

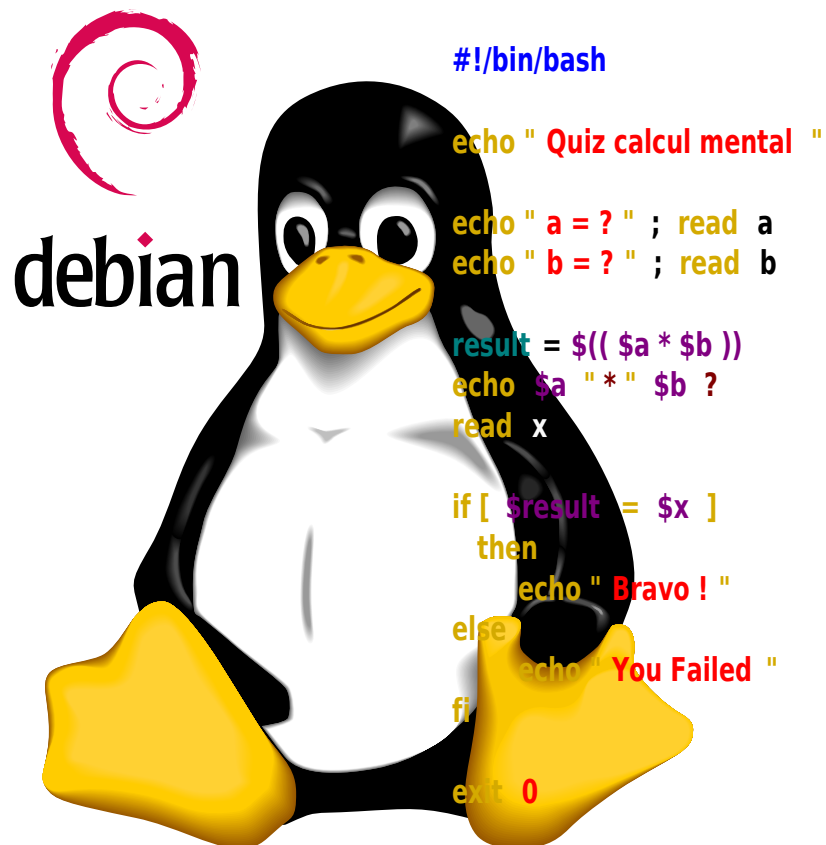
Dossier Projet BTS IRIS 2013

Projet Maths-Linux

NOM Prénom

CLASSE

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



Logo Debian
Tux

Software in the Public Interest, Inc
Wikipedia.org (Auteur : Larry Ewing)

1999 CC By-SA 3.0
1996 Domaine public

Table des matières

Serveur de clients légers GNU/Linux Debian d'applications pour QCM mathématiques.....	1
I) Rôles.....	3
II) Analyse des Besoins.....	3
-A.Présentation.....	3
-B.Principe de l'architecture clients légers/serveur.....	3
III) Fonctions de services.....	4
-A.Fonctions Principales.....	4
-B.Fonctions Secondaires.....	4
IV) Contraintes.....	5
V) Spécifications (UML).....	6
-A.Cas d'Utilisation.....	6
-B.Diagramme d'activité.....	7
-C.Diagramme de déploiement.....	8
VI) Conception architecturale.....	9
-A.Présentation.....	9
-B.Modèles de conceptions.....	10
-B.1.Modèles Entité-Relation Étendue.....	10
VI.Conception Détaillée.....	10
-A.Préparation du matériel.....	10
-B.Installation et configuration du serveur Debian.....	10
-B.1.Préparation de Debian.....	11
-B.2.Vérification et configuration des cartes réseaux.....	12
-C.Installation et configuration de LTSP 5.....	13
-C.1.Installation de LTSP et configuration réseau.....	13
-C.2.Gestion des utilisateurs LTSP.....	15
-D.Bases de Données et Scripts SQL.....	20
-D.1.Présentation script SQL pour les tables de quiz de Maths et les utilisateurs.....	20
-D.2.Explications du script SQL.....	20
-F.Conclusion.....	22

I) Rôles

Noms	Rôles
	Réaliser le logiciel produisant les exercices interactifs, en collaboration avec Madame Audibert ainsi que l'application élève
	Réaliser le logiciel permettant l'affichage en temps réel des scores et exercices de la compétition.
Moi-même	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'entraîner 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 Interactif).

Les exercices, scores, et informations sur les participants seront stockés dans des bases de données différentes.

2. Le serveur

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

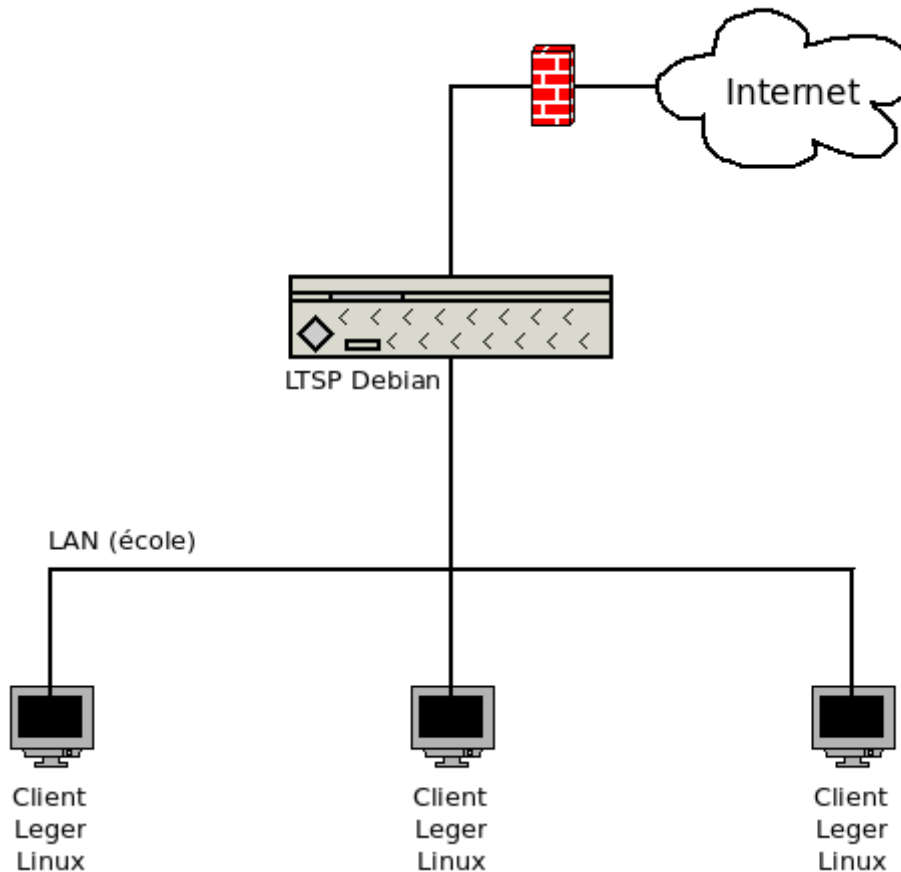
3. Les clients

Il s'agit d'anciens PC utilisés en tant que clients légers, qui permettront aux élèves d'accéder aux applications pour effectuer les entraînements et la compétition.

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

Toutes les applications seront exécutées sur le serveur, l'intérêt est de pouvoir utiliser des machines de faible puissance, anciennes et/ou sans disque dur, qui serviront uniquement d'interface pour les différents utilisateurs (chacun son écran, clavier, souris) afin d'accéder aux ressources (matérielles et logicielles) du serveur

Serveur de clients légers GNU/Linux



Dia : Thin Clients/Server architecture

III) Fonctions de services

-A.Fonctions Principales

- Les logiciels développés lors de ce projet devront être faciles d'utilisation et permettront de
- Créer les exercices QCM
 - Gérer la liste des élèves
 - Étudier les résultats des exercices
 - Faire les exercices

-B.Fonctions Secondaires

- Sauvegarder les scores dans une base de données

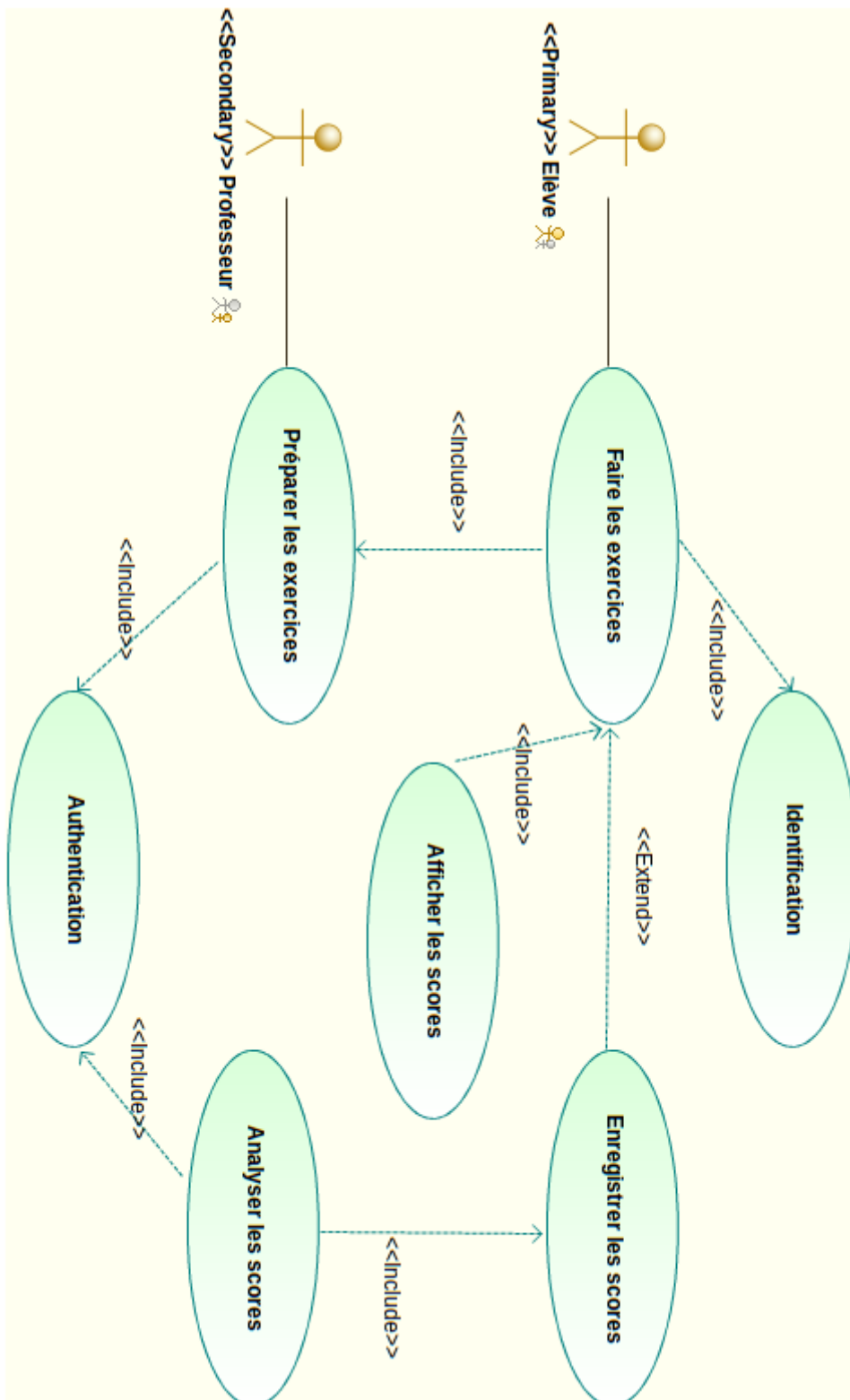
- Éventuellement proposer une interactivité
- Changement de positionnement des bonnes réponses
 - Modification automatique de mauvaises réponses à chaque répétition d'un même exercice

IV) Contraintes

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

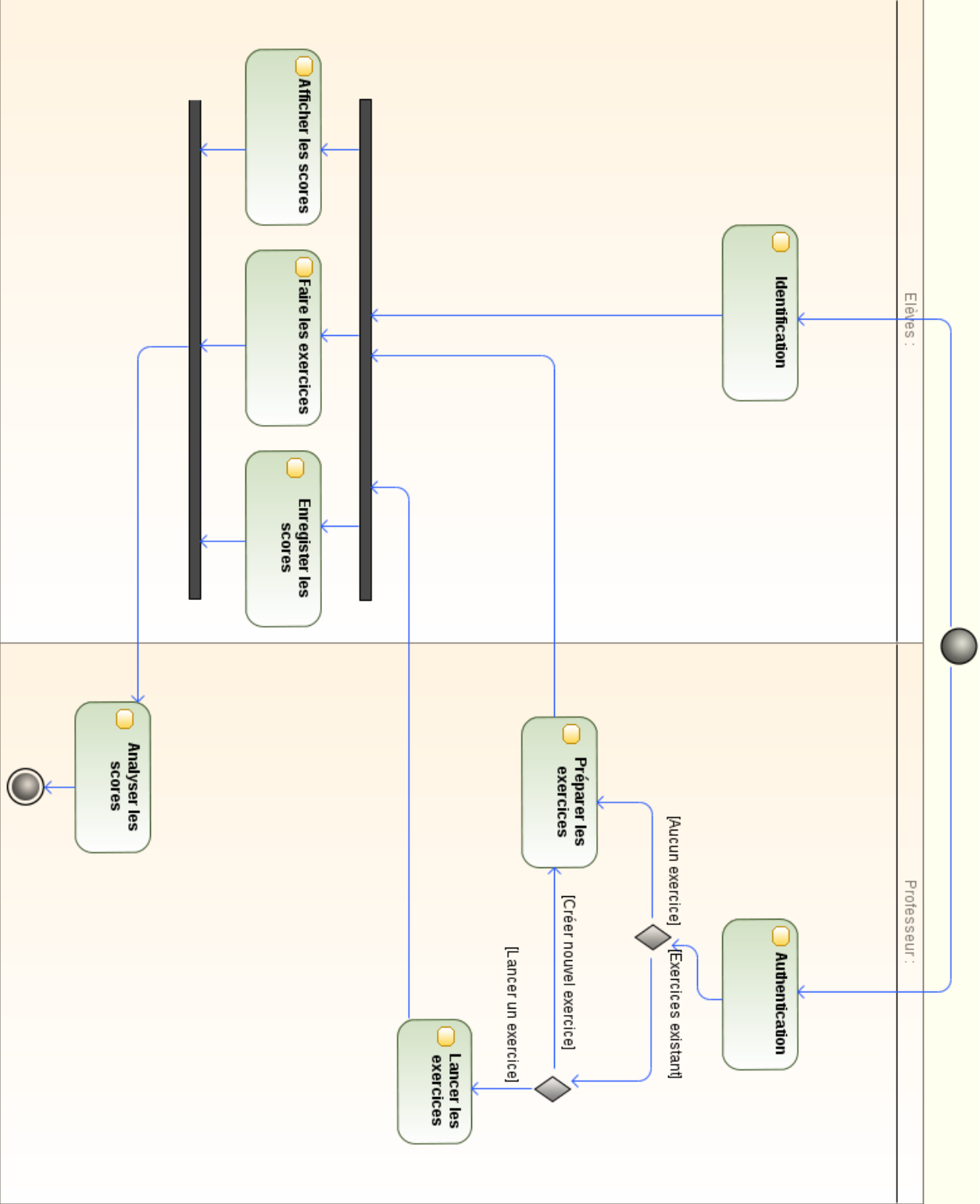
V) Spécifications (UML)

-A.Cas d'Utilisation



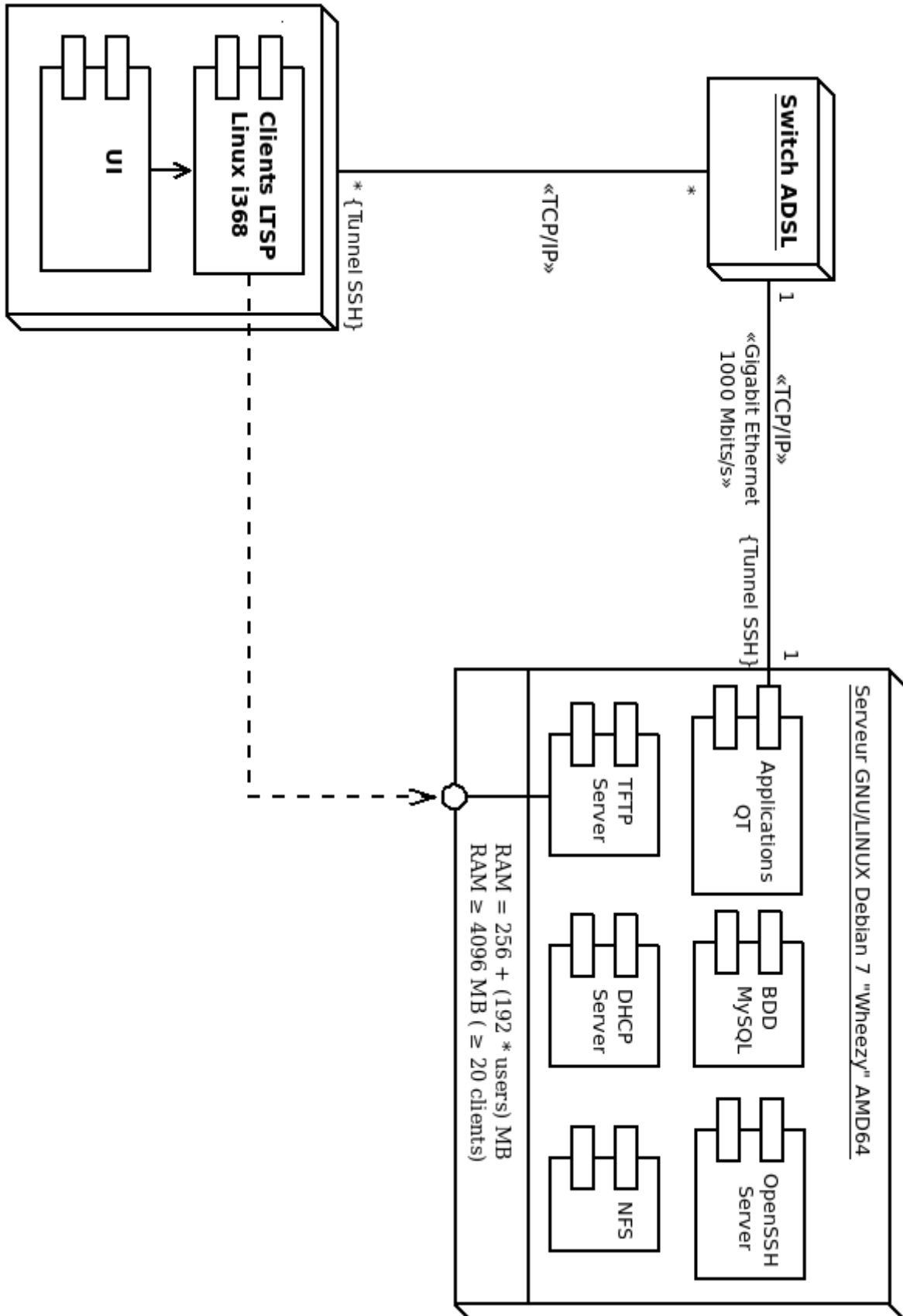
Modelio : Use Case Diagram

-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 asymétrique bout à bout

-Pour des raisons de rapport qualité/prix, on s'orientera vers le logiciel libre, accessible gratuitement sans contrat 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'être plus flexibles/évolutifs pour être utilisés dans un environnement professionnel (GNU/Linux, BSD, MySQL, QT, PHP, Apache...)

Le maintien de ces logiciels étant assuré par une large communauté de développeurs 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.

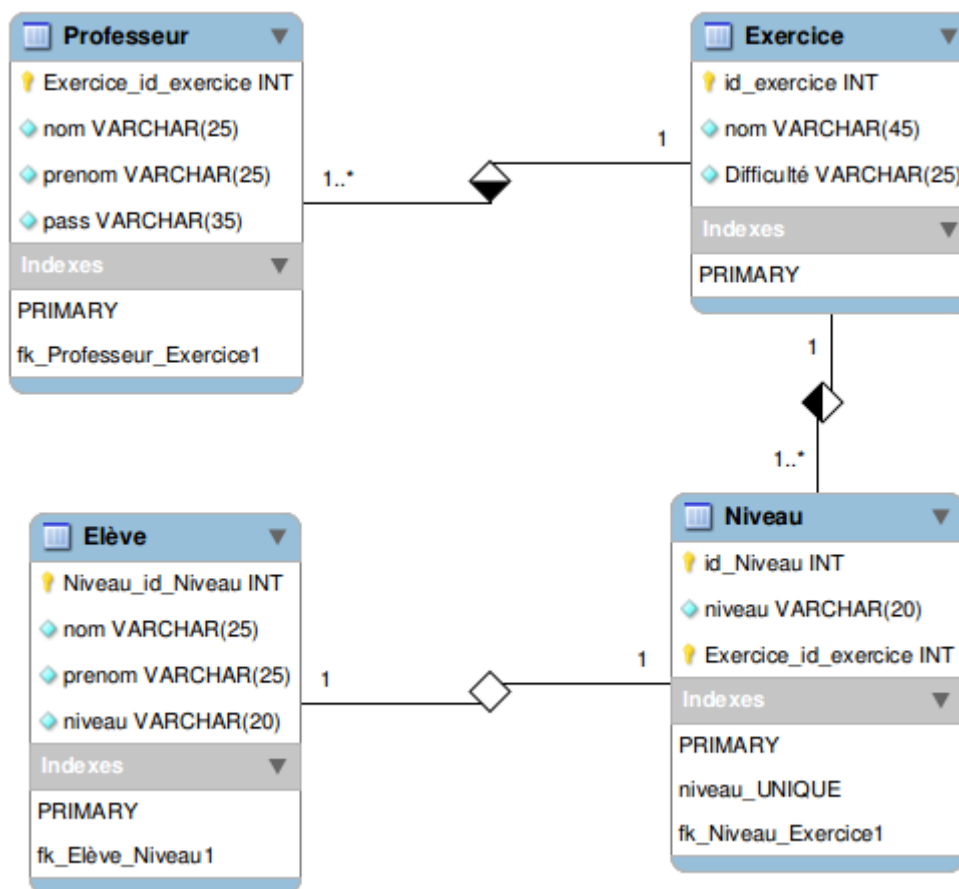
L'accès au code source permet aussi l'analyse du code par des parties tierces, facilitant la découverte et la correction de bugs/failles, améliorant la sécurité et la stabilité des logiciels libres encore maintenus, les avantages du logiciel libre sont donc multiples

Nos applications seront écrites en C++ sur l'IDE (Integrated Development Environment) **QT 4.8.2**, et compilées pour une distribution **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.Modèles de conceptions

-B.1.Modèles Entité-Relation Étendue



MySQL-Workbench : EER (Extended Entity-Relationship)

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 réservée 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'accéder à Internet

Une autre avec une IP fixe en 10.0.0.1, qui permettra 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 récupérer l'image correspondante depuis le [Serveur Debian](#)
Une fois l'image récupérée, 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 grâce à la commande **df -h** ou sudo **fdisk -l**

df permet d'obtenir 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 différentes branchés en même temps, on identifiera le bon support grâce à sa capacité mémoire

fdisk est un outil de partitionnement en ligne de commande, et l'option **-l** permet de lister les toutes partitions présentes 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 support 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/Downloads # 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 dépendant 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'installation

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

-B.2.Vérification et configuration des cartes réseaux

Une fois l'installation de Debian finie, on vérifie 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* : boucle locale)

Dans les rares cas, où 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 équipées les différentes 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 différences entre majuscules et minuscules)

Attention un seul modèle (nom commercial) d'une même carte (Ethernet ou autre) ou d'un périphérique USB (**lusb**) WiFi/Bluetooth peut-être dérivé en plusieurs versions, utilisant différents circuits et nécessitant potentiellement différents drivers en fonction de driver, d'où 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 différents 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 grâce 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 autoconditionnement 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 présent, en validant la commande, on devrait donc pas avoir d'erreurs en retour

En cas de problèmes « exotiques » ou d'un simple manque d'expérience 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          # Spécifie au système « 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 diffusion générale
gateway 192.168.1.1   # Définit la passerelle, ici l'IP réservée à l'interface eth0
```

Ne pas oublier d'activer le routage sur le serveur, grâce 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 différents paquets (TFTP, DHCP...) en installant LTSP tout seul (ltsp-server) sur un serveur, et les autres paquets séparément sur les autres serveurs

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

```
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" # Répertoire 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 DAEMON¹ 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 fois les clients connectés au switch branché sur la carte réseau du serveur eth1 (IP = 10.0.0.1)

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

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

```
tuxmaths@debianserver:~$ who --ips
bob      pts/0      2014-05-15 15:05 10.0.0.5
tuxmaths pts/2      2014-05-15 15:06 192.168.1.16
tuxmaths@debianserver:~$
```

L'utilisateur bob est connecté en tant que client léger

L'utilisateur tuxmaths (moi) est connecté en SSH afin d'administrer le serveur

Cette image a été prise après l'exécution du script de créations d'utilisateurs

La valeur pts correspond au pseudo-terminal slave, des pseudo terminaux² émulés logiquement tel que xterm, mais ici il s'agit respectivement du client LTPS (encapsulé dans du SSH) et de serveur SSHd (Secure Shell Daemon) pour accès (et l'administration) à distance via un protocole chiffré

-C.2.Gestion des utilisateurs LTSP

L'étape suivante consiste 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 prédéterminés, 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 créé par le script aussi longtemps qu'il en aura besoin. Par conséquent il ne devra stocker ce userlist et n'exécuter le script que depuis un répertoire auquel il est le seul à avoir accès, et donner à ces fichiers les droits d'accès adéquats (**ls -l** pour voir et **chmod** et **chown** pour changer les permissions) 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 posera évidemment des problèmes de sécurité en cas de fuite desdits fichiers

Le userlist devra se trouver dans le même répertoire que le script à des fins pratiques

Le users.log servira à des fins debugging uniquement et sera détruit 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> <nomdefichier>

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 zéro afin de cacher l'effacement 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

² Terminaux : ensemble de périphériques d'entrées/sorties communiquant avec l'ordinateur/calculateur, tels l'écran, le clavier, l'imprimante.
Sous les systèmes UNIX, le terme désigne un terminal de commande (SHELL UNIX) tel que XTerm ou Gnome Terminal

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 différents utilisateurs, et le groupe secondaire (qui doit être créé avant l'exécution du script avec **addgroup** <groupname>) correspond au type d'utilisateurs, professeurs ou élèves, afin de pouvoir attribuer des droits différents en fonction 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 réservés comme caractères pour séparer les trois champs, donc les espaces dans les noms, les mots de passe seront substitués par _ (caractère underscore)

Exemple : nom_prenom

Le script BASH

```
1          #*****
2          # Nom: create_users_v1.sh
3          # Auteur: Moi-même
4          # Version: 3
5          #*****
6
7          #!/bin/bash
8
9          #set -x
10
11         function create
12         {
13         select=0
14
15         for i in `cat userlist`
16
17         do
18         case $select in
19           0 ) u=$i;;
20           1 ) p=$i;;
21           2 ) g=$i;;
22         esac
23         ((select=select+1))
24
25         if [ $select -eq 3 ]
26         then
27           echo $u >> users.log
```



```

28     echo $p >> users.log
29     echo $g >> users.log
30     ((select=0))
31
32     useradd -m -G $g -p $(mkpasswd -m sha-512 $p) $u
33     fi
34
35     done
36     }
37
38     function delete
39     {
40
41         echo "Veuillez spécifiez le nom de l'utilisateur que vous voulez supprimer"
42         read u
43
44         echo "Voulez-vous faire un backup des fichiers de l'utilisateur qui vous allez
45         supprimer : O/N"
46         read x
47
48         case $x in
49             O|o)
50                 deluser --backup --remove-home $u
51                 ;;
52             N|n)
53                 deluser --remove-home $u
54                 ;;
55             *)
56                 echo "Erreur du choix - Réessayez"
57         esac
58     }
59
60     echo "Choisissez votre opération (A)jouter ou (S)upprimer des utilisateurs sinon
61     (Q)uitter"
62     read choice
63
64     case $choice in
65         A|a)
66             echo "Ajouter les utilisateurs"
67             create
68             ;;
69         S|s)
70             echo "Supprimer des utilisateurs"
71             delete
72             ;;
73         Q|q)
74             echo "Vous venez de quitter"
75             ;;
76         *)
77             echo "Erreur du choix - Réessayez"

```

76 **esac**
77 **exit 0**

La ligne commentée `# set -x` permet d'afficher le déroulement de l'opération (à des fins de debugging) si on la dé-commente

Ce script est constitué de deux fonction, `create` pour créer et `delete` pour supprimer des utilisateurs, ainsi que d'un case permettant de choisir l'une des deux fonctions ou de quitter le script

Fonction delete

Cette fonction permet de supprimer un seul utilisateur par exécution du script, afin d'alléger le script, la fonction peut néanmoins être reprise dans une boucle de type `do` avec en parcourant les lignes d'un fichier contenant les noms d'utilisateurs, de la même façon que la fonction `create` amis avec un seul champs

Fonction create

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 troisième 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 relecteur de champs (pour pouvoir traiter la ligne suivante) et utilise l'outil `useradd` pour créer les comptes utilisateurs, `useradd` vérifie 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 hashage³ 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

3 Hash : 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 cryptanalyse 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 réversible
- 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'algorithme est victime de collision

Quelques exemples de préfixe **\$id\$** et les algorithmes de hashage correspondants

\$1\$ → **MD5** Victime de collisions

\$2\$ → **BlowFish et dérivés (\$2a\$, \$2b\$, \$2x\$, \$2y\$)** Obsolète

\$3\$ → **NT Hash (basé sur MD4)** Obsolète, notation sous FreeBSD

\$5\$ → **SHA-2/256**, statut de sécurité inconnu mais pas 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 algorithme de hashage, va se rabattre sur la fonction **crypt** de la [GNU C Library](#) (glibc) avec un algorithme basé sur DES (Data Encryption Standard)

Tout système UNIX-like utilise des bibliothèques 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

La fonction crypt n'a pas de préfixe **\$idi\$** et est **obsolète** car très **vulnérable** aux attaques par force brute

La fonction crypt fonctionne de la manière suivantes

```
#define _XOPEN_SOURCE    /* See feature_test_macros(7) */
#include <unistd.h>
char *crypt(const char *key, const char *salt);
#define _GNU_SOURCE     /* See feature_test_macros(7) */
#include <crypt.h>

char *crypt_r(const char *key, const char *salt,
              struct crypt_data *data);
```

La valeur key est le mot de passe donné par l'utilisateur

La valeur salt (sel⁴) est dans CE CAS (il existe plusieurs algorithmes, plus complexes et plus sécurisés) une donnée de deux caractères compris dans les valeurs [a-zA-Z0-9./] qui influence l'empreinte du mot de passe de 4096 façon différentes

La structure data est le résultat chiffré

Avant le premier appel de crypt_r, on initialise data avec **data -> initialized**

L'implémentation 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.

En effet l'espace de la clé est codé sur 2⁵⁶ soit 7,2e16 possibilités seulement

Par contre il semblerait que les clients LTSP ont du mal à se connecter aux comptes ayant un mot de passe stockés en SHA2/512, le client continue à charger alors qu'ils arrivent à se connecter en local sans problème, et le serveur indique qu'il est connecté (visible dans les commandes **w** et **who**) aussi longtemps que le client charge, puis le client revient sur l'interface de connexion et disparaît du tableau visible dans les sorties des commandes **who** et **w**

Les clients dont le mot de passe est stocké en utilisant les paramètres par défauts de **mkpasswd**

4 Salt/sel : donnée aléatoire, et théoriquement unique, souvent stockée en clair, qui en étant concaténée au mot de passe, modifie son empreinte, deux utilisateurs ayant le même mots de passent se retrouvent avec deux empreintes différentes car le sel est différent

Au mot de la connexion, on rajoute de nouveau le sel au mot de passe rentré avant de vérifier son empreinte

(crypt) arrivent à se connecter via LTSP
A ce jour je n'ai pas pu identifier l'origine de ce problème

-D.Bases de Données et Scripts SQL

-D.1.Présentation script SQL pour les tables de quiz de Maths et les utilisateurs

Un exemple avec une base de données maths constituée de
Une table `exercice_test` qui sera créée par le professeur via l'application de création de l'utilisateur avec un exercice pour tester l'application, et qui contiendra l'identifiant de l'exercice (clé primaire), les questions, et les 4 réponses possibles, ainsi que le nombre de questions
Une table `Exercices` répertoriant les exercices avec l'identifiant unique de l'exercice (clé primaire), le nom, le niveau niveau de difficulté, et le nombre de questions pour chaque exercice
Deux tables pour les utilisateurs (professeurs et élèves) avec des clés primaires, noms prénoms pour les deux tables, mot de passe pour les professeurs uniquement et niveau pour les élèves uniquement
Une table `niveau` pour lister les exercices accessibles par l'élève d'un niveau donné, avec un champs niveau, un identifiant et un INDEX unique permettant de trier facilement les exercices par niveau de difficulté

-D.2.Explications du script SQL

Le Script SQL

```
1      CREATE SCHEMA IF NOT EXISTS `maths` DEFAULT CHARACTER SET
      latin1 ;
2      USE `maths` ;
3
4      -----
5      -- Table `exercice_test`
6      -----
7      CREATE TABLE IF NOT EXISTS `exercice_test` (
8          `id_exercice` INT NOT NULL AUTO_INCREMENT ,
9          `questions` VARCHAR(30) NULL ,
10         `bonne_rep` INT NULL ,
11         `rep1` INT NULL ,
12         `rep2` INT NULL ,
13         `rep3` INT NULL ,
14         PRIMARY KEY (`id_exercice`))
15     ENGINE = InnoDB;
16
17
18     -----
19     -- Table `Exercice`
20     -----
21     CREATE TABLE IF NOT EXISTS `Exercice` (
22         `id_exercice` INT NOT NULL AUTO_INCREMENT ,
23         `nom` VARCHAR(45) NOT NULL ,
24         `difficulté` VARCHAR(25) NOT NULL ,
25         `nb_questions` VARCHAR(45) NOT NULL ,
26         `exercice_test_id_exercice` INT NOT NULL ,
27         PRIMARY KEY (`id_exercice`, `exercice_test_id_exercice`))
28     ENGINE = InnoDB;
```

```

29
30
31 -----
32 -- Table `Professeur`
33 -----
34 CREATE TABLE IF NOT EXISTS `Professeur` (
35     `Exercice_id_exercice` INT NOT NULL ,
36     `nom` VARCHAR(25) NOT NULL ,
37     `prenom` VARCHAR(25) NOT NULL ,
38     `pass` VARCHAR(35) NOT NULL ,
39     PRIMARY KEY (`Exercice_id_exercice`))
40 ENGINE = InnoDB;
41
42 -----
43 -- Table `Élevé`
44 -----
45
46 CREATE TABLE IF NOT EXISTS `Eleve` (
47     `Niveau_id_Niveau` INT NOT NULL ,
48     `nom` VARCHAR(25) NOT NULL ,
49     `prenom` VARCHAR(25) NOT NULL ,
50     `niveau` VARCHAR(20) NOT NULL ,
51     PRIMARY KEY (`Niveau_id_Niveau`))
52 ENGINE = InnoDB
53
54 -----
55 -- Table `Niveau`
56 -----
57
58 CREATE TABLE IF NOT EXISTS `Niveau` (
59     `id_Niveau` INT NOT NULL AUTO_INCREMENT ,
60     `niveau` VARCHAR(20) NOT NULL ,
61     `Exercice_id_exercice` INT NOT NULL ,
62     PRIMARY KEY (`id_Niveau`, `Exercice_id_exercice`),
63     UNIQUE INDEX `niveau_UNIQUE` (`niveau` ASC))
64 ENGINE = InnoDB;
65 PACK_KEYS = DEFAULT;

```

CREATE SCHEMA est équivalent de **CREATE DATABASE** donc la ligne
« **CREATE SCHEMA IF NOT EXISTS `maths` DEFAULT CHARACTER SET latin1** » crée
une table nommée ``maths`` SEULEMENT IS elle n'existe pas encore
L'encodage par défaut est « latin1 », les caractères français sont supportés sauf Œ, œ et ÿ (très peu
utilisés)

CREATE TABLE IF NOT EXISTS `table_name` crée une table au nom spécifié seulement s'il
n'existe pas, avec les différents champs spécifiés entre parenthèses

INT NOT NULL AUTO_INCREMENT signifie que le champ est un entier (INT) non-nul, qui
s'incrémente tout seul en passant d'une valeur à l'autre contenu dans ce champ (1...n)

`VARCHAR(30)` est un champs contenu une chaîne de caractères (`CHAR`) de longueur variable (`VAR`) pouvant atteindre au maximum 30 caractères (chiffre indiqué entre parenthèses)

`NULL` indique que ce champs peut prendre une valeur nulle et `PRIMARY KEY ('id_exercice')` indique que la clé primaire est `'id_exercice'`, ou le champs donc le nom est spécifié (`'ici'`)

`UNIQUE INDEX 'niveau_UNIQUE' ('niveau' ASC)` est un index permettant de trier les données, ici on trie les exercices par niveau de difficulté, ici par ordre ASCendant, donc les premiers exercices affichés sont les plus faciles, suivis de ceux ayant une difficulté moyenne et enfin les plus difficiles

`ENGINE = InnoDB`; spécifie le moteur SQL, ici InnoDB (l'un des moteur du SGBD⁵ MySQL)

-F.Conclusion

En tant que passionné des réseaux et de systèmes UNIX-like libres, et utilisateur régulier de GNU/Linux (Mint et Debian), et ayant également un certain intérêt pour l'administration de bases de données, ce projet m'a permis de mettre en pratique et d'améliorer mes connaissances dans les environnements GNU/Linux et dans la conception et le déploiement de services en réseau

A ce titre j'ai trouvé ce projet très intéressant, bien qu'un peu déséquilibré au niveau de la répartition des tâches, la conception architecturale, la recherche de solutions, les tests et d'autres tâches que j'ai effectuée m'ont pas laissé le temps de m'occuper de la partie programmation QT/C++, ni même d'encapsuler mes scripts dans une GUI, en utilisant Zenity

Néanmoins les scripts restent documentés et simples à utilisés même sans interface graphique

5 SGBD : Système de Gestion de Base de Données, permet de stocker et de communiquer données organisés dans une base de données