutes les solutions trouvées sur internet ne marchaient pas et j'ai dû trouver une solution. J'ai donc trouvé la librairie « E-iceblue » qui permet justement de pouvoir accéder à toutes les fonctionnalités des appareils Office. J'allais donc pouvoir tout modifier dans mon Excel. Après avoir téléchargé le fichier d'installation, inséré la référence de librairie dans mon fichier et testé les premières lignes de code, je me suis rendue compte que ça n'allait pas être facile. J'ai essayé plusieurs manières de coder avec cette librairie sans succès. J'ai même essayé de créer un nouveau projet vide pour tester simplement cette librairie, mais rien n'a fonctionné. J'ai donc décidé, après plusieurs heures de vaines tentatives, de laisser tomber cette librairie.

Après plusieurs recherches, j'ai réussi à résoudre certains de mes problèmes comme l'insertion des formules directement calculées dans Excel, directement avec la librairie utilisée de base, Office.Interop. J'ai donc décidé de laisser mon fichier Excel en l'état et d'abandonner le côté design du fichier.

4.4 Différences des interfaces entre le corrigé et le fichier élève

Un problème actuellement non résolu, mais qui est en recherche de solutions, est le changement des interfaces entre les élèves et le corrigé. En effet dans ma vérification des interfaces, on va comparer les interfaces au corrigé et vérifier leur adresse ip. Il faut pouvoir vérifier que c'est la bonne adresse ip pour vérifier l'ensemble du fonctionnement du schéma. Le problème que j'ai remarqué est que sur certains énoncés de contrôle continu, chaque élève va pouvoir choisir ces différentes interfaces pour les liaisons entre les appareils. Cela signifie que les interfaces du corrigé ne seront pas les mêmes pour chaque élève. L'algorithme mettra donc des fautes là où peut-être les interfaces seront justes. La première solution que j'ai essayé de faire a été de faire une vérification générale des interfaces sans regarder leur nom, mais je me suis rendue compte que je ne pourrais pas faire de vérification par rapport au schéma. Je pouvais simplement vérifier la configuration, l'adresse ip, mais pas si cette interface est juste par rapport à l'énoncé. Je n'ai pas trouvé de solutions en code pour gérer ce problème. La solution actuelle décidée avec mon professeur est de mettre les interfaces sur les schémas des énoncés pour forcer tout le monde (élèves et corrigé) à avoir la même configuration avec le même nom des interfaces.

Ce problème est donc non résolu pour le moment, mais entre dans la liste des problèmes à résoudre à la suite de mon travail de bachelor.

4.5 Adressage ip

Le dernier problème rencontré a été la compatibilité des adressages ip. En effet, le corrigé va forcément proposer des adresses ip pour l'ensemble des interfaces utilisées, pour les ACL, et pour plusieurs autres configurations. Le problème est qu'il n'existe pas forcément une unique adresse ip possible. Sur des réseaux de grande taille, il existe une multitude d'adresses ip disponibles pour chaque type d'appareils. La vérification directe avec le corrigé n'est donc pas totalement juste. Si l'élève a la même adresse que le corrigé, les points seront donc directement attribués, mais si ce n'est pas la même adresse, cela ne veut pas dire qu'il a pour autant faux. J'ai donc dû créer une classe qui permet la recherche des adresses ip en fonction du plan d'adressage de la HEG.

Dans la classe de recherche ip il me fallait donc créer l'ensemble du plan d'adressage ip en mode algorithme. Chaque taille est représentée avec les adresses ip qui lui

correspond, séparée par le type d'appareils. De plus, il a fallu faire une méthode pour la recherche de réseaux pour chaque type d'appareils. En effet, nous allons avoir les informations de l'adresse ip de l'élève et du masque, mais pour savoir si cette adresse correspond bien au plan d'adressage, il nous faut une adresse réseau. Cette adresse réseau est recherchée sur la base des informations de l'adresse ip et du masque du corrigé.

Pour chaque vérification de l'ip on va entrer le réseau, l'adresse ip donnée par l'élève, le masque inscrit par l'élève qui correspond et le type d'appareils vérifié. Le type d'appareils a été enregistré en énumération pour faire une double vérification. Avec toutes ces informations, l'algorithme va faire une recherche dans l'ensemble du plan d'adressage ip pour vérifier que l'adresse proposée par l'élève, qui est différent du corrigé, peut quand même être correct.

5. Présentation de l'installation et du mode d'emploi de ResCorTxt

5.1 Installation et prérequis

Il y a plusieurs prérequis pour l'installation et le fonctionnement de l'application.

Dans un premier temps, il faut obligatoirement avoir l'application Office Excel installée sur l'ordinateur utilisé pour le logiciel. En effet, la sortie fichier de l'application est un Excel, et son enregistrement en .xlsx ne pourra pas fonctionner sans Excel. Si Excel n'est pas présent, l'application pourra tourner pour le calcul de la note et le téléchargement de la grille d'évaluation ne sera pas disponible.

Le deuxième prérequis pour le fonctionnement de l'application est net 6. En effet, l'application est en Windows Form qui utilise net 6. Comme j'ai décidé de créer le fichier d'installation, le téléchargement et l'installation de net 6 se fait en même temps que l'installation du logiciel. Mais il arrive que des problèmes surviennent pour cette installation. En effet, certaines configurations sont en (x64) et d'autre en (x86). Ces configurations peuvent ne pas être compatibles. Avec l'installeur créé pour cette application, c'est net 6 (x86) qui est directement téléchargé. En l'absence de connaissance de la configuration ou de sa compatibilité, il faut télécharger le logiciel et lancer l'installation. Si à l'ouverture de l'application cela ne fonctionne pas et que vous avez une erreur il vous faudra aller directement télécharger net 6 sur la doc Microsoft[1].

5.2 Mode d'emploi

5.2.1 Application

L'application est très simple d'utilisation.

L'utilisateur devra entrer les informations (nom et prénom) de l'élève évalué, télécharger le fichier de l'élève et télécharger le corrigé. Il devra ensuite cocher les différents critères d'évaluation.

1 https://docs.microsoft.com/fr-fr/dotnet/core/install/windows?tabs=net60

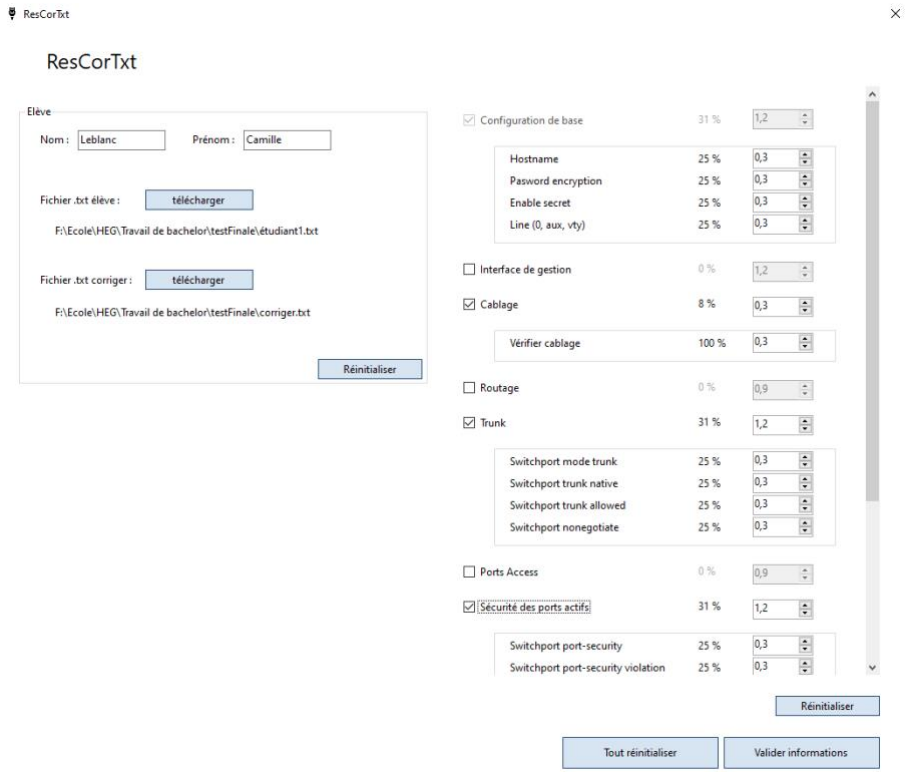
Pour avoir une meilleure vision des coefficients, chaque critère d'évaluation possède son propre pourcentage en fonction de son propre coefficient par rapport au coefficient général. Et dans chaque critère d'évaluation, pour chaque sous-critère, il y a un pourcentage pour ce sous-critère par rapport au critère associé. Cette vision en pourcentage est donc plus claire et plus simple pour imaginer la note finale.

Après avoir sélectionné toutes les données relatives à l'élève et aux critères, l'utilisateur n'a plus qu'à cliquer sur le bouton « calculer ». Attention, s'il manque des informations sur l'élève le calcul ne sera pas initié et un message d'erreur sera affiché. Il suffit donc de remplir ce qu'il manque et de relancer le calcul.

Le bouton « annuler » en bas à droite permet de remettre l'application dans son état initial. Les informations de l'élève seront supprimées et les critères seront remis en mode initial.

Le bouton « annuler » dans la partie de l'élève permet uniquement de remettre en mode initial les informations de l'élève. Les critères d'évaluation ne seront pas touchés.

Figure 2 : Interface ResCorTxt après remplissage des informations

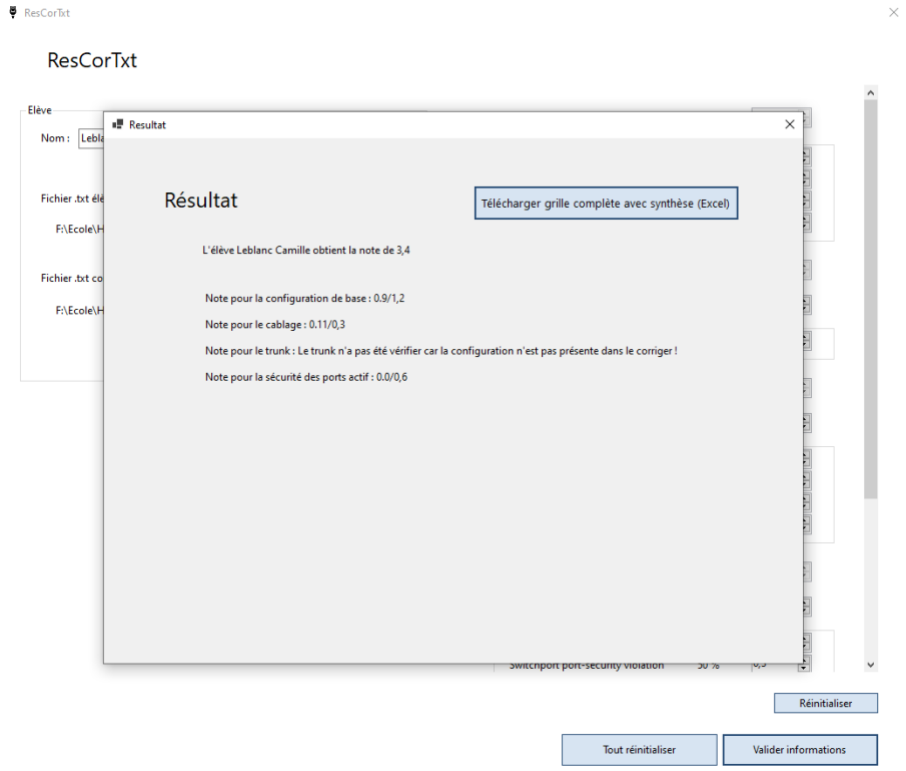


Après lancement du calcul, l'algorithme va tourner en arrière-plan. Dès que l'algorithme a fini son analyse, une nouvelle fenêtre est affichée, la page de résultat. Sur cette page, on pourra voir la note finale attribuée à l'élève, ainsi qu'un détail des notes par critères d'évaluation. En haut à droite, il y a une possibilité de téléchargement d'Excel de retour pour l'élève. Cette Excel contient l'intégralité des lignes de code vérifié par l'algorithme ainsi que le calcul pour la note.

Cette fenêtre est un singleton, c'est-à-dire qu'elle ne peut être ouverte qu'une seule fois et qu'on ne peut rien faire d'autre sur l'application avant de l'avoir fermé. En effet, c'est obligatoire sur cette application, car l'algorithme ne stocke pas les données des notes autre part que sur l'Excel pour chaque session. On ne peut donc pas ouvrir plusieurs sessions en même temps.

Pour continuer, et passer à la notation d'un prochain élève, il suffit de fermer la page « Résultat », après avoir téléchargé l'Excel. La page principale de l'application ne se réinitialise pas à chaque utilisation. Ce qui veut dire que si des coefficients ont été modifiés pour cet examen en particulier, il n'y aura pas besoin de remodifier ces mêmes coefficients. Tant que l'application n'est pas fermée ou que vous ne cliquez pas sur « annuler », les coefficients ne changeront pas.

Figure 3 : Page de résultat



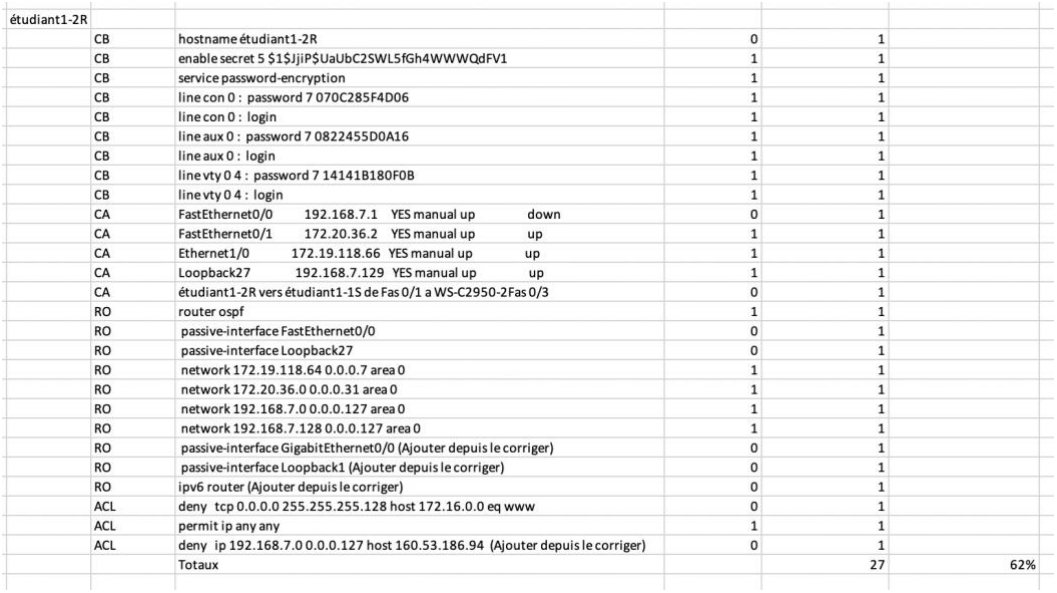
5.2.2 Fichier Excel

L'application va créer un Excel de sortie à télécharger. Cet Excel est très important, car c'est la seule sauvegarde de la note de l'élève après avoir fermé la page de résultat.

Cet Excel va contenir deux feuilles :

La première est la grille d'évaluation ; on va retrouver une grille d'évaluation classée par type d'appareils, puis par appareils. Les différentes lignes qui ont été notées y seront indiquées. On pourra voir les points attribués pour chaque ligne. À la fin de chaque appareil, on pourra voir le total ainsi qu'un pourcentage de réussite.

Figure 4 : Exemple fichier Excel (routeur 2 pour l'étudiant 1)

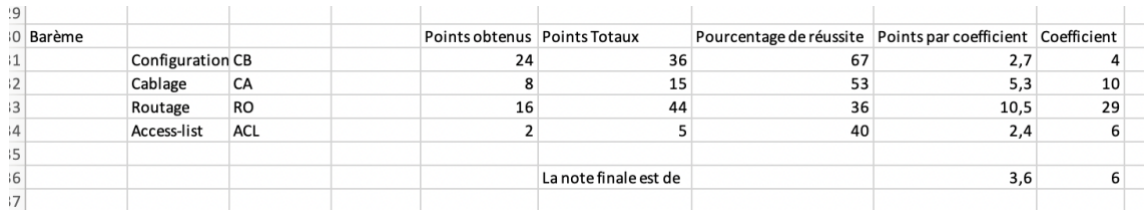


étudiant1-2R					
	CB	hostname étudiant1-2R	0	1	
	CB	enable secret 5 1JjiP$UaUbC2SWL5fGh4WWWQdFV1	1	1	
	CB	service password-encryption	1	1	
	CB	line con 0 : password 7 070C285F4D06	1	1	
	CB	line con 0 : login	1	1	
	CB	line aux 0 : password 7 0822455D0A16	1	1	
	CB	line aux 0 : login	1	1	
	CB	line vty 0 4 : password 7 14141B180F0B	1	1	
	CB	line vty 0 4 : login	1	1	
	CA	FastEthernet0/0 192.168.7.1 YES manual up down	0	1	
	CA	FastEthernet0/1 172.20.36.2 YES manual up up	1	1	
	CA	Ethernet1/0 172.19.118.66 YES manual up up	1	1	
	CA	Loopback27 192.168.7.129 YES manual up up	1	1	
	CA	étudiant1-2R vers étudiant1-1S de Fas 0/1 a WS-C2950-2Fas 0/3	0	1	
	RO	router ospf	1	1	
	RO	passive-interface FastEthernet0/0	0	1	
	RO	passive-interface Loopback27	0	1	
	RO	network 172.19.118.64 0.0.0.7 area 0	1	1	
	RO	network 172.20.36.0 0.0.0.31 area 0	1	1	
	RO	network 192.168.7.0 0.0.0.127 area 0	1	1	
	RO	network 192.168.7.128 0.0.0.127 area 0	1	1	
	RO	passive-interface GigabitEthernet0/0 (Ajouter depuis le corriger)	0	1	
	RO	passive-interface Loopback1 (Ajouter depuis le corriger)	0	1	
	RO	ipv6 router (Ajouter depuis le corriger)	0	1	
	ACL	deny tcp 0.0.0.0 255.255.255.128 host 172.16.0.0 eq www	0	1	
	ACL	permit ip any any	1	1	
	ACL	deny ip 192.168.7.0 0.0.0.127 host 160.53.186.94 (Ajouter depuis le corriger)	0	1	
		Totaux		27	62%

À la fin de la grille d'évaluation, le récapitulatif des différentes notes obtenues pour chaque critère d'évaluation et enfin la note finale obtenue apparaîtront

La deuxième feuille contient une synthèse des différentes fautes que l'élève a commises. On aura donc pour chaque appareil l'ensemble des lignes de code que l'élève n'a pas bien fait. Cette feuille est principalement pour l'élève, pour qu'il puisse mieux comprendre ses erreurs et qu'il ne les reproduise pas.

Figure 5 : Exemple barème fichier Excel



Barème			Points obtenus	Points Totaux	Pourcentage de réussite	Points par coefficient	Coefficient
	Configuration	CB	24	36	67	2,7	4
	Cablage	CA	8	15	53	5,3	10
	Routage	RO	16	44	36	10,5	29
	Access-list	ACL	2	5	40	2,4	6
				La note finale est de		3,6	6

6. Conclusion

Pour conclure ce projet, je souhaite préciser que j'ai particulièrement apprécié travailler sur un travail de bachelor pratique. Cela m'a permis d'avoir un projet individuel plutôt conséquent et de planifier sa bonne réalisation. Cela m'a permis une autonomie dans mon travail, une rigueur importante pour arriver avec une application et un rapport finalisés à la date prévue.

Ce travail pratique m'a permis de mettre en pratique les trois dernières années réalisées à la HEG. J'ai pu récupérer les différentes pratiques et informations accumulée au cours des différents cours de réseau, mais aussi de programmation. J'ai notamment pu approfondir mes connaissances en réseau, car j'ai dû réfléchir d'une manière bien plus large sur le travail des corrections d'examens. La vision de l'enseignant qui corrige, l'élève qui fait son contrôle continu correctement, mais aussi l'élève qui fait un peu n'importe comment, les spécificités du projet avant même de coder ont été les lignes directrices de ma réflexion dans ce projet

Il est déjà convenu que ce projet, très complexe dans l'analyse de toutes les différentes situations possibles, n'est pas complétement opérationnel à ce stade. C'est pourquoi il va se poursuivre de manière personnelle pour l'améliorer et le compléter afin de pouvoir le mettre à disposition des professeurs de réseau dès son fonctionnement optimal.

Bibliographie

Activation de ssh sur un switch Cisco. [en ligne]. Disponible à l'adresse : https://www.clemanet.com/activation-ssh.php

ADEGEO. Installer .NET sur Windows - .NET. [en ligne]. Disponible à l'adresse : https://docs.microsoft.com/fr-fr/dotnet/core/install/windows

BILLWAGNER. Guide pratique pour accéder à Office objets d'interopérabilité - Guide de programmation C#. [en ligne]. Disponible à l'adresse : https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/interop/how-to-access-office-onterop-objects

DOTNET-BOT. Microsoft.Office.Interop.Excel Espace de noms. [en ligne]. Disponible à l'adresse : https://docs.microsoft.com/fr-fr/dotnet/api/microsoft.office.interop.excel

HEG GENÈVE, 2019. *Aide-Memoire*. Version -2020 2019.

Rider : L'IDE .NET multiplateforme de JetBrain. *JetBrains* [en ligne]. Disponible à l'adresse : https://www.jetbrains.com/fr-fr/rider/

Rj45 free icons designed by Freepik. *Flaticon* [en ligne]. Disponible à l'adresse : https://www.flaticon.com/free-icon/rj45_5232750

Spire.XLS for .NET. [en ligne]. Disponible à l'adresse : https://www.e-iceblue.com/Introduce/excel-for-net-introduce.html?gclid=Cj0KCQjwjIKYBhC6ARIsAGEds-IvyLNA-J4saxFQINhlVokC6slp5BuDIXiqwM2H_7jY7aO3-QWuBgcaAvUaEALw_wcB

Workbooks.Open, Microsoft.Office.Interop.Excel C# (CSharp) Exemples de code - HotExamples. [en ligne]. Disponible à l'adresse : https://csharp.hotexamples.com/fr/examples/Microsoft.Office.Interop.Excel/Workbooks/Open/php-workbooks-open-method-examples.html