

Université du Havre
Master Matis
Spécialisation SIREs

TeXcloud

Des documents L^AT_EX dans le Cloud

RÉFÉRENT : Y. PIGNÉ

Rapport

Adrien Bruyère
David Ducatel
Meva Rakotondratsima
Sidina Biha
Zakaria Bouchakor

21 février 2012

Table des matières

1	Introduction	2
1.1	Rappel du sujet	2
1.2	L'équipe	2
2	Choix techniques	3
2.1	Architecture générale	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	7
2.6	Serveur de compilation	8
3	Utilisation des applications	9
3.1	Application Web	9
3.1.1	Création de compte	9
3.1.2	Création de projet	11
3.1.3	Création de fichier	12
3.1.4	Compilation	13
3.1.5	Fonctions annexes	16
3.2	Application Android	17
3.2.1	Page d'accueil	17
3.2.2	Création de compte	18
3.2.3	Création de projet	19
3.2.4	Choisir un projet	20
3.2.5	Synchronisation	21
3.2.6	Compilation	22
3.2.7	Ouverture des PDF	24
3.2.8	Fenêtres de chargement	24
4	Perspectives d'évolution	25
4.1	Support des images	25
4.2	Gestion des groupes	25
4.3	Coloration syntaxique sur Android	25
4.4	Mot de passe oublié	25
4.5	Utilisation d'autres supports de stockage	25
4.6	Sécurisation des transferts de données	26
4.7	Mode offline pour Android	26
5	Conclusion	27

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 de tout OS) 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 générale

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

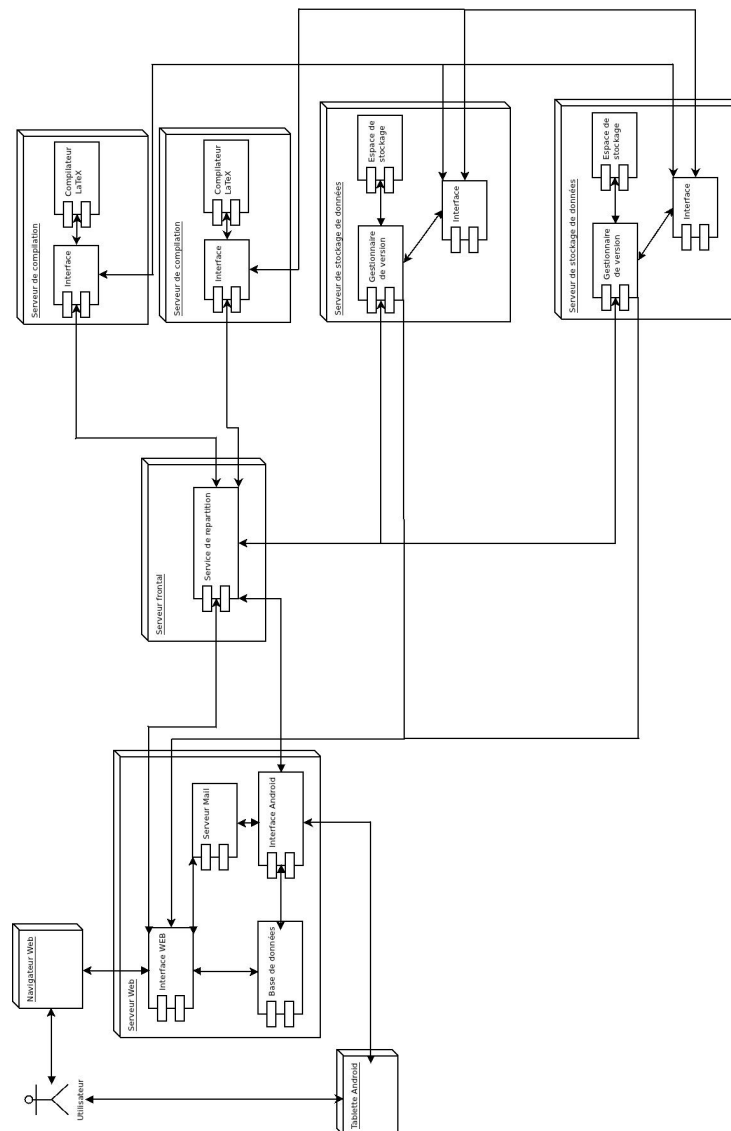


FIGURE 1 – Diagramme de déploiement

Ce type d'architecture a été choisi car il permet une extension du nombre de serveurs de compilation et de données théoriquement infinie. Aussi, elle permet de limiter au maximum les échanges "lourds"¹. L'ensemble des communications (sauf envoi de binaire) est 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'attaque 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 ses informations personnelles ou changer son mot de passe.

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

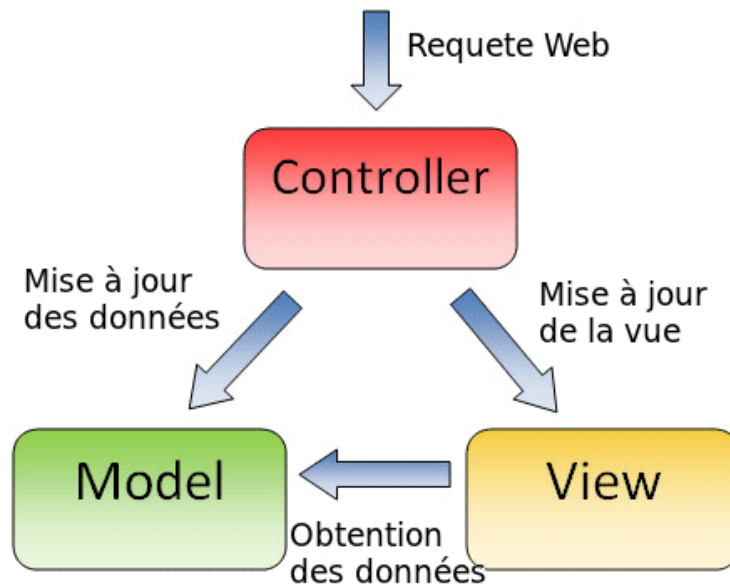


FIGURE 2 – Diagramme MVC

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 pour 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 là aussi grâce à des sockets TCP.

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

Il faut aussi indiquer que ce serveur génère des logs afin que le suivi de l'exploitation de cette partie puisse être effectué le plus simplement possible. Les logs générés sont en rotation, c'est-à-dire qu'ils ont une taille maximale fixée et qu'ils s'écrivent sur plusieurs fichiers².

2. Sur trois fichiers actuellement

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 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 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 projet³
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 là aussi développé en Python et les communications sont basées sur des socket TCP. De même que pour le service de routage et d'ordonnancement, le service de compilation peut gérer plusieurs connexions en simultané.

Ici aussi des logs en rotation sont générés afin de faciliter l'exploitation de ce service de compilation.

3. 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

Pour commencer à utiliser le projet TeXloud, un utilisateur doit forcément passer par l'inscription sur le service. Pour ce faire, il doit se rendre sur la page d'accueil du site et cliquer sur le lien "s'inscrire".

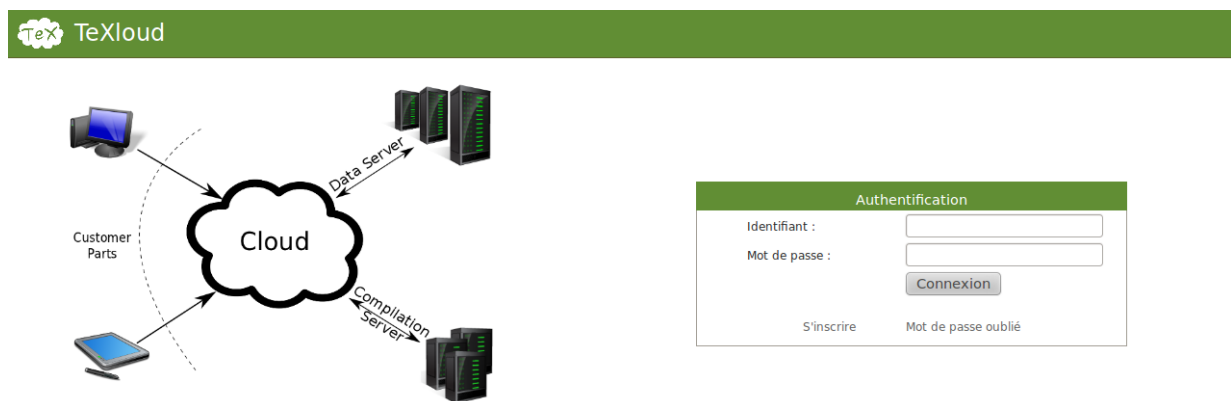
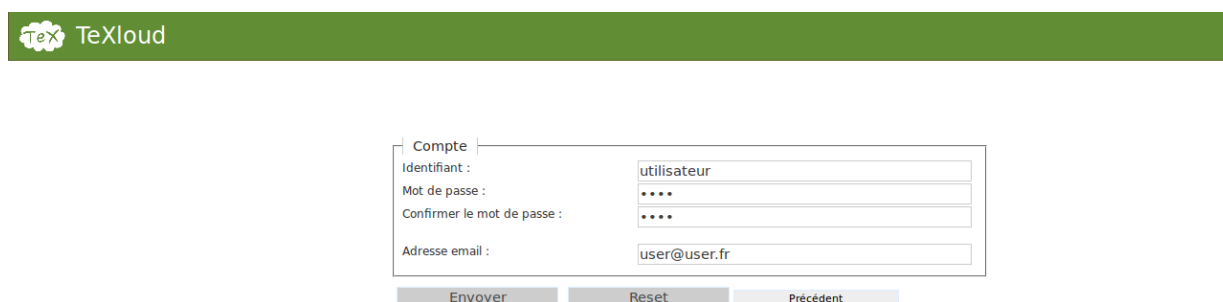


FIGURE 3 – Accueil, application web

L'utilisateur est alors redirigé vers la page d'inscription qui lui demande quelques informations :

- Un identifiant (Login)
- Un mot de passe
- Une adresse email

Une fois ces informations renseignées, l'utilisateur peut valider son inscription grâce au bouton prévu à cet effet. Si il y a une erreur sur ses informations, l'utilisateur est redirigé sur la page d'inscription avec une indication sur la provenance de l'erreur.



The screenshot shows the registration form on the TeXcloud website. At the top is a green header with the TeXcloud logo. Below it is a form titled 'Compte' with four input fields: 'Identifiant' (containing 'utilisateur'), 'Mot de passe' (containing four dots), 'Confirmer le mot de passe' (containing four dots), and 'Adresse email' (containing 'user@user.fr'). At the bottom of the form are three buttons: 'Envoyer', 'Reset', and 'Précédent'.

FIGURE 4 – Inscription, application web

Une fois son inscription terminée, l'utilisateur est redirigé vers la page d'accueil ou il doit se connecter à l'interface. La connexion redirige l'utilisateur vers l'interface principale de l'application. À ce moment, l'utilisateur ne dispose d'aucun projet. Nous allons voir par la suite comment faire pour créer un projet et des fichiers LaTeX.

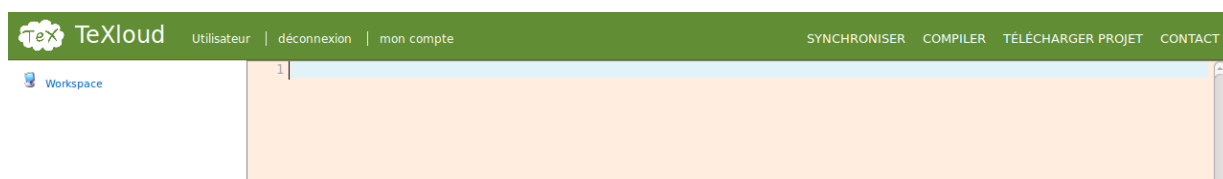


FIGURE 5 – Première page, application web

3.1.2 Création de projet

La création d'un projet s'effectue via la partie de gauche de l'application, l'arbre. Pour créer un nouveau projet, il suffit de faire un clic droit sur la racine de l'arbre "workspace" et de choisir dans la popup "créer un projet".

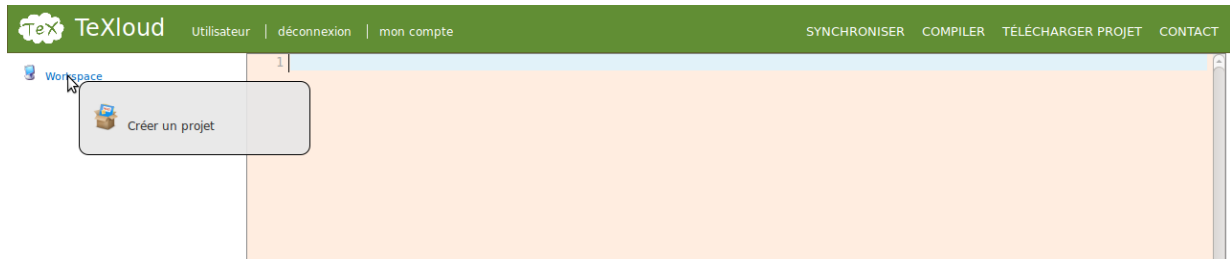


FIGURE 6 – Création projet, application web

Une fois que l'utilisateur a cliqué sur "créer un projet", la popup change afin que l'utilisateur puisse entrer le nom qu'il souhaite pour son projet. Une fois ceci terminé, il suffit de valider.

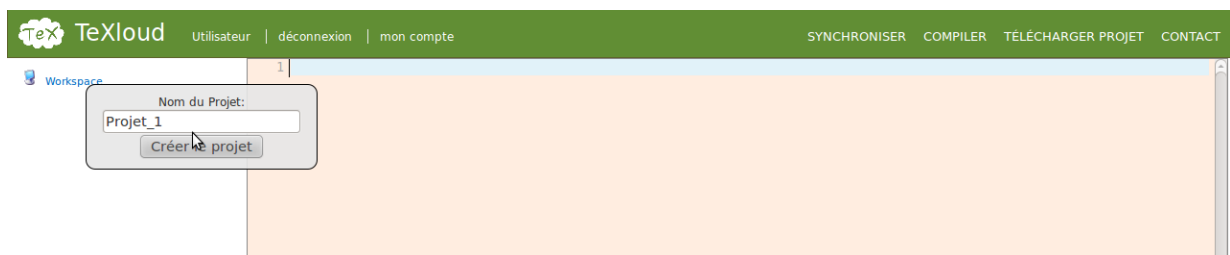


FIGURE 7 – Création projet (nom du projet), application web

Une fois que l'utilisateur a créé son projet, il peut le voir afficher dans l'arbre sous la racine "workspace". A partir de ce point l'utilisateur peut créer des fichiers latex et des sous-dossiers s'il le souhaite.

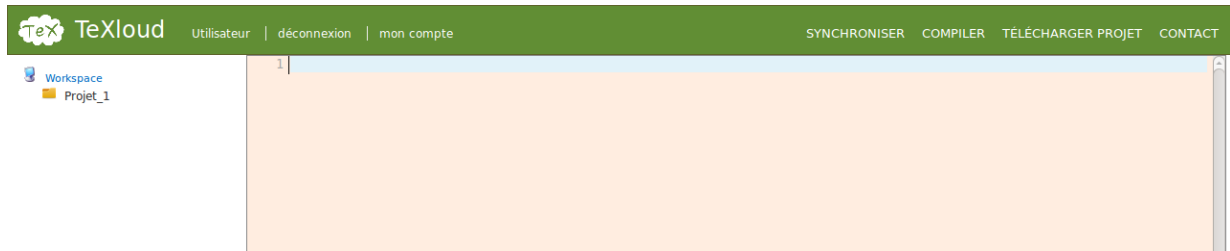


FIGURE 8 – Création projet terminé, application web

3.1.3 Création de fichier

Nous avons vu dans la partie précédente comment créer un projet, donc nous allons voir maintenant comment faire pour créer un fichier latex. Pour ce faire, il suffit de faire un clic droit sur le projet et un popup apparaît. Dans celle-ci, il suffit de choisir "créer un fichier".

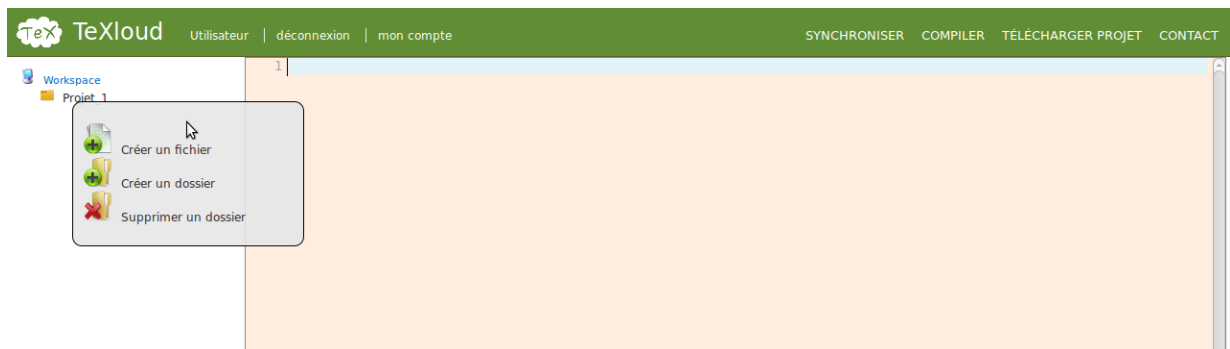


FIGURE 9 – Création de fichier, application web

Une fois ceci fait, l'utilisateur doit entrer le nom de son fichier *avec son extension .tex* puis valider. Enfin l'utilisateur voit apparaître sous son projet son nouveau fichier.tex.

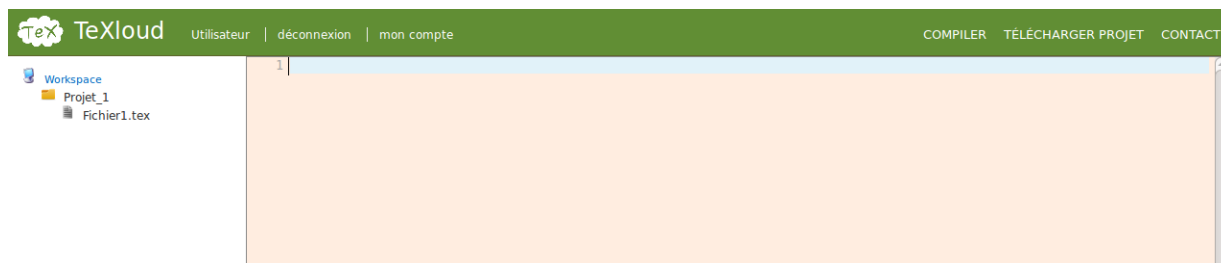


FIGURE 10 – Création de fichier terminée, application web

3.1.4 Compilation

À la création d'un nouveau fichier, le document courant reste en cours d'édition (à la connexion l'interface d'édition est désactivée), l'utilisateur clique donc sur le nouveau fichier qui sera rapatrié depuis le serveur de données et chargé dans l'interface. L'utilisateur peut compiler son projet en cliquant sur le lien *compiler* et accède au menu contextuel de compilation proposant une liste des différents projets disponibles.

La sélection d'un projet dans la liste charge automatiquement la liste des différents fichiers disponibles dans laquelle l'utilisateur choisit le fichier principal, incluant éventuellement d'autres fichiers du projet.

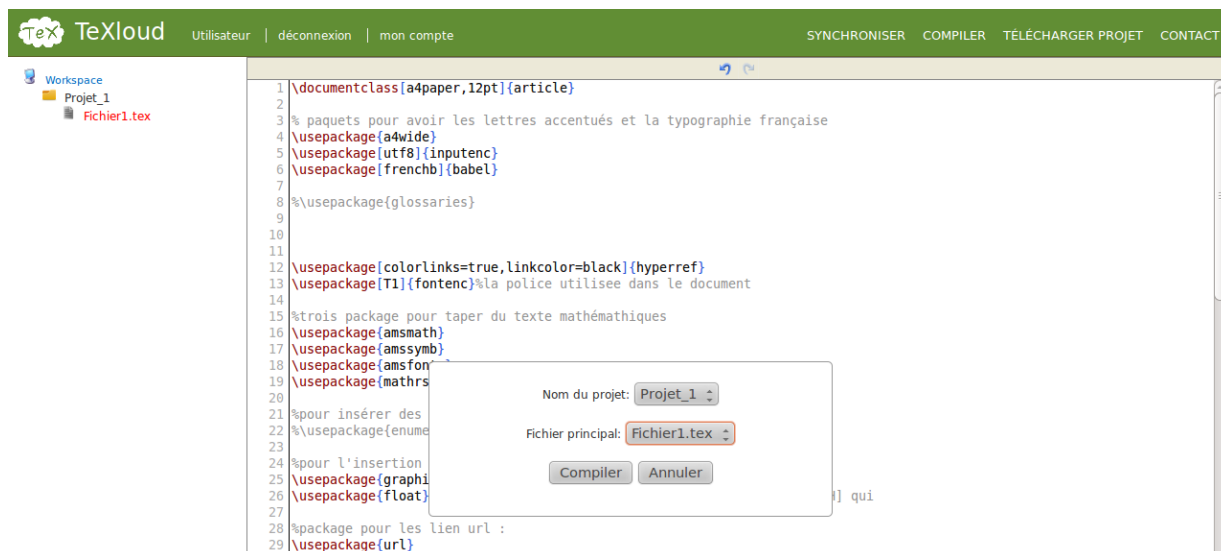


FIGURE 11 – Compilation, application web

Enfin, la soumission du formulaire lance la requête de compilation à proprement parler au travers du réseau et rapatrie le fichier de log d'erreur et le fichier pdf si la compilation est un succès.

La terminaison de la requête retourne à l'interface web les informations sur la compilation (log) et affiche le pdf dans l'interface web si il existe.

The screenshot shows the TeXcloud web interface. The top navigation bar includes the TeXcloud logo, a user menu (Meva | déconnexion | mon compte), and action buttons (SYNCHRONISER, COMPILER, TÉLÉCHARGER PROJET, CONTACT). On the left, a sidebar shows the workspace with 'Projet_1' and 'Fichier1.tex'. The main editor displays a LaTeX document with lines 105 to 141. The document content includes paragraphs about sorting and conditional coloring of fields based on values. The compilation log at the bottom shows two warnings: 'message : underfull \hbox (badness 10000) in paragraph at lines 143–144' occurring at lines 143 and 144. A 'fermer' button is located in the top right corner of the log area.

```

105 entre les pages est possible grâce aux liens présent dans les entêtes de la
106 table de présentation.
107 \paragraph*{}
108 Les tris sont obtenue grâce à une commande de ce type:
109 \begin{verbatim}
110 <xsl:sort select="expression" data-type="text|number"
111 order="ascending|descending"/>
112 \end{verbatim}
113
114
115
116 \paragraph*{}
117 D'autres traitements sont prévu dans chacun des fichiers XSLT. En effet certain
118 champs subissent une coloration en fonction de leurs valeurs:
119 \begin{itemize}
120 \item Si $pp<0$ le champs est coloré en orange
121 \item Si $pp>100$ le champs est coloré en vert
122 \item Si $pp\geq 0 \wedge pp\leq 100$ le champ n'est pas coloré
123 \item Si $ic<0$ le champs est coloré en orange
124 \item Si $ic\geq 0$ le champs est coloré en vert
125 \end{itemize}
126
127 \paragraph*{}
128 Les if sont obtenue grâce à une commande de ce type (ici un if/else):
129 \begin{verbatim}
130 <xsl:if test="expression">
131   du contenu
132 </xsl:if>
133 <xsl:if test="not(expression)">
134   du contenu
135 </xsl:if>
136 \end{verbatim}
137
138 \paragraph*{}
139 Enfin le dernier traitement est l'addition des trois champs PP,PV,PA afin de
140 générer une dernière données sur laquelle nous effectuons aussi les tris.
141

```

Position: Ln 113, Ch 15 Total: Ln 150, Ch 3709

Warning

- Lignes : 143 -> 144
• message : underfull \hbox (badness 10000) in paragraph at lines 143–144
- Lignes : 143 -> 144
• message : underfull \hbox (badness 10000) in paragraph at lines 143–144
- Lignes : 143 -> 144

fermer

FIGURE 12 – Log d'erreur, application web

L'accès au pdf se fait systématiquement par la même url `/ajax/getPdf`. En fait au niveau du serveur le fichier pdf est écrit dans `/tmp/tecloud` et l'adresse du pdf est stockée en session, l'action `getPdf` force le téléchargement du fichier.

En réalité, nous n'utilisons pas directement l'action de téléchargement du pdf, en revanche, la librairie de visualisation de pdf *pdfjs* fait un appel sur l'url de téléchargement puis convertit le pdf pour un affichage dans des canvas HTML5.

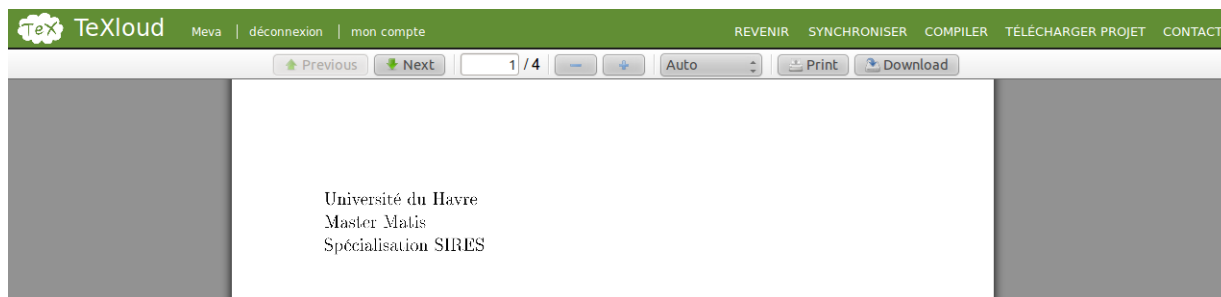


FIGURE 13 – Visualisation du PDF, application web

3.1.5 Fonctions annexes

L'application inclut également des fonctionnalités secondaires, notamment en ce qui concerne la gestion et les modifications de documents.

Suppression de dossiers et de fichiers

Les utilisateurs peuvent supprimer les fichiers et les dossiers *via* le menu contextuel sur l'arbre de navigation de gauche. Les requêtes de suppressions sont effectués en base de donnée et la requête est envoyée au serveur de donnée correspondant qui met à jour le dépôt de version.

Renommage des documents

À la manière de la suppression de document, le renommage s'effectue par l'arbre de navigation et remonte les ordres en base de données et au serveurs de stockage.

Information sur les utilisateurs

Les utilisateurs peuvent compléter les informations sur leur profils afin pouvoir s'identifier dans la communauté (non implémentée).

3.2 Application Android

Le fonctionnement général de l'application Android est proche de celui de l'application web.

3.2.1 Page d'accueil

La page de login est assez intuitive. Après avoir tapé son identifiant et son mot de passe, une requête de vérification est envoyée, et si l'authentification échoue, un message en rouge apparaît sur la page. La page d'accueil permet également de s'inscrire.

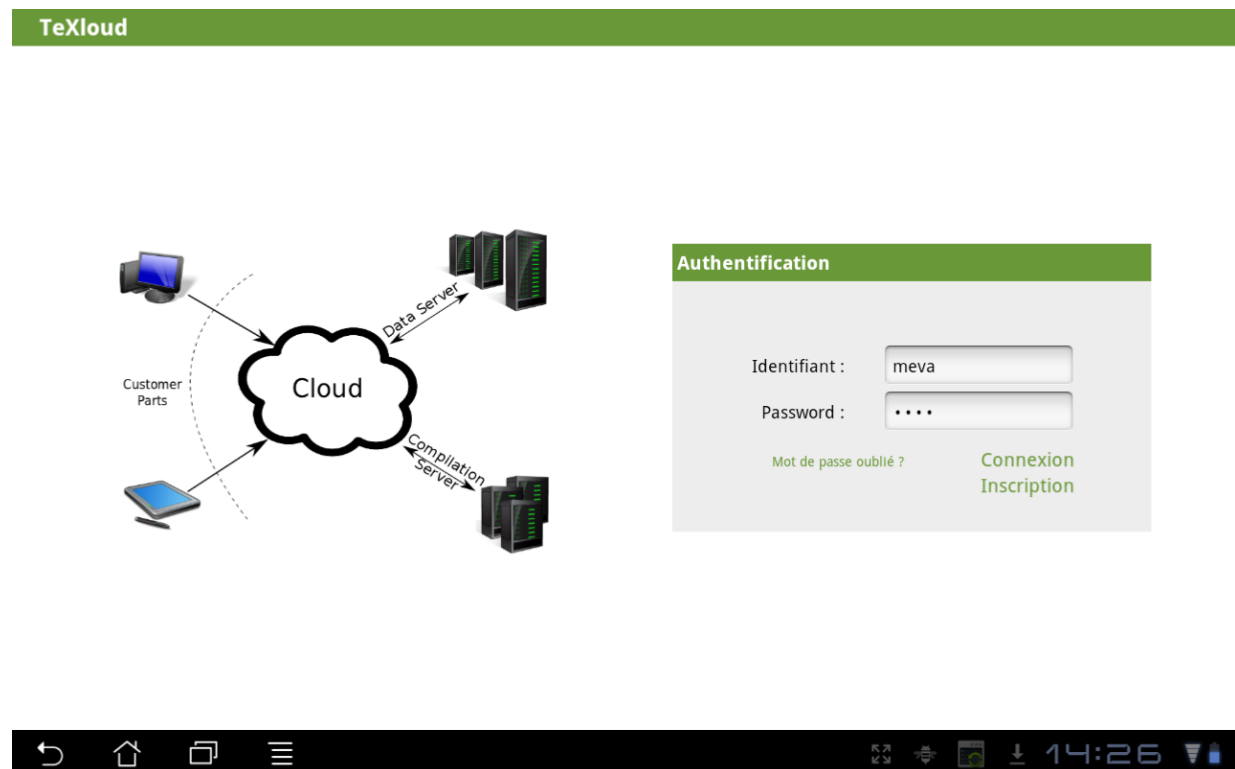


FIGURE 14 – Page d'accueil Android

3.2.2 Création de compte

Les champs obligatoires sont en italique. L'application Android, contrairement à l'application Web, n'implémente pas encore de moyen de pouvoir modifier son compte ultérieurement. L'utilisateur doit donc aller sur l'interface web s'il souhaite modifier un élément de son profil.

The screenshot displays the TeXloud Android application interface. A central white dialog box titled "Rejoignez TeXloud !" is overlaid on a dark grey background. The background features a diagram on the left with a cloud labeled "Cloud" connected to a desktop monitor and a smartphone, with the text "Customer Parts" nearby. On the right, a semi-transparent login box is visible with fields for "meva" and "...." and buttons for "Connexion" and "Inscription". The registration form itself contains the following fields, with the first one highlighted by an orange border:

- Nom d'utilisateur* (highlighted with an orange border)
- Mot de passe*
- Confirm*
- E-Mail*
- Prénom*
- Nom*
- Sexe* (with a dropdown menu showing "M")
- Adresse*
- Ville*
- Pays*
- Zip Code*
- Date de naissance* (with sub-fields for "Année", "Janvier" (dropdown), and "Jour")

At the top right of the form, it says "Champs italiques requis". The bottom of the screen shows the Android navigation bar with icons for back, home, recent apps, and a menu, along with the status bar showing the time as 17:29.

FIGURE 15 – Inscription via Android

3.2.3 Création de projet

La création de projet est très simple puisqu'il suffit de toucher le bouton "Créer projet" et d'entrer un nom de projet. Le projet est ensuite immédiatement disponible dans la liste. C'est la première étape pour un nouvel utilisateur, puisqu'il est impossible de créer des fichiers tex qui ne seraient dans aucun projet.

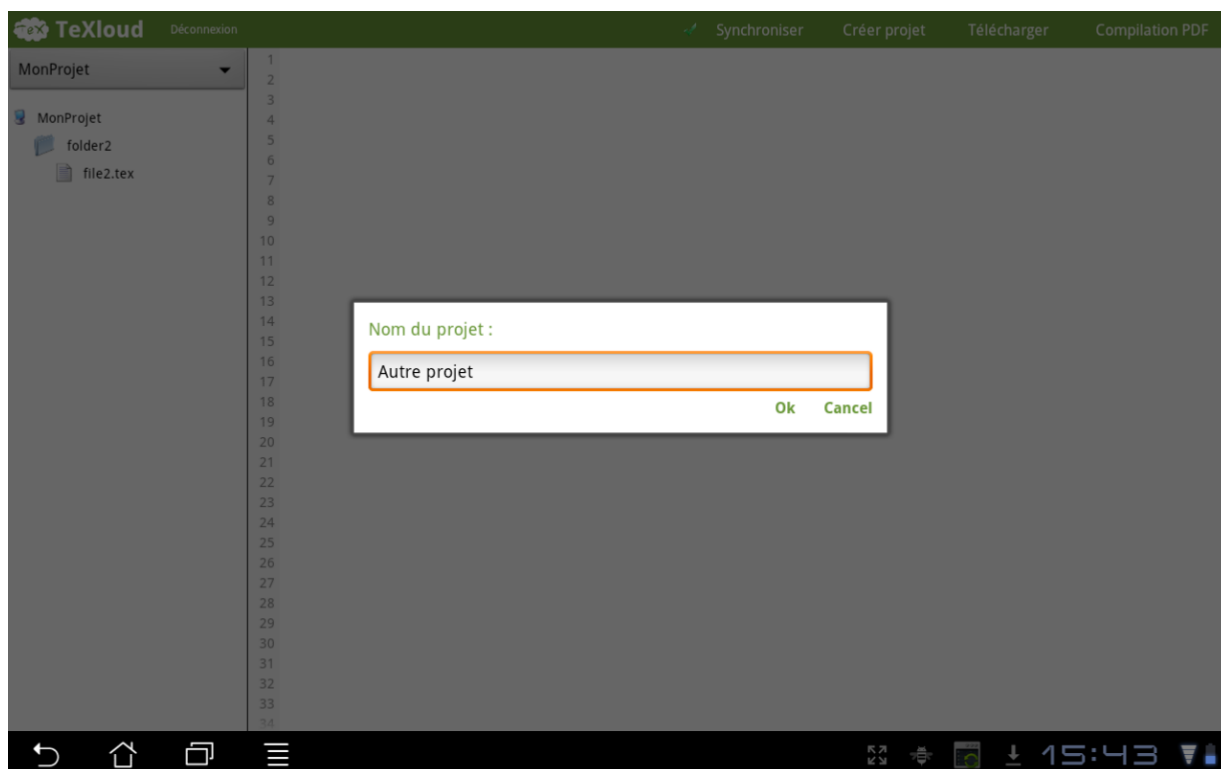


FIGURE 16 – Création de projet

3.2.4 Choisir un projet

La sélection de projet se fait par un objet Spinner (ou liste déroulante) en haut de la partie gauche. Le spinner contient le nom du projet en cours, et lorsqu'on appuie dessus, un écran de choix apparaît au milieu de l'écran.

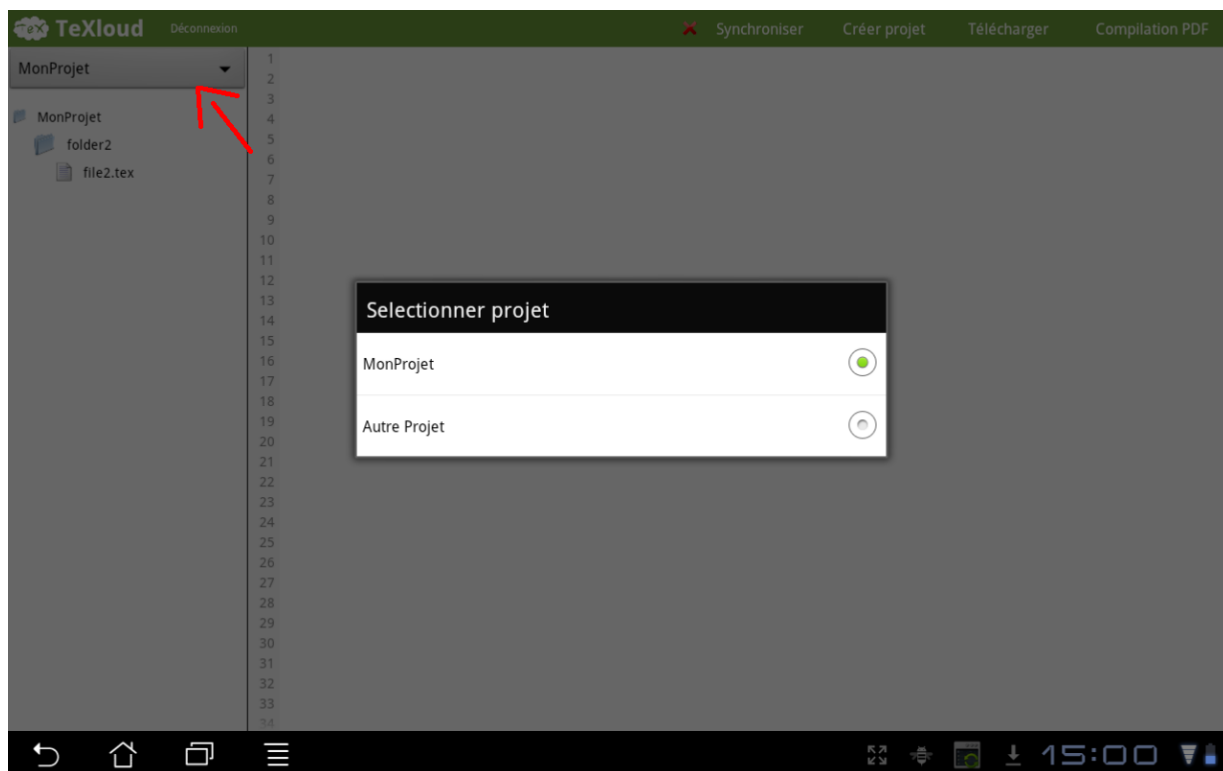


FIGURE 17 – Selection de projet

3.2.5 Synchronisation

La synchronisation est fondamentale dans l'application TeXloud, et il est très important pour l'utilisateur de savoir à tout moment où il en est. Une icône est située juste à côté du bouton de synchronisation. Par défaut, elle est verte et signale que tous les fichiers sont synchronisés.

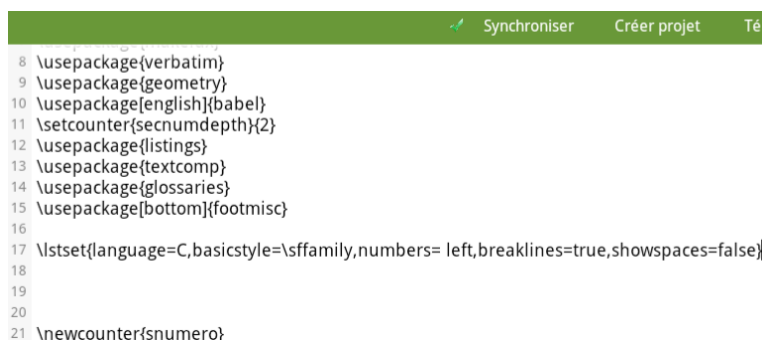


FIGURE 18 – Fichier synchronisé

Cependant, dès qu'un fichier est modifié, le logo devient rouge, signalant qu'une synchronisation doit être faite pour que les modifications soient enregistrées. Lorsque la synchronisation est faite, tous les fichiers modifiés sont enregistrés.

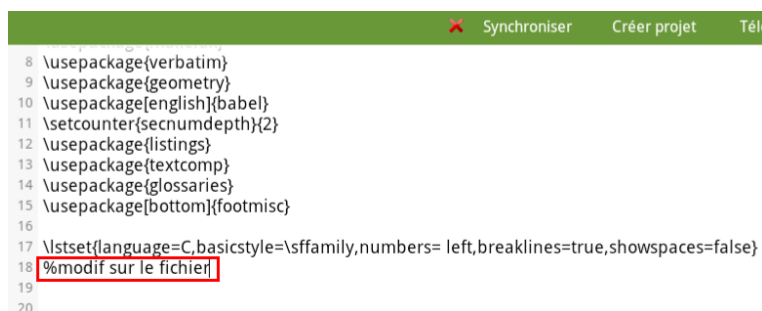


FIGURE 19 – Fichier non synchronisé

3.2.6 Compilation

La compilation passe par le bouton situé dans le bandeau supérieur. Lorsque l'ordre est lancé, tous les fichiers sont synchronisés puis la compilation est traitée. Chaque étape est décrite par une boîte de dialogue signalant à l'utilisateur qu'une opération est en cours. Lorsque la compilation s'est déroulée avec succès, une boîte de dialogue s'ouvre, affichant les warnings éventuels ainsi que le lien local du fichier PDF sur la tablette. Un bouton propose d'ouvrir immédiatement le PDF.

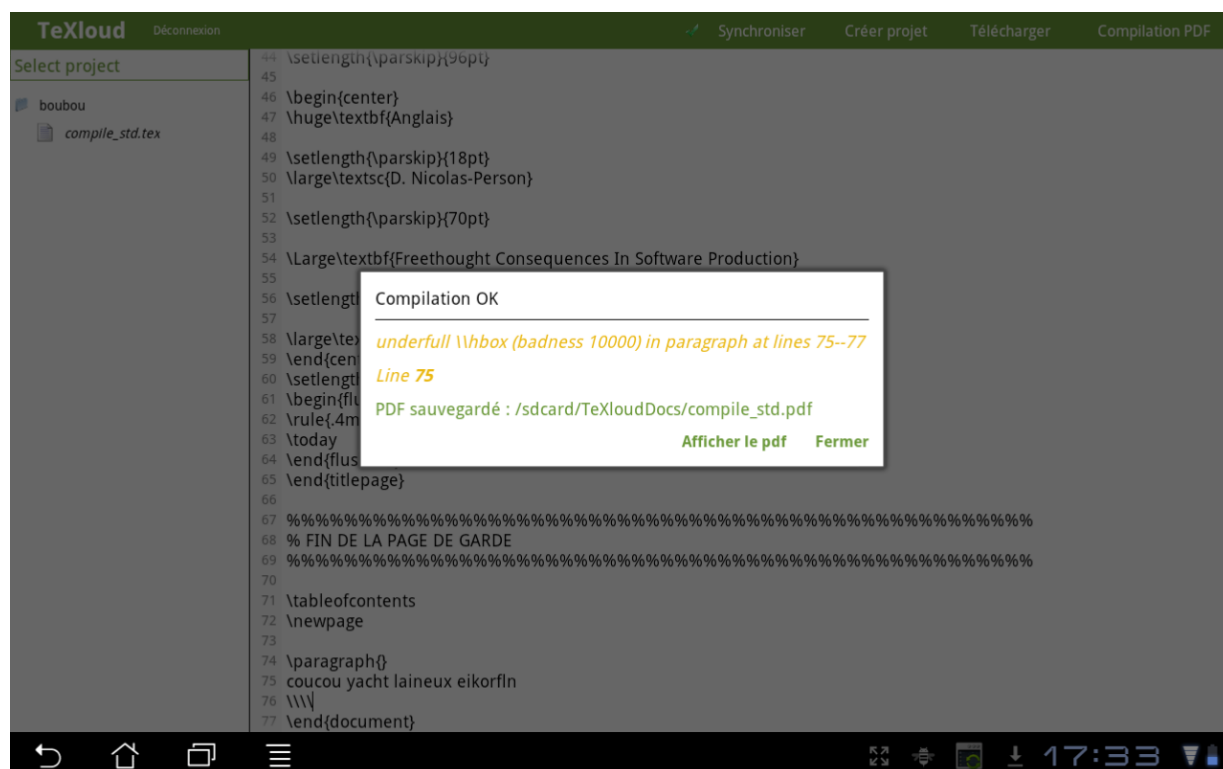


FIGURE 20 – Compilation via Android

Les erreurs de compilation apparaissent en rouge dans la boîte de dialogue (bien évidemment, aucun fichier PDF n'est généré).

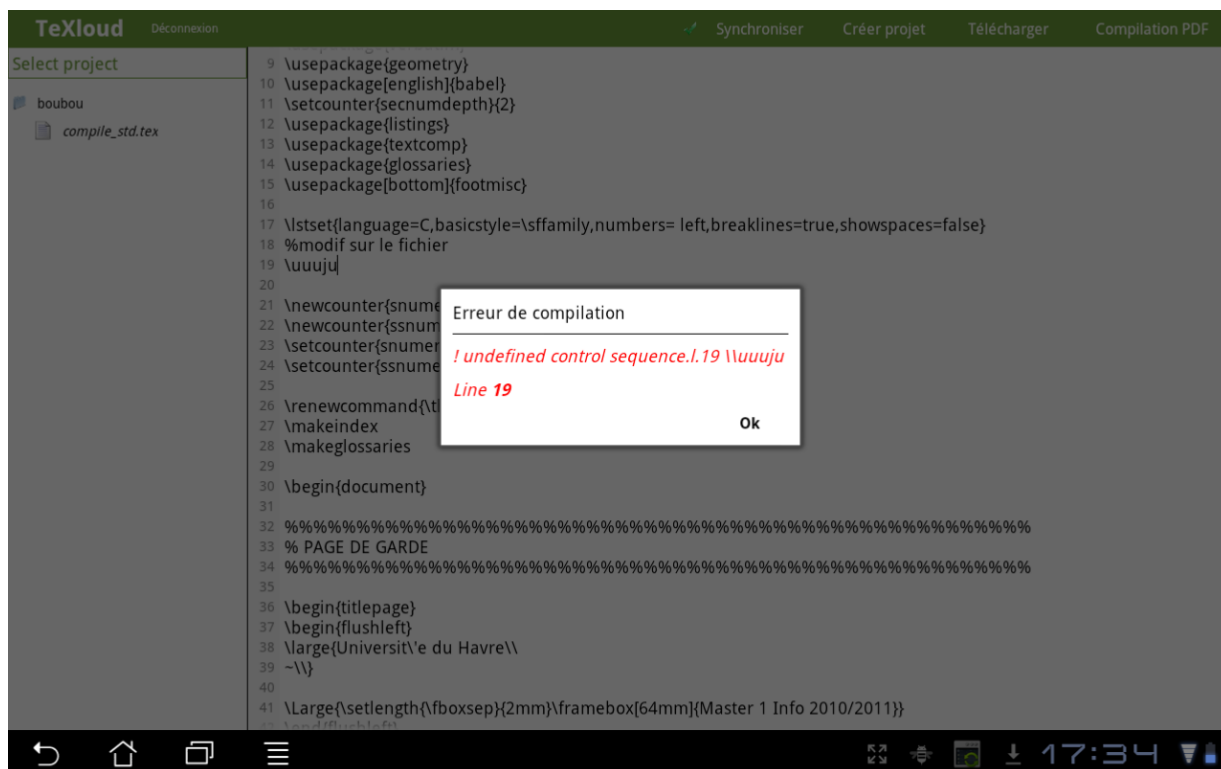


FIGURE 21 – Compilation via Android

3.2.7 Ouverture des PDF

L'ouverture des PDF est gérée par un programme tiers propre à l'environnement présent sur la tablette. Quand l'utilisateur clique sur "Afficher le PDF" après avoir compilé le fichier, l'application TeXloud signale à la tablette qu'il faut ouvrir un fichier PDF, et la tablette peut ensuite ouvrir le programme par défaut.

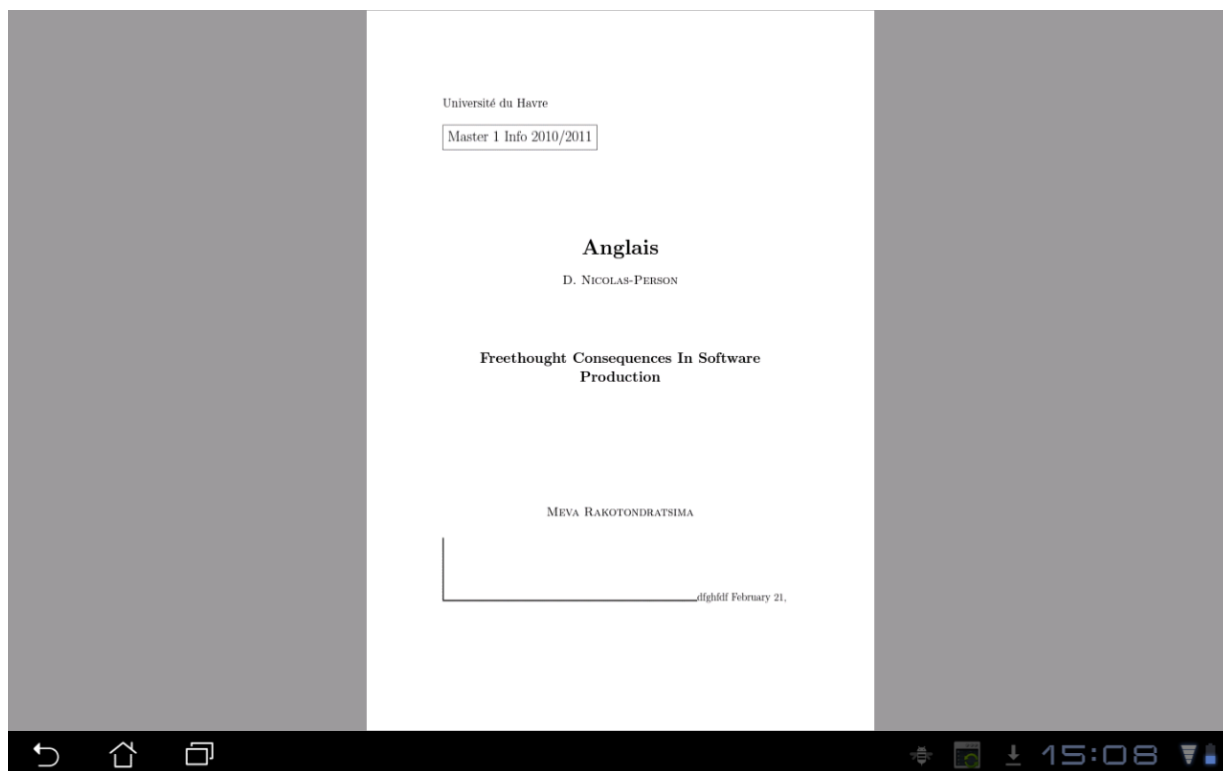


FIGURE 22 – Visualisation PDF

3.2.8 Fenetres de chargement

Toutes ces actions Android transitionnent avec une boîte de dialogue, indiquant à l'utilisateur qu'une action est en cours :

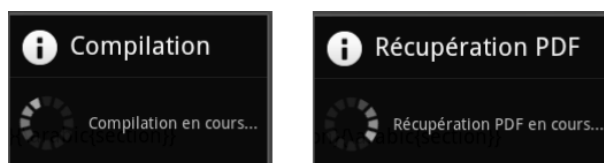


FIGURE 23 – Exemples de chargement

4 Perspectives d'évolution

4.1 Support des images

Le supports des images dans l'application permettrait d'ajouter des images au sein des PDF générés. 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 nécessaires.

4.2 Gestion des groupes

La gestion des groupes dans l'application permettrait la modification simultanée de documents par plusieurs utilisateurs. Cette modification est possible à intégrer par la suite car la base de données et le stockage des informations ont été prévus à cet effet⁴. Comme pour le support des images, les modifications de l'application pour supporter cette fonctionnalité sont minimales. 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'a pas été implémentée car nous n'avons pas trouvé de solution simple qui ne soit pas trop lourde en ressource pour une tablette. Ceci est sûrement possible si nous déclenchons 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 Mot de passe oublié

La gestion des mots de passe 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 et la boîte de dialogue sont déjà implémentés sur l'interface Android. La gestion des e-mails n'a pas été possible car nous n'avons pas l'architecture physique nous permettant de faire fonctionner convenablement un serveur mail⁵.

4.5 Utilisation d'autres supports 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

4. Le stockage des données étant effectué sur des serveurs SubVersion et la gestion des conflits étant implémentée, la gestion des groupes ne posera aucun problème

5. Pour être dans de bonnes conditions, il aurait fallu que nous puissions avoir un accès vers internet afin de contacter les serveurs SMTP ou POP de différents fournisseurs d'adresses mail

géré, mais grâce à 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.

4.6 Sécurisation des transferts de données

Tous les échanges entre socket ne sont pas sécurisés. Plus précisément ils ne sont pas cryptés et donc vulnérable aux écoutes sur le réseau⁶. Pour sécuriser ceci, il suffit d'utiliser un cryptage type SSL/TLS⁷. Cette solution est très simple à mettre en place en python, il suffit de remplacer les objets socket par des objets SSLSocket⁸.

4.7 Mode offline pour Android

Le mode offline pour Android est, à l'heure actuelle, à un stade très prématuré. Cependant, il sera peut-être développé entre la date de rédaction de ce document et le moment de rendu du projet.

6. Attaque communément appelée sniffing

7. Technique utilisée pour les transferts HTTPS

8. Il faut bien sûr au préalable générer les certificats SSL

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 former sur une application réelle complexe et complète.

Le projet a subi 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 dû travailler sur les horaires d'ouverture de la faculté et lorsque la salle n'était pas occupée pour des cours. Aussi nous avons eu des problèmes quand à la formation de notre "cloud". En effet, nous avons dû utiliser les PC de la salle A104 (Résultat 2 serveurs en moins par rapport a notre architecture de base). De plus il 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 a été très intéressant à mettre en place. Aussi, nous avons pris 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
3	Accueil, application web	9
4	Inscription, application web	10
5	Première page, application web	10
6	Création projet, application web	11
7	Création projet (nom du projet), application web	11
8	Création projet terminé, application web	12
9	Création de fichier, application web	12
10	Création de fichier terminée, application web	13
11	Compilation, application web	14
12	Log d'erreur, application web	15
13	Visualisation du PDF, application web	16
14	Page d'accueil Android	17
15	Inscription via Android	18
16	Création de projet	19
17	Selection de projet	20
18	Fichier synchronisé	21
19	Fichier non synchronisé	21
20	Compilation via Android	22
21	Compilation via Android	23
22	Visualisation PDF	24
23	Exemples de chargement	24