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

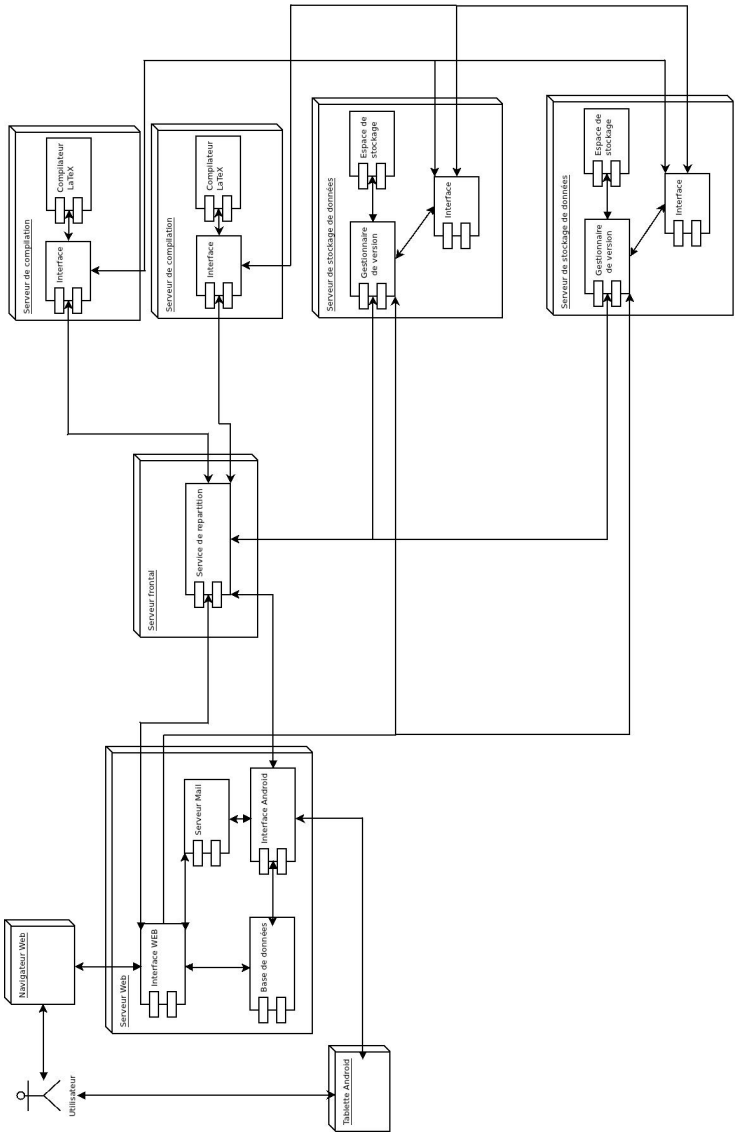

FIGURE 1 – Diagramme de déploiement

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 deux 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.

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

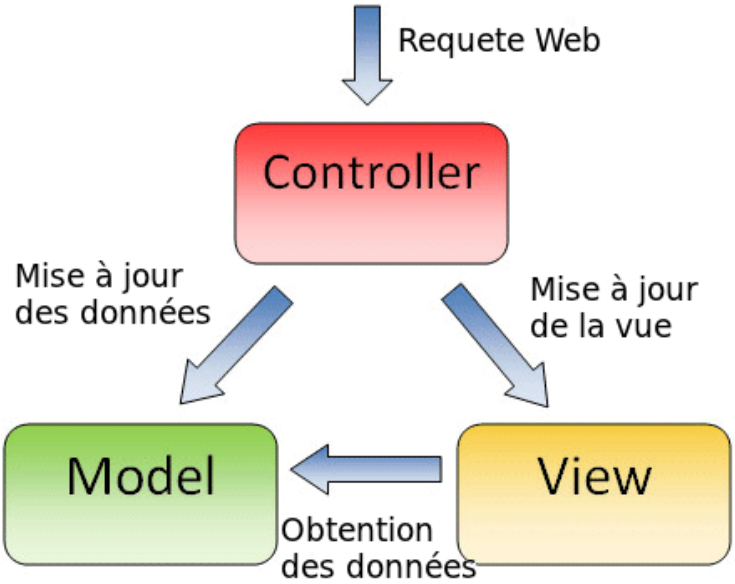

FIGURE 2 – Diagramme MVC

2.3 Application Android

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

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]
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.

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

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.

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

Table des figures