

# Systeme d'hebergement domestique

Projet de fin d'etudes - Rapport Final



P9

Romain LIBAERT

Timothée TENEUR

IMA5

23 Février 2016

# Sommaire

[Sommaire](#)

[Contexte & Objectifs](#)

[Contexte](#)

[Objectifs](#)

[Cahier des Charges](#)

[Matériel et logiciels utilisés](#)

[Dans le détail](#)

[Au coeur du projet](#)

[Gestion des utilisateurs - Le serveur LDAP](#)

[Serveurs SMTP et DNS](#)

[Serveur Web et Webmail](#)

[Un serveur web léger](#)

[Une architecture optimisée](#)

[Envoyer des courriels](#)

[Mais aussi en recevoir](#)

[Interface d'administration](#)

[Architecture du système de fichiers](#)

[Firewall](#)

[Gestion de l'énergie](#)

[Évolutions possibles](#)

[Webmail](#)

[Système](#)

[Partie énergétique](#)

[Conclusion](#)

[Sources](#)

# Contexte & Objectifs

## Contexte

Aujourd'hui avec l'arrivée de l'Internet dans notre quotidien, quasiment tout le monde possède une adresse mail. Ce système fut une véritable révolution dans la manière dont les Hommes communiquent entre eux, augmentant de façon drastique la vitesse des échanges et la productivité des Entreprises.

Après 20 ans d'existence, les services de mail ont fleuri sur la toile. Dorénavant, il est possible de créer une boîte de messagerie en quelques instants, et d'envoyer des messages à l'infini.

Quelques statistiques sur le mail en 2015, d'après The Radicati Group :

- 4 353 millions de comptes
- 2 586 millions d'utilisateurs
- 205 milliards de mails envoyés par jour

Le problème, c'est qu'aujourd'hui toutes les grosses entreprises du domaine des technologies de l'information rognent de plus en plus sur la vie privée de leurs utilisateurs. Le contenu de nos conversations n'est parfois plus notre propriété.

L'objectif de notre projet est de donner la possibilité à n'importe qui de déployer son propre service de mail sécurisé, hébergé sur sa propre machine.

## Objectifs

Ce projet doit permettre à tout utilisateur de créer quelques comptes de messagerie sur un système embarqué de type Raspberry et permettre de conserver les données à la maison.

Le système doit être constitué à base de standards (Base LDAP, serveur de messagerie connu et maintenu, pareillement pour le client de messagerie). L'interface d'administration doit être elle aussi très simple d'utilisation.

En outre, le système devra être capable de distinguer des niveaux de privilèges entre les utilisateurs, gérer des groupes d'utilisateurs, les quotas, et devra rester le moins énergivore et le plus léger possible. Nous devons de plus rendre le code le plus générique possible afin de l'empaqueter et le déployer sur un système quelconque sans soucis.

Enfin, un effort particulier doit être porté sur l'alimentation du système embarqué. Il est notamment demandé de mettre au point une alimentation à base d'énergie renouvelable (comme un panneau solaire) permettant d'alimenter totalement ou partiellement le système.



*Fig. 1 : Une Raspberry Pi*

# Cahier des Charges

## Matériel et logiciels utilisés

Pour ce projet, nous avons utilisé jusqu'ici le matériel suivant :

- 1 Raspberry Pi 2
- 1 Carte micro SD 8 Go
- 1 Câble USB/micro USB
- 1 Câble RJ45

En outre, nous avons installé les logiciels suivants :

- Une distribution Debian Jessie pour Raspberry
- Un serveur web : Lighttpd
- Un serveur mail : Postfix
- Un serveur DNS : Bind
- Un serveur d'annuaire : OpenLDAP
- Le module PHP5 pour Lighttpd

Nous avons développé nos différents codes en Shell ou Perl directement sous Debian, et le code pour la partie web est écrit en PHP sous l'IDE NetBeans.

## Dans le détail

Dans le détail, l'idée est de développer un système autonome et fonctionnel. Autrement dit les différents packages et fonctionnalités installées sur la Raspberry ainsi que l'interface utilisateur sur le site web, et l'aspect énergétique.

Voici les différentes fonctionnalités que nous avons implémentées :

- Serveur Mail
  - Postfix pour le protocole SMTP
  - Système léger, pas de protocoles IMAP/POP en plus, tout en web
- Gestion particulière des gros mails et notamment de leurs pièces jointes
  - Décodage et encodage base 64 des pièces jointes
  - Support de la réception de pièces jointes multiples

- Gestion des utilisateurs
  - Plusieurs comptes, en utilisant LDAP
  - La possibilité de s'identifier en tant qu'administrateur
  - Listes de diffusion et groupes
  
- Antivirus & gestion des spams
  - iptables : scripts shell/perl de détection, par les logs, d'attaques, puis mise en place de règles iptables via ces mêmes scripts
  - Antispam maison, synchronisation sur les blacklists existantes et fiables, rejet d'un spam.
  
- Nom de domaine et enregistrements associés
  - Enregistrement A : définit l'adresse de nom d'un serveur web
  - Enregistrement MX : définit l'adresse de nom d'un serveur mail
  
- Sauvegarde automatique périodique de la Raspberry Pi
  
- Interface web
  - Gestion des comptes (création, modification, suppression...)
  - Gestion des listes de diffusions
  - Gestion des quotas
  - Affichage du courrier
  
- Paquets .deb installable sur tout système Debian.
  - Incluant un document explicatif pour l'installation et la configuration particulière à apporter
  
- Système pouvant être autonome en énergie



# Au coeur du projet

## Gestion des utilisateurs - Le serveur LDAP

Comme il l'était suggéré dans l'énoncé du sujet, nous nous sommes intéressés à la solution LDAP. Il s'agit d'un protocole de gestion d'annuaire, basé sur TCP/IP. Dans un premier temps, nous avons pensé à créer un système de base de données utilisant un simple système de fichiers, car nous avons trouvé ce système trop compliqué à configurer, à utiliser et trop lourd, bien que plus léger qu'une Base de Données classique.

Cependant, LDAP s'avère être un système parfaitement adapté pour la gestion d'informations utilisateurs. En effet cette tâche requiert beaucoup plus de lectures que d'écritures, ce pourquoi LDAP est optimisé. Ce programme présente l'avantage de privilégier la lecture, et dispose de nombreuses API dans les langages de programmation divers, dont PHP que nous utilisons, c'est pourquoi nous avons décidé de l'utiliser.

Le modèle d'information de LDAP est basé sur des entrées. Chacune d'entre elles est un ensemble d'attributs. Les informations sont organisées sous la forme d'un arbre. Pour notre projet, nous avons retenu la structure suivante :

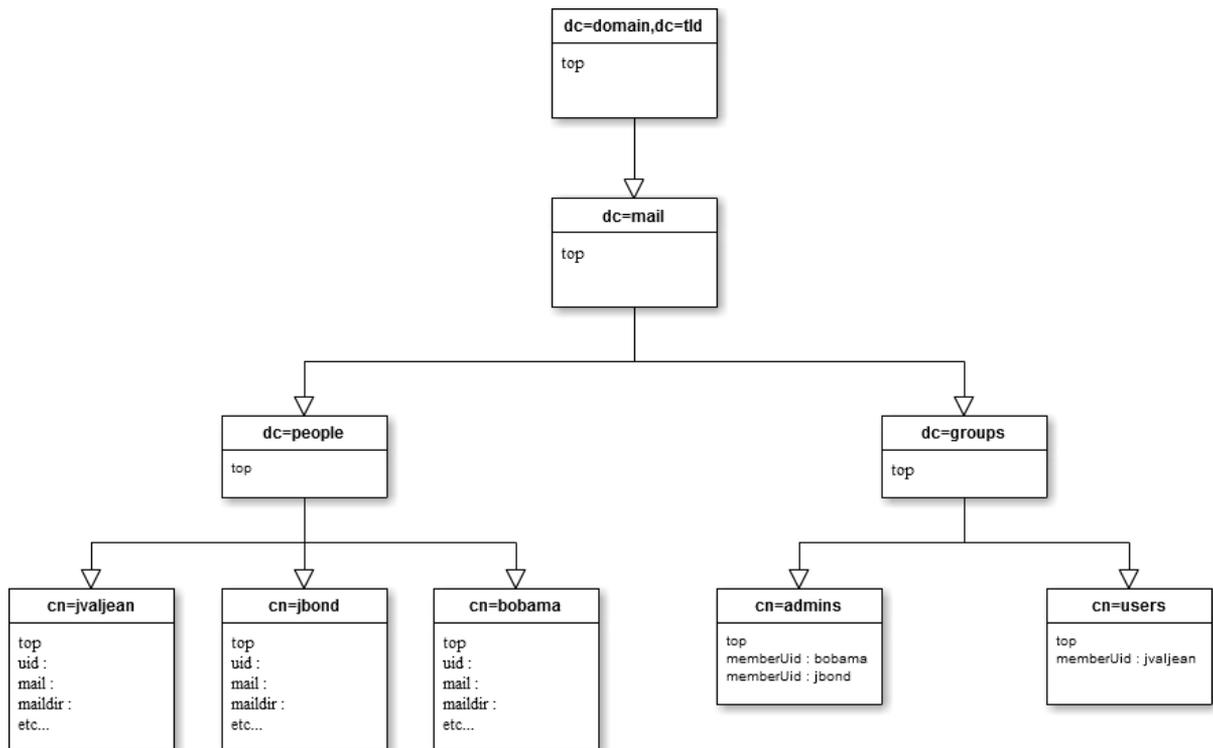


Fig. 2 : Structure de l'annuaire LDAP

Nous avons ici déterminé des champs à renseigner pour constituer un utilisateur. En particulier, un utilisateur dispose des attributs suivants :

- Son identité (nom, prénom(s))
- Son identifiant utilisateur (UID), sous la forme comme à Polytech, *initiale+nom(+chiffre)*.
- Son adresse mail complète sous la forme *UID@domaine.tld*.
- Le chemin de sa boîte mail sur le système */domaine/UID/*.
- Son quota (en octets)
- Son mot de passe, encrypté en SSHA.

```
# jvaljean, people.mail.intimail.pw
dn: cn=jvaljean,dc=people,dc=mail,dc=intimail,dc=pw
uid: jvaljean
mail: jvaljean@intimail.pw
sn: Valjean
givenName: Jean
displayName: Jean Valjean
mailbox: intimail.pw/jvaljean/
homeDirectory: /home/vmail/
objectClass: top
objectClass: inetOrgPerson
objectClass: CourierMailAccount
cn: jvaljean
userPassword:: e1NTSEF9ZGdMcFEwSWdoYVpLQS9EV1d5aVBEanNvcWxXU1h1TjI=
quota: 50000000
```

Fig. 3 : Une personne dans l'annuaire LDAP

Les utilisateurs sont enregistrés dans la branche "people" de notre annuaire. Nous avons, de plus, une seconde branche "groups". Celle-ci contient les listes de diffusion. Elle contient aussi le groupe particulier *admins*, utilisateurs qui ont droit à des options d'administration sur le webmail.

```
# admins, groups.mail.intimail.pw
dn: cn=admins,dc=groups,dc=mail,dc=intimail,dc=pw
uid: admins
mail: admins@intimail.pw
sn: Administrators
objectClass: top
objectClass: inetOrgPerson
objectClass: CourierMailAlias
cn: admins
maildrop: jvaljean@intimail.pw,jbond@intimail.pw
```

Fig. 4 : Un groupe dans l'annuaire LDAP

On peut remarquer que pour la gestion des listes de diffusion, nous avons choisi de relier une personne à un groupe par son *UID*. Ainsi, l'appartenance d'un utilisateur à un groupe se traduit par la présence de son adresse mail dans l'attribut *maildrop* du groupe.

Cela nous permet de gérer de manière pratique les groupes auxquels appartient un utilisateur, et inversement de sortir la liste des utilisateurs d'un groupe, avec une simple commande de type *ldapsearch* sur le serveur LDAP. Pour ajouter ou supprimer un utilisateur à un groupe, il suffira de modifier l'attribut maildrop du groupe.

## Serveurs SMTP et DNS

Pour cette partie, nous avons utilisé l'excellent Postfix. C'est une solution de messagerie électronique libre, rapide, légère, sécurisée et facile à administrer. C'est aussi le serveur mail par défaut dans plusieurs systèmes de type UNIX. Il permet notamment d'éviter une bonne partie du spam avec une configuration simple et accessible.

Il est important aussi de respecter les spécifications de la RFC. Nous avons eu la surprise de voir qu'il est "impératif" d'avoir deux adresses particulières sur le serveur : postmaster et abuse. Ce sont les premières victimes des attaques par force brute.

La configuration de Postfix se fait de manière étroite avec Bind, le serveur de nom de domaine que nous avons installé. En effet, pour que Postfix fonctionne, il est essentiel que l'enregistrement MX du serveur DNS fonctionne. C'est un type d'enregistrement particulier qui permet la mise en place d'un serveur mail.

De plus, il est nécessaire de faire un enregistrement PTR, qui correspond à au DNS inverse. Il n'est pas indispensable au fonctionnement du serveur mail en soit, mais la plupart des fournisseurs de ce type de service se servent de cet enregistrement pour contrer une partie du spam. Il est donc possible que les mails d'un serveur soient bloqués si l'enregistrement PTR est mal configuré.

Parallèlement, nous avons aussi procédé à l'enregistrement A qui permet la mise en place d'un serveur Web, nécessaire à notre Webmail. Nous avons utilisé des outils en ligne tel que DNSStuff pour tester notre configuration, dont un exemple se trouve en annexe. Notons par contre que la technologie DNSSEC n'est pas mise en place sur les domaines .pw par Gandi.net, notre fournisseur. Nous n'avons donc pas pu l'implémenter.

Nous avons configuré Postfix pour qu'il utilise notre annuaire LDAP. Il est possible à ce stade de tester la configuration avec telnet et d'envoyer un mail. Après cela, nous avons testé le bon fonctionnement du serveur en envoyant avec succès un mail vers une boîte mail extérieure de type gmail.

Nous nous sommes alors heurtés aux protections du Service Informatique de l'Université (CRI), qui bloque le port 25 en sortie et en entrée en dehors de l'université. Pour nous astreindre de ces restrictions, nous avons migré notre serveur vers une nouvelle connexion SDSL extérieure au réseau de Lille 1. Cela a été pour nous un moyen de vérifier en partie la portabilité de notre système, puisque nous utilisons une toute nouvelle connexion, avec une nouvelle adresse IP. Nous avons donc dû mettre à jour notre serveur de nom pour refléter ces changements. La migration s'est passée sans soucis avec un délai de propagation des changements d'environ 1 heure. Le système est resté fonctionnel à 100%, notamment parce que dans notre code nous faisons appel au serveur par *localhost* ou une variable et jamais directement l'adresse IP ou le domaine. Cependant, nous n'avons pas pu vérifier la migration vers un domaine différent, bien que cela ne reste normalement qu'une variable dans les fichiers de configuration de l'interface.

## Gestion du Spam

Comme prévu, nous avons apporté une solution à la gestion du spam sur notre serveur. Postfix propose dans ses paramètres l'utilisation de services de DNSBL (DNS Black List). Cette technologie reprend le même principe que les serveurs de noms classiques. Postfix soumet une requête de type DNS aux serveurs de liste noire définis dans sa configuration, et le serveur indique si ce nom de domaine est présent dans sa liste ou non. On peut de cette manière rejeter une bonne partie du trafic mail correspondant au Spam, et ce très facilement.

Malgré cela, nous pensons qu'une partie du Spam passera à travers les mailles du filet. Il aurait été intéressant de traiter ceux-ci pour les classer comme indésirables et pouvoir ainsi les afficher dans l'interface web.

## Serveur Web et Webmail

### Un serveur web léger

Initialement, nous sommes partis sur une solution classique à base d'un serveur web Apache, qui nous a permis de débiter le développement de l'interface. Cependant, comme nous cherchons à alléger un maximum le système, nous nous sommes penchés sur une autre solution : Lighttpd. Ce serveur web vise, comme son nom l'indique, la performance et la légèreté. Sur le même contenu, avec une connexion Très Haut Débit (100Mbps) nous chargeons désormais la page de vue

générale en 0,42s (Lighttpd) contre 1,70s avant (Apache 2) avec le navigateur Chrome, sans cache.

## Une architecture optimisée

L'utilité de réaliser notre propre webmail est, dans un premier temps, de s'abstenir de l'installation d'un serveur IMAP/POP, puisque nous retrouvons nous-mêmes le contenu. Dans un second temps il nous permet de ne garder que les fonctionnalités nécessaires à notre webmail, sans superflu.

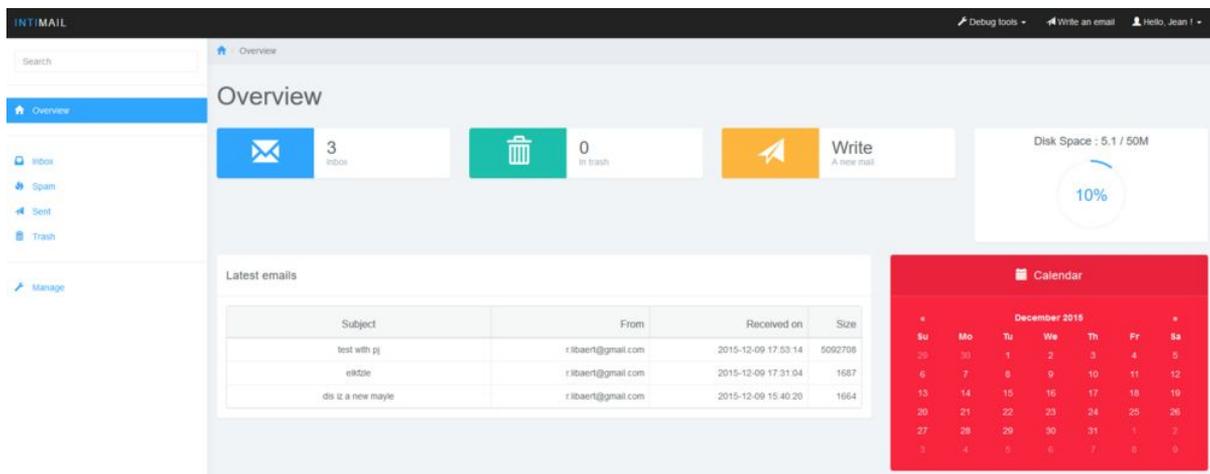


Fig. 5 : La vue générale de notre webmail

Grâce à PHP et à son module d'interface avec LDAP, nous pouvons identifier un utilisateur avec son nom d'utilisateur et son mot de passe sur le serveur local LDAP, et ainsi retrouver nombreuses informations utiles comme le quota actuel autorisé, son nom entier, le chemin d'accès à son répertoire dans lesquels sont ses mails sur le système de fichiers...

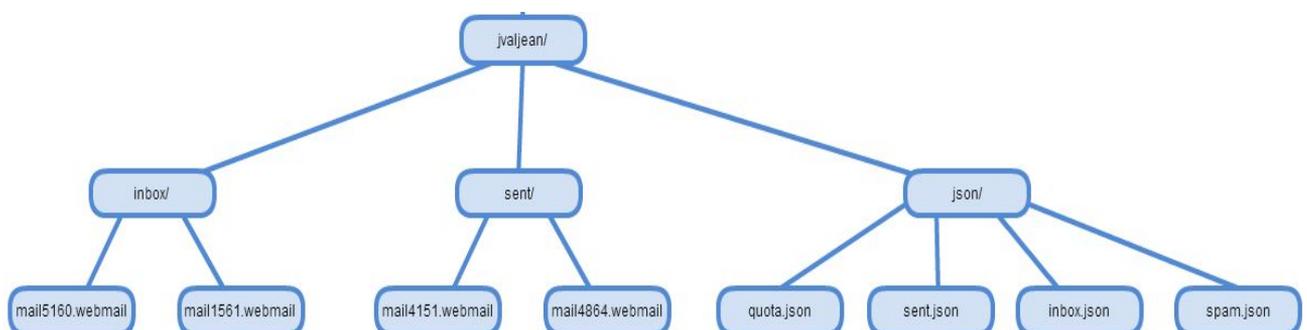


Fig. 6 : L'architecture du dossier d'un utilisateur : des dossier sent et inbox contenant les mails bruts, un dossier json dans lequel sont placés les entêtes des mails ainsi que le quota

Chaque en-tête de mail est écrite dans un fichier .json distinct par boîte (un fichier pour la boîte de réception, un fichier pour les mails envoyés...) ce qui nous permet de montrer à l'utilisateur la liste des mails d'une boîte sans traitement lourds en arrière plan pour le serveur. Ces fichiers sont renseignés par un script lancé à la réception d'un mail par postfix du côté réception, et par l'interface web elle même pour l'émission. L'en-tête d'un mail est écrite dans un conteneur JSON de la manière suivante :

```
2 {
3   "from": "r.libaert@gmail.com",
4   "subject": "pls work now",
5   "timestamp": "2015-12-09 15:48:23",
6   "unixtimestamp": "1449672503",
7   "queueid": "78BDB3FA8A",
8   "size": "1687",
9   "status": "0",
10  "pj": "0",
11  "id": "1",
12  "to": "jbond@intimail.pw"
13 },
```

Fig. 7 : Exemple d'entrée dans un fichier .json

Grâce tout cela, nous pouvons afficher simplement le contenu des différentes boîtes mail dans des tableaux sur l'interface grâce aux en-têtes, et par la suite retrouver et afficher le contenu d'un mail, à partir de son fichier brut, sur lequel l'utilisateur aura cliqué, comme on le verra un peu plus loin.

Toujours dans un souci d'amélioration des performances, le strict nécessaire est généralement chargé sur les pages web : par exemple, en venant de la vue générale, lorsqu'on clique sur *Inbox*, seule la partie intérieure de la page est chargée, on ne recharge pas le bandeau du haut de page ou le menu de gauche, à la manière d'un MVC classique.

## Envoyer des courriels

Toute cette architecture ne doit pas nous faire oublier le but premier : envoyer et recevoir des mails. Pour envoyer des mails, c'est très simple, on peut soit cliquer dans le bandeau du haut sur "Quick Email", qui permet, sur la même page, d'envoyer un email rapide et sans pièce-jointe, soit cliquer dans la barre de navigation sur le côté sur "Write a new email", ce qui nous mène vers une page dédiée permettant d'écrire un email, incluant une pièce jointe.

# New mail

**Recipient:**

**Subject:**

**Attach a file:**

 Aucun fichier sélectionné.  
(Accepted : .jpg, .png, .gif, .pdf, .zip)

**Message:**

*Fig. 8 : Envoyer un mail, c'est simple comme bonjour !*

On peut remarquer sur cette capture d'écran que l'on ne peut envoyer que certains types de pièces jointes. Nous supportons JPG, PNG, GIF, PDF et ZIP. Si la pièce jointe est d'un autre format, elle est rejetée. La vérification du type s'effectue sur l'extension du fichier et sur son type MIME, ce qui évite par exemple que quelqu'un modifie l'extension d'un fichier.exe en fichier.jpg pour l'envoyer. La pièce jointe doit, de plus, avoir une taille inférieure à la taille maximale d'une pièce jointe paramétrée par l'administrateur, comme nous le verrons dans l'interface d'administration.

Si tous ces critères sont bons, on écrit le mail au format brut décrit par la RFC, puis on l'expédie à l'aide de la fonction `mail` de PHP. Si cette fonction n'échoue pas, c'est que le mail est expédié avec succès, on peut alors mettre à jour le quota d'espace disque maximum autorisé à l'utilisateur en incrémentant celui-ci avec la

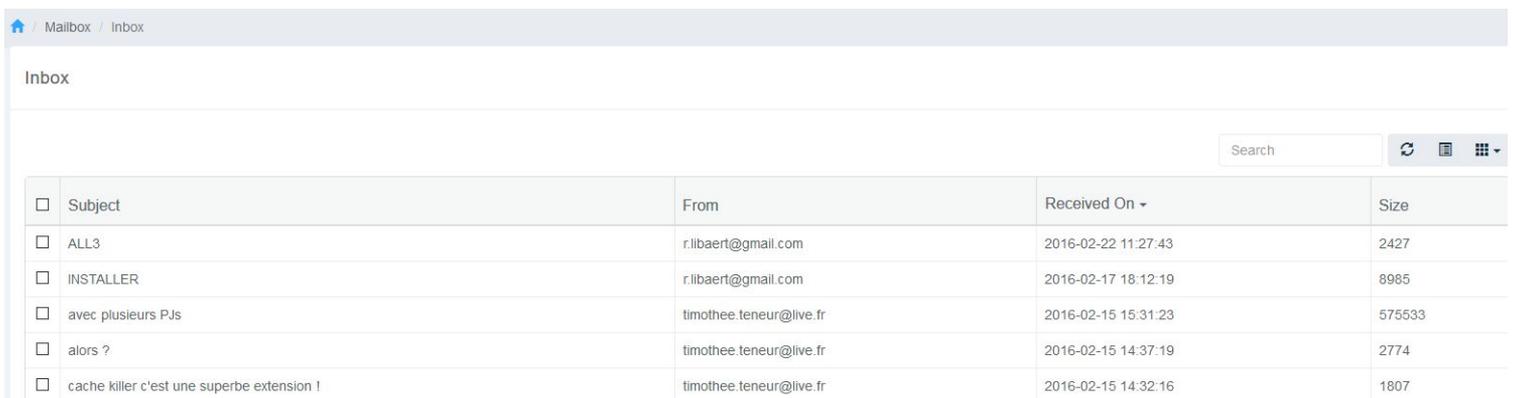
taille du mail qui vient d'être envoyé, ainsi que créer un fichier mail brut sur le disque pour pouvoir afficher le mail dans la boîte d'envoi et en garder une trace.

Notons que comme c'est l'utilisateur web du système (*www-data*) qui exécute la fonction, nous modifions les en-têtes du mail afin que le destinataire reçoive un mail de la part de "*Prénom Nom*" depuis l'adresse "*pnom@domaine.tld*" plutôt que "*www-data@domain.tld*" par défaut.

La migration vers une IP externe à Lille 1 étant réussie, nous pouvons bien envoyer des mails depuis notre interface qui sont reçus à l'extérieur sans soucis.

## Mais aussi en recevoir

Recevoir des mails est tout aussi important. Nous avons vu l'architecture en arrière-plan, mais au final, à quoi cela ressemble-t-il pour l'utilisateur ? Il dispose en fait, en cliquant sur Inbox (ou Sent, Spam ou Trash également) d'un accès à sa boîte de réception. D'un clic s'affichent alors à lui tous les mails qu'il a reçu :



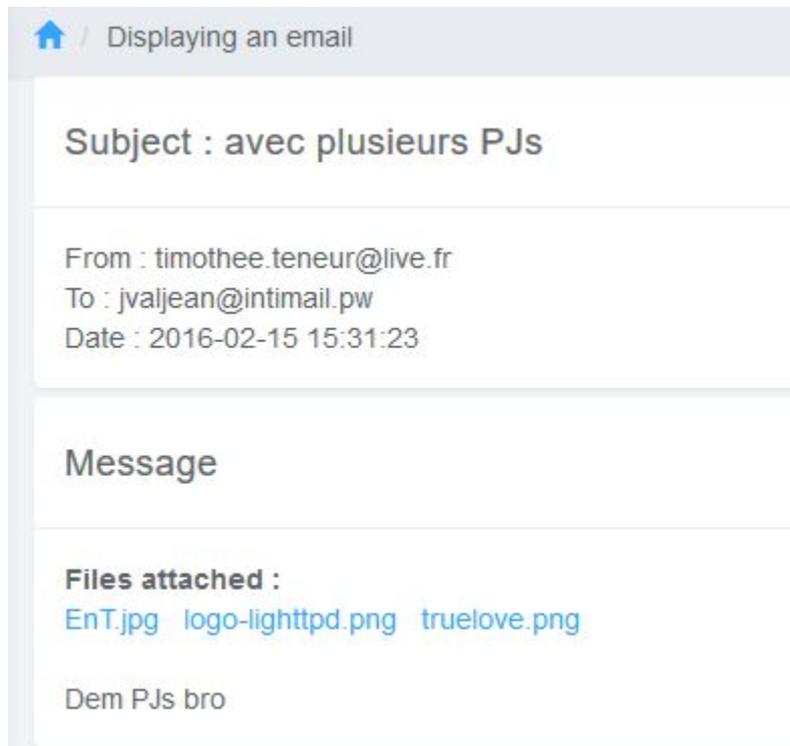
The screenshot shows a webmail interface with a header bar containing a home icon, the text 'Mailbox / Inbox', and a search box. Below the header is a table of emails. Each row has a checkbox on the left, followed by the subject, the sender's email address, the received date and time, and the size in bytes. The table contains six rows of email data.

<input type="checkbox"/>	Subject	From	Received On	Size
<input type="checkbox"/>	ALL3	r.libaert@gmail.com	2016-02-22 11:27:43	2427
<input type="checkbox"/>	INSTALLER	r.libaert@gmail.com	2016-02-17 18:12:19	8985
<input type="checkbox"/>	avec plusieurs PJs	timothee.teneur@live.fr	2016-02-15 15:31:23	575533
<input type="checkbox"/>	alors ?	timothee.teneur@live.fr	2016-02-15 14:37:19	2774
<input type="checkbox"/>	cache killer c'est une superbe extension !	timothee.teneur@live.fr	2016-02-15 14:32:16	1807

Fig. 9 : La boîte de réception

On remarque notamment les fonctions de recherche, d'actualisation du tableau, de choix des colonnes à afficher, on peut classiquement choisir combien de lignes on affiche par page, changer de pages, etc...

Lorsque l'on clique sur un des lignes, les données de l'en-tête du mails sont passés en paramètres à une autre page, qui d'après ces informations retrouve le fichier mail brut, le traite et l'affiche :



*Fig. 10 : Un mail contenant 3 pièces jointes et un message*

Comme on peut le voir, nous pouvons recevoir des mails avec plusieurs pièces jointes. Nous avons ici présenté la boîte de réception, mais les autres boîtes fonctionnent de manière similaire.

## Interface d'administration

Le système de webmail possède plusieurs variables, des listes de diffusion, des comptes... Mais on veut pouvoir donner les droits à un utilisateur de modifier tout cela. En effet, si nous trouvons cela abordable d'ajouter un utilisateur avec une commande dans le shell, il faut aussi penser à l'utilisateur final qui préfère peut-être le confort d'une interface dédiée pour gérer son serveur mail plutôt que le terminal.

De manière très simple, un utilisateur appartenant au groupes d'utilisateurs "Administrateurs" est reconnu en tant qu'administrateur par l'interface, et peut cliquer sur un bouton "Manage" dans le panneau latéral qui n'est visible qu'aux administrateurs. De là, il arrive sur la page suivante :

Modify a user :

Existing users :

---

<p>Max attachment size</p> <p>Size : <input type="text" value="5000000"/> <input type="button" value="Change Size"/></p> <p><small>(in bytes, for example 5 Mb = 5000000)</small></p>	<p>Change domain name</p> <p>New domain name : <input type="text" value="intimail.pw"/></p> <p><small>(i.e. intimail.pw)</small></p> <p><input type="button" value="Change Domain"/></p>
---	--

Create a new account

First name :  Last name :

Quota :  Password :

---

<p>Add a group</p> <p>Group name : <input type="text" value="Write the group name"/></p> <p><small>(i.e. assistants)</small></p> <p><input type="button" value="Add this group"/></p>	<p>Remove a group</p> <p>Choose the group to remove :</p> <p><input type="text"/></p> <p><input type="button" value="Remove this group"/></p>
---	---

Fig. 11 : l'interface d'administration

Sur cette interface, il peut choisir la taille maximale des pièces jointes, changer le nom de domaine sur lequel est installé l'interface et dont dépendent toutes les adresses (@domaine). Il est aussi possible de modifier directement ces variables en modifiant le fichier `.../include/config.inc.php`. Pratique notamment si l'on a renseigné un mauvais nom de domaine à la configuration et que l'on arrive plus à accéder à l'interface.

Il peut aussi créer un nouveau compte en renseignant les champs prénom, nom, mot de passe et quota. L'utilisateur peut ensuite, une fois son compte créé, directement se connecter avec son identifiant (initiale prénom + nom) et mot de passe. Lors de la création du compte, un mail de bienvenue est envoyé au nouvel utilisateur, ce qui permet en arrière-plan de créer tous les dossiers utilisateurs et fichiers en-têtes utiles à l'interface une fois pour toutes.

L'administrateur peut aussi ajouter une liste de diffusion (assimilée à un groupe) ou en supprimer une parmi les listes existantes.

En outre, l'administrateur peut aussi modifier les données d'un utilisateur. Pour cela, sur la présente page, il peut sélectionner un utilisateur parmi la liste des utilisateurs existants, ce qui le redirige vers la page de gestion des comptes :

## Administration Panel - jbond

<b>Change User ID</b> Change UID : <input type="text" value="Write the new User ID"/> <input type="button" value="Change UID"/>	<b>Change User Full Name</b> Change user's name : <input type="text" value="jbond"/> <input type="button" value="Change Name"/>
<b>Change Password</b> Change user's password : <input type="password" value="....."/> <input type="button" value="Change Password"/>	<b>Change Quota</b> Change user's quota : <input type="text" value="Write the new quota here"/> <input type="button" value="Change Quota"/>
<b>Add user to a group</b> Choose the group to add to : <input type="text"/> <input type="button" value="Add jbond to this list"/>	<b>Remove user from group</b> Choose the group to remove user from : <input type="text"/> <input type="button" value="Remove jbond from this list"/>
<input type="button" value="Delete account"/>	
Yes, I want to delete jbond	

Fig. 12 : La gestion d'un compte sélectionné, ici celui de James Bond

A ce stade, il peut changer l'UID de l'utilisateur (utile en cas de conflit), le nom, le mot de passe, le quota de cet utilisateur, et l'ajouter ou le retirer d'une liste de diffusion. Accessoirement, il peut aussi supprimer le compte en cliquant sur l'énorme bouton rouge, qui lui demandera deux fois de suite de confirmer son action, afin d'éviter les erreurs de clic. Cette action entraîne la suppression de l'utilisateur de toutes les listes de diffusion auxquelles il appartient, la suppression de son compte dans la branche *people* de l'arbre LDAP, et la suppression définitive de son dossier utilisateur sur le disque/système de fichiers, impliquant donc la perte de ses mails.

## Architecture du système de fichiers

Comme nous l'avons dit plus haut, l'idée est de se passer des mécanismes classiques avec les clients mails habituels. Ces serveurs s'alourdissent généralement en utilisant des serveurs IMAP/POP3 et un mécanisme d'authentification SMTP.

Pour adapter ces fonctionnalités à notre système, nous avons donc constitué une architecture pour organiser les fichiers qui transitent. Vous pourrez voir un schéma récapitulatif en annexe 1.

Dans la première moitié du projet, les mails étaient délivrés par Postfix. Nous utilisions un script exécuté lors de la réception d'un mail pour les traiter de la manière que nous voulions. Cette méthode avait les inconvénients d'être lourde et pas efficace à 100%. En effet seuls les mails reçus depuis l'extérieur étaient correctement traités.

Désormais, les mails sont traités totalement par notre script. Cela nous permet de gérer les mails dans des dossiers organisés à notre manière, et de recevoir les mails locaux c'est-à-dire entre utilisateurs du même domaine. Cette nouvelle méthode règle aussi d'autres problèmes notamment au niveau des permissions sur les fichiers.

Pour réduire le nombre d'opérations, et pour profiter des fonctionnalités offertes par le framework utilisé pour l'interface web, nous regroupons des informations sur chacun des mails reçus sous la forme de fichiers JSON. Pour ce faire, l'idée est d'exécuter un script lorsque l'évènement "réception d'un mail" survient. Postfix offre cette possibilité dans sa configuration. Or, il se trouve qu'il n'est pas si simple de récupérer le chemin du fichier contenant le mail, car le nom du fichier crée est sous la forme *Timestamp UNIX.10 Caractères aléatoires*. (ex: *1456233618.5A8D43E97F*).

La seconde problématique est que le webmail doit avoir la possibilité de pouvoir lire les fichiers pour afficher les mails sur son interface. Or, il semble que Postfix les écrit en autorisant uniquement son utilisateur en lecture. Il nous a fallu trouver une solution pour remédier à cela. Nous avons choisi d'ajouter `www-data` (utilisateur du serveur web) au groupe `vmail` et de modifier les droits sur les fichiers, de sorte que les utilisateurs de ce groupe puissent lire les fichiers.

Nous avons revu une partie de l'organisation des fichiers pour récupérer les pièces jointes si elles sont présentes. Le script que nous avons écrit récupère le fichier original du mail et le découpe en éléments simples dans des fichiers distincts : corps du mail et différentes pièces jointes. De cette manière, pour afficher un mail, nous n'avons pas besoin de charger la totalité du fichier. Dans le cas où il y a des pièces jointes, cela aurait ralenti l'affichage des informations.

Nous avons également implémenté la gestion des quotas. Dans l'état actuel des choses, s'il n'y a plus de place disponible pour l'utilisateur, le mail n'est tout bonnement pas délivré à ce destinataire. L'idéal aurait été de notifier l'émetteur avec un mail, ce qui n'est pour l'instant pas le cas, mais qui pourrait être résolu en renvoyant à postfix un code d'erreur défini.

Pour terminer, nous avons revu la totalité du code développé. Nous sommes passé du Bash au Perl. Ce dernier est plus performant et plus simple à utiliser, et à lire.

## Firewall

Un autre gros morceau du projet, mais qui n'est pas spécifiquement lié à la problématique du mail, c'est la sécurisation du serveur. En effet, après avoir mis en ligne notre serveur, des pirates ont très vite tenté de pénétrer le système. Globalement, les attaques étaient de type force brute sur les services hébergés sur notre Raspberry nécessitant une identification, tels que SSH, SMTP, ...

On retrouve ici un petit point faible du protocole SMTP. Étant donné que chaque serveur mail doit avoir une adresse postmaster et abuse, l'attaquant a à sa disposition deux adresses à attaquer. Ceci dit ce type d'attaque est simple à contrecarrer car il suffit de limiter dans le temps le nombre de tentatives de connexions.

La contre-mesure que nous avons mis en place est simple : une série de règles iptables. En résumé, nous avons opté pour un système de liste blanche et noire. Les adresses IPs de la liste blanche sont autorisées à utiliser la totalité des services (notamment SSH), tandis que celles de la liste noire sont totalement ignorées. Les autres IPs n'ont accès qu'au webmail.

Étant donné que le nombre d'IPs attaquant notre serveur n'a cessé d'augmenter, nous avons automatisé la mise à jour de la liste de noire en développant un script pour Cron. Ce script est exécuté une fois par heure. A ce jour, notre liste noire contient 47 adresses. Bien sûr, ce système est susceptible d'être modifié à convenance pour passer sur une solution dédiée telle que Fail2ban, qui fait en fait ce qu'on fait déjà (étudier les logs et bannir des IP avec iptables), avec des règles prédéfinies par la communauté et personnalisables. En effet, nous nous attendons à voir la variété et le nombre d'attaques augmenter après avoir migré sur une IP non protégée par le firewall de l'Université, puisque nous n'avons plus la première ligne de défense du CRI, en particulier pour le port 389 (LDAP) qui était bloqué par l'université et s'est retrouvé en première ligne après la migration. Nous avons donc ajouté une règle bloquant tout le trafic entrant vers le port 389 hormis *localhost*.

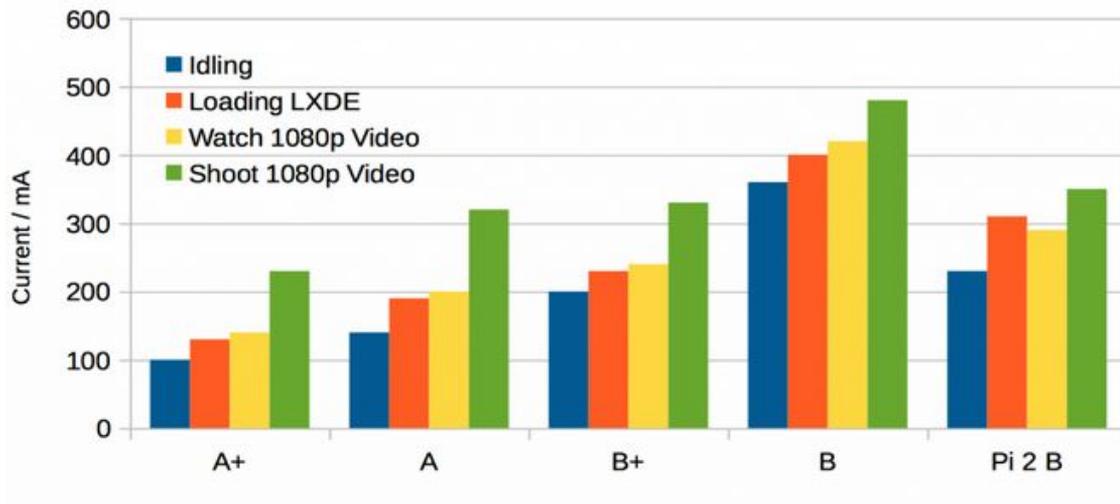
## Package d'installation

Nous avons créé un package `.deb` contenant les sources du webmail ainsi que des scripts utiles à l'installation et au fonctionnement du serveur. Nous avons développé des scripts de configuration automatique pour LDAP et Postfix. Comme `dpkg` ne supporte pas directement l'installation automatique des dépendances, nous recommandons d'installer avant tout le paquet `gdebi-core` qui est en fait un `dpkg` amélioré. Grâce à la commande `gdebi paquet.deb`, toutes les dépendances seront installées, en particulier postfix, le serveur ldap, le serveur web lighttpd entre autres, sans soucis. Il est à noter que comme les scripts de configuration des différents services ne sont que partiellement implémentés dans le paquet `.deb`, nous fournissons en annexe un README, complet, et illustré, qui explique pas à pas comment faire fonctionner notre système sur n'importe quel Linux, en particulier les lignes de configuration des différents services à changer, et un exemplaire des fichiers de configuration fonctionnels de ces services.

## Gestion de l'énergie

Une considération importante pour un système autonome. D'après les fiches techniques, notre modèle de Raspberry Pi (B 2) est officiellement évalué avec un fonctionnement à  $\sim 650\text{mA}$  pour 5V. De nombreux propriétaires de ce système équipés se sont amusés à vérifier la consommation électrique réelle de leur Pi 2 B sous différentes charges. On pourra retenir cette étude ([lien en sitegraphie](#)), qui relève la consommation en courant en fonction de l'activité lancée sur la Pi, mais aussi en charge sur 1 à 4 CPU :

## Raspberry Pi Power Usage



Raspberry Pi 2 model B Power Usage Chart

	A+	A	B+	B	Pi 2 B
<b>Idling</b>	100	140	200	360	230
<b>Loading LXDE</b>	130	190	230	400	310
<b>Watch 1080p Video</b>	140	200	240	420	290
<b>Shoot 1080p Video</b>	230	320	330	480	350

Raspberry Pi 2 model B Power Usage Comparison

Current Draw vs CPU use /mA					
Cores under load	0	1	2	3	4
<b>BCM2836 Pi2</b>	230	280	320	380	420

BCM2836 current usage with varying number of cores under load

*Ces chiffres sont relevés sur un système avec caméra connectée, HDMI connecté, et câble ethernet branché et actif.*

On peut constater que globalement, notre système Pi 2 B tournant sans interface graphique, sans composant supplémentaire, sans sortie vidéo ni caméra avec plusieurs serveurs tournant tout en restant le plus léger possible, et demeurant ~99% idle la plupart du temps, ne devrait pas dépasser une consommation de 300mA grand maximum :

moth@webmail: ~

```
top - 16:41:09 up 10 days, 59 min, 1 user, load average: 0,00, 0,01, 0,05
Tasks: 97 total, 1 running, 96 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,2 us, 0,2 sy, 0,0 ni, 99,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 948108 total, 381796 used, 566312 free, 59024 buffers
KiB Swap: 102396 total, 0 used, 102396 free. 249160 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
20896	root	20	0	5224	2580	2156	R	1,0	0,3	0:00.27	top
14869	root	20	0	94664	7376	4908	S	0,3	0,8	0:30.18	php5-fpm
19997	moth	20	0	11608	3916	3232	S	0,3	0,4	0:01.30	sshd
1	root	20	0	3028	1504	1364	S	0,0	0,2	0:15.80	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:01.22	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	0:11.62	rcu_preempt
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_sched
9	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.25	migration/0
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.31	migration/1
12	root	20	0	0	0	0	S	0,0	0,0	0:00.16	ksoftirqd/1
14	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/1:0H

Fig. 13 : Un processeur très très peu chargé et une mémoire relativement libre

Nous avons, l'année dernière déjà, réalisé un circuit permettant d'adapter un panneau solaire sur une batterie USB classique de 9000mAh. Nous avons alors pu mesurer l'intensité délivrée par le panneau solaire. En cas d'ensoleillement moyen habituel dans nos contrées, le panneau solaire débitait 400mA, et pouvait atteindre le double quand le soleil était directement présent, pour environ 800mA débité.

Ce panneau solaire accompagné de son circuit d'adaptation devrait donc suffire pour alimenter notre système la journée, le brancher ensuite sur une batterie portable permet de stocker le surplus pour la nuit.

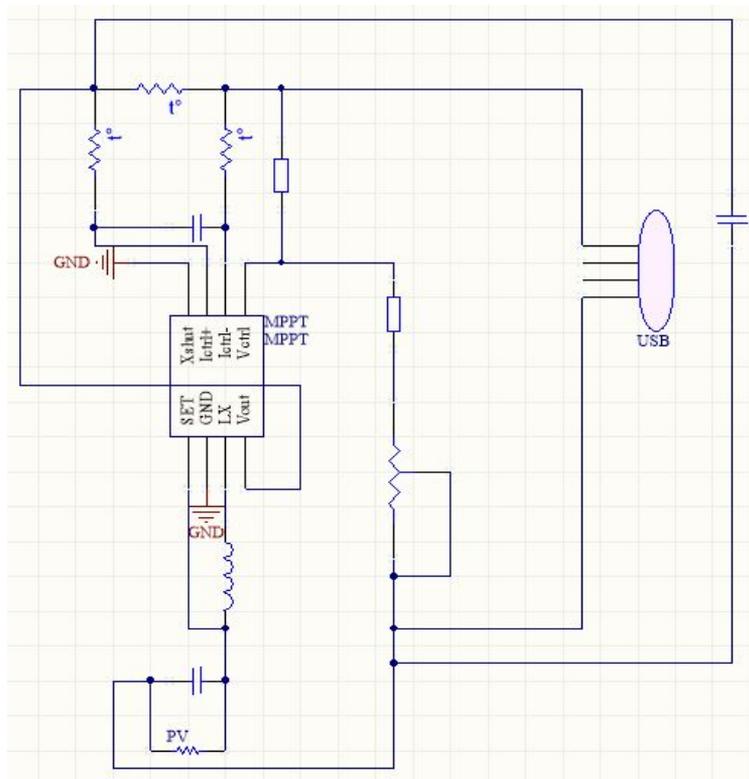


Fig. 14 : le circuit d'adaptation

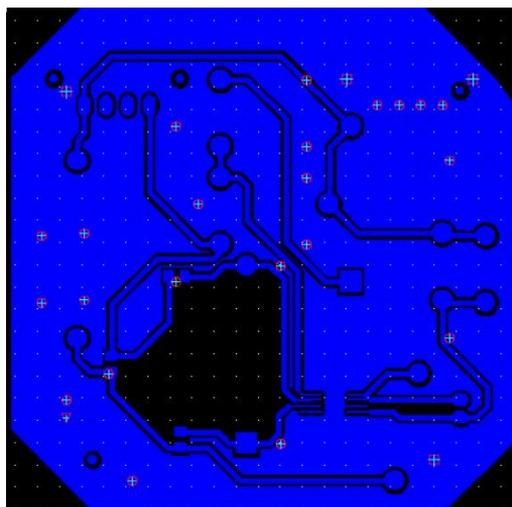


Fig. 15 : et son PCB

## Évolutions possibles

### Webmail

Concernant les évolutions possibles que nous aurions aimé apporter à notre projet avec plus de temps, commençons par le web. Nous aurions souhaité apporter

la gestion de l'envoi de plusieurs pièces jointes, comme nous l'avons fait pour la réception.

De plus, nous aurions voulu intégrer une corbeille, où les messages pourraient aller avant d'être définitivement supprimés du système, d'autant que nous avons déjà écrites les fonctions PHP permettant de retirer une en-tête d'un fichier JSON et le fichier mail associé.

Enfin, nous aurions aimé, en plus du blocage déjà présent, implémenter une gestion plus profonde des pièces jointes, permettant notamment le remplissage de la boîte de spams.

## Systeme

Il aurait été intéressant de notifier l'expéditeur lorsque le mail n'a pas pu être délivré à un utilisateur lorsque son quota est plein. En toute logique, cela devrait être fait simplement avec la commande Sendmail.

Une meilleure gestion du trafic mail indésirable aurait été la bienvenue. En effet, malgré la limitation du spam, il restera toujours une partie du volume de mail qui sera indésirable (pub, phishing, scam, ...).

Étant donné la taille du système, nous avons pensé à une approche "machine learning". L'utilisateur indiquerait pourquoi il considère le mail comme indésirable, constituant des règles refusant certains mails. Les mails provenant de cet expéditeur pourraient être détectés par le script à la réception et automatiquement envoyés dans la boîte de spams.

## Partie énergétique

Même si la solution au panneau solaire ne pose aucun problème en théorie, nous aurions déjà dans un premier temps aimé l'implémenter en pratique. Nous n'avons finalement pas réalisé notre objectif de détecter deux cas d'alimentation différents : soit l'alimentation via panneau solaire est suffisante, dans ce cas il alimente la batterie et donc la Raspberry, soit quand ce courant n'est pas suffisant, on alimente la batterie par le secteur.

## Conclusion

En conclusion, nous pensons avoir répondu au cahier des charges initial dans sa majeure partie. Nous sommes aujourd'hui en mesure de proposer une solution mail domestique, privée, et personnalisable par un particulier.

Notre installation de Debian a été au maximum allégée pour notre utilisation sans interface graphique, et nous avons mis en place une réponse aux diverses attaques que nous avons pu subir lors de la mise en place de notre solution logicielle. En outre, le pare-feu, avec notre système de liste noire, bloque une partie des attaques qui peuvent nous parvenir.

Nos différents scripts de pare-feu, de gestion d'un mail entrant, associé au webmail, permettent à un utilisateur d'accéder aux fonctions usuelles associées à l'envoi et la réception de mails.

Grâce à sa configuration personnalisable, notre projet devrait pouvoir être porté sur un domaine et une adresse IP différente sans soucis, et être déployé partout où des gens souhaitent conserver leur vie privée.

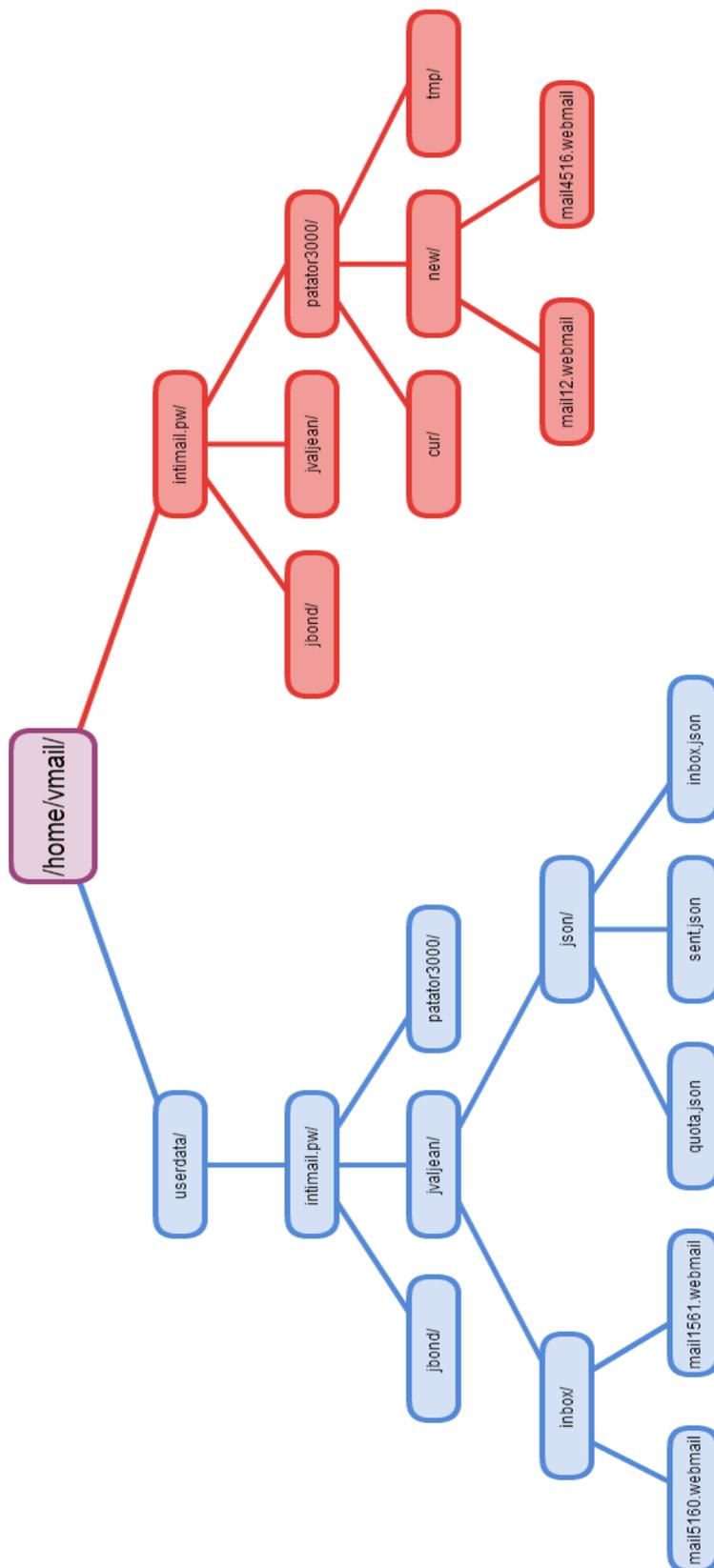
Nous regretterons cependant de ne pas avoir exploité la partie énergétique de ce projet qui portait une problématique intéressante et qui nous paraît importante dans le monde actuel.

Ce projet nous aura occupé pendant très longtemps et nous sommes fiers du travail que nous avons accompli. Il est à nos yeux la preuve de ce que nous savons et pouvons faire, et nous donnera confiance pour notre début de carrière d'ingénieur.

## Sources

- <http://www.radicati.com/wp/wp-content/uploads/2014/01/Email-Statistics-Report-2014-2018-Executive-Summary.pdf>
- <http://www.commentcamarche.net/contents/525-le-protocole-ldap>
- [https://fr.wikipedia.org/wiki/Simple\\_Mail\\_Transfer\\_Protocol](https://fr.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol)
- <http://www.commentcamarche.net/contents/536-pop3-smtp-imap-protocoles-de-messagerie>
- <https://tools.ietf.org/html/rfc2821>
- [https://fr.wikipedia.org/wiki/Internet\\_Message\\_Access\\_Protocol](https://fr.wikipedia.org/wiki/Internet_Message_Access_Protocol)
- <https://tools.ietf.org/html/rfc3501>
- <http://arstechnica.com/information-technology/2014/02/how-to-run-your-own-mail-server-with-your-own-domain-part-1/>
- <https://www.raspberrypi.org/forums/viewtopic.php?p=136912>
- <http://stackoverflow.com/questions/17394356/how-can-i-make-a-bash-command-run-periodically>
- <http://untroubled.org/nullmailer/>
- [http://elinux.org/RPi\\_Hardware](http://elinux.org/RPi_Hardware)
- <http://raspi.tv/2015/raspberry-pi2-power-and-performance-measurement>

## Annexe 1 : Architecture du système de fichiers



## Annexe 2 : Scan DNSSTUFF

DNSreport Results for intimail.pw		Export
Overall Results:		
0	4	29
FAIL	WARNING	PASS
		3
		INFO
▼ PARENT		
Status	Test Name	Information
WARN	Parent zone provides NS records	<p>Parent zone does not provide glue for nameservers, which will cause delays in resolving your domain name. The following nameserver addresses were not provided by the parent 'glue' and had to be looked up individually. This is perfectly acceptable behavior per the RFCs. This will usually occur if your DNS servers are not in the same TLD as your domain (for example, a DNS server of "ns1.example.org" for the domain "example.com"). In this case, you can speed up the connections slightly by having NS records that are in the same TLD as your domain.</p> <p>ns6.gandi.net.   No Glue   TTL=3600            ns.intimail.pw.   193.48.57.171   TTL=3600</p>
PASS	Number of nameservers	<p>At least 2 (RFC2182 section 5 recommends at least 3), but fewer than 8 NS records exist (RFC1912 section 2.8 recommends that you have no more than 7). This meets the RFC minimum requirements, but is lower than the upper limits that some domain registrars have on the number of nameservers. A larger number of nameservers reduce the load on each and, since they should be located in different locations, prevent a single point of failure. The NS Records provided are:</p> <p>ns6.gandi.net.   No Glue   TTL=3600            ns.intimail.pw.   193.48.57.171   TTL=3600</p>
▼ NS		
Status	Test Name	Information
PASS	Unique nameserver IPs	<p>All nameserver addresses are unique. The Nameservers provided are nameservers that supply answers for your zone, including those responsible for your mailservers or nameservers A records. If any are missing a name (No Name Provided), it is because they did not send an A record when asked for data or were not specifically asked for that data:</p> <p>ns.intimail.pw.   193.48.57.171            ns6.gandi.net.   No Glue</p>
PASS	All nameservers respond	<p>All nameservers responded. We were able to get a timely response for NS records from your nameservers, which indicates that they are running correctly and your zone (domain) is valid. The Nameservers provided are nameservers that supply answers for your zone, including those responsible for your mailservers or nameservers A records. If any are missing a name (No Name Provided), it is because they did not send an A record when asked for data or were not specifically asked for that data:</p> <p>ns.intimail.pw.   193.48.57.171            ns6.gandi.net.   No Glue</p>
PASS	Open DNS servers	<p>Nameservers do not respond to recursive queries. Your DNS servers do not announce that they are open DNS servers (i.e. answering recursively). Although there is a slight chance that they really are open DNS servers, this is very unlikely. Open DNS servers increase the chances of cache poisoning, can degrade performance of your DNS, and can cause your DNS servers to be used in an attack, so it is important that externally facing DNS servers do not recursively answer queries.</p>

Depuis : <http://www.dnsstuff.com/tools#dnsReport?type=domain&value=intimail.pw>



## Annexe 4 : Envoyer un mail

INT MAIL

Search

Overview

Write a new mail

Inbox

Spam

Sent

Trash

Manage

Writing a new mail

Quick email

Debug tools

Hello, Jean 1

### New mail

**Recipient:** Who do you want to send this email to ?

**Subject:** What's the subject of your email ?

**Attach a file:** Parcourir... Aucun fichier sélectionné.  
(Accepted : .jpg, .png, .gif, .pdf, .zip)

**Message:** Write your email here

Send message

# Annexe 5 : Boite de réception

INTIMAIL

Mailbox / Inbox

Debug tools Quick email Hello, Jean !

Search

Overview

Write a new mail

**Inbox**

Spam Sent Trash Manage

Inbox

<input type="checkbox"/>	Subject	From	Received On	Size
<input type="checkbox"/>	ALL3	r.libaert@gmail.com	2016-02-22 11:27:43	2427
<input type="checkbox"/>	INSTALLER	r.libaert@gmail.com	2016-02-17 18:12:19	8985
<input type="checkbox"/>	avec plusieurs P/s	timothee.teneur@live.fr	2016-02-15 16:31:23	575533
<input type="checkbox"/>	alors ?	timothee.teneur@live.fr	2016-02-15 14:37:19	2774
<input type="checkbox"/>	cache killer c'est une superbe extension !	timothee.teneur@live.fr	2016-02-15 14:32:16	1807
<input type="checkbox"/>	lelelelel	r.libaert@gmail.com	2016-02-15 14:16:32	2522
<input type="checkbox"/>	Lelel	r.libaert@gmail.com	2016-02-12 15:52:32	2562
<input type="checkbox"/>	RE: oui	timothee.teneur@live.fr	2016-02-12 15:50:53	2175
<input type="checkbox"/>	dismik	r.libaert@gmail.com	2016-02-11 15:10:49	2478
<input type="checkbox"/>	lskdjgs	r.libaert@gmail.com	2016-02-11 15:09:22	2474

Showing 1 to 10 of 13 rows 10 records per page

# Annexe 6 : Interface d'administration

**INTMAIL**    Search    Overview    Write a new mail    Manage

**Administration Panel**    Debug tools    Quick email    Hello, Jean !

---

**Modify a user :** Existing users :  [Go 1](#)

---

**Max attachment size**    **Change domain name**

Size :  [Change Size](#)    New domain name :  [Change Domain](#)  
(in bytes, for example 5 Mb = 5000000)    (i.e. intmail.pw)

---

**Create a new account**

First name :     Last name :     Password :  [Create User](#)  
Quota :

---

**Add a group**    **Remove a group**

Group name :  [Add this group](#)    Choose the group to remove :  [Remove this group](#)  
(i.e. assistants)

# Annexe 7 : Gestion d'un utilisateur

INTMAIL

Search

Overview

Write a new mail

Inbox

Spam

Sent

Trash

Manage

Admin Panel

Debug tools

Quick email

Hello, Jean 1

## Administration Panel - jbond

### Change User ID

change uid :

[Change UID](#)

### Change Password

Change user's password :

[Change Password](#)

### Add user to a group

Choose the group to add to :

[Add jbond to this list](#)

### Change User Full Name

change user's name :

[Change Name](#)

### Change Quota

Change user's quota :

[Change Quota](#)

### Remove user from group

Choose the group to remove user from :

[Remove jbond from this list](#)

### Delete account

Yes, I want to delete jbond

# Tutorial installation intimail™

## Table des matières

Après une nouvelle installation de l'OS .....	2
Sécurisation du serveur .....	2
Changement de l'utilisateur par défaut.....	2
Changement du port SSH.....	2
Interdire l'authentification SSH pour root .....	2
Installation de LDAP .....	3
Configuration de base.....	3
Configuration des Schémas.....	4
Constitution de la base de données LDAP .....	5
Installation de Postfix.....	7
Installation de Bind9 .....	8
Installation du paquet Debian et configuration de lighttpd .....	9
Dépendances.....	11
Sources.....	12

## Après une nouvelle installation de l'OS

Exécutez ces commandes pour mettre à jour à la dernière version. Ces opérations peuvent prendre un certain temps. Nous avons utilisé pour cet exemple Raspbian Jessie Lite sur une Raspberry Pi 2.

```
apt-get update
apt-get upgrade && apt-get dist-upgrade
apt-get install linux-headers-$(uname -r)
reboot
```

Puis lancer :

```
raspi-config
```

Étendre au maximum l'espace de stockage en navigant dans les menus :

```
Expand Filesystem
```

Régler une ip statique avec la box, fichier `/etc/network/interfaces` :

```
iface eth0 inet static
    address 192.168.1.31
    netmask 255.255.255.0
    gateway 192.168.1.1
```

Nous avons aussi supprimé `wpa_supplicant`.

## Sécurisation du serveur

### Changement de l'utilisateur par défaut

Changez le nom d'utilisateur par défaut ainsi que son mot de passe.

```
sudo passwd root
```

Sortie et login en tant que root, puis :

```
usermod -l myuname pi
usermod -m -d /home/myuname myuname
groupmod -n myuname pi
passwd myuname
```

Voir [cette adresse](#) pour l'utilisation de clés privées/publiques.

### Changement du port SSH

```
sed -i 's/Port 22/Port 2222/g' /etc/ssh/sshd_config
```

### Interdire l'authentification SSH pour root

```
sed -i 's/PermitRootLogin yes/PermitRootLogin no/g' /etc/ssh/sshd_config
```

# Installation de LDAP

## Configuration de base

Installation du package :

```
apt-get install slapd
```

Choisir un mot de passe administrateur, puis lancer la commande :

```
dpkg-reconfigure slapd
```

Répondre non à la question 1.

Entrez le nom de domaine utilisé à la question 2, sous la forme : domaine.tld.

Entrez ce que vous voulez au 3<sup>ème</sup> écran, le nom de l'organisation ne sera pas utilisé.

Entrez un mot de passe.

Choisissez HDB.

Choisissez si la base de données est supprimée à la désinstallation de slapd.

Déplacez l'ancienne base de données.

N'utilisez pas LDAPv2.

Installer les outils de conversion pour ldap (slaptest, ldapmodify notamment) :

```
apt-get install ldap-utils
```

On peut maintenant vérifier que la configuration est valable :

```
ldapsearch -x -h localhost -b "dc=domaine,dc=tld" -LLL "dc=domaine" dn
```

Cette commande doit retourner quelque chose comme :

```
dn: dc=domaine,dc=tld
```

Il faut maintenant sécuriser le serveur LDAP, en effet on peut y accéder en anonyme avec la commande :

```
ldapsearch -Y EXTERNAL -H ldapi:// -b cn=config  
" (&(objectClass=olcDatabaseConfig) (olcSuffix=dc=domaine,dc=tld)) "
```

Cette commande retourne un certain nombre d'informations, notamment à propos des champs suivants :

```
olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by  
anonymous auth by * none  
olcAccess: {1}to dn.base="" by * read  
olcAccess: {2}to * by * read
```

On va modifier ces champs avec la commande ldapmodify. Pour cela on va créer un fichier ldif :

```
cat > changeAccess.ldif << EOF  
dn: olcDatabase={1}hdb,cn=config  
changetype: modify  
delete: olcAccess  
-  
add: olcAccess  
olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by  
anonymous auth by dn="cn=admin,dc=domaine,dc=tld" write by * none  
-  
add: olcAccess  
olcAccess: {1}to dn.base="" by * read  
-
```

```
add: olcAccess
olcAccess: {2}to * by self write by dn="cn=admin,dc=domaine,dc=tld" write by
* none
-
EOF
```

On peut maintenant appliquer les changements avec ldapmodify :

```
ldapmodify -c -Y EXTERNAL -H ldapi:/// -f changeAccess.ldif
```

La commande devrait vous confirmer que la modification s'est bien effectuée :

```
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={1}hdb,cn=config"
```

On peut vérifier qu'on ne peut plus accéder aux informations avec la commande :

```
ldapsearch -x -c -h localhost -b dc=domaine,dc=tld
```

On peut vérifier la configuration en se connectant en tant qu'administrateur :

```
ldapsearch -c -h localhost -b dc=domaine,dc=tld -D
"cn=admin,dc=domaine,dc=tld" -W
```

## Configuration des Schémas

Nous allons utiliser les schémas de courier-ldap. Pour éviter une manipulation compliquée, vous trouverez directement le schéma avec le fichier authldap.schema. On place ce fichier dans le dossier :

```
mv authldap.schema /etc/ldap/schema
```

Nous créons un fichier qui inclura tous les schémas de base en plus du nouveau. On peut les afficher avec la commande :

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=schema,cn=config" -LLL
"(objectClass=*)" cn
```

Création du fichier de configuration, pour notre exemple cela donne :

```
mkdir /tmp/ldapconf
cat > /tmp/ldapconf/ldap.conf << EOF
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/authldap.schema
EOF
```

On lance ensuite la commande :

```
slaptest -f /tmp/ldapconf/ldap.conf -F /tmp/ldapconf
```

Elle renvoie un message de succès. Cette commande a créé dans le répertoire /tmp/ldapconf une arborescence correspondant à la configuration de notre LDAP.

On modifie le chemin de configuration :

```
sed -i '0,/dn:.*authldap/ s/authldap/authldap,cn=schema,cn=config/'
cn\=\{*}\authldap.ldif
```

On supprime ensuite les 7 dernières lignes du fichier :

```
head -n -7 cn\=\{*}\authldap.ldif > tmp.ldif
mv tmp.ldif cn\=\{*}\authldap.ldif
```

On peut maintenant importer notre fichier de configuration. Pour ce faire :

```
ldapadd -Y EXTERNAL -H ldapi:/// -f
/tmp/ldapconf/cn=config/cn=schema/cn={*}\authldap.ldif
```

Si tout va bien, l'entrée devrait avoir été ajoutée. On peut vérifier avec la commande vue tout à l'heure :

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=schema,cn=config" -LLL
"(objectClass=*)" cn
```

On devrait voir en plus le nouveau schéma authldap.

## Constitution de la base de données LDAP

Le remplissage de la base se fait à l'aide de fichiers ldif. Nous donnons ici un exemple de structure.

Fichier pour la structure de base :

```
dn: dc=mail,dc=domaine,dc=tld
o: intimail.pw
description: Global mail tree
dc: mail
objectClass: top
objectClass: dcObject
objectClass: organization

dn: dc=people,dc=mail,dc=domaine,dc=tld
description: Informations of all users
o: people
dc: people
objectClass: top
objectClass: dcObject
objectClass: organization

dn: dc=groups,dc=mail,dc=domaine,dc=tld
description: All groups of users
o: groups
dc: groups
objectClass: top
objectClass: dcObject
objectClass: organization
```

Fichier pour ajouter un utilisateur :

```
dn: cn=jbond,dc=people,dc=mail,dc=domaine,dc=tld
uid: jbond
mail: jbond@domaine.tld
sn: Moulin
givenName: James
displayName: James Bond
mailbox: domaine.tld/jbond/
quota: 50M
homeDirectory: /home/vmail/
objectClass: top
objectClass: inetOrgPerson
objectClass: CourierMailAccount
userPassword: {SSHA}367PSXiU//Aegy8dpJbPU8OepEf8L5ye
```

Pour obtenir un hash du mot de passe :

```
slappasswd -s password -h {SSHA}
```

On peut par la suite ajouter la configuration à la base de données avec la commande :

```
ldapadd -D "cn=admin,dc=domaine,dc=tld" -W -h localhost -f path/to/file
```

Si les champs proposés ne sont pas suffisants, il faudra alors effectuer les modifications dans le fichier autldap.schema proposé plus haut, et refaire la manipulation de configuration.

De plus, les standards du Mail décrit par l'IETF (RFC822 6.3, RFC1123 5.2.7 and RFC2821 4.5.1) impliquent d'avoir une adresse Postmaster et une Abuse. Pour cela, nous pourrions créer ces adresses avec la méthode vue au-dessus, mais ce n'est pas la meilleure. Il est préférable d'utiliser un alias, comme nous le ferons avec les listes de diffusion. Ici, nous proposons de rediriger ces deux adresses vers une adresse administrateur. Pour cela, la méthode est la même, sauf que le fichier Idif n'est pas tout à fait identique. En voici un exemple :

```
dn: cn=postmaster,dc=groups,dc=mail,dc=domaine,dc=tld
uid: postmaster
mail: postmaster@domaine.tld
sn: Postmaster
displayName: Postmaster
maildrop: admin@domaine.tld
objectClass: top
objectClass: inetOrgPerson
objectClass: CourierMailAlias
```

Pour faire une liste de diffusion, la méthode est la même. Il suffit d'ajouter toutes les adresses destinataires dans le champ maildrop, séparées par des virgules (,).

## Installation de Postfix

Installation des packages :

```
apt-get install postfix postfix-ldap
```

Lors de l'installation, deux écrans défilent :

- Configuration Internet Site
- Valeur par défaut pour le mail name

On crée l'utilisateur qui servira à exécuter Postfix :

```
groupadd vmail  
useradd -g vmail -d /home/vmail -s /bin/false -m vmail
```

Nous aurons besoin dans la suite des GID et UID créés à cet instant. Ils peuvent être affichés avec les commandes :

```
cat /etc/group | grep vmail | cut -d: -f3  
cat /etc/passwd | grep vmail | cut -d: -f3
```

On ajoute les paramètres au fichier `/etc/postfix/main.cf` (remplacer les valeurs de GID et d'UID avec vos valeurs) :

```
smtpd_banner = mail.domaine.tld  
virtual_mailbox_base = /home/vmail  
virtual_mailbox_domains = domaine.tld  
virtual_mailbox_maps = ldap:/etc/postfix/ldap_accounts.cf  
virtual_alias_maps = ldap:/etc/postfix/ldap_aliases.cf  
virtual_minimum_uid = 100  
virtual_gid_maps = static:1001 # Remplacez ici par votre valeur de GID  
virtual_uid_maps = static:1001 # Remplacez ici par votre valeur d'UID
```

Il nous faut maintenant constituer nos fichiers de configuration, `ldap_accounts.cf` :

```
cat > /etc/postfix/ldap_accounts.cf << EOF  
server_host = localhost  
server_port = 389  
search_base = dc=people,dc=mail,dc=domaine,dc=tld  
query_filter = (&(objectClass=CourierMailAccount) (mail=%s))  
result_attribute = mailbox  
bind = yes  
bind_dn = cn=admin,dc=domaine,dc=tld  
bind_pw = <mdp_en_clair>  
version = 3  
EOF
```

Et pour `ldap_aliases.cf` :

```
cat > ldap_aliases.cf << EOF  
server_host = localhost  
server_port = 389  
search_base = dc=groups,dc=mail,dc=domaine,dc=tld  
query_filter = (&(objectClass=CourierMailAlias) (mail=%s))  
result_attribute = maildrop  
bind = yes  
bind_dn = cn=admin,dc=domaine,dc=tld  
bind_pw = <mdp_en_clair>  
version = 3  
EOF
```

On peut maintenant vérifier la configuration de Postfix avec (cette commande ne doit rien retourner si c'est bon) :

```
postfix check
```

Si c'est bon, alors on recharge le serveur :

```
postfix reload
```

On peut alors faire un envoi de mail test via telnet :

```
telnet localhost 25
```

Si vous souhaitez délivrer les mails à votre manière, il faut dans le fichier `/etc/postfix/main.cf` la ligne :

```
virtual_transport = nomdevotreregle
```

Et de même, dans le fichier `/etc/postfix/master.cf` :

```
nomdevotreregle    unix    -    n    n    -    5    pipe
  flags=Rq user=vmail null_sender=
  argv=/chemin/de/votre/programme
```

Il nous reste maintenant à configurer les différents enregistrements DNS nécessaire au fonctionnement d'un serveur mail. Vous pouvez utiliser la solution de votre choix. Dans la suite vous trouverez une méthode utilisant Bind9.

## Installation de Bind9

Dans toute cette partie, on se situera dans le dossier `/etc/bind/`. Dans la suite, notre IP sera a.b.c.d.

Dans le fichier `named.conf.options`, on ajoutera dans le champ options :

```
version "Not supported";
```

Dans un dossier `zones`, on écrit le fichier `db.domaine.tld` :

```
$TTL      604800
@         IN      SOA      ns.domaine.tld. root.domaine.tld. (
                        2015122111      ; Serial
                        43200             ; Refresh
                        3600              ; Retry
                        2419200           ; Expire
                        86400 )          ; Negative Cache TTL
@         IN      NS       ns.domaine.tld.
@         IN      NS       ns11.ovh.net.
ns        IN      A        a.b.c.d
www       IN      A        a.b.c.d
@         IN      A        a.b.c.d

@         IN      MX       10 mail.domaine.tld.
mail     IN      A        a.b.c.d
```

Dans le même dossier `zones`, on a `c.b.a.in-addr.arpa` :

```

$TTL      604800

@         IN          SOA      ns.domaine.tld. root.domaine.tld.  (
          2015122101      ;serial
          14400           ;refresh
          3600            ;retry
          604800          ;expire
          10800           ;minimum
)

c.b.a.in-addr.arpa.      IN      NS      ns.domaine.tld.
c.b.a.in-addr.arpa.      IN      NS      ns11.ovh.net.

d                       IN      PTR     mail.domaine.tld.

```

Il faut faire attention à bien incrémenter le serial pour chaque modification des fichiers.

Dans le fichier named.conf.local :

```

zone "domaine.tld" {
    type master;
    file "/etc/bind/zones/db.domaine.tld";
    allow-transfer { 213.251.128.130; };
    notify yes;
};

zone "c.b.a.in-addr.arpa" IN {
    type master;
    file "/etc/bind/zones/c.b.a.in-addr.arpa";
    allow-transfer { 213.251.128.130; };
};

```

Ici, 213.251.128.130 est l'adresse IP du serveur ns11.ovh.net.

Il faut bien sûr configurer le DNS avec votre fournisseur (Gandi, OVH, ...). Pour ça, amusez-vous bien. Vous pouvez aussi tester votre configuration avec des outils en ligne tels que DNSstuff.

En test final, si votre configuration est bonne, vous devriez pouvoir envoyer un mail depuis un service neutre (Gmail, yahoo, ...) et le voir dans votre arborescence (ici /home/vmail/domaine.tld/).

## Installation du paquet Debian et configuration de lighttpd

Commencez par récupérer le paquet .deb à l'adresse suivante : <https://www.intimail.pw/intimail.deb>

Installez ensuite GDebi. Il s'agit d'un *dpkg* amélioré pour télécharger les dépendances automatiquement, ce que *dpkg* ne gère pas :

```

apt-get update
apt-get install gdebi-core

```

Vous pouvez alors installer le paquet :

```
gdebi intimail.deb
```

Vous aurez alors le site web d'intiMail déployé au chemin */var/www/webmail/*

Il ne vous reste plus qu'à changer la configuration de `lighttpd`, votre serveur web, pour mettre en ligne le site. Vous pouvez pour cela copier-coller la configuration suivante, mais veillez à bien changer l'adresse IP présente au début de la partie SSL CONFIG avec votre adresse IP :

```
server.modules = (
    "mod_access",
    "mod_accesslog",
    "mod_alias",
    "mod_compress",
    "mod_redirect",
    "mod_rewrite",
)

#Hide server version
server.tag = "Such Headers. Very try. Now close telnet."

server.document-root      = "/var/www/webmail/"
server.upload-dirs        = ( "/var/www/webmail/uploads" )
server.errorlog           = "/var/log/lighttpd/error.log"
server.pid-file           = "/var/run/lighttpd.pid"
server.username           = "www-data"
server.groupname          = "www-data"
server.port               = 80
accesslog.format          = "%V %h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User-Agent}i\""
accesslog.filename        = "/var/log/lighttpd/access.log"

index-file.names          = ( "index.php", "index.html",
"index.lighttpd.html" )
url.access-deny           = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )

compress.cache-dir       = "/var/cache/lighttpd/compress/"
compress.filetype         = ( "application/javascript", "text/css",
"text/html", "text/plain" )

# default listening port for IPv6 falls back to the IPv4 port
include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
include_shell "/usr/share/lighttpd/create-mime.assign.pl"
include_shell "/usr/share/lighttpd/include-conf-enabled.pl"

#####
# SSL CONFIG
#####

$SERVER["socket"] == "5.23.44.85:443" {
    ssl.engine = "enable"
```

```
ssl.pemfile = "/etc/lighttpd/ssl/intimail.pw.pem"
ssl.ca-file = "/etc/lighttpd/ssl/intimail.pw.crt"
server.name = "intimail.pw"

# votre configuration habituelle pour ce domaine
server.document-root      = "/var/www/webmail/"
server.errorlog           = "/var/log/lighttpd/error_ssl.log"
server.pid-file           = "/var/run/lighttpd.pid"
server.username           = "www-data"
server.groupname          = "www-data"

#Rewrite des URL pour masquer .php
url.rewrite-once = (  "^(.*)/$" => "$1/" )
url.rewrite-if-not-file = (  "^([^?]*)(\?.*)?$" => "$1.php$2" )

server.network-backend = "writev"
}
```

## Dépendances

Slapd

Ldap-utils

Postfix

Postfix-ldap

Bind9

Lighttpd

## Sources

<https://wiki.gandi.net/fr/hosting/using-linux/tutorials/debian/mail-server-ldap>

<http://www.postfix.org/pipe.8.html>

[http://www.postfix.org/FILTER\\_README.html](http://www.postfix.org/FILTER_README.html)

<http://www.postfix.org/>

<http://www.postfix.org/transport.5.html>