Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Automatisation des tâches, développement d'outils de gestion de serveurs

Lorenzo Albanese

Office Center

Responsable entreprise : Jérémy Raspail

Responsable académique : Roland Depeyre

2019

Sommaire

I. Introduction

Afin de compléter ma formation de **DUT**, Diplôme Universitaire de Technologie spécialisé en **R&T**, Réseaux et Télécommunications, j'ai eu l'occasion d'effectuer un stage de 10 semaines en entreprise dans le but de mettre en pratique les compétences acquises ces deux dernières années. J'ai notamment pu vivre ma première vraie expérience en entreprise dans le domaine de l'informatique et des réseaux.

J'ai ainsi réalisé mon stage du 8 avril au 14 juin 2019 chez la société Office Center située à Le Chaffaut-Saint-Jurson, dans le 04. Au sein de cette société de services et de vente de matériel informatique, j'ai intégré le service technique de l'entreprise. J'avais pour but principal de développer des outils permettant la mise en place, la gestion et le dépannage de serveurs Windows Exchange[1] et ordinateurs.

Mon travail peut donc être décrit par deux grandes catégories qui sont le dépannage d'ordinateurs et la gestion de serveurs Exchange. La partie dépannage d'ordinateurs a nécessité l'acquisition de nouvelles connaissances en informatique. La partie gestion de serveurs a, quant à elle, sollicité les compétences PowerShell que j'ai développé pendant les cours du DUT.

Pour présenter mon stage, nous verrons premièrement une description de l'entreprise plus détaillée, puis le cadre général du sujet. Enfin, je décrirais le travail réalisé.

II. Présentation d'Office Center

1. Description

Office Center est une SARL créée en 1973 qui proposait initialement de la fourniture de machines à écrire. Dans un secteur à faible attraction économique, l'entreprise a dû se diversifier vers l'offre d'une large gamme de services afin de prospérer. Elle compte aujourd'hui 7 salariés dont les co-gérants, Jérémy et Sylvain Raspail.

L'entreprise se situe à 15 minutes de Digne-les-Bains, à Le Chaffaut-Saint-Jurson, au centre du département des Alpes-de-Haute-Provence (04) (voir Figure 1).

Figure 1 - Localisation de l'entreprise

2. Activités

Comme dit précédemment, la société propose une large gamme de services, en plus de la vente, pour l'informatique, les copieurs, caisses de paiement, et caméras.

On peut citer comme exemple l'infogérance[2] et l'audit[3] de parcs informatique, la maintenance d'installations, le développement de logiciels personnalisés, l'assistance technique et un centre de formation agrée pour la prise en main des logiciels Microsoft Office (niveau débutant ou confirmé).

Est aussi proposé l'installation de copieurs, de solution **GED**, Gestion Électronique des Documents[4], le paramétrage de caisse et la mise en place de caméras IP, de systèmes d'alarmes.

Les clients peuvent être des particuliers ou des professionnels, il peut s'agir d'entreprises dans le Bâtiment, de cabinets notariaux, juridiques, administrations, associations…

Office Center a donc clientèle très variée, couvrant la majorité du département des Alpes-de-Haute-Provence (04), mais aussi dans les autres départements de la région PACA, regroupant le 05, 06, 13, 83 et 84 (voir Figure 2).

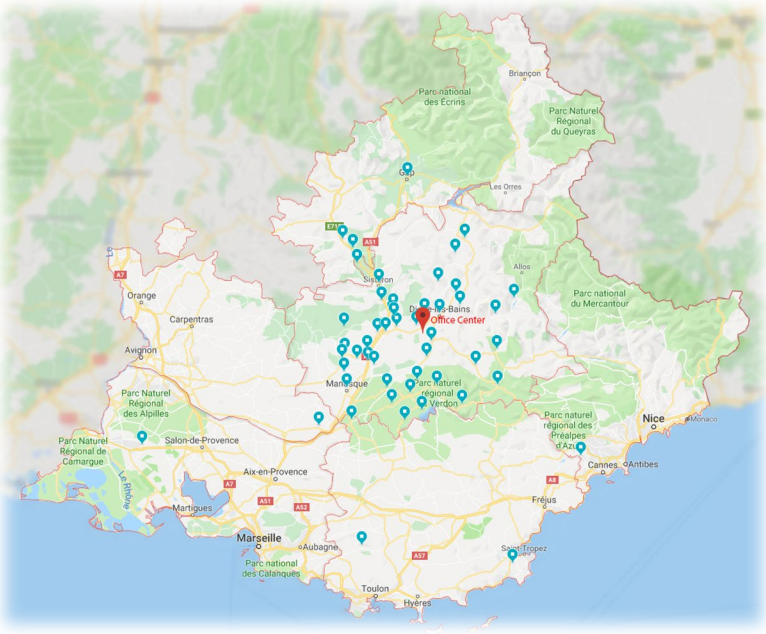

Figure 2 - Principales villes des interventions

3. Organisation

L'entreprise étant petite, l'équipe se limite à un service technique, un commercial et la secrétaire qui est aussi comptable. Comme mentionné précédemment, Office Center est co-gérée par Jérémy et Sylvain Raspail (voir Figure 3).

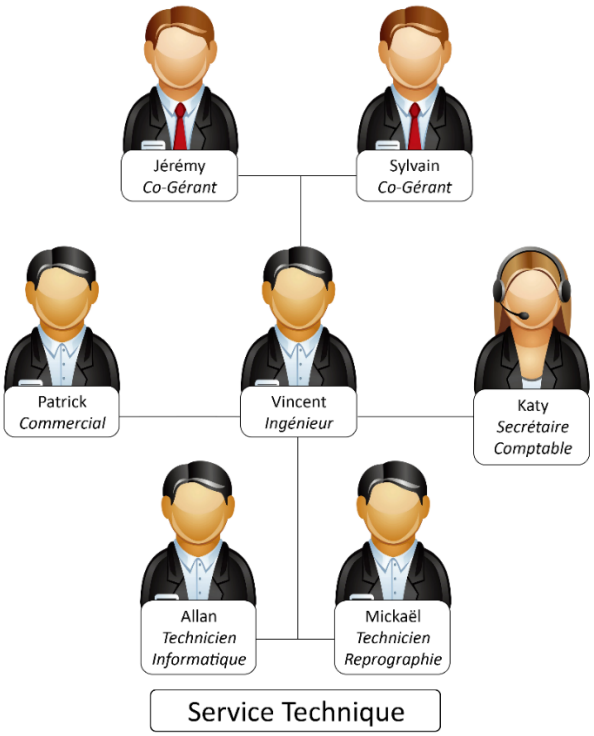

Figure 3 - Organigramme d'Office Center

J’avais à ma disposition un ordinateur, une adresse mail professionnelle et un téléphone. J’ai travaillé dans l’atelier (voir Figure 4), au sein du service technique composé de Vincent, Allan et Mickael, qui partait régulièrement en intervention sur site. Jérémy participe également lorsqu’il n’anime pas de formations. Le but du service technique est de prendre en charge les appels de clients enregistrés par la secrétaire. Ils peuvent ensuite soit prendre la main sur l’équipement à distance via TeamViewer[5], soit se déplacer si la prise en charge à distance n’est pas possible.

L’équipe travaille dans une ambiance conviviale et collaborative, si bien que l’on ne ressent pas de liens hiérarchiques stricts.

Figure 4 - Photographie de l'atelier

III. Cadre technique général du sujet

Comme précisé en introduction, j'ai travaillé sur 2 domaines qui sont le dépannage d'ordinateurs et la gestion de serveurs Windows Exchange.

Pour la première catégorie, il s'agissait de mettre à jour une clé USB **HBCD**, Hiren's Boot CD et de trouver un moyen de chiffrer une clé USB.
À propos du serveur Exchange, j'avais pour missions de trouver un moyen de vérifier la taille disponible sur les disques durs, et de gérer des scripts PowerShell permettant de créer des comptes Exchange.

1. Dépannage d'ordinateurs

Hiren's Boot CD est un logiciel gratuit qui permet de démarrer un ordinateur à partir d'une clé USB pour accéder à divers outils (logiciels) de diagnostics informatiques.

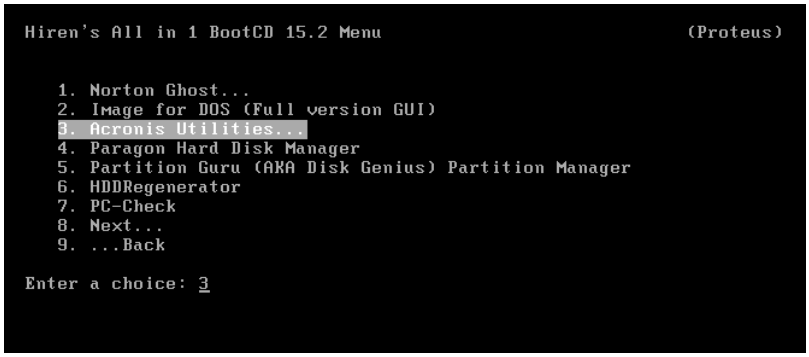

Figure 5 - Menu HBCD au démarrage sur clé USB

Il est par exemple possible d'analyser un disque dur pour savoir s'il est défectueux ou le cloner à l'identique (données et partitions créées). L'interface lance les logiciels rapidement par des lignes de commandes (voir Figure 5).

On peut également lancer un mini Windows XP ou mini Windows 7 pour utiliser des logiciels ou accéder à des données se trouvant sur un disque ou une clé USB (voir Figure 6 et 7).

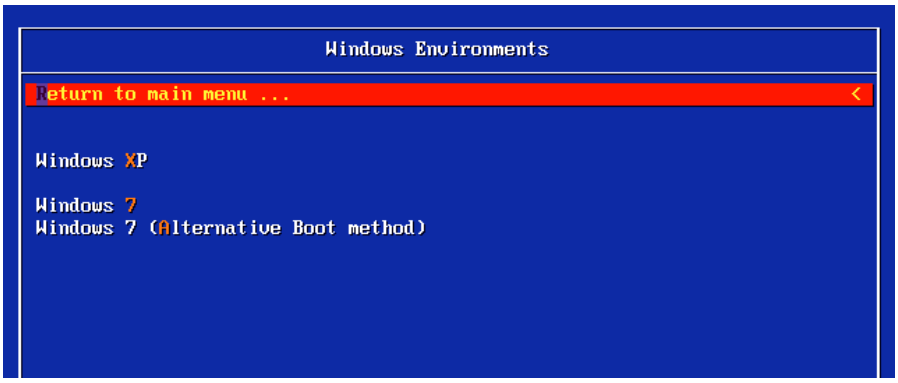

Figure 6 - Menu de sélection des mini Windows

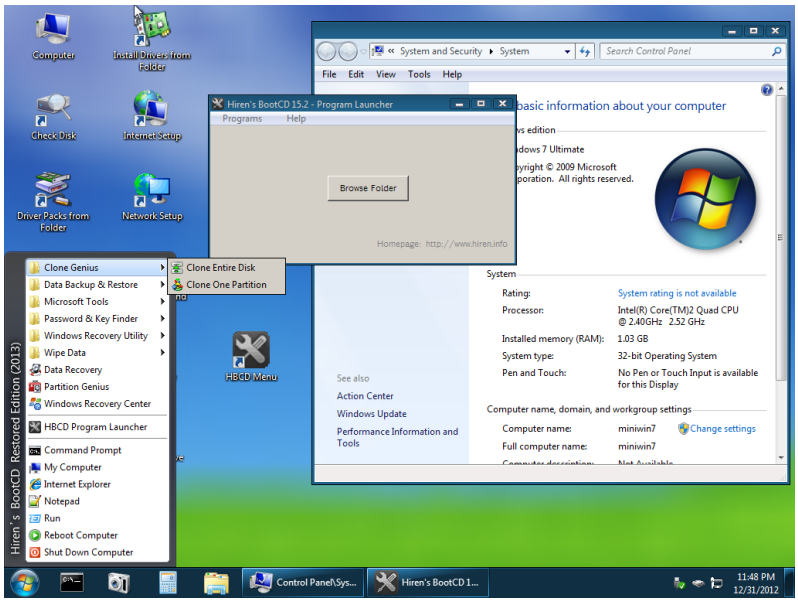

Figure 7 - Mini Windows 7 avec menu HBCD

L'intérêt d'avoir ce logiciel sur clé USB est qu'il est transportable. On peut l'utiliser aussi bien sur un ordinateur test en atelier, que directement chez le client en intervention. Office Center utilisait déjà une version de l'HBCD datant de 2012 (version 15.2 Restored edition).

Ma première mission était de trouver/concevoir une mise à jour ou une version améliorée qui comprenait une version plus récente des logiciels de diagnostics et un mini Windows 10 afin d'accéder aux nouvelles applications non compatibles avec les versions 7 et XP.

Ma deuxième mission était de trouver un moyen de chiffrer une clé USB (ou une partition) pour limiter l'accès aux données par un mot de passe. En cas d'oubli de la clé USB chez le client, il faut éviter qu'il accède aux données, notamment les feuilles de suivis des autres clients qui contiennent des identifiants de connexions, adresses, coordonnées et autres informations sensibles.

2. Gestion de serveurs Exchange

Le premier point concerne la surveillance d'espace disque restant. Le but est de surveiller et alerter par mail en cas d'espace disque faible. Cette solution sera déployée sur les serveurs de leurs clients pour les avertir de manière automatique.

Par la suite Vincent a mis au point des scripts PowerShell qui créent des comptes Exchange complets. Cela comprend la création d'adresses mail, d'utilisateurs dans l'Active Directory[6], la possibilité d'ajouter des groupes de diffusion mail, des salles et équipements pour la programmation de réunions, ou encore l'ajout de contacts externes pour la redirection de mails professionnels.

Le but de ces scripts est qu'Office Center installe Microsoft Exchange sur un de ses serveurs pour ensuite vendre la solution à ses clients. Au lieu d'acheter un serveur dédié avec des comptes Exchange (qui se chiffre très vite dans les milliers d'euros) le client achète le service à moindre coût à la société.

Le problème rencontré par l'entreprise lors des tests de mise place est que les clients potentiels peuvent accéder aux ressources des autres clients (boites mail, équipements, groupes). En utilisant les interfaces de configuration Exchange proposées par Microsoft, il est impossible de réaliser ce projet.

L’utilisation de scripts PowerShell permet de compartimenter chaque client, en exécutant des commandes qui ne sont pas accessibles via les interfaces, de sorte qu’ils ne puissent pas se voir entre eux.

La mission qui m’a été confiée était de développer une interface permettant de lancer automatiquement les scripts que Vincent a fait, en remplissant seulement des informations comme le nom du client, nom de l’utilisateur, nom d’équipement…

À ce moment du stage, j’avais déjà réalisé la première partie de cette catégorie concernant la surveillance de disque en utilisant un script PowerShell. Comme mon tuteur de stage était satisfait de ce que j’avais produit, il m’a demandé de gérer cette partie de la même manière. J’ai donc dû mobiliser mes connaissances en PowerShell et en web pour cette mission puisque j’ai également dû mettre en forme une interface web.

3. Étude de configuration de switchs

La dernière semaine, j’ai étudié le réseau d’un client qui a de gros problèmes e ralentissements. Depuis quelques semaines l’origine du problème n’a pas été identifiée par les techniciens, j’ai donc étudié le réseau pour tenter de trouver des pistes et relever des informations pour les futures interventions de l’entreprise.

IV. Travail réalisé

1. Dépannage d'ordinateurs

a. Clé USB HBCD

Dans un premier temps, j'ai recherché une nouvelle version de HBCD. J'ai pu trouver que le projet avait été relancé en 2018 après 6 ans sans mises à jour. Cette nouvelle version (v1.0.1) proposait un mini Windows 10, comme demandé, mais n'intégrait pas certain logiciels professionnels de la version précédente. Il aurait été possible de les rajouter manuellement mais cette version a comme autre défaut de démarrer directement sur le mini Windows. Cela signifie une perte de temps puisqu'au démarrage sur la clé il faut attendre le lancement de Windows alors qu'il était jusque-là possible de lancer un programme directement depuis le menu HBCD.

Après discussion avec Vincent, j'ai décidé de créer mon propre Windows 10 **PE SE**, Preinstallation Environnement Special Edition[7] pour ajouter une entrée au menu HBCD et avoir la possibilité de démarrer un mini Windows 10. J'ai téléchargé et utilisé l'outil Win10PE SE (voir Figure 8) qui permet de créer et personnaliser une version allégée de Windows 10 à partir d'un fichier d'image disque (.iso). On peut choisir les composants Windows que l'on souhaite intégrer et le fichier ISO est récupérable via un outil spécialisé dans le téléchargement de fichiers ISO pour Windows et Office.

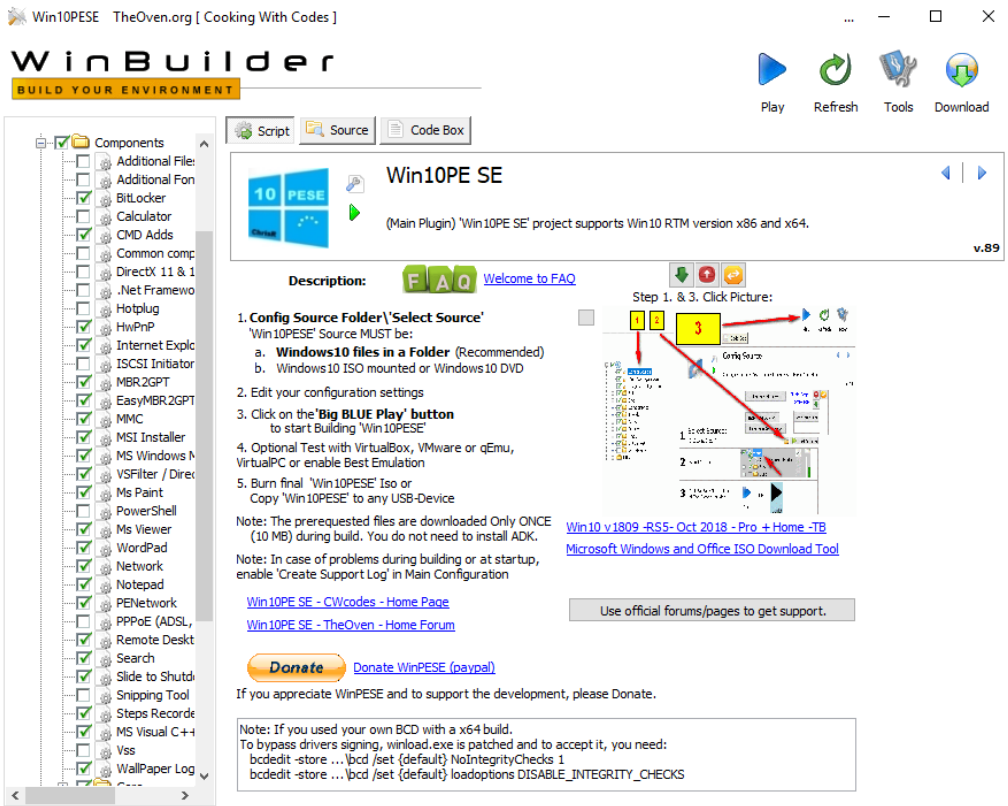

Figure 8 - Interface WinBuilder de Win10PE SE

L'inconvénient de cette solution est que le mini Windows 10 créé ne comprend pas l'intégration du menu HBCD avec tous les logiciels.

Au cours de recherches j'ai pu trouver un autre utilitaire appelé Medicat, et créé par des fans de HBCD pour remplacer celui-ci qui n'était plus mis à jour. On y retrouve un menu permettant à la fois de lancer des programmes et un mini Windows 10 (voir Figure 9).

Figure 9 - Menu principal de Medicat

La configuration de Medicat (v18.10) correspondait parfaitement aux besoins puisqu'il y a un menu au démarrage et la possibilité de lancer un mini Windows. Une fois lancé le mini Windows 10 permet d'accéder à divers programmes grâce à PortableApps (voir Figure 10) qui permet d'embarquer des versions allégées de programmes sur des clés USB.

Figure 10 - Menu PortableApps

Cette solution paraissait être la plus adaptée aux besoins et la plus simple à mettre en place, je l'ai donc légèrement modifié afin d'y rajouter quelques logiciels dans PortablesApps et le mini Windows 7 qui était sur HBCD. Les techniciens avaient besoin de garder la possibilité de démarrer sur Windows 7 car certains adaptateurs qu'ils utilisent de sont plus compatibles sous Windows 10 ou fonctionnent mal.

Pour l'ajout de logiciels cela se fait très simplement en ajoutant les programmes dans le dossier PortableApps, puis en modifiant le menu qui est en format .txt pour les faire apparaître.

Pour l'ajout du mini Windows 7, il a fallu modifier le bootloader[8] de Windows à l'aide de la commande bcdedit qui s'exécute dans l'invite de commandes Windows.

```
c:\>bcdedit

Gestionnaire de démarrage Windows
---------------------------------
identificateur          {bootmgr}
description             Windows Boot Manager
locale                  fr-FR
inherit                 {globalsettings}
default                 {default}
displayorder            {default}
toolsdisplayorder       {memdiag}
timeout                 1

Chargeur de démarrage Windows
-----------------------------
identificateur          {default}
device                  ramdisk=[boot]\HBCD\10\sources\boot.wim
path                    \windows\system32\boot\winload.exe
description             Win10PESE x64
locale                  fr-FR
inherit                 {bootloadersettings}
osdevice                ramdisk=[boot]\HBCD\10\sources\boot.wim
systemroot              \windows
bootmenupolicy          Legacy
detecthal               Yes
winpe                   Yes
ems                     No

c:\>bcdedit /export c:\ bcd10
La commande d'exportation de magasin n'est pas valide.
Exécutez « bcdedit /? » pour obtenir de l'aide sur la ligne de commande.
Paramètre incorrect.

c:\>bcdedit /export c:\bcd10
L'opération a réussi.

c:\>bcdedit /import c:\boot\bcd
L'opération a réussi.

c:\>_
```

Figure 11 - Fichier avant ajout de Windows 7

```
c:\>bcdedit

Gestionnaire de démarrage Windows
---------------------------------
identificateur          {bootmgr}
description             Windows Boot Manager
locale                  fr-FR
inherit                 {globalsettings}
default                 {default}
displayorder            {default}
toolsdisplayorder       {memdiag}
timeout                 1

Chargeur de démarrage Windows
-----------------------------
identificateur          {default}
device                  ramdisk=[boot]\HBCD\10\sources\boot.wim
path                    \windows\system32\boot\winload.exe
description             Win10PESE x64
locale                  fr-FR
inherit                 {bootloadersettings}
osdevice                ramdisk=[boot]\HBCD\10\sources\boot.wim
systemroot              \windows
bootmenupolicy          Legacy
detecthal               Yes
winpe                   Yes
ems                     No

Chargeur de démarrage Windows
-----------------------------
identificateur          {db9e49af-70a3-11e9-8661-080027536770}
device                  ramdisk=[boot]\HBCD\7\7.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
path                    \windows\system32\boot\winload.exe
description             Mini Windows 7
locale                  fr-FR
inherit                 {bootloadersettings}
osdevice                ramdisk=[boot]\HBCD\7\7.wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
systemroot              \windows
bootmenupolicy          Legacy
detecthal               Yes
winpe                   Yes
ems                     No
```

Figure 12 - Fichier après ajout de Windows 7

On peut voir entre la Figure 11 et 12 l'ajout d'une entrée dans le fichier de démarrage bootmgr, pointant vers le chargeur du mini Windows 7. Lorsque l'on choisit de démarrer Windows dans le menu Medicat on peut ainsi choisir entre Windows 7 et 10.

Pour terminer cette mission j'ai dû modifier la langue de clavier utilisé par défaut par les versions de Windows. L'interface habituelle de Windows pour changer la langue de clavier était inaccessible sur ces versions allégées. De plus, le Windows chargé provient d'une clé USB et non d'un disque dur ce qui veut dire qu'aucunes actions menées ne sont enregistrées, les modifications apportées au système disparaissent lors de l'arrêt de l'ordinateur. Il a fallu accéder au registre du fichier .wim qui contient l'image de Windows. Le logiciel NTLite propose cette fonctionnalité (voir Figure 13).

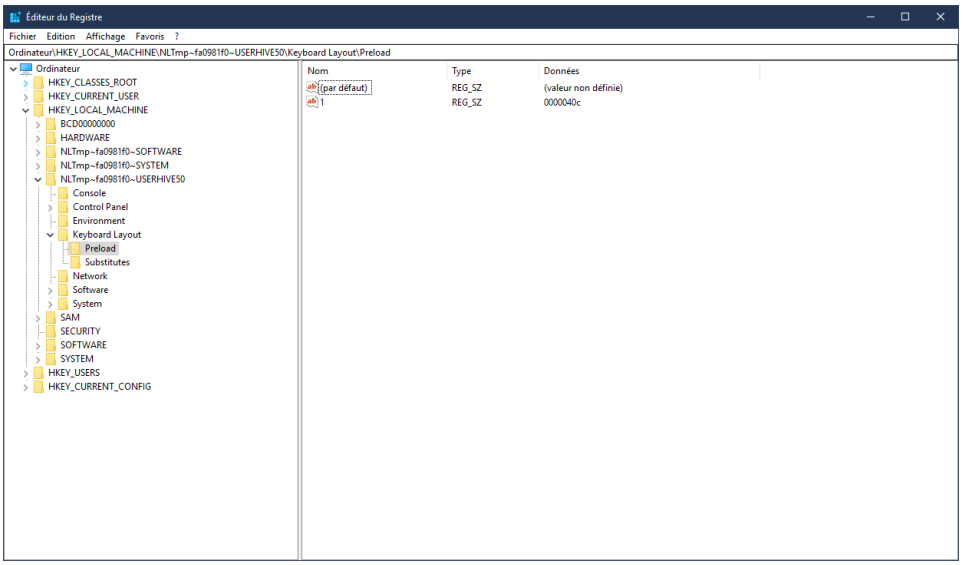

Figure 13 - Éditeur de registre NTLite

C’est la valeur 0000040c qui définit la langue française du clavier par défaut. À l’origine cette valeur était à 00000400, ce qui correspond au clavier qwerty américain.

Pour conserver et transférer le travail effectué plus simplement, j’ai transformé tous les dossiers et fichiers de Medicat en fichier ISO. J’ai pour cela utilisé le logiciel ImgBurn (voir Figure 14). Il suffit de sélectionner un dossier source et une destination sur l’ordinateur.

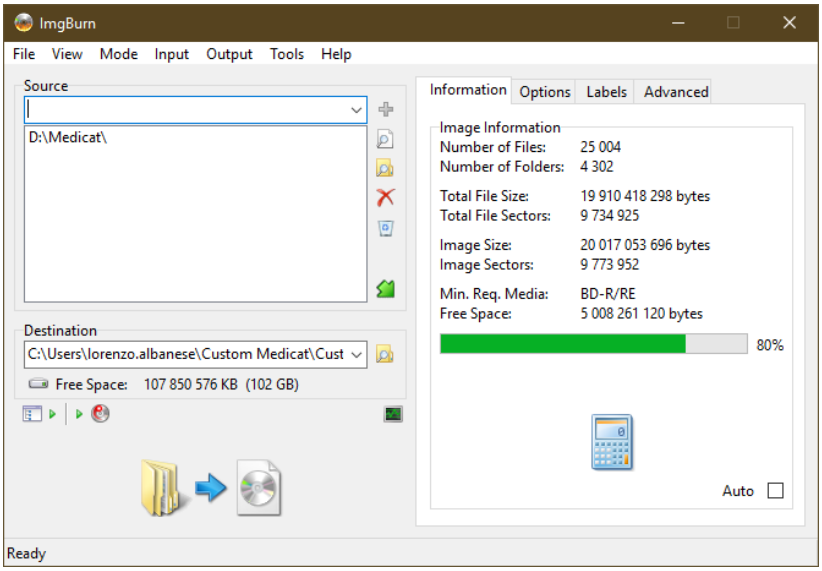

Figure 14 - Interface d’ImgBurn

Pour transférer ce fichier ISO sur une clé USB j’ai utilisé le logiciel Rufus, il suffit de sélectionner le fichier et la clé USB (voir Figure 15).

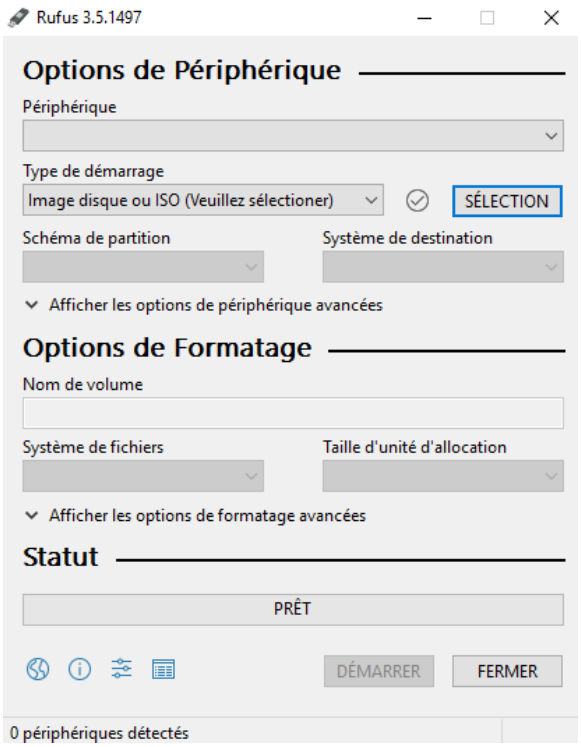

Figure 15 - Interface de Rufus

b. Chiffrement de clés USB

Pour compléter les nouvelles clé USB Medicat, je devais chercher un moyen de chiffrer celles-ci. Le but final était d'avoir une partition avec Medicat et une autre contenant des données de clients, inaccessible sans connaître le mot de passe. J'avais comme contrainte que le logiciel utilisé devait être entièrement embarqué sur la clé USB. Cela signifie que la demande de mot de passe doit se faire sur n'importe quel ordinateur, sans nécessiter l'installation de logiciels.

Après diverses recherches les quelques logiciels complètement embarqués étaient BitLocker (intégré à Windows), SecurStcik, Kakasoft USB Security et Rohos. Sur cette première base j'ai effectué des tests de performances en copiant des fichiers (un gros de 10 Go et plusieurs petits, 7000 de 89 o), et j'ai cherché des failles de sécurité. Grace au logiciel TreeSize, j'ai pu retrouver des fichiers cachés non chiffrés sur la clé USB (voir Figure 16).

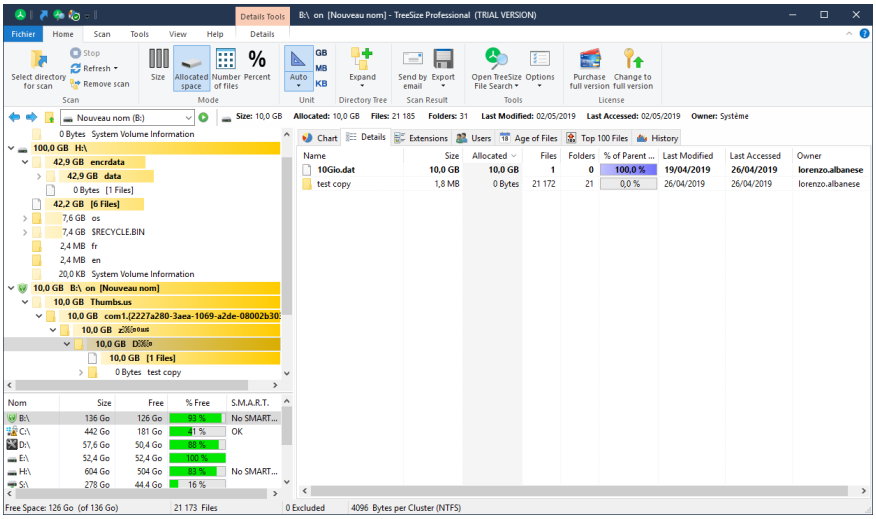

Figure 16 - Faille de sécurité d'USB Security

Au terme de ces tests nous avons retenu, avec mon tuteur, d'envisager d'utiliser BitLocker ou Rohos. Dans le cas e Bitlocker, Windows rend le lecteur inaccessible sans avoir écrit le bon mot de passe. Pour Rohos, on peut voir sur la Figure 17 que les données ne sont pas inscrites en clair, elles sont enregistrées sous la forme d'une image disque d'un format spécifique à Rohos (.rdi).

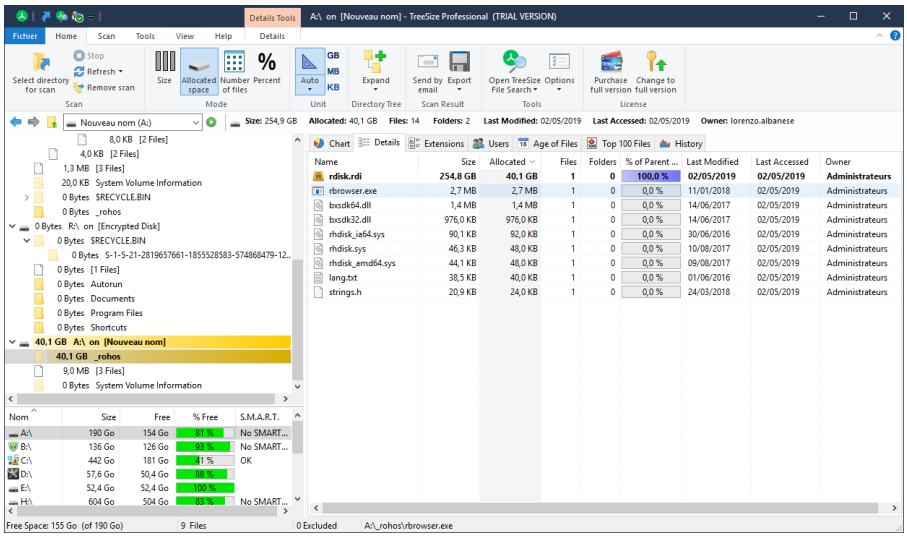

Figure 17 - Données converties par Rohos

Bitlocker et Rohos ayant les mêmes performances, il a été décidé de garder BtiLocker puisqu'il est gratuit sur les versions Windows 10 Pro et qu'il est très simple à configurer. Une licence Rohos Disk Encryption coûte 80 € pour une entreprise.

2. Gestion de serveurs Exchange

a. Vérification d'espace disque

J'ai donc commencé par chercher comment avertir d'un espace de disque faible. Rapidement dans mes recherches, j'ai remarqué qu'il existe déjà dans Windows un programme appelé Observateur d'évènements qui permet de contrôler tout ce qui peut se faire sur l'appareil. Il y avait donc déjà la possibilité de vérifier la taille restante des disques.

Le premier problème que j'ai rencontré est que lorsqu'on met en place l'observation des disques, il faut choisir les lecteurs à vérifier. Cela veut dire que si l'on ajoute des disques par la suite, l'observation ne se modifie pas automatiquement pour prendre en compte ces nouveaux disques.

Le deuxième problème est qu'il n'est pas possible d'envoyer des alertes par mail directement après avoir observé l'évènement, la seule action possible est de démarrer un programme. J'ai alors pensé à utiliser un script PowerShell pour gérer l'envoi de mails. En poursuivant mes recherches dans cette direction, j'ai trouvé un script qui vérifie l'espace disponible de tous les disques et génère un rapport qui est envoyé par mail, en cas de dépassement des limites. Ces limites sont à définir par l'utilisateur, ex : seuil d'avertissement à 15% restants et seuil critique à 7% restants.

J'ai premièrement modifié ce script pour mettre en évidence les variables à définir. Il fallait maintenant trouver un moyen de donner automatiquement les valeurs nécessaire à ce script pour pouvoir le lancer. J'ai donc commencé à développer un script PowerShell qui fait office de menu pour récupérer les informations en posant des questions et en lisant les réponses de l'utilisateur (voir Figure 18).

Figure 18 - Résumé des informations fournies

Après avoir répondu aux questions l'utilisateur à la possibilité de vérifier ses réponses. Lorsqu'il valide ses réponses, mon script programme une tâche dans le planificateur de tâches qui est un autre outil intégré à Windows (voir Figure 19).

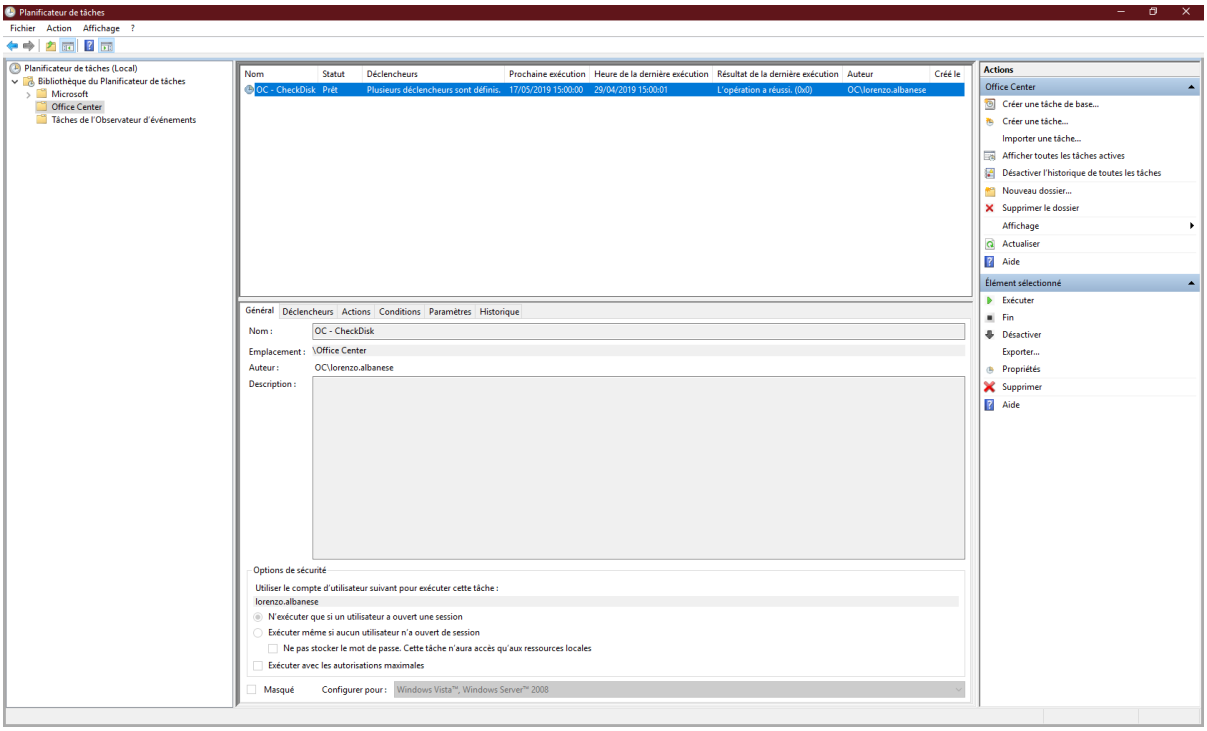

Figure 19 - La tâche crée dans le planificateur de tâches de Windows

On peut voir sur la capture qu'il est possible d'ajouter des déclencheurs et des actions pour la tâche créée. Mon script ajoute comme déclencheurs une exécution journalière (à raison de 5 fois par jours). Pour les actions, il programme le lancement de powershell.exe avec comme argument le script que j'ai trouvé et toutes les variables définies par l'utilisateur (voir Figure 20).

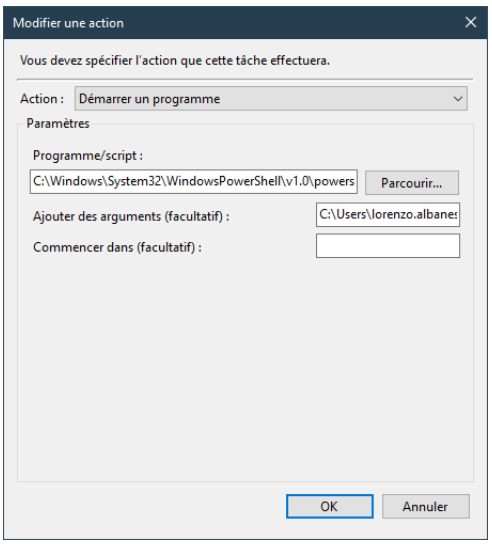

Figure 20 - Action de la tâche planifiée

J'ai donc mis au point un script PowerShell qui interroge l'utilisateur et programme l'exécution automatique d'une vérification des disques connectés à l'ordinateur. En cas de dépassement des seuils programmés, un mail sera envoyé avec un rapport sur l'espace restant des disques concernés (voir Figure 21). L'envoi de mails se fait en utilisant une adresse Gmail et les serveurs mail de Google.

Ce système pourra ainsi être déployé sur des serveurs mis en place par Office Center pour éviter des problèmes d'espace disques insuffisants.

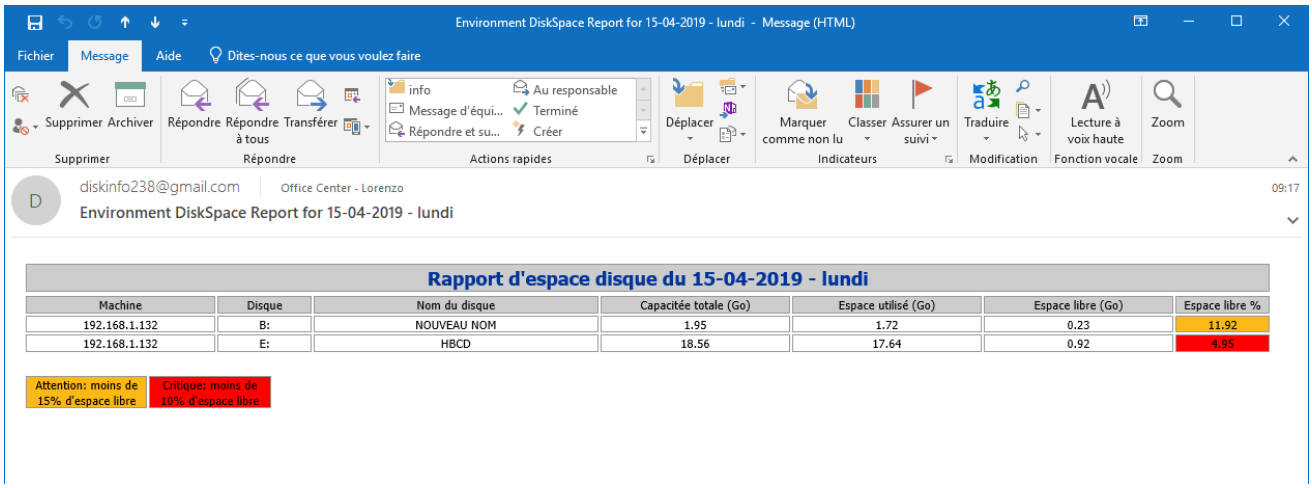

Figure 21 - Rapport d'espace disque restant envoyé par mail

b. Gestion de comptes Exchange

Comme expliqué précédemment, mon tuteur de stage était très satisfait du travail que je venais de réaliser pour créer un menu sous PowerShell et lancer des scripts de manière automatisée. Il m'a donc demandé de faire un autre menu en PowerShell afin d'initialiser des scripts préparés par Vincent, ce que j'ai fait (voir Figure 22).

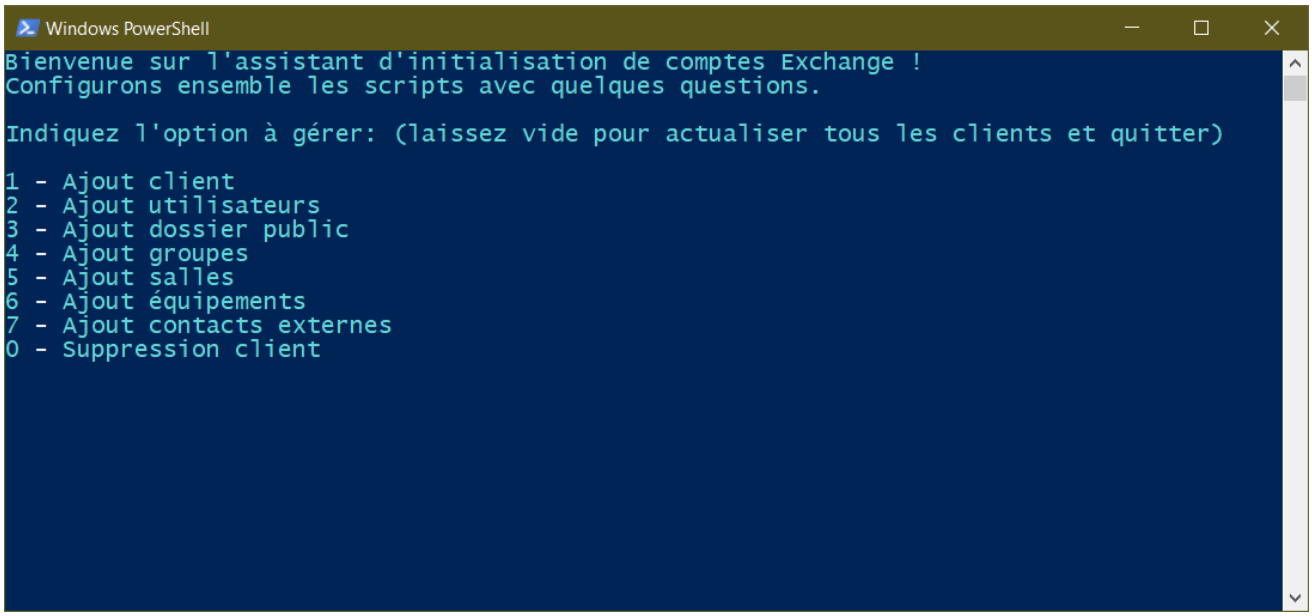

Figure 22 - Menu d'initialisation des scripts Exchange

De la même manière, le menu demande des informations et lance des scripts individuels pour par exemple ajouter des utilisateurs ou des groupes de diffusion à un client donné. Le lancement des scripts se fait directement par PowerShell, il n'y a cette fois pas de système de tâches planifiées.

Dans un deuxième temps, mon tuteur m'a demandé de reprendre ces éléments pour les intégrer à des pages web. J'ai donc du mobiliser mes connaissances en PHP afin de créer une interface utilisateur, et gérer l'exécution de scripts PowerShell (voir Figure 23).

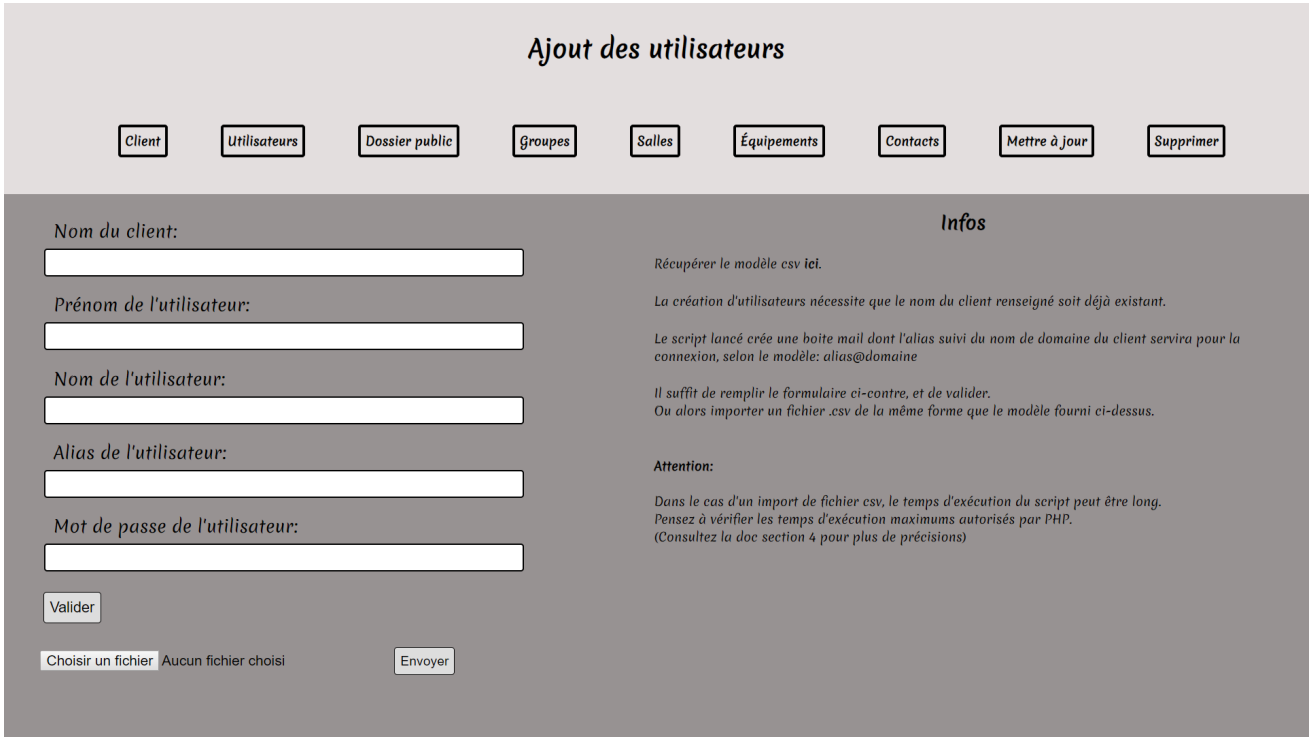

Figure 23 - Page web pour la création d'utilisateurs Exchange

On peut voir sur cet exemple le formulaire à remplir pour l'ajout d'utilisateurs, ainsi que la possibilité d'ajouter un fichier. Après avoir fini l'interface web, Vincent m'a demandé de rajouter la possibilité d'envoyer un fichier. Le but est de préparer un fichier au format CSV (tableau de données), et de l'envoyer pour que le serveur traite plusieurs utilisateurs en même temps. Il s'agit d'un gain de temps puisque les techniciens qui mettront en place des serveurs Exchange n'auront pas besoin d'attendre pour ajouter chaque utilisateur un à un.

3. Étude de configuration de switchs

La dernière semaine du stage, j'ai dû réfléchir à un problème de lenteur réseau chez un client d'Office Center. De manière aléatoire, les communications entre ordinateurs et serveurs sont très lentes, un logiciel communicant avec un serveur interne mais du temps à s'initialiser et la copie de fichiers sur le serveur prend beaucoup de temps. Ces lenteurs varient d'un ordinateur à un autre et peuvent n'être que temporaire sur un ordinateur. Tous les ordinateurs ont les mêmes caractéristiques techniques et les mêmes performances.

Je me suis rendu rapidement sur site pour constater les lenteurs, puis j'ai travaillé à distance en me connectant sur le serveur GED grâce à TeamViewer. J'ai pu lancer des analyses de trames en utilisant Wireshark mais je n'ai rien trouvé d'anormal. Comme j'avais peu de temps pour étudier le sujet, mon tuteur m'a demandé de me connecter aux switchs pour relever des informations qui leur seront utile par la suite. Le réseau de leur client se compose principalement de 2 switchs 48 ports administrable, sur lesquels de connectent tous les ordinateurs, serveurs, copieurs, caméras et 2 points d'accès vers des fournisseurs d'accès à internet.

Sur les 2 switch administrables de marques TP Link et Netgear j'ai pu tester l'état des câbles connectés et consulter les logs grâce à une interface web intégrée. J'ai ensuite produit un rapport de mes observations que j'ai remis à mon tuteur avant de partir (voir annexes).

V. Conclusion

Lors de ce stage j'ai pu produire des outils utiles à Office Center qui fera gagner du temps au service technique.

Ils ont désormais accès à une nouvelle version de leur HBCD qui prend en charge les nouvelles technologies compatibles Windows 10. En complétant avec mes recherches sur le chiffrement des partitions, ils ont maintenant des clés USB contenants divers utilitaires qu'ils peuvent utiliser sur site chez des clients, sans craindre de perdre des données sensibles.

Grâce aux menu PowerShell et web que j'ai mis au point, ils ont à présent des moyens simples de gérer la configuration de serveurs Exchange et la surveillance d'espace disque.

Les missions que j'ai réalisées correspondent à leurs objectifs et répondent donc au cahier des charges établi avant et pendant mon travail de développement.

Je pourrais citer comme point d'amélioration la demande d'aide technique. En effet, j'aurais pu gagner plus de temps en expliquant les problèmes rencontrés à mon tuteur, plutôt qu'en cherchant moi-même une solution.

Je suis cependant très satisfait de mon stage car il m'a permis de me familiariser avec le monde professionnel et le travail en équipe de manière très concrète.

Sur le plan personnel, j'ai pu me rendre compte de mon intérêt réel pour les réseaux et le développement web ou PowerShell au travers des tâches effectuées. Cela correspond d'ailleurs à mon projet professionnel qui est de m'orienter vers l'architecture réseau. Cette catégorie de métier comprend les réflexions que j'ai eues pour le cas concret chez le client, ainsi que le concept de développement dans le but d'une automatisation des tâches.

VI. Remerciements

Je souhaiterais remercier chaleureusement Jérémy Raspail pour m'avoir accepté en stage, ainsi que Vincent pour m'avoir pris en charge pendant tout la durée de mon stage. Également, de manière plus générale, je remercie toute l'équipe d'Office Center pour leur accueil convivial.

Cette première expérience en entreprise m'a beaucoup appris sur le travail en équipe, aussi bien en termes de travail que pour le comportement à adopter.

Grâce aux conseils de Jérémy, Vincent et Allan, j'ai pris connaissance de mes pistes d'améliorations personnelles et professionnelles, et j'ai acquis diverses compétences en informatique qui me seront utiles tout au long de ma vie professionnelle.

J'ai à présent en ma possession divers éléments qui me permettront de réussir au mieux ma poursuite d'études dans une école d'ingénieur en alternance.

VII. Glossaire

DUT, Diplôme Universitaire de Technologie

GED, Gestion Électronique des Documents

HBCD, Hiren's Boot CD

R&T, Réseaux et Télécommunications

Windows PE SE, Preinstallation Environnement Special Edition

[1] Microsoft Exchange est un serveur de messagerie qui s'installe sur un serveur Windows. Il a pour but de connecter chaque membre d'une société pour faciliter la collaboration. Il assure également la gestion d'agenda, de contacts et tâches. Il offre aux entreprises une organisation simplifiée.

[2] L'infogérance est de fait de confier la gestion de tâches informatiques à un prestataire extérieur.

[3] L'audit est la procédure de contrôle et de gestion d'équipements informatiques.

[4] La gestion électronique des documents est le procédé visant à organiser et référencer les documents numérisés pour en faciliter l'accès.

[5] TeamViewer est logiciel permettant de prendre le contrôle d'un ordinateur à distance.

[6] L'Active Directory est un système d'annuaire pour les systèmes d'exploitation Windows. Il fournit des services d'identifications et authentification d'utilisateurs à un réseau d'ordinateurs.

[7] Il s'agit d'une version de Windows 10 allégée et capable de démarrer à partir d'une clé USB.

[8] Le bootloader Windows est un logiciel qui définit, au démarrage de Windows, la manière dont l'ordinateur doit démarrer. Ici, est utilisé pour choisir entre 2 systèmes d'exploitation.

VIII. Sitographie

Documentation officielle de PowerShell :

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/

Win10PE SE Builder :

http://theoven.org/index.php?topic=1336.0

Divers autres sites consultés très rapidement au cours de mes recherches

IX. Annexes

1. Script de vérification d'espace disque

```
################################################################################
#
#  Check disk space and send an HTML report as the body of an email.
#  Reports only disks on computers that have low disk space.
#  Author: Mike Carmody
#  Some ideas extracted from Thiyagu's Exchange DiskspaceHTMLReport module.
#  Date: 8/10/2011
#  I have not added any error checking into this script yet.
#
################################################################################
param(
    [int]$percentWarning,
    [int]$percentCritical,
    [string]$listPath,
    [string]$reportPath,
    [string]$reportName,
    [string]$Daysback,
    [string]$filter1,
    [string]$filter2,
    [string]$filter3,
    [array]$mails
)

# Continue even if there are errors
$ErrorActionPreference = "Continue";

#############################################################################################
# Items to change to make it work for you.
#
# EMAIL PROPERTIES
#  - the $mails that this report will be sent to.
#  - near the end of the script the smtpserver, From and Subject.
#
# REPORT PROPERTIES
#  - you can edit the report path and report name of the html file that is the report.
#############################################################################################

                    # Réglage des pourcentages d'alerte d'espace disque
#-------------------------------------------------------------------------------------------------------#
#
# Set your warning and critical thresholds
#$percentWarning = 15;
#$percentCritcal = 10;
#
#-------------------------------------------------------------------------------------------------------#

# EMAIL PROPERTIES
 # Set the recipients of the report.

                    # Liste des adresses mail pour envoi de rapports
#-------------------------------------------------------------------------------------------------------#
#
#  $mails = "lorenzo@officecenter.fr"
#
#-------------------------------------------------------------------------------------------------------#
```

```
   #$mails = "You@company.com" # I use this for testing by uing my email address.
  #$mails = "you@company.com", "manager@company.com", "etc@company.com";  # can be sent to individuals.

# REPORT PROPERTIES
 # Path to the report

                    # Emplacement du dossier où seront stockés les rapports
#-------------------------------------------------------------------------------------------------------------#
#
#  $reportPath = "C:\Users\lorenzo.albanese\checkdisk\";
#
#-------------------------------------------------------------------------------------------------------------#

 # Report name
  $reportName += "_$(get-date -format ddMMyyyy_HHmm).html";

# Path and Report name together
$diskReport = $reportPath + $reportName

#Set colors for table cell backgrounds
$redColor = "#FF0000"
$orangeColor = "#FBB917"
$whiteColor = "#FFFFFF"

# Count if any computers have low disk space.  Do not send report if less than 1.
$i = 0;

# Get computer list to check disk space

                    # Emplacement de la liste des adresses IP à vérifier
#-------------------------------------------------------------------------------------------------------------#

$computers = Get-Content $listPath;  #"C:\Users\lorenzo.albanese\servers_c.txt";

#-------------------------------------------------------------------------------------------------------------#

$datetime = Get-Date -Format "dd-MM-yyyy_HHmmss";

# Remove the report if it has already been run today so it does not append to the existing report
If (Test-Path $diskReport)
    {
       #Remove-Item $diskReport
    }

           # Vérifier les besoins pour ajuster le temps avant suppression des anciens rapports
#/!\!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!#/!\#

# Cleanup old files..
#$Daysback = "-7"
$CurrentDate = Get-Date;
$DateToDelete = $CurrentDate.AddDays($Daysback);
Get-ChildItem $reportPath | Where-Object { $_.LastWriteTime -lt $DatetoDelete } | Remove-Item;

#/!\!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!#/!\#

# Create and write HTML Header of report
$titleDate = get-date -uformat "%d-%m-%Y %H:%M - %A"
$header = "
  <html>
  <head>
  <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-1'>
  <title>Rapport d'espace disque</title>
  <STYLE TYPE='text/css'>
  <!--
  td {
```

```
  font-family: Tahoma;
  font-size: 11px;
  border-top: 1px solid #999999;
  border-right: 1px solid #999999;
  border-bottom: 1px solid #999999;
  border-left: 1px solid #999999;
  padding-top: 0px;
  padding-right: 0px;
  padding-bottom: 0px;
  padding-left: 0px;
 }
 body {
  margin-left: 3%;
  margin-top: 2%;
  margin-right: 3%;
  margin-bottom: 10%;
  table {
  border: thin solid #000000;
 }
 -->
 </style>
 </head>
 <body>
 <table width='100%'>
 <tr bgcolor='#CCCCCC'>
 <td colspan='7' height='25' align='center'>
 <font face='tahoma' color='#003399' size='4'><strong>Rapport d'espace disque du $titledate</strong></font>
 </td>
 </tr>
 </table>
"
 Add-Content $diskReport $header

# Create and write Table header for report
 $tableHeader = "
 <table width='100%'><tbody>
 <tr bgcolor=#CCCCCC>
 <td width='10%' align='center'>Machine</td>
 <td width='5%' align='center'>Disque</td>
 <td width='15%' align='center'>Nom du disque</td>
 <td width='10%' align='center'>Capacit&eacute;e totale (Go)</td>
 <td width='10%' align='center'>Espace utilis&eacute; (Go)</td>
 <td width='10%' align='center'>Espace libre (Go)</td>
 <td width='5%' align='center'>Espace libre %</td>
 </tr>
"
Add-Content $diskReport $tableHeader

# Start processing disk space reports against a list of servers
  foreach($computer in $computers)
 {

                                # Filtrage des disques à vérifier
#-------------------------------------------------------------------------------------------------------------#

$disks
if ($filter1 -eq 6) {
    $disks = Get-WmiObject -ComputerName $computer -Class Win32_LogicalDisk
} else {
    if (-not ($filter1 -eq "")) {
        $disks += Get-WmiObject -ComputerName $computer -Class Win32_LogicalDisk -Filter $filter1
    }
    if (-not ($filter2 -eq "")) {
        $disks += Get-WmiObject -ComputerName $computer -Class Win32_LogicalDisk -Filter $filter2
    }
```

```
    if (-not ($filter3 -eq "")) {
        $disks += Get-WmiObject -ComputerName $computer -Class Win32_LogicalDisk -Filter $filter3
    }
}

#-----------------------------------------------------------------------------------------------------------#

 $computer = $computer.toupper()
  foreach($disk in $disks)
 {
  $deviceID = $disk.DeviceID;
       $volName = $disk.VolumeName;
  [float]$size = $disk.Size;
  [float]$freespace = $disk.FreeSpace;
  $percentFree = [Math]::Round(($freespace / $size) * 100, 2);
  $sizeGB = [Math]::Round($size / 1073741824, 2);
  $freeSpaceGB = [Math]::Round($freespace / 1073741824, 2);
       $usedSpaceGB = $sizeGB - $freeSpaceGB;
       $color = $whiteColor;

# Set background color to Orange if just a warning
 if($percentFree -lt $percentWarning)
  {
    $color = $orangeColor

# Set background color to Red if space is Critical
      if($percentFree -lt $percentCritical)
       {
       $color = $redColor
       }

 # Create table data rows
    $dataRow = "
  <tr>
  <td width='10%' align='center'>$computer</td>
  <td width='5%' align='center'>$deviceID</td>
  <td width='15%' align='center'>$volName</td>
  <td width='10%' align='center'>$sizeGB</td>
  <td width='10%' align='center'>$usedSpaceGB</td>
  <td width='10%' align='center'>$freeSpaceGB</td>
  <td width='5%' bgcolor=`'$color`' align='center'>$percentFree</td>
  </tr>
"
Add-Content $diskReport $dataRow;
#Write-Host -ForegroundColor DarkYellow "$computer $deviceID percentage free space = $percentFree";
    $i++
  }
 }
}

# Create table at end of report showing legend of colors for the critical and warning
 $tableDescription = "
 </table><br><table width='20%'>
 <tr bgcolor='White'>
    <td width='10%' align='center' bgcolor='#FBB917'>Attention: moins de $percentWarning% d'espace libre</td>
 <td width='10%' align='center' bgcolor='#FF0000'>Critique: moins de $percentCritical% d'espace libre</td>
 </tr> </table>
"
 Add-Content $diskReport $tableDescription
 Add-Content $diskReport "<br><br><br>Ce rapport est g&eacute;n&eacute;r&eacute; par un script mis en place par la
soci&eacute;t&eacute; Office Center. tel: 04 92 31 39 75"
 Add-Content $diskReport "</body></html>"

# Send Notification if alert $i is greater than 0
if ($i -gt 0)
```

```
{
   foreach ($mail in $mails)
{
      #Write-Host "Sending Email notification to $mail"

                  # Informations de connexions au serveur mail SMTP
#-----------------------------------------------------------------------------------------------------#

  $smtpServer = "smtp.gmail.com"
  $smtpmail = "diskinfo238@gmail.com"
  $pass =
"76492d1116743f0423413b16050a5345MgB8AEoAZAB0AHMAYQAxAGgAUgB3AGwASABQADQAWQBYAGQ
AaQBNAEcAeABBAEEAPQA9AHwAMABiADAAMwBlAGEANgA1ADcANgAxAGEANAAxADMANABkAGQA
ZgBhAGQANAA1ADEAOAA5ADAAMwA2ADAANwBhADAAYQA1ADAAYQAzAGQANQAzADgAYgBjADAA
NAAxAGEANAAxADUAMQA3AGQAYgA3AGYANABjADAANwA5ADYAYwA="
  [Guid]$Key = "e8bdc7c5-a62c-4fa3-9228-5abe22488141"
  $smtpPass = ConvertTo-SecureString -String $pass -Key $Key.ToByteArray()

#-----------------------------------------------------------------------------------------------------#

  $credentials = New-Object System.Net.NetworkCredential($smtpmail,$smtpPass);
  $smtp = New-Object Net.Mail.SmtpClient($smtpServer)

  $smtp.EnableSsl = $true                  # $true ou $false si ssl requis pour le serveur mail

  $smtp.Credentials = $credentials
  $msg = New-Object Net.Mail.MailMessage
  $msg.From = "OC CheckDisk<diskinfo238@gmail.com>"
  $msg.To.Add($mail)
  $msg.Subject = "[OC]-Rapport d'espace disque du $titledate"
  $msg.IsBodyHTML = $true
  $msg.Body = get-content $diskReport
  $smtp.Send($msg)
    }
  }
```

2. Script d'initialisation de tâche planifiée

```
#####################################################################
#
#  Asistant d'initialisation de tâche
#   vérifiant l'espace disques disponible,
#   et gérant les alertes par mail en cas de dépassement.
#  Autheur: Lorenzo Albanese
#  Sur base du script de Mike Carmody.
#  Date: 17/04/2019
#
#####################################################################

Write-Host -ForegroundColor cyan -Object "Bienvenue sur l'assistant d'initialisation de CheckDisk !
Configurons ensemble le script avec quelques questions.

Commencons !
"

# Seuil d'avertissement $percentWarning
Read-Host
clear
Write-Host -ForegroundColor cyan -Object "[1/8] Indiquez le seuil d'avertissement d'espace disque (%):"
$percentWarning = Read-Host

if ($percentWarning -eq "") {
   while ($percentWarning -eq "") {
      Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
```

```
        $percentWarning = Read-Host
    }
}

# Seuil critique $percentCritical
clear
Write-Host -ForegroundColor cyan -Object "[2/8] Indiquez le seuil critique d'espace disque (%):"
$percentCritical = Read-Host

if ($percentCritical -eq "") {
    while ($percentCritical -eq "") {
        Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
        $percentCritical = Read-Host
    }
}

# Chemin vers liste pour scan $listPath
clear
Write-Host -ForegroundColor cyan -Object "[3/8] Indiquez le chemin d'accès au fichier contenant la liste des machines à
scanner (fichier .txt):"
$listPath = Read-Host

if ($listPath -eq "") {
    while ($listPath -eq "") {
        Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
        $listPath = Read-Host
    }
}

if (-not (Test-Path -Path $listPath -PathType Leaf)) {
    Write-Host "`nErreur: Ce ficihier n'exite pas !" -ForegroundColor Red;
    Write-Host "Pas de stress, nous allons le créer ensemble." -ForegroundColor Green
    Read-Host
    clear
    $lines = ""
    Write-Host -ForegroundColor Cyan -Object "Indiquez une adresse IP ou un nom de machine suivi de la touche Entrer
(laisser vide pour terminer):"
    while (-not ($line -eq "")) {
        $line = Read-Host
        $lines += "$line`n"
    }
    if (New-Item -Path $listPath -ItemType file -Value $lines -Force) {
        Write-Host -ForegroundColor Green -Object "Le fichier à été créé !"
    } else {
        Write-Host -ForegroundColor Red -Object "Une erreur est survenue lors de la création du fichier."
        Read-Host
        exit 2
    }
    Read-Host
}

# Chemin vers dossier qui stocke les rapports $reportPath
clear
Write-Host -ForegroundColor cyan -Object "[4/8] Indiquez le chemin d'accès au dossier qui contiendra les rapports (il
sera créé si inexistant):"
$reportPath = Read-Host

if ($reportPath -eq "") {
    while ($reportPath -eq "") {
        Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
        $reportPath = Read-Host
    }
}

if (-not (Test-Path -Path $reportPath -PathType Container)) {
```

```
    if (New-Item -Path $reportPath -ItemType directory -Force) {
        Write-Host -ForegroundColor Green -Object "Le dossier à été créé !"
    } else {
        Write-Host -ForegroundColor Red -Object "Une erreur est survenue lors de la création du dossier."
        Read-Host
        exit 2
    }
    Read-Host
}

# Nom des rapports générés $reportName
clear
Write-Host -ForegroundColor Cyan -Object "[5/8] Indiquez le nom des rapports qui seront générés (la date sera ajoutée):"
$reportName = Read-Host

if ($reportName -eq "") {
    while ($reportName -eq "") {
        Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
        $reportName = Read-Host
    }
}

# Delai avant suppression rapports
clear
Write-Host -ForegroundColor Cyan -Object "[6/8] Indiquez la durée de conservation des rapports en jours (laisser vide
pour sans limite):"
$days = Read-Host
$Daysback = "-$days"

if ($days -eq "") {
    $Daysback = "-100000"
    $days = "indéfini"
}

# Filtrage des disques à vérifier $filter[]
clear
Write-Host -ForegroundColor Cyan -Object "[7/8] Les filtres de sélection des types de disques:
En fonction des besoins, il est possible de ne sélectionner que certains types de disques.
Entrez les numéros correspondants pour filtrer.
2: Disques amovibles (clés USB)
3: Disques locaux (interne ou externe)
4: Disques réseau
5: CD/DVD
6: Tout
"

$filter = @("","","","")
$filtre = @("","","","")
$f = 0

for ($i = 1; $i -le 3; $i++ ) {
    Write-Host -ForegroundColor Cyan -Object "Entrez la valeur du filtre [$i/3]"
    $filtre[$i] = Read-Host
    if (-not ($filtre[$i] -eq "")) {
        $f++
        if ($filtre[$i] -match "6") {
            $i = 4
            $filter = @("","6","","")
            $filtre = @("","Tout","","")
        } else {
            if (-not ($filtre[$i] -match "[2-5]")) {
                while (-not $filtre[$i] -match "[2-5]") {
                    Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur correcte`n"
                    Write-Host -ForegroundColor Cyan -Object "Entrez la valeur du filtre [$i/3]"
                    $filtre[$i] = Read-Host
```

```
            }
        }
        $filter[$i] = "DriveType = "+$filtre[$i]
        switch ($filtre[$i]) {
            2 {
                $filtre[$i] = "Disques amovibles (clés USB)"
                break
            }
            3 {
                $filtre[$i] = "Disques locaux (interne ou externe)"
                break
            }
            4 {
                $filtre[$i] = "Disques réseau"
                break
            }
            5 {
                $filtre[$i] = "CD/DVD"
                break
            }
        }
    }
  } else {
    $filtre[$i] = "Aucun"
  }
}

while ($f -eq 0) {
  Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
  Read-Host

  for ($i = 1; $i -le 3; $i++ ) {
  Write-Host -ForegroundColor Cyan -Object "Entrez la valeur du filtre [$i/3]"
  $filtre[$i] = Read-Host
  if (-not ($filtre[$i] -eq "")) {
    $f++
    if ($filtre[$i] -match "6") {
      $i = 4
      $filter = @("","6","","")
      $filtre = @("","Tout","","")
    } else {
      if (-not ($filtre[$i] -match "[2-5]")) {
        while (-not $filtre[$i] -match "[2-5]") {
          Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur correcte`n"
          Write-Host -ForegroundColor Cyan -Object "Entrez la valeur du filtre [$i/3]"
          $filtre[$i] = Read-Host
        }
      }
      $filter[$i] = "DriveType = "+$filtre[$i]
      switch ($filtre[$i]) {
        2 {
          $filtre[$i] = "Disques amovibles (clés USB)"
          break
        }
        3 {
          $filtre[$i] = "Disques locaux (interne ou externe)"
          break
        }
        4 {
          $filtre[$i] = "Disques réseau"
          break
        }
        5 {
          $filtre[$i] = "CD/DVD"
          break
```

```
            }
        }
    }
  } else {
    $filtre[$i] = "Aucun"
  }
}
}

# Définissions adresses mail pour envois des rapports $mails
clear
Write-Host -ForegroundColor Cyan -Object "[8/8] Entrez une par une les adresses mail sur lesquelles seront envoyés les rapports (laisser vide pour terminer):"
$mails = @()
$mail = Read-Host

if ($mail -eq "") {
  while ($mail -eq "") {
    Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
    $mail = Read-Host
  }
}

$mails += $mail
while (-not $mail -eq "") {
  $mail = Read-Host
  if (-not $mail -eq "") {
    $mails += ",$mail"
  }
}

# Résumé des valeurs
clear
Write-Host -ForegroundColor Green -Object "C'est la fin des questions !`n"
Write-Host -ForegroundColor Cyan -Object "Voila un résumé des données saisies :
"
Write-Host -ForegroundColor Cyan -Object "Seuil d'avertissement espace disque:          " -NoNewline; Write-Host -Object "$percentWarning%";
Write-Host -ForegroundColor Cyan -Object "Seuil critique espace disque:               " -NoNewline; Write-Host -Object "$percentCritical%";
Write-Host -ForegroundColor Cyan -Object "Fichier contenant la liste des machines:      " -NoNewline; Write-Host -Object $listPath;
Write-Host -ForegroundColor Cyan -Object "Dossier contenant les rapports:              " -NoNewline; Write-Host -Object $reportPath;
Write-Host -ForegroundColor Cyan -Object "Nom des rapports générés:                    " -NoNewline; Write-Host -Object $reportName;
Write-Host -ForegroundColor Cyan -Object "Durée de conservation des rapports (jours):   " -NoNewline; Write-Host -Object $days;
Write-Host -ForegroundColor Cyan -Object "Filtre de disque 1:                          " -NoNewline; Write-Host -Object $filtre[1];
Write-Host -ForegroundColor Cyan -Object "Filtre de disque 2:                          " -NoNewline; Write-Host -Object $filtre[2];
Write-Host -ForegroundColor Cyan -Object "Filtre de disque 3:                          " -NoNewline; Write-Host -Object $filtre[3];
Write-Host -ForegroundColor Cyan -Object "Adresses mail pour envoi de rapports:        " -NoNewline; Write-Host -Object $mails -NoNewline;

Write-Host -ForegroundColor Cyan -Object "`n`n`nPour valider cette configuration et mettre en place la tâche entrez V (la tâche existante sera remplacée)."
$check = Read-Host

if (-not ($check -match "[vV]")) {
  exit 1
}
```

```
Write-Host -ForegroundColor Green -Object "`nConfiguration validée !"
Read-Host
clear

Write-Host -ForegroundColor Cyan -Object "Dernière étape, entrez l'emplacement du dossier contenant CheckDisk.ps1
(Il doit pouvoir être accessible à tout moment):"
$scriptPath = Read-Host

if ($scriptPath -eq "") {
   while ($scriptPath -eq "") {
      Write-Host -ForegroundColor Red -Object "Erreur: Veuillez entrer une valeur:`n"
      $scriptPath = Read-Host
   }
}

if (-not (Test-Path -Path "$scriptPath\CheckDisk.ps1" -PathType Leaf)) {
   while (-not (Test-Path -Path "$scriptPath\CheckDisk.ps1" -PathType Leaf)) {
      Write-Host -ForegroundColor Red -Object "Erreur: Script CheckDisk.ps1 non détecté dans le dossier
indiqué.`nVeuillez réessayer: `n"
      $scriptPath = Read-Host
   }
}

$filter1 = $filter[1]
$filter2 = $filter[2]
$filter3 = $filter[3]

Write-Host -ForegroundColor Cyan -Object "`nConfiguration de la tâche en cours" -NoNewline; Sleep 1; Write-Host -
ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

$action = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-ExecutionPolicy Bypass
$scriptPath\Checkdisk.ps1 -percentWarning $percentWarning -percentCritical $percentCritical -listPath '$listPath' -
reportPath '$reportPath\' -reportName '$reportName' -Daysback $Daysback -filter1 '$filter1' -filter2 '$filter2' -filter3
'$filter3' -mails '$mails'"
$trigger = @(
   $(New-ScheduledTaskTrigger -At 5AM -Daily),
   $(New-ScheduledTaskTrigger -At 10AM -Daily),
   $(New-ScheduledTaskTrigger -At 3PM -Daily),
   $(New-ScheduledTaskTrigger -At 8PM -Daily),
   $(New-ScheduledTaskTrigger -At 12AM -Daily)
)

if (Register-ScheduledTask -TaskName "CheckDisk" -TaskPath "\Office Center\" -Action $action -Trigger $trigger -
Force) {
   Write-Host -ForegroundColor Green -Object "`nFélicitations ! Vous venez de mettre en place la vérification
automatique de disques."
   Write-Host -ForegroundColor Cyan -Object "`n`nMerci d'avoir utilisé cet assistant. :)"
   Read-Host
   exit 0
} else {
   Write-Host -ForegroundColor Red -Object "Une erreur est survenue lors de la création de la tâche."
   Read-Host
   exit 2
}
```

3. Script d'initialisation des scripts Exchange

```
#####################################################################
#
#  Asistant d'initialisation des scripts
#   permettant la mise en place de comptes exchange.
#  Autheur: Lorenzo Albanese
#  Sur base des scripts de Vincent Charles.
#  Date: 20/05/2019
```

```
#
####################################################################

$script = @'
  $player = New-Object -ComObject 'MediaPlayer.MediaPlayer'
  $player.Open("C:\ScriptsExchange\background.mp3")
  $player
'@
$runspace = [RunspaceFactory]::CreateRunspace()
$runspace.ApartmentState = "MTA"
$bgPowerShell = [PowerShell]::Create()
$bgPowerShell.Runspace = $runspace
$runspace.Open()
$player = @($bgPowerShell.AddScript($script).Invoke())[0]

Import-Module ActiveDirectory

Add-PSSnapin Microsoft.Exchange.Management.PowerShell.SnapIn

while ($true) {
  clear

  Write-Host -ForegroundColor Cyan -Object "Bienvenue sur l'assistant d'initialisation de comptes Exchange !"
  Write-Host -ForegroundColor Cyan -Object "Configurons ensemble les scripts avec quelques questions.`n"
  Write-Host -ForegroundColor Cyan -Object "Indiquez l'option à gérer: (laissez vide pour actualiser tous les clients et
quitter)`n"

  Write-Host -ForegroundColor Cyan -Object "1 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout client"
  Write-Host -ForegroundColor Cyan -Object "2 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout utilisateurs"
  Write-Host -ForegroundColor Cyan -Object "3 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout dossier public"
  Write-Host -ForegroundColor Cyan -Object "4 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout groupes"
  Write-Host -ForegroundColor Cyan -Object "5 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout salles"
  Write-Host -ForegroundColor Cyan -Object "6 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout équipements"
  Write-Host -ForegroundColor Cyan -Object "7 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Ajout contacts externes"
  Write-Host -ForegroundColor Cyan -Object "0 " -NoNewLine; Write-Host -Object "-" -NoNewline; Write-Host -
ForegroundColor Cyan -Object " Suppression client`n"

  $menu = Read-Host

  switch ($menu) {
    '1' {
      # Création société
      clear
      Write-Host -ForegroundColor Cyan -Object " [1/7] Ajout d'un client:`n"
      Write-Host -ForegroundColor Cyan -Object "[1/2] Indiquez le nom du client: (laissez vide pour annuler)"
      $clientname = Read-Host
      if ($clientname -eq "") {break}

      if (Get-AcceptedDomain -Filter "name -like '$clientname'") {
        Write-Host -ForegroundColor Red -Object "Le client $clientname existe déjà, retour au menu..."
        Read-Host
        break
      }

      Write-Host -ForegroundColor Cyan -Object "[2/2] Indiquez le nom de domaine du client:"
      $domainname = Read-Host
      if ($domainname -eq "") {
        while ($domainname -eq "") {
```

```
            Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
            $domainname = Read-Host
        }
    }

    $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
    clear
    Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :
    "
    Write-Host -ForegroundColor Cyan -Object "Nom du client:      " -NoNewline; Write-Host -Object $clientname;
    Write-Host -ForegroundColor Cyan -Object "Nom de domaine:     " -NoNewline; Write-Host -Object
$domainname;

    Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
    $check = Read-Host

    if ($check -match "[vV]") {
        Write-Host -ForegroundColor Cyan -Object "`nAjout du client $clientname en cours" -NoNewline; Write-Host
-ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -NoNewline;
Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

        if (-not (& '.\1_multitenant_company.ps1' -clientname $clientname -domainname $domainname)) {
            Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
            Read-Host
            break
        }

        Write-Host -ForegroundColor Green -Object "Le client à correctement été ajouté !"
        Read-Host
    }
    break
}

'2' {
    # Ajout utilisateurs
    do {
        clear
        Write-Host -ForegroundColor Cyan -Object " [2/7] Ajout des utilisateurs:`n"

        Write-Host -ForegroundColor Cyan -Object "[1/5] Indiquez le nom du client: (laissez vide pour terminer)"
        $clientname = Read-Host

        if (-not ($clientname -eq "")) {
            if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
                Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
                Read-Host
                break
            }
            Write-Host -ForegroundColor Cyan -Object "[2/5] Indiquez un prénom:"
            $firstname = Read-Host
            if ($firstname -eq "") {
                while ($firstname -eq "") {
                    Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                    $firstname = Read-Host
                }
            }
            Write-Host -ForegroundColor Cyan -Object "[3/5] Indiquez un nom:"
            $lastname = Read-Host
            if ($lastname -eq "") {
                while ($lastname -eq "") {
                    Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                    $lastname = Read-Host
                }
            }
            Write-Host -ForegroundColor Cyan -Object "[4/5] Indiquez un alias:"
```

```
            $alias = Read-Host
            if ($alias -eq "") {
               while ($alias -eq "") {
                  Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                  $alias = Read-Host
               }
            }

            Write-Host -ForegroundColor Cyan -Object "[5/5] Indiquez un mot de passe:"
            $password = Read-Host
            if ($password -eq "") {
               while ($password -eq "") {
                  Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                  $password = Read-Host
               }
            }

            $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
            $firstname = (Get-Culture).TextInfo.ToTitleCase($firstname)
            $lastname = (Get-Culture).TextInfo.ToTitleCase($lastname)
            clear
            Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :`n"
            Write-Host -ForegroundColor Cyan -Object "Nom du client:              " -NoNewline; Write-Host -Object
$clientname;
            Write-Host -ForegroundColor Cyan -Object "Prénom utilisateur:         " -NoNewline; Write-Host -Object
$firstname;
            Write-Host -ForegroundColor Cyan -Object "Nom utilisateur:            " -NoNewline; Write-Host -Object
$lastname;
            Write-Host -ForegroundColor Cyan -Object "Alias utilisateur:          " -NoNewline; Write-Host -Object
$alias;
            Write-Host -ForegroundColor Cyan -Object "Mot de passe utilisateur:   " -NoNewline; Write-Host -Object
$password;

            Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
            $check = Read-Host

            if ($check -match "[vV]") {
               Write-Host -ForegroundColor Cyan -Object "`nAjout de l'utilisateur $alias en cours" -NoNewline; Write-
Host -ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

               if (-not (& '.\2_multitenant_mailbox.ps1' -clientname $clientname -firstname $firstname -lastname
$lastname -alias $alias -password $password)) {
                  Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
                  Read-Host
                  break
               }

               & '.\8_multitenant_update.ps1' -clientname $clientname
               Write-Host -ForegroundColor Green -Object "L'utilisateur à correctement été ajouté !"
               Read-Host
            }
         }
      } until ($clientname -eq "")
      break
   }

   '3' {
      # Ajout dossier public
      clear
      Write-Host -ForegroundColor Cyan -Object " [3/7] Ajout de dossier public: (facultatif)"
      Write-Host -ForegroundColor Cyan -Object "Indiquez le nom du client pour qui créer un dossier public: (laissez
vide pour annuler)"
      $clientname = Read-Host
      if ($clientname -eq "") {break}
```

```
        $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)

        if (Get-Mailbox -Filter "name -like '$clientname*'" -PublicFolder) {
            Write-Host -ForegroundColor Red -Object "Le dossier publique existe déjà pour le client $clientname, retour au menu..."
            Read-Host
            break
        }

        if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
            Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
            Read-Host
            break
        }

        Write-Host -ForegroundColor Cyan -Object "`nAjout du dossier public en cours" -NoNewline; Write-Host -ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

        if (-not (& '.\3_multitenant_publicfolder.ps1' -clientname $clientname)) {
            Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
            Read-Host
            break
        }

        & '.\8_multitenant_update.ps1' -clientname $clientname
        Write-Host -ForegroundColor Green -Object "Le dossier publique à correctement été ajouté !"
        Read-Host
        break
    }

    '4' {
        # Ajout groupes
        do {
            clear
            Write-Host -ForegroundColor Cyan -Object " [4/7] Ajout de groupes:`n"

            Write-Host -ForegroundColor Cyan -Object "[1/2] Indiquez le nom du client: (laissez vide pour terminer)"
            $clientname = Read-Host

            if (-not ($clientname -eq "")) {
                if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
                    Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
                    Read-Host
                    break
                }
                Write-Host -ForegroundColor Cyan -Object "[2/2] Indiquez un alias:"
                $alias = Read-Host
                if ($alias -eq "") {
                    while ($alias -eq "") {
                        Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                        $alias = Read-Host
                    }
                }

                $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
                clear
                Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :`n"
                Write-Host -ForegroundColor Cyan -Object "Nom du client:       " -NoNewline; Write-Host -Object $clientname;
                Write-Host -ForegroundColor Cyan -Object "Alias du groupe:     " -NoNewline; Write-Host -Object $alias;

                Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
                $check = Read-Host
```

```
            if ($check -match "[vV]") {
                Write-Host -ForegroundColor Cyan -Object "`nAjout du groupe $alias en cours" -NoNewline; Write-Host
-ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

                if (-not (& '.\4_multitenant_groups.ps1' -clientname $clientname -alias $alias)) {
                    Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
                    Read-Host
                    break
                }

                & '.\8_multitenant_update.ps1' -clientname $clientname
                Write-Host -ForegroundColor Green -Object "Le groupe à correctement été ajouté !"
                Read-Host
            }
        }
    } until ($clientname -eq "")
    break
}

'5' {
    # Ajout salles
    do {
        clear
        Write-Host -ForegroundColor Cyan -Object " [5/7] Ajout de salles:`n"

        Write-Host -ForegroundColor Cyan -Object "[1/2] Indiquez le nom du client: (laissez vide pour terminer)"
        $clientname = Read-Host

        if (-not ($clientname -eq "")) {
            if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
                Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
                Read-Host
                break
            }
            Write-Host -ForegroundColor Cyan -Object "[2/2] Indiquez un numéro de salle:"
            $number = Read-Host
            if ($number -eq "") {
                while ($number -eq "") {
                    Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                    $number = Read-Host
                }
            }

            $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
            clear
            Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :`n"
            Write-Host -ForegroundColor Cyan -Object "Nom du client:       " -NoNewline; Write-Host -Object
$clientname;
            Write-Host -ForegroundColor Cyan -Object "Numéro de salle:     " -NoNewline; Write-Host -Object
$number;

            Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
            $check = Read-Host

            if ($check -match "[vV]") {
                Write-Host -ForegroundColor Cyan -Object "`nAjout de la salle $number en cours" -NoNewline; Write-
Host -ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

                if (-not (& '.\5_multitenant_room.ps1' -clientname $clientname -number $number)) {
                    Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
                    Read-Host
                    break
                }
```

```
                & '.\8_multitenant_update.ps1' -clientname $clientname
                Write-Host -ForegroundColor Green -Object "La salle à correctement été ajouté !"
                Read-Host
            }
        }
    } until ($clientname -eq "")
    break
}

'6' {
    # Ajout équipements
    do {
        clear
        Write-Host -ForegroundColor Cyan -Object " [6/7] Ajout d'équipements:`n"

        Write-Host -ForegroundColor Cyan -Object "[1/2] Indiquez le nom du client: (laissez vide pour terminer)"
        $clientname = Read-Host

        if (-not ($clientname -eq "")) {
            if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
                Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
                Read-Host
                break
            }
            Write-Host -ForegroundColor Cyan -Object "[2/2] Indiquez un nom d'équipement:"
            $name = Read-Host
            if ($name -eq "") {
                while ($name -eq "") {
                    Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                    $name = Read-Host
                }
            }

            $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
            clear
            Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :`n"
            Write-Host -ForegroundColor Cyan -Object "Nom du client:        " -NoNewline; Write-Host -Object
$clientname;
            Write-Host -ForegroundColor Cyan -Object "Nom d'équipement:     " -NoNewline; Write-Host -Object
$name;

            Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
            $check = Read-Host

            if ($check -match "[vV]") {
                Write-Host -ForegroundColor Cyan -Object "`nAjout de l'équipement $name en cours" -NoNewline;
Write-Host -ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

                if (-not (& '.\6_multitenant_equipement.ps1' -clientname $clientname -name $name)) {
                    Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
                    Read-Host
                    break
                }

                & '.\8_multitenant_update.ps1' -clientname $clientname
                Write-Host -ForegroundColor Green -Object "L'équipement à correctement été ajouté !"
                Read-Host
            }
        }
    } until ($clientname -eq "")
    break
}
```

```
    '7' {
        # Ajout contacts
        do {
            clear
            Write-Host -ForegroundColor Cyan -Object " [7/7] Ajout de contacts:`n"

            Write-Host -ForegroundColor Cyan -Object "[1/2] Indiquez le nom du client: (laissez vide pour terminer)"
            $clientname = Read-Host

            if (-not ($clientname -eq "")) {
                if (-not (Get-AcceptedDomain -Filter "name -like '$clientname'")) {
                    Write-Host -ForegroundColor Red -Object "Le client $clientname n'existe pas encore, retour au menu..."
                    Read-Host
                    break
                }
                Write-Host -ForegroundColor Cyan -Object "[2/4] Indiquez un nom de contact:"
                $name = Read-Host
                if ($name -eq "") {
                    while ($name -eq "") {
                        Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                        $name = Read-Host
                    }
                }
                Write-Host -ForegroundColor Cyan -Object "[3/4] Indiquez un alias:"
                $alias = Read-Host
                if ($alias -eq "") {
                    while ($alias -eq "") {
                        Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                        $alias = Read-Host
                    }
                }
                Write-Host -ForegroundColor Cyan -Object "[4/4] Indiquez l'adresse mail:"
                $emailaddress = Read-Host
                if ($emailaddress -eq "") {
                    while ($emailaddress -eq "") {
                        Write-Host -ForegroundColor Red -Object "Veuillez entrer une valeur:`n"
                        $emailaddress = Read-Host
                    }
                }

                $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
                $name = (Get-Culture).TextInfo.ToTitleCase($name)
                clear
                Write-Host -ForegroundColor Cyan -Object "Veuillez vérifier les valeurs saisies :`n"
                Write-Host -ForegroundColor Cyan -Object "Nom du client:        " -NoNewline; Write-Host -Object
$clientname;
                Write-Host -ForegroundColor Cyan -Object "Nom du contact:       " -NoNewline; Write-Host -Object $name;
                Write-Host -ForegroundColor Cyan -Object "Alias du contact:     " -NoNewline; Write-Host -Object $alias;
                Write-Host -ForegroundColor Cyan -Object "E-mail du contact:    " -NoNewline; Write-Host -Object
$emailaddress;

                Write-Host -ForegroundColor Cyan -Object "`n`nPour valider ces données, entrez V."
                $check = Read-Host

                if ($check -match "[vV]") {
                    Write-Host -ForegroundColor Cyan -Object "`nAjout du contact $alias en cours" -NoNewline; Write-Host
-ForegroundColor Cyan -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Cyan -Object "." -
NoNewline;Sleep 1; Write-Host -ForegroundColor Cyan -Object ".";

                    if (-not (& '.\7_multitenant_contacts.ps1' -clientname $clientname -name $name -alias $alias -emailaddress
$emailaddress)) {
                        Write-Host -ForegroundColor Red -Object "Une erreur est survenue, retour au menu..."
                        Read-Host
                        break
                    }
```

```
                & '.\8_multitenant_update.ps1' -clientname $clientname
                Write-Host -ForegroundColor Green -Object "Le contact à correctement été ajouté !"
                Read-Host
            }
        }
    } until ($clientname -eq "")
    break
}

'0' {
    # Supression de client
    clear
    Write-Host -ForegroundColor Cyan -Object " Suppression de client:`n"

    Write-Host -ForegroundColor Cyan -Object "Vous pouvez ici supprimer tous les fichiers qui définissent un client."
    Write-Host -ForegroundColor Cyan -Object "Indiquez le nom du client: (laissez vide pour annuler)"
    $clientname = Read-Host

    if (-not ($clientname -eq "")) {
        $clientname = (Get-Culture).TextInfo.ToTitleCase($clientname)
        clear
        Write-Host -ForegroundColor Red -Object "Êtes-vous certain de vouloir supprimer le client $clientname ?`n"

        Write-Host -ForegroundColor Cyan -Object "`nPour continuer, entrez V."
        $check = Read-Host

        if ($check -match "[vV]") {
            Write-Host -ForegroundColor Red -Object "`nSuppression du client $clientname en cours" -NoNewline; Write-Host -ForegroundColor Red -Object "." -NoNewline; Sleep 1; Write-Host -ForegroundColor Red -Object "." -NoNewline;Sleep 1; Write-Host -ForegroundColor Red -Object ".";

            & '.\9_multitenant_removecompany.ps1' -clientname $clientname
            Write-Host -ForegroundColor Green -Object "Toutes les entrées associées au nom $clientname ont été supprimées !"
            Read-Host
        }
    }
    break
}

Default {
    & '.\8_multitenant_update_all.ps1'
    exit 0
}
}

}
```

4. Rapport d'étude de la configuration des switchs

TP Link T2600G-52TS

Connexion au switch via interface web : 192.168.0.1
La dernière version du firmware est déjà installée.
Charge du CPU : env. 30%. Charge mémoire RAM : env. 80%
Il est possible de mettre en place de la QOS dans le menu QOS. Elle s'applique sur les ports avec 8 niveaux de priorité possibles.
Il est possible de programmer un redémarrage automatique dans le menu System --> System Tools --> Reboot Schedule. Il faut alors choisir entre un reboot à intervalle de temps ou (à priori)

occasionnel, à une date et heure fixée (à tester pour en être sûr). Ne pas oublier de cocher Save Before Reboot pour que la planification de redémarrage soit enregistrée après le reboot.
Via le test de câbles qui est proposé par le switchs, des câbles associés à certains ports sont défaillants. Les ports **8**, **9**, **24**, **27**, **32**, **34**, **45**, **48** ont des câbles défaillants sur la paire C et D (blanc bleu/bleu, blanc marron/marron). Les ports **17**, **18** ont des câbles défaillants sur la paire D (blanc marron/marron) et le port **33** à un câble défaillant sur la paire A, C et D (blanc orange/orange, blanc bleu/bleu et blanc marron/marron). La paire A est essentielle à la bonne communication entre 2 équipements contrairement aux paires C et D qui ne sont généralement pas utilisées.
Tous les ports cités, à part le **8** et le **32**, communiquent en 100 Mb/s au lieu de 1000 Mb/s. A vérifier si cela vient du câble qui est de mauvaise qualité ou de l'équipement à l'extrémité qui limite la vitesse de transmission.
Les ports **22**, **35**, **40**, **44**, **46** se déconnectent/reconnectent occasionnellement. Vérifier que les équipements concernés prévoient des déconnexions ou que la déconnexion du réseau ne pose pas de problèmes (que l'équipement ne soit pas un pc à l'accueil).

Infos :
D'après les adresses MAC associées aux ports, il semblerait que le switch Netgear 48 ports soit connecté au port **7**.
D'après la longueur des câbles, les équipements connectés aux ports **1**, **2**, et **12** sont proches du switch.
Le port 12 est relié à la GEDBOX, le 41 au switch Netgear 5 ports, le 5 au ZyXEL (128.1.1.254), le **8** au ZyXEL (128.1.1.253) et le **9** au routeur Wifi (128.1.1.252).

Netgear GS748Tv4

Connexion via interface web : 128.1.1.62
Version du software : 5.4.1.11 Version disponible : 5.4.2.18 sur le site Netgear Pas d'infos dans le Release notes concernant une amélioration de performances.
La mise en place de QOS est possible avec des niveaux de priorité et des limites d'octets à appliquer sur les ports.
Via les logs, le port **36** subit des déconnexion/reconnexion fréquentes.
L'interface de configuration du Netgear est beaucoup moins complète que celle du TP Link.
Il est également possible que toutes les informations ne s'affichent pas (table d'adresse MAC notamment).

Infos :
D'après les adresses MAC associées aux ports, il semblerait que le switch TP Link 48 ports soit connecté au port 4.
1 Synology et 2 Mobotix sont connectés au port **24**, il semble que cela soit le switch 16 ports dédié aux caméras.
La longueur de câble estimée entre les 2 switch 48 ports est de 0m, et 2-3m entre les 2 Netgear.

Le switch 16 ports, le 8 ports et le 5 ports ne sont pas administrables. Le Netgear 5 ports à tout de même une adresse IP : 192.168.0.239.