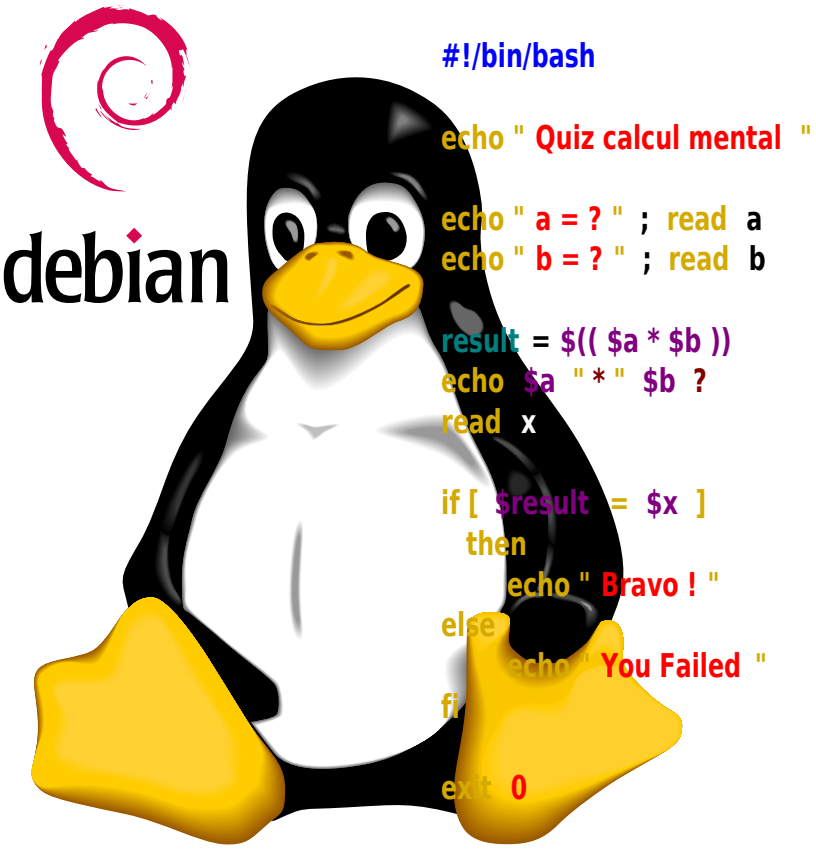
Dossier Projet BTS IRIS 2013
Projet Maths-Linux

NOM Prénom BTS IRIS 2ème année

Serveur de clients légers GNU/Linux Debian d'applications pour QCM mathématiques



Logo Debian	*Software in the Public Interest, Inc*	*1999*	*CC By-SA 3.0*
Tux	*Wikipedia.org (Auteur : Larry Ewing)*	*1996*	*Domaine public*

Table des matières

I) Roles

Noms	Roles
	Réaliser le logiciel produisant les exercices interactifs, en colaboration avec Madame Audibert ainsi que l'application élève
	Réaliser le logiciel permettant l'affichage en temps réel des scores et exercices de la compétition.
	Réaliser le socle applicatif de l'ensemble permettant de gérer les clients légers.Les clients légers seront réalisés avec des PC obsolètes (ou des cartes Raspberry Pi) On réalisera aussi une interface de gestion des participants pilotant la compétition.,

II) Analyse des Besoins

-A.Présentation

1.Le projet consiste en un système clients/serveur permettant aux professeurs d'écoles primaires de créer des exercices de maths, et aux élèves de s'entrainer de façon interactive (QCM), en comptabilisant les scores à la fin de chaque exercice, ainsi que d'effectuer une compétition, avec affichage des scores et classements sur un TBI (Tableau Blanc Intercatif).
Les exercices, scores, et informations sur les participants seront stockés dans des bases de données diffrentes.

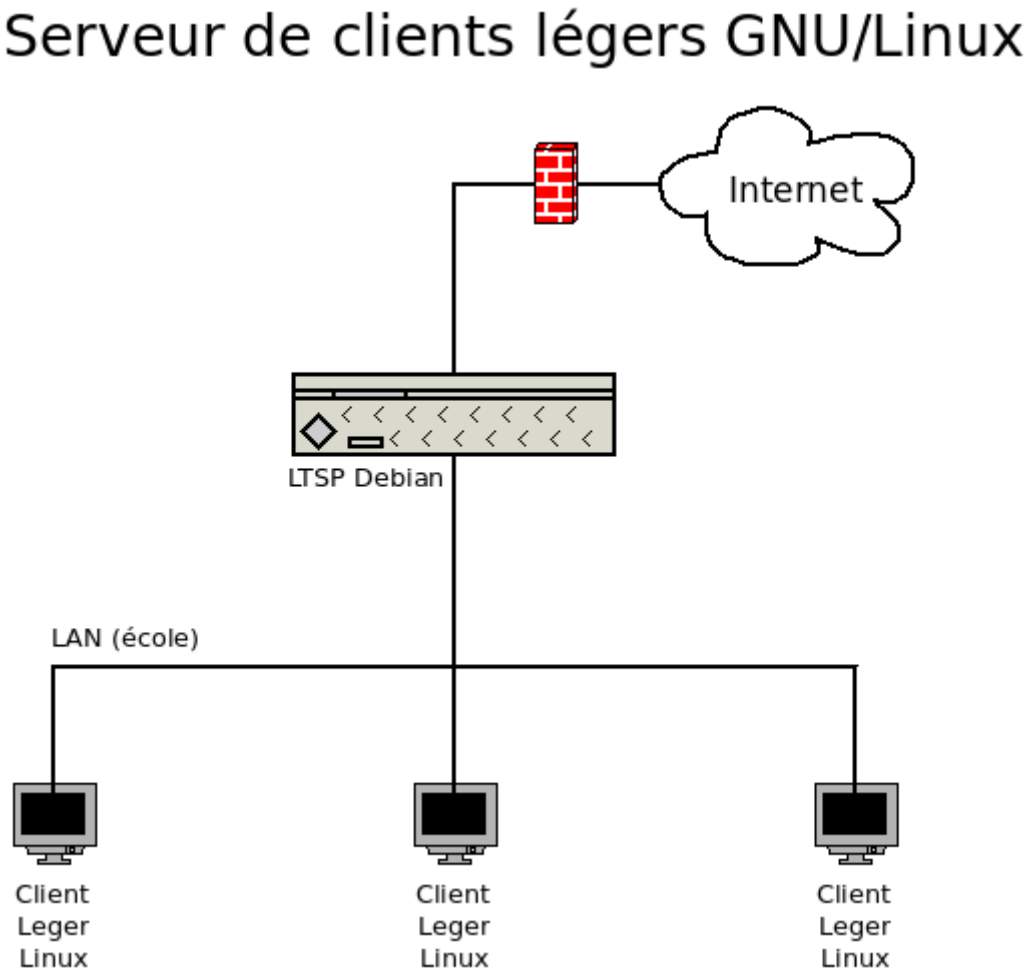
2.Le serveur
Le serveur utilise une distribution **Debian** devra heberger les bases de données, et fournir le système et les applications aux clients légers.

3.Les clients
Il s'agit d'anciens PC utilisés en tant que clients légers, qui permetteront aux élèves d'acceder aux applications pour effectuer les entrainements et la compétion.

-B.Principe de l'architecture clients légers/serveur

Toutes les applications seront éxecutés sur le serveur, l'intérêt est de pouvoir utilisé des machines de faibles puissance, anciennes et/ou sans disque dur, qui serviront uniquement d'interface pour les

différents utlisateurs (chacun son écran, clavier, souris) afin d'accéder aux ressources (matérielles et logicielles) du serveur



Dia : Thin Clients/Server architecture

III) Fonctions de services

-A.Fonctions Principales

Les logiciels developpés lors de ce projet deveront êtres faciles d'utilisation et permetteront de

-Créer les exercices QCM
-Gérer la liste des élèves
-Étudier les résultats des exercices
-Faire les exercices

-B.Fonctions Secondaires

-Sauvergarder les scores dans une base de données

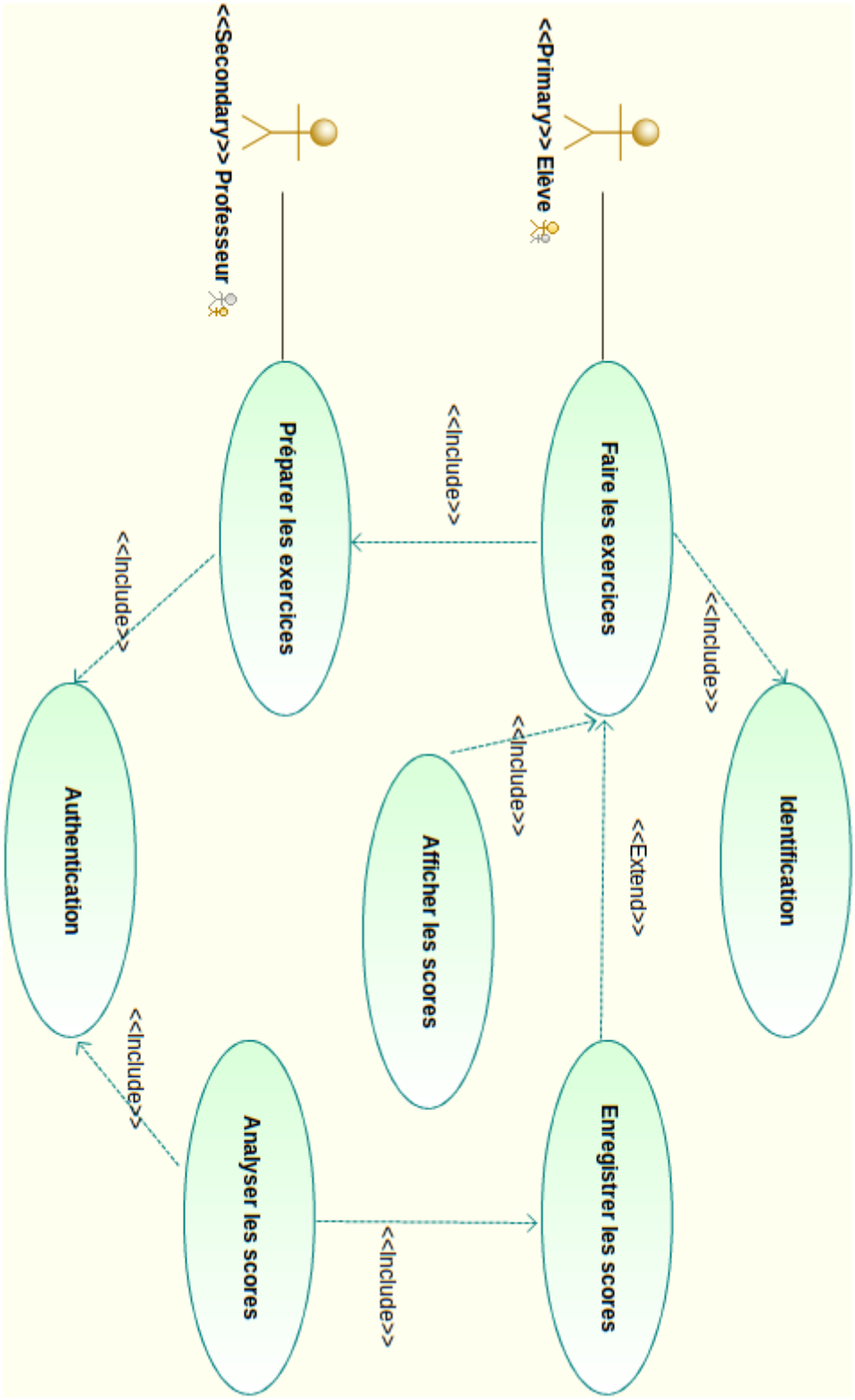
Eventuellement proposer une interactivité
-Changement de positionnement des bonnes réponses
-Modification automatique de mauvaises réponses à chaque répétition d'un même exercice

IV) Contraintes

-Coûts maitrisés
-Fonctionnement sur PC obsolètes (faible puissance)

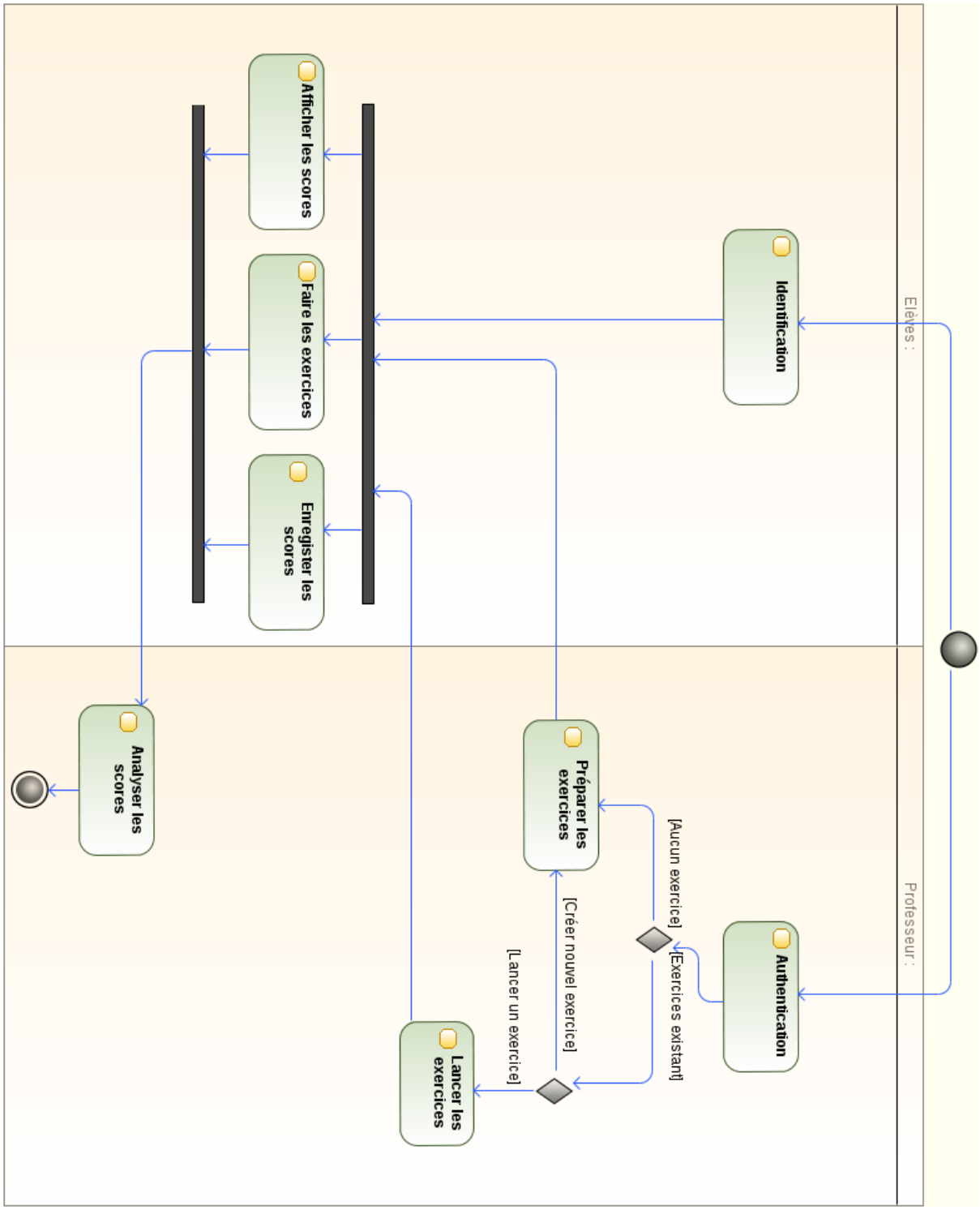
V) Specifications (UML)

-A.Cas d'Utilisation



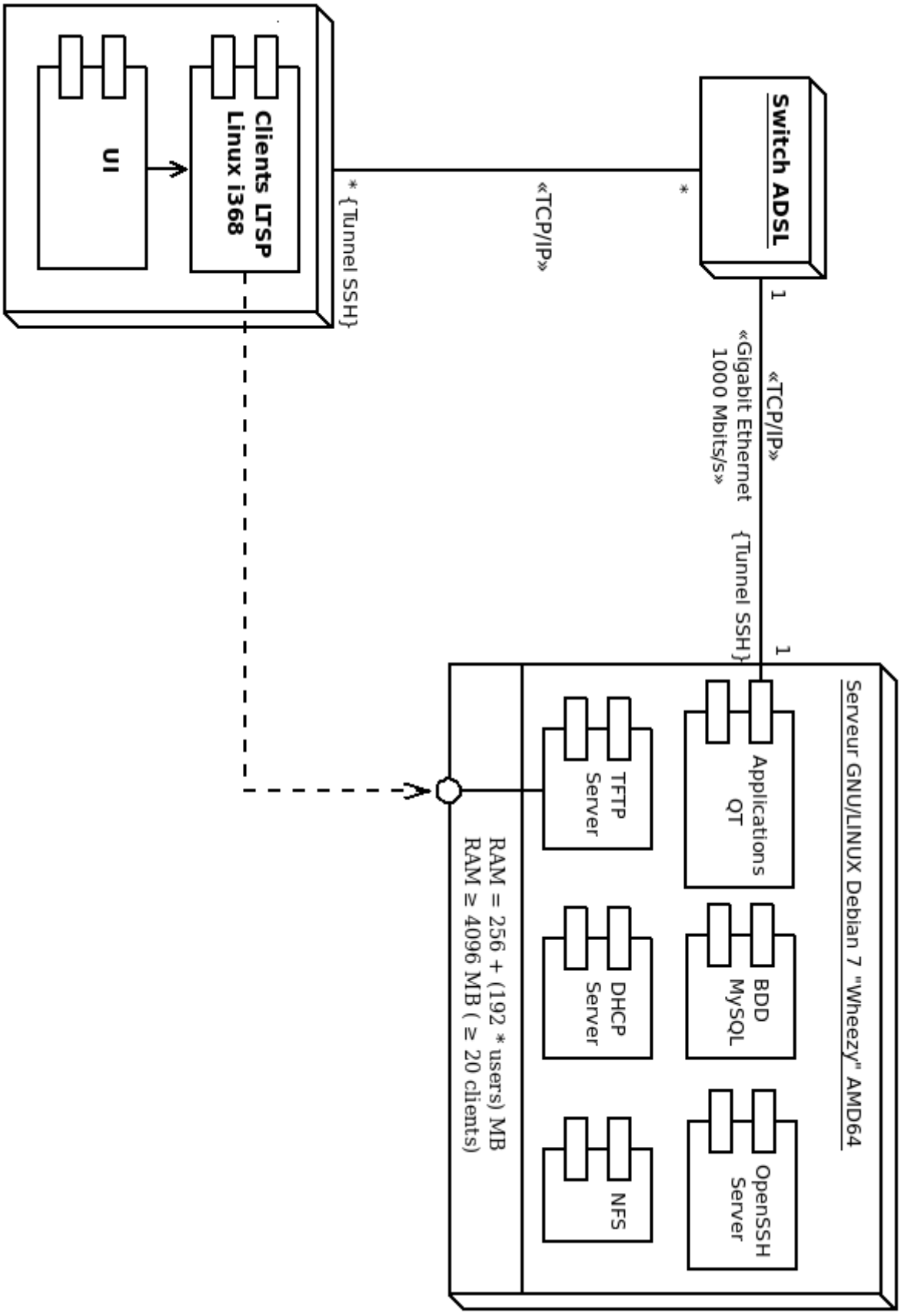
Modelio : Use Case Diagram

-B.Diagramme d'activité



Modelio : Activity Diagram

-C.Diagramme de déploiement



Dia : Deployment Diagram

VI) Conception architecturale

-A.Présentation

-Le système fonctionnera sur une architecture clients/légers, serveurs
Pour la gestion des clients, on utilisera Linux Terminal Server Project (LTSP), qui est une suite de logiciels libres permettant à plusieurs personnes d'utiliser les ressources matérielles et logiciels d'une même machine serveur à partir de clients légers

LTSP est composé de

- -Un serveur DHCP pour l'attribution d'adresses IP aux postes clients légers
- -Un serveur TFTP pour fournir les images systèmes aux clients légers
- -NFS : Network FileSystem et NBD Network Block Device (un au choix avec LTSP v5)
- -Serveur SSHv2 pour une communication sécurisée par chiffrement asymetrique bout à bout

-Pour des raisons de rapport qualité/prix, on s'orientera vers le logiciel libre, accessible gratuitement sans contrat de de support technique pour les particuliers, car pas de frais de licences à payer (support payant pour les entreprises)
Les logiciels libres ont, par rapport aux freewares (logiciels propriétaires gratuits), l'avantage d'êtres plus flexibles/évolutifs pour être utilisés dans un environnement professionnel (GNU/Linux, BSD, MySQL, QT, PHP, Apache...)

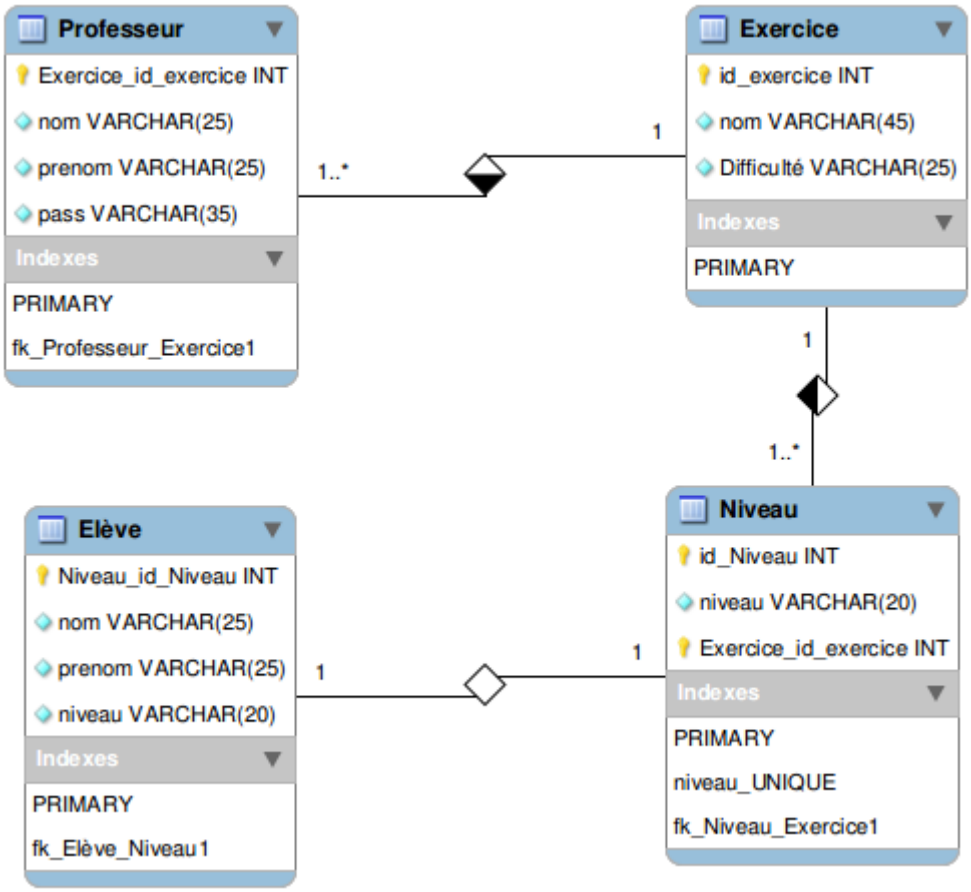
Le maintien de ces logciels étant assuré par une large communauté de developpeurs généralement très active, permise par le libre accès au code source, rendant l'évolution des logiciels indépendante de leurs créateurs initiaux, donc plus faciles à adapter à ses propres besoins, pour des coûts relativement faibles.
L'accès au code source permet aussi l'analyse du code par des parties tierces, facilitant la découverte et la correction de bugs/failles, améliorant la sécurité et la stabilité des logiciels libres encore maintenus, les avantages du logiciel libre sont donc multiples

Nos applications seront écrites en C++ sur l'IDE (Integrated Development Environment) **QT 4.8.2**, et compilées pour une distriibution **Linux Debian 7 « Wheezy »**, distribution réputée pour sa stabilité.

Le SGBD (Système de Gestion de Base de Données) utilisé sera **MySQL**, de Oracle et l'interfaçage avec **QT4** nécessite une librairie particulière, le paquet Linux correspondant est nommé **libqt4-sql-mysql**.

-B.Modélès de conceptions

-B.1.Modèles Entité-Relation Etendue



MySQL-Workbench : EER (Extended Entity–Rlationship)

VI.Conception Détaillée

-A.Préparation du matériel

Notre serveur aura besoin de deux interfaces réseaux et d'un switch

-Une permettant de connecter le serveur au réseau de l'école, pour accéder à internet dont l'IP sera attribué par le DHCP du LAN de l'école
Mais cette IP sera reservée au serveur par une association IP/adresse MAC dans les réglages du DHCP de l'école, afin de faciliter la configuration de la passerelle, permettant au clients légers d'acceder à Internet

Une autre avec une IP fixe en 10.0.0.1, qui permettera de connecter les clients légers au serveur via un switch
On commence donc par installer une deuxième carte Ethernet sur le serveur

-B.Installation et configuration du serveur Debian

-B.1.Préparation de Debian

Pour installer Debian, on doit d'abord connaître l'architecture de la machine sur laquelle on va installer Debian et on doit choisir une méthode d'installation (par réseau, CD/DVD d'installation, Live USB...)

Dans mon cas, il s'agit d'un PC avec un CPU 64 bit donc il faut utiliser l'image Debian amd64, on choisit une installation à partir d'une clé USB

On va donc récuperer l'image correspondante depuis le Serveur Debian
Une fois l'image récupée, disposant déjà d'une machine sous Linux (ou un autre système UNIX-like (BSD, OS X...), on utilisera l'outil dd pour la copier vers une clé USB
Une fois la clé USB branchée, identifie le « device » correspondant à notre clé USB graçe à la commande **df-h** ou sudo **fdisk-l**

df permet d'ontenir une liste de toutes partitions avec la mémoire utilisée et la mémoire libre
L'option-**h** ou-**-human-readable** permet de faciliter la lecture des capacités mémoires en affichant des valeurs en puissances 1024 (MebiByte, GibiByte...) plutôt que des valeurs en octets
Note :-**H** ou –**si** affichera des valeurs en puissance de 10, comme pour le système international (MegaByte, GigaByte...)
Dans le cas ou en a plusieurs supports de stockage de taille differentes branchés en même temps, on identifiera le bon support graçe à sa capacité mémoire

fdisk est un outil de partitionnement en ligne de commande, et l'option-**l** permet de lister les toutes partitions presentes sur le systèmes, y compris les volumes externes montées, tout comme pour **df**

Une fois la clé USB formellement identifiée, on se place dans le dossier contenant l'image de Debian et on utilise la commande **dd**, soit

user@pc:/home/user # **sudo dd if=<nom_image> of=/dev/sdX**

En remplaçant X par la lettre correspondante à notre clé USB, a étant déjà prise par le disque dur interne de la machine en cours d'utilisation, la lettre peut être b ou autre si b est déjà pris par un autre sopport de stockage
Dans le cas d'une installation à partir d'une carte mémoire, lue et écrite à partir d'un lecteur de carte interne, le nom du « device » pourra être sous la forme **mmcblkX**

Dans le cas de Debian 7.2 avec le bureau KDE, et la clé USB associée au device sdb, on obtient

user@pc:/home/user # ***sudo dd if=debian-live-7.2-amd64-lxde-desktop of=/dev/sdb***

On patiente jusqu'à la fin de la copie, et pendant ce temps, si ce n'est pas déjà fait, on active le boot sur support USB sur la machine qu'on utilisera comme serveur
Ce réglage dependant du BIOS/UEFI, donc de la machine/carte mère utilisée, on abordera pas ce sujet dans ce document

On démarrage sur la clé USB pour commencer l'installation, évidemment la version de Debian à été mise à jour depuis l'instatllation

Il suffit de suivre les instructions à l'écran pour installer Debian, quelques bases en systèmes GNU/Linux facilitent la tâches, l'opération reste cependant accessible aux novices, en utilisant eventuellement un tutoriel

-B.2.Verification et configuration des cartes réseaux

Une fois l'installation de Debian finie, on verifie bien avec la commande **ifconfig** pour voir si les deux interfaces sont reconnues, on doit donc avoir *eth0* et *eth1 (*en plus *de lo/loopback : bouclei locale*)

Dans les rares cas, ou une des deux cartes Ethernet n'est pas reconnue, on utilise **lspci** pour lister toutes les cartes PCI, avec les noms de fabricants et de circuits intégrés dont sont équippés les differentes cartes, afin de déterminer le driver nécessaire

Pour faciliter l'identification des cartes, on peut utiliser le filtre **grep** avec la commande **lspci**, ce qui donne **lspci | grep-i ether** afin de n'afficher que les cartes dont la ligne correspondante contient le mot ethernet, l'option-i permet d'ignorer la case (c'est à dire les differences entre majuscules et miuscules)

Attention un seul model (nom commercial) d'une même carte (Ethernet ou autre) ou d'un périph USB (**lsusb**) WiFi/Bluetooth peut-être dérivé en plusieurs versions, utilisants différents circuits et nécessitant potentiellement differents drivers en fonction de driver, d'ou l'importance de passer par la commande **lspci** et ne pas se contenter du nom commercial des périphériques qui fait souvent abstraction des circuits internes

L'erreur du choix du drivers ou la cohabitation entre differents drivers est tolérée pour ce type de périphériques, mais se limiter au nom commercial ne permet pas de trouver (rapidement) le bon driver
On obtiens donc le résultat suivant

*root@debianserver:/home/tuxmaths # **lspci | grep-i ether***
02:00.0 Ethernet controller: Atheros Communications Inc.AR8151 v2.0 Gigabit Ethernet (rev c0)
04:01.0 Ethernet controller: VIA Technologies, Inc.VT6102 [Rhine-II] (rev 42)

On sait que nous avons une première carte fabriquée par ***Atheros*** et équipée d'un circuit ***AR8151***, reconnue automatiquement et fonctionnelle au démarrage de Debian dans mon cas
Et une deuxième carte fabriquée par ***VIA Technologies***, et équipée d'un circuit ***VT6102 (Rhine-II)***, dont le bon driver n'a pas été chargée automatiquement au démarrage de Debian

L'étape suivante consiste à déterminer graçe à une recherche sur Internet quel est le nom du driver Linux/Debian pour les cartes équipés du circuit *Rhine II*, et s'il inclut dans la version du noyau **Linux** utilisée (3.2.0 dans mon cas) ou s'il faut l'installer manuellement et comment procéder

Dès fois on dispose que du drivers mais il faut ajouter manuellement le firmware correspondant à son périphérique (fichier en .fw généralement situé dans /lib/firmware)
Dans mon cas, tout ce dont j'avais besoin pour cette carte était inclus dans mon installation actuelle, mais le bon module n'était pas chargé automatiquement
Une fois le nom du module déterminé, ici **via-rhine**, on utilise la commande **modprobe**

<modname> soit ***modprobe via-rhine*** pour charger (ou décharger avec l'option-r) un module du noyau à chaud

L'utilisation de la touche tab permet d'utiliser l'autocomplemention pour voir si le bon module est présent ou si on s'est pas trompé dans l'orthographe du nom, si le nom est complété, ça veut dire que le module est bien present, en validant la commande, on devrait donc pas avoir d'erreurs en retour

En cas de problèmes « exotiques » ou manque d'experience en installation/configurations de systèmes UNIX/Linux, dispose de plusieurs sources d'informations, notamment les forums/wikis/salons IRC dédiés à la distribution Linux choisie
On va ensuite éditer le fichier /etc/network/interfaces afin de fixer l'adresse de eth1, on y rajoutera les lignes suivantes, attention aux erreurs d'orthographe/syntaxe !

vim /etc/network/interfaces

```
auto eth1               # Specifie au systeme « d'activer » (ifconfig iface up) automatiquement eth1
iface eth1 inet static  # eth1 utilise une IP statique

address 10.0.0.1            # Définit l'IP de eth1
netmask 255.0.0.0           # Définit le masque-réseau de eth1, ici ip/8
network 10.0.0.0            # Définit l'adresse du sous-réseau
broadcast 10.0.0.255        # Définit l'IP de défusion générale
gateway 192.168.1.1         # Définit la passerelle, ici l'IP reservé à l'interface eth0
```

Ne pas oublier d'activer le routage sur le serveur, graçe à la commande suivante (en root)
echo 1>/proc/sys/net/ipv4/ip_forward cas d'un réseau en IP v4

-C.Installation et configuration de LTSP 5

-C.1.Installation de LTSP et configuration réseau

On commence par installer Linux Terminal Server Project, on a le soit entre tout mettre sur même serveur (ltsp-server-standalone), ou séparer les differents paquets (TFTP, DHCP...) en installant LTPS tout seul (ltsp-server) sur un serveur, et les autres paquets séparemment sur les autres serveurs

On préferera la méthode standalone dans notre cas, car on dispose que d'un seul serveur

```
user@debianserver ~ $ su
root@debianserver ~ # apt-get install ltsp-server-standalone mysql-server
root@debianserver ~ # ltsp-build-client --arch i386
root@debianserver ~ # vim /etc/dhcp3
```

Ajouter (touche i) la ligne include "/etc/ltsp/dhcpd.conf";
Valider (echap :wq)

```
root@debianserver ~ # vim /etc/ltsp/dhcpd.conf
```

Ajouter (et adapter au réseau privé voulu) les lignes suivantes

```
authoritative;

subnet 10.0.0.0 netmask 255.0.0.0 {
        range 10.0.0.2 10.0.0.254;
```

```
        option domain-name-servers 10.0.0.1;
        option broadcast-address 10.0.0.255;
        option routers 10.0.0.1;
        Next-server 10.0.0.1; # Adresse TFTP

        option subnet-mask 255.0.0.0;
        option root-path "/opt/ltsp/i386";
        if substring( option vendor-class-identifier, 0, 9 ) = "PXEClient" {
                filename "/ltsp/i386/pxelinux.0";
        } else {
                filename "/ltsp/i386/nbi.img";
        }
}
```

```
root@debianserver ~ # invoke-rc.d dhcp3-server restart
root@debianserver ~ # vim /etc/exports
/opt/ltsp *(ro,no_root_squash,async,no_subtree_check)
```

Redémarrer nfs-kernel-server

```
root@debianserver ~ # invoke-rc.d nfs-kernel-server restart
root@debianserver ~ # vim /etc/default/tftpd-hpa
```

```
RUN_DAEMON="yes" # Exécution automatique
TFTP_USERNAME="tftp" # Nom d'utilisateur par défaut
TFTP_DIRECTORY="/srv/tftp" # Repertoire des images systèmes dans le TFTP
TFTP_ADDRESS="10.0.0.1:69" # IP de l'interface TFTP vers les clients légers
TFTP_OPTIONS="--secure" # Chiffrement par tunnel SSH
```

```
root@debianserver ~ # vim /etc/inetd.conf
#tftp    dgram          udp            wait    root    /usr/sbin/in.tftpd /usr #commenter
```

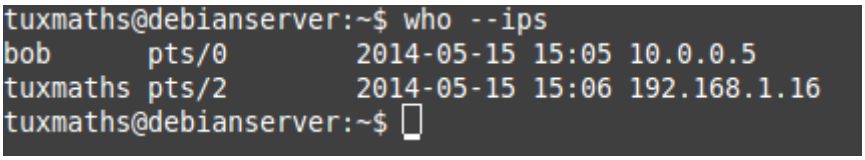
Redémarrer les DAEMON[1] TFTP et inet pour prendre en compte les modifications effectuées dans les fichiers de configurations

```
root@debianserver ~ # invoke-rc.d openbsd-inetd restart
root@debianserver ~ # invoke-rc.d tftpd-hpa restart
```

Il faut ensuite accéder au BIOS des machines clients, et activer le boot sur le réseau/PXE valider et démarrer le système une fis les clients connecter au switch branché sur la carte réseau du serveur eth1 (IP = 10.0.0.1)

Le client devrait booter sur une image Debian, crée sur le serveur, et avoir accès aux comptes crées sur le serveurs comme l'indique l'image suivante

1 Disk And Execution MONitor: programme s'excutant en tâche de fond,, sous les sytêmes UNIX-like (Linux, BSD, OS X...), le nom des services correpondants aux DAEMON finit souvent en *d (inetd, ftpd, httpd...)



L'utilisateur bob est connecté en tant que client léger
L'utilisateur tuxmaths (moi) est connecté en SSH afin d'adminitrer le serveur
Cette image à été prise après l'éxecution du script de créations d'utilisateurs

-C.2.Créations des utilisateurs LTSP

L'étape suivante consite justement à créer un script shell afin d'automatiser la création d'utilisateurs à partir d'une liste contenu dans un fichier texte userlist

Les utilisateurs étant des enfants, on n'utilisera pas d'outils les invitant à créer leurs mots de passes eux-mêmes, à la première connexion, des mots de passes seront prédeterminés, et contenus dans ledit fichier

NOTE IMPORTANTE : L'administrateur devra s'assurer du maintien de la confidentialité de ce userlist, ainsi que du fichier users.log crée par le script aussi longtemps qu'il en aura besoin
Par conséquent il ne devra stocker ce userlist et n'executer le script que depuis un répertoire auquel il est le seul à avoir accès, et donner à ces fichiers les droits d'accès adéquats (**chmod** et **chown**) car les mots de passes contenus dans ces fichiers sont en claires
Les mots de passes sont stockés ainsi à des fins pratiques mais ça poserai évidemment des problèmes de sécurité en cas de fuite desdits fichiers
Le userlist devra se trouver dans le même repertoire que le script à des fins pratiques
Le users.log servira à des fins debugging uniquement et sera detruit ainsi que userlist avec les noms et mots de passes après chaque vérification du bon fonctionnement de nouveaux utilisateurs
Pour effacer un fichier de façon sécurisée, on utilisera la commande **shred -vuzn <nombre_entier>** <nomdefichiers>

v → Mode verbeux, montrer la progression de la tâche
u → tronquer et effacer le fichier après l'écrasement
z → remplir la zone mémoire du fichiers avec des zero afin de cacher l'éffacement sécurisé (shredding)
n → Nombre d'itérations de l'opération

En cas de perte de mots de passe par un élève, l'administrateur devra changer le mot de passe de l'utilisateur concerné en utilisant **passwd <username>**, en remplaçant **<username>** par le vrai nom d'utilisateur concerné

La syntaxe de userlist sera :
nom1 motdepasse_en_claire1 eleve
nom2motdepasse_en_claire2 prof
...

La syntaxe de users.log sera :

nom_utilisateur1
mot_de_passe1
eleve

nom_utilisateur2
mot_de_passe2
prof
…

Les chiffres indiquent differents utilisateurs, et le groupe secondaire correspond au type d'utilisateurs, professeurs ou élèves, afin de pouvoir attribuer des droits differents en fonctions des deux groupes, plutôt que par utilisateurs au cas par cas (plus lent et moins pratique)
Attention, les noms ne doivent pas contenir de majuscules ni d'accents et les espaces sont reservés comme caratères pour séparer les trois champs, donc les espaces dans les noms, les mots de passe seront subtitués par _ (caratère underscore)
Exemple : nom_prenom

Script create_users_v1.sh

```
#************************************************************
# Nom: create_users_v1.sh
# Auteur: me
# Version: 1
#************************************************************

#!/bin/bash

# set -x
select=0

for i in `cat userlist`

do
 case $select in
  0 ) u=$i;;
  1 ) p=$i;;
  2 ) g=$i;;
 esac
 ((select=select+1))

 if [ $select -eq 3 ]
 then
  echo $u >> users.log
  echo $p >> users.log
  echo $g >> users.log
  ((select=0))

  useradd -m -G $g -p $(mkpasswd -m sha-512 $p) $u
 fi

done
exit 0 # fin du script
```

La ligne commentée # set -x permet d'afficher le deroulement de l'opération (à des fins de débuuging) si on la décommente

Ce script est constitué d'une boucle for permettant de lire toutes les lignes d'un fichier nommé userlist
Pour chaque champs de chaque ligne, il stocke l'information dans la variable correspondante
Le premier champs, nom d'utilisateur est stocké dans la variable $u, le second champs, le mot de passe dans la variable $p et le troième champs, nom du groupe secondaire dans la variable $g
Avant de passer à la ligne suivante, il stocke les informations dans un fichier users.log, réinitialise le selecteur de champs (pour pouvoir traiter la ligne suivante) et utilise l'outil useradd pour créer les comptes utilisateurs, useradd verifie automatiquement que l'utilisateur crée n'existe pas déjà

Il prend comme paramètre

-m crée un dossier /home/username
-G spécifie le nom d'un groupe secondaire
-p mot de passe chiffré par l'outil mkpasswd

Le mot de passe étant contenu en claire dans la variable $p, il faudra le chiffré et le remettre dans une autre variable $(mkpasswd $p) qui sera utilisée comme argument à l'option -p

Quand on crée un compte utilisateur sous un système GNU/Linux en utilisant l'interface graphique, par défaut le mot de passe est stocké après en utilisant un algorithme de hashage[2]
SHA-512 (Secure Hash Algorithm 2/512 bits)
La commande ***sudo cat /etc/shadow*** montre que les hash de mots de passes en SHA-512 sont précédés d'un préfixe **6**, même si ce fichier est accessible n'est accessible qu'à l'admin, un algorithme fiable permet une meilleure protection des mots de passes, notamment en cas de faille permettant une élévation des privilèges

Quelques exemples de préfix **id** et les algorithmes de hashage correspondants
1 → **MD5** Victime de collisions
2 → **BlowFish et dérives ($2a$, $2b$, $2x$, $2y$)** Obsolète
3 → **NT Hash (basé sur MD4)** Obsolète, notation sous FreeBSD
5 → **SHA-2/256**, status inconnu mais de collision
6 → **SHA-2/512**, actuellement considéré comme relativement fiable

Contrairement aux créations de comptes en GUI, **mkpasswd** sans l'option -m (method) pour spécifier un algo de hashage, va se rabattre sur la fonction crypt de la GNU C Library (glibc) avec un algorithm basé sur DES (Data Encryption Standard)

Tout système UNIX-like utilise des libraries C pour fonctionner, la glibc est une library C standard utilisée dans les systèmes GNU et/ou basés sur un noyau Linux, donc GNU/Linux, GNU/Hurd, mais aussi des systèmes Linux sans la partie GNU

2 Empreinte théoriquement unique permettant l'identification d'une donnée, ici un mot de passe.
La différence par rapport au chiffrement, c'est que une fonction de hashage ne dispose de clé de « dé-hashage », l'opération est supposée être irréversible, sauf faille, mais des techniques de crypanalyse contre les fonctions de hashage existent cependant, par exemple la Rainbow table
Une fonction de hashage est considérée comme fiable si :
- L'opération n'est pas reversible
- Un message ne peut avoir qu'une et qu'une seule empreinte
- Un empreinte ne correspond qu'à une seule et unique donnée, sinon on dit que l'algo est victime de collision

La fonction crypt n'a pas de préfixe **idi** et est **obsolète** car très **vulnérable** aux attaques par force brute, l'implementation DES dans crypt utilise une empreinte de 56-bit calculée à partir des 8 premiers caractères du mot de passe (7 bit/char), ce qui en fait un algorithme faible