

Université du Havre  
Master Matis  
Spécialisation SIREs

# TeXcloud

## Des documents L<sup>A</sup>T<sub>E</sub>X dans le Cloud

RÉFÉRENT : Y. PIGNÉ

### Rapport

Adrien Bruyère  
David Ducatel  
Meva Rakotondratsima  
Sidina Biha  
Zakaria Bouchakor

---

15 février 2012

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Rappel du sujet . . . . .	2
1.2	L'équipe . . . . .	2
<b>2</b>	<b>Choix techniques</b>	<b>3</b>
2.1	Architecture général . . . . .	3
2.2	Serveur web . . . . .	4
2.3	Application Android . . . . .	5
2.4	Serveur de routage et d'ordonnancement . . . . .	6
2.5	Serveur de stockage de données . . . . .	6
2.6	Serveur de compilation . . . . .	7
<b>3</b>	<b>Utilisation des applications</b>	<b>9</b>
3.1	Application Web . . . . .	9
3.1.1	Création de compte . . . . .	9
3.1.2	Création de projet . . . . .	9
3.1.3	Compilation . . . . .	9
3.1.4	Fonctions annexes . . . . .	9
3.2	Application Android . . . . .	9
3.2.1	Création de compte . . . . .	9
3.2.2	Création de projet . . . . .	9
3.2.3	Compilation . . . . .	9
3.2.4	Fonctions annexes . . . . .	10
<b>4</b>	<b>Perspective d'évolution</b>	<b>11</b>
4.1	Support des images . . . . .	11
4.2	Gestion des groupes . . . . .	11
4.3	Coloration syntaxique sur Android . . . . .	11
4.4	Intégration de PDFJS . . . . .	11
4.5	Mot de passe oublié . . . . .	11
4.6	Utilisation d'autre support de stockage . . . . .	12
<b>5</b>	<b>Conclusion</b>	<b>13</b>

# 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

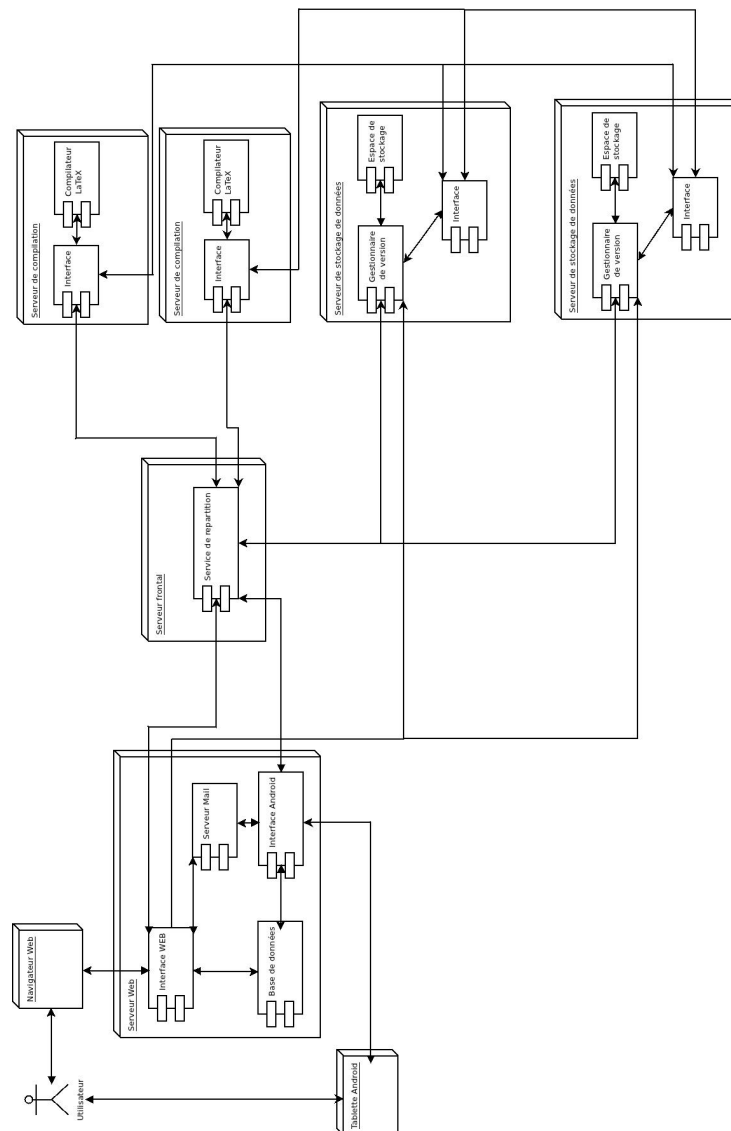


FIGURE 1 – Diagramme de déploiement

Ce type d'architecture a été choisie car elle permet une extension du nombre de serveurs de compilation et de données théoriquement infinie. Aussi, elle permet de limiter au maximum les échanges "lourds"<sup>1</sup>. L'ensemble des communications (sauf envoi 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 inclut plusieurs services :

- La base de données de l'application. Le SGBD MySQL a été choisi pour supporter cette tâche.
- L'interface côté serveur a été développée en PHP5. Le serveur HTTP Apache2 est utilisé pour gérer les connexions.
- L'interface cliente utilise les technologies HTML5, CSS3 et javascript (Via le framework 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 effectuées 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 ouverts sur le serveur et ainsi limiter les risques d'attaques sur ce service.

L'application Web est divisée en trois parties :

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

---

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

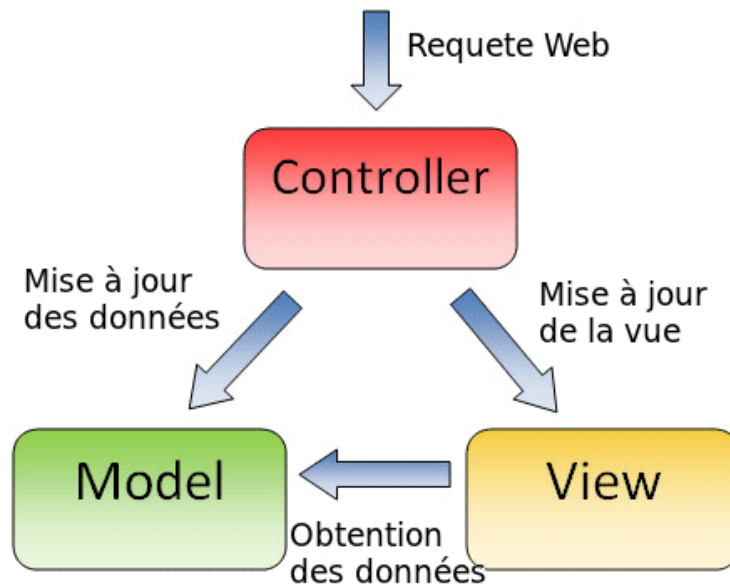


FIGURE 2 – Diagramme MVC

## 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 apparaît 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 envoient les données via POST.

## 2.4 Serveur de routage et d'ordonnancement

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

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

Le service de routage permet de diriger les requêtes vers les bons serveurs en fonction de l'action voulue (compilation vers un serveur 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 leur charge propre. L'ordonnancement permet ainsi d'éviter d'avoir des serveurs totalement au repos et d'autres au maximum de leur capacité.

Ce service permet de gérer plusieurs connexions simultanées. Il a été mis en charge avec plus de 1000 connexions simultanées et aucun problème n'a été constaté.

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

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

## 2.5 Serveur de stockage de données

Lorsqu'un projet est créé 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 part 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 composants de l'application permet de donner une certaine scalabilité de l'application, les méthodes d'accès aux données peuvent donc différer selon le

serveur concerné. De plus, le stockage des fichiers 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 deux 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 spécifiques pour un type de stockage différent en développant les fonctions nécessitant un traitement adapté à celui-ci.

Les fonctions du connecteur subversion sont intégrées par l'utilisation de la librairie Python PySvn permettant de faire les opérations proposées 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 gestion 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 créé correspond à une copie de travail, lorsque des modifications sont faites sur le projet, elles sont ajoutées puis mises à jour sur le serveur, créant ainsi une nouvelle version du projet.

À chaque connexion d'un utilisateur, une requête est envoyée par le serveur web jusqu'au serveur de données 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 gestionnaires 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<sup>2</sup>

---

2. La gestion du nombre de compilation (édition des liens, bibtex, etc.) nécessaire est effectuée 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 simultanée 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 à cet effet<sup>3</sup>. 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 trouvé 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 à chaque fin de ligne et juste sur la ligne en cours. Malheureusement nous n'avons pas eu le temps de mettre en place cette idée.

### 4.4 Intégration de PDFJS

La librairie PDFJS<sup>4</sup> étant pour l'instant en version de développement, il nous a été impossible de l'intégrer directement au sein d'une page de TeXcloud. 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 sera possible mais demande des modifications très lourdes des scripts de PDFJS. De ce fait nous pensons que lorsque une release stable sera produite 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 effectué sur des serveurs SubVersion et la gestion des conflits étant implémenté, la gestion des groupes ne posera aucun problème

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

et la boîte 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<sup>5</sup>.

## 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

Le projet, bien qu'il ne soit pas totalement terminé, est stable, viable et utilisable. Ce projet nous a permis de nous familiariser avec de nouvelles technologies, notamment en PHP5 et en Python. Aussi cela nous a permis de nous impliquer dans des conceptions distribuées et ainsi nous formé sur une application réelle complexe et complète.

Le projet a subit de fortes contraintes au niveau de l'architecture physique qui limitait grandement notre possibilité de travail. Ces contraintes nous ont forcé à travailler en local. Par conséquent nous avons du travailler sur les horaires d'ouverture de la faculté et lorsque la salle n'était pas occupé pour des cours. Aussi nous avons eu des problème quand à la formation de notre "cloud". En effet, nous avons du utilisé les PC de la salle A104 (Résultat 2 serveurs en moins par rapport à notre architecture de base). De plus ils nous a fallu trouver un routeur WIFI afin de faire fonctionner la tablette avec notre "Cloud".

Pour finir nous pouvons dire que ce projet complexe et complet a été très intéressant à mettre en place. Aussi, nous avons prêté un soin particulier à produire un code propre, compréhensible, documenté et donc facilement réutilisable afin que de nombreuses fonctionnalités puissent être facilement intégrables.

## Table des figures

1	Diagramme de déploiement . . . . .	3
2	Diagramme MVC . . . . .	5