Université du Havre
Master Matis
Spécialisation SIRES

TeXloud
Des documents LaTeX dans le Cloud

Référent : Y. Pigné

Rapport

Adrien Bruyère
David Ducatel
Meva Rakotondratsima
Sidina Biha
Zakaria Bouchakor

15 février 2012

Table des matières

1 Introduction

1.1 Rappel du sujet

Ce projet propose la création et la gestion collaborative de documents Latex. Le but est de proposer à des plateformes dépourvues de distribution Latex (tablettes, smartphones, desktops), de se connecter au Web et d'accéder à ces service de gestion et de compilation de documents.

Les utilisateurs seront authentifiés au service et bénéficieront d'un espace de stockage privé. L'application facilitera le partage de documents et le travail collaboratif entre utilisateurs du service.

Coté client, deux types d'applications seront développés :
- Un service Web permettra l'accès au service à partir de n'importe quelle machine (desktop, tablette non-Android) pourvue d'un navigateur Web et d'une connexion internet
- Une application Android, permettra une certaine autonomie avec le stockage temporaire d'une copie de travail des documents, permettant un mode d'édition non-connecté.

1.2 L'équipe

L'équipe est composé de 5 personnes :
- Adrien Bruyère est responsable du développement de l'application Android
- David Ducatel (chef de projet) est responsable du développement de la frontale et du service de compilation.
- Meva Rakotondratsima est responsable du développement du service de stockage de données.
- Sidina Biha, Zakaria Bouchakor sont responsables du développement de l'application WEB

2 Choix techniques

2.1 Architecture général

Le projet est divisé en 5 parties minimum :
- Un serveur web
- Une application Android
- Un serveur de routage et d'ordonnancement
- Un à N serveur(s) de stockage de données
- Un à N serveur(s) de compilation

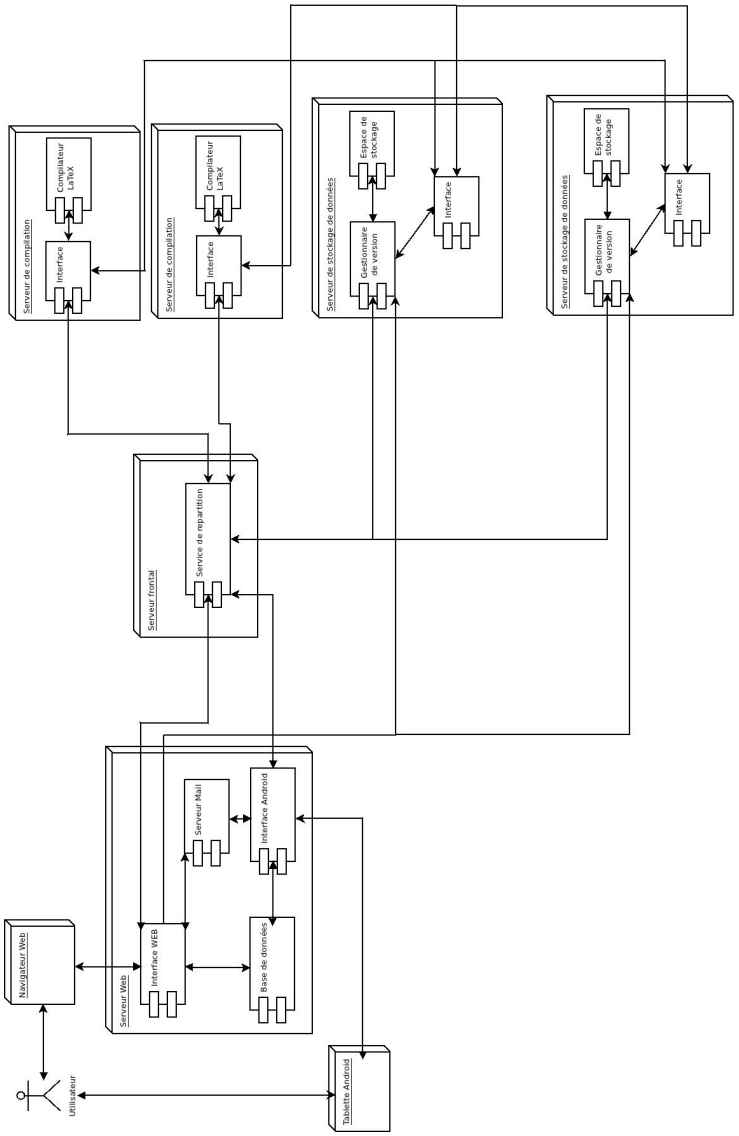

FIGURE 1 – Diagramme de déploiement

Ce type d'architecture à été choisie car elle permet une extension du nombre de serveurs de compilation et de données théoriquement infinie. Aussi, elle permet de limité au maximum les échange "lourd" [1]. L'ensemble des communications (sauf envoie de binaire) sont au format JSON.

Typiquement une compilation de documents passe par l'ensemble des serveurs :

1. Serveur HTTP → serveur de routage et d'ordonnancement
2. Serveur routage et d'ordonnancement → serveur de stockage de données
3. Serveur de stockage de données → serveur de compilation
4. Serveur de compilation → serveur de stockage de données
5. Serveur de stockage de données → serveur HTTP

2.2 Serveur web

Le serveur web inclue plusieurs services :

- La base de données de l'application. Le SGBD MySQL a été choisie pour supporté cette tâche.
- L'interface coté serveur a été développé en PHP5. Le serveur HTTP Apache2 est utilisé pour géré les connexions.
- L'interface cliente utilise les technologies HTML5, CSS3 et javascript (Via le frameworks JQuery).
- L'interface de communication entre l'application Android et le reste de l'architecture.

Un modèle de conception MVC a été utilisé afin de rendre le code de l'application plus clair, plus facilement maintenable et évolutif. Les communications avec le serveur de routage et d'ordonnancement sont effectués grâce à des sockets TCP qui ont une durée de vie égale à une boucle entre tous les serveurs. Ceci permet de ne pas laisser en permanence de ports ouvert sur le serveur et ainsi limité les risques d'attaques sur ce service.

L'application Web est divisé en trois parties :

1. La partie développement, où l'utilisateur peut écrire sont code LaTex, consulter les logs d'erreurs/warning (s'il existe) après la compilation.
2. La partie gestion des projets. L'utilisateur peut crée des projets, des dossier (sous dossiers) et des fichiers LaTex. il peut aussi renommer, supprimer un projet, un dossier ou un fichier.
3. La barre supérieur, où l'utilisateur peut synchronisées les fichiers qui ont été modifier, les compilées, télécharger sous format de PDF. Éventuellement l'utilisateur peut modifier ces informations personnels ou changer son mot de passe.

1. Transfert de nombreux fichiers et donc potentiellement lourd pour le réseau

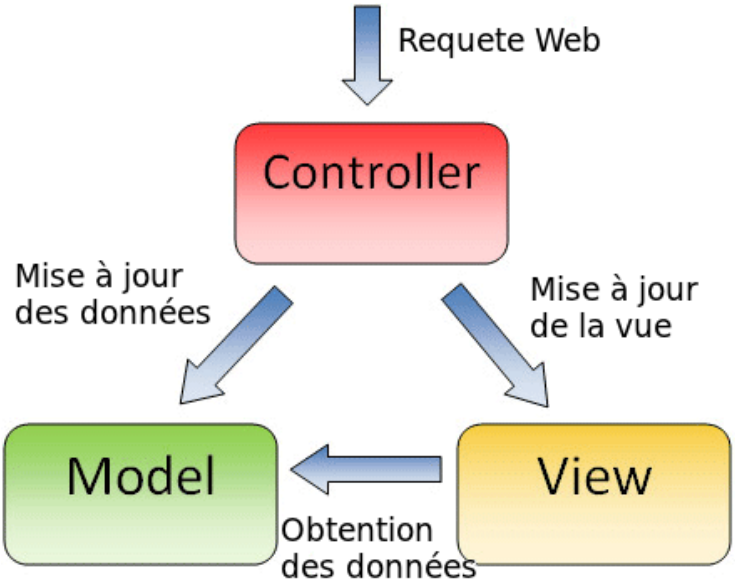

FIGURE 2 – Diagramme MVC

2.3 Application Android

L'application Android comporte deux activités (écrans) principales.

- La première est l'écran d'authentification, qui permet à l'utilisateur de s'authentifier afin d'accéder à son espace personnel, ou de s'inscrire afin de rejoindre le service TeXloud.
- La deuxième activité est lancée lorsque l'utilisateur est identifié. C'est l'écran principal de l'application.

L'écran principal est divisé en trois parties : le bandeau supérieur, qui permet à l'utilisateur quelques interactions indispensables, comme la création de projet, la synchronisation ou encore la compilation. Lorsque l'utilisateur a fait des modifications qui n'ont pas été enregistrées (synchronisées), un logo rouge apparait pour rappeler à l'utilisateur de sauvegarder ses données. Une fois que les documents sont synchronisés, le logo rouge devient vert, l'utilisateur peut alors quitter l'application en étant sûr que ses modifications ont été prises en compte.

La partie de gauche contient deux éléments : une liste déroulante (Spinner), qui permet à l'utilisateur de naviguer entre ses différents projets, et une arborescence qui permet à l'utilisateur de parcourir ses dossiers et fichiers. Un clic (appui) sur un dossier permet de masquer/afficher tous les fichiers et dossiers enfants (récursivement), et un clic sur un fichier le télécharge et l'affiche, permettant à l'utilisateur de le modifier. Les long clics

sur les éléments de l'arborescence ouvrent un menu contextuel permettant, par exemple, d'ajouter un fichier, renommer, supprimer etc...

La partie de droite est la saisie de texte. Le contenu de cette zone de texte est le contenu du fichier ouvert, affiché en italique dans l'arborescence de droite.Pour des raisons de simplicité, les ordres de synchronisation et de compilation sont bloquants, c'est-à-dire que l'utilisateur ne peut effectuer aucune modification tant que ces actions ne sont pas terminées.Toutes les requêtes utilisent le protocole HTTP et envoie les données via POST.

2.4 Serveur de routage et d'ordonnancement

Le serveur de routage et d'ordonnancement permet de géré les communications entre le serveur web et les différents serveur de données et de compilation. Ce service à été développé en python. Il fourni deux services :

1. Un service de routage
2. Un service d'ordonnancement

Le service de routage permet de dirigé les requêtes vers les bon serveurs en fonctions de l'action voulu (compilation vers un serveurs de compilation,gestion des données vers le bon serveur de données). Le service d'ordonnancement permet de répartir la charge sur les différents serveurs en fonction de leurs charge propre. L'ordonnancement permet ainsi d'éviter d'avoir des serveurs totalement au repos et d'autres au maximum de leurs capacité.

Ce service permet de gérer plusieurs connexion simultané. Il a été mit en charge avec plus de 1000 connexions simultané et aucun problème n'as été constaté.

Les communications entre le serveur HTTP, les serveurs de compilation et les serveurs de données sont effectués la aussi grâce à des socket TCP.

L'ensemble des données sur les serveurs (adresse IP, port du service, type de service, charge maximal) sont stocké au sein d'un fichier XML sur ce serveur.

2.5 Serveur de stockage de données

Lorsqu'un projet est crée sur TeXloud, un serveur de stockage lui est attribué lors de la création du projet par l'ordonnanceur. Il s'agit en fait d'un parc de serveurs physiques disposant d'un démon serveur destiné à recevoir de la frontale des requêtes standardisées permettant de faire les traitements demandés.

L'application serveur est développée en python et dispose d'une socket en écoute sur un port spécifique. La séparation des composant de l'application permet de donner un certaine scalabilité de l'application, les méthodes d'accès aux données peuvent donc différer selon le

serveur concerné. De plus, le stockage des fichier se fait à l'aide de gestionnaires de versions afin de permettre le travail en équipe et une gestion de version éventuelle.

Le service d'accès aux données se compose principalement de eux parties :
- Un démon serveur recevant les requêtes
- Un connecteur permettant de faire les actions (notamment le stockage)

Chacune de ces parties sont implémentées par héritage, les deux classes *DataSocket* et *GenericConnector* définissent les méthodes nécessaires pour le fonctionnement complet du système et font éventuellement les traitements non spécifiques au type de stockage. L'application est donc potentiellement capable d'ajouter des connecteurs spécifique pour un type de stockage différent en développant les fonctions nécessitant un traitement adapté à celui-ci.

Les fonctions du connecteur subversion sont intégrés par l'utilisation de la librairie Python PySvn permettant de faire les opération proposé par le système de gestion de version telle que les *commit*, les *merge* et les *updates*.

Le concept de *copies de travail* est au centre du système de gestion des données. Les systèmes de gestions de versions permettent de récupérer une version d'un projet depuis un serveur distant en créant un *miroir* du projet dans un état spécifique (généralement le dernier).

Le répertoire ainsi crée correspond à une copie de travail, lorsque des modification sont faites sur le projet, elles sont ajoutées puis mise à jour sur le serveur, créant ainsi une nouvelle version du projet.

À chaque connexion d'un utilisateur, une requête est envoyé par le serveur web jusqu'au serveur de donnée visé qui initialise une nouvelle copie de travail sur le serveur, il s'agit du répertoire de travail associé à l'utilisateur. Les modifications effectuées au cours de la session de l'utilisateur sont synchronisées sur ce répertoire par écrasement des fichiers puis sont transmises aux gestionnaire de version.

2.6 Serveur de compilation

Le serveur de compilation, comme son nom l'indique permet de compiler le projet LaTeX afin de générer un fichier PDF. Le processus du service de compilation se déroule en 5 étapes :

1. Réception d'une archive qui contient tous les fichiers latex du projet
2. Décompression de l'archive
3. Compilation du projet [2]

2. La gestion du nombre de compilation (édition des liens, bibtex,etc.) nécessaire est effectués par le script Perl latexmk

4. Génération d'un fichier de log au format XML
5. Renvoi au serveur de données du log et du fichier PDF

Ce service est la aussi développé en Python et les communications sont basé sur des socket TCP. De même que pour le service de routage et d'ordonnancement, le service de compilation peut gérer plusieurs connexion en simultané.

3 Utilisation des applications

3.1 Application Web

3.1.1 Création de compte

3.1.2 Création de projet

3.1.3 Compilation

- Lancement compilation
- log
- visu du pdf
- telechargement du pdf

3.1.4 Fonctions annexes

- Suppression de dossier/fichier
- Rename dossier/fichier
- Info supp sur user

3.2 Application Android

3.2.1 Création de compte

3.2.2 Création de projet

3.2.3 Compilation

- Lancement compilation
- log
- visu du pdf
- telechargement du pdf

3.2.4 Fonctions annexes

- Suppression de dossier/fichier
- Rename dossier/fichier
- Info supp sur user

4 Perspective d'évolution

4.1 Support des images

Le supports des images dans l'application permettrai d'ajouter des images au sein des PDF généré. L'ajout de cette fonctionnalité est très simple et rapide à mettre en œuvre mais malheureusement, nous n'avons pas eu le temps de la mettre en place. Pour intégrer ce support, une légère modification de la frontale, du service web et de l'application Android sont nécessaire.

4.2 Gestion des groupes

La gestion des groupes dans l'application permettrai la modification simultané de document par plusieurs utilisateurs. Cette modification est possible à intégrer par la suite car la base de données et le stockage des informations ont été prévue a cette effet [3]. Comme pour le support des images, les modifications de l'application pour supporté cette fonctionnalité sont minime. Il s'agit seulement de modifier légèrement l'application Web et l'application Android.

4.3 Coloration syntaxique sur Android

La coloration syntaxique sur tablette Android n'as pas été implémenté car nous n'avons pas trouvez de solution qui ne soit pas trop lourde en ressource pour une tablette. Ceci est sûrement possible si nous déclenchions la coloration syntaxique moins fréquemment, comme par exemple a chaque fin de ligne et juste sur la ligne en cour. Malheureusement nous n'avons pas eu le temps de mettre en place cette idée.

4.4 Intégration de PDFJS

La librairie PDFJS [4] étant pour l'instant en version de développement, il nous a été impossible de l'intégrer directement au sein d'une page de TeXloud. Nous avons cherché pendant plus de 10 Heures pour l'intégrer mais le code ne contient aucune documentation et évolue de jour en jour.Le support serai possible mais demande des modifications très lourde des script de PDFJS. De ce fait nous pensons que lorsque une release stable sera produit par l'équipe de Mozilla, il sera plus simple de l'intégrer au sein de notre design.

4.5 Mot de passe oublié

La gestion des mots de passes oubliés n'est pas encore implémentée puisqu'elle dépend de la gestion des e-mails, et nous n'avons pas développé cette fonction. Cependant, le bouton

3. Le stockage des données étant effectués sur des serveur SubVersion et la gestion des conflit étant implémenté, la gestion des groupe ne posera aucun problème

4. PDFJS est un librairie permettant d'afficher un PDF dans un canvas HTML5 grâce à du JavaScript

et la boite de dialogue sont déjà implémentés sur l'interface Android. La gestion des e-mails n'as pas été possible car nous n'avions pas l'architecture physique nous permettant de faire fonctionné convenablement un serveur mail [5].

4.6 Utilisation d'autre support de stockage

Il serait possible d'utiliser plusieurs gestionnaire de version différents, des systèmes de fichiers,etc. pour stocker les données des utilisateurs. Pour l'instant seul SubVersion est géré, mais grâce a une interface générique, il est possible de développer rapidement des connecteurs pour n'importe quel gestionnaire de version. Grâce à cette interface, il suffit juste de développer une classe fille de cette interface sur un serveur de stockage de données et faire les appels spécifique au gestionnaire de version voulu.

5. Pour être dans de bonne condition, il aurai fallu que nous puissions avoir un accès vers internet afin de contacté les serveur SMTP ou POP de différents fournisseur d'adresse mail

5 Conclusion

Table des figures