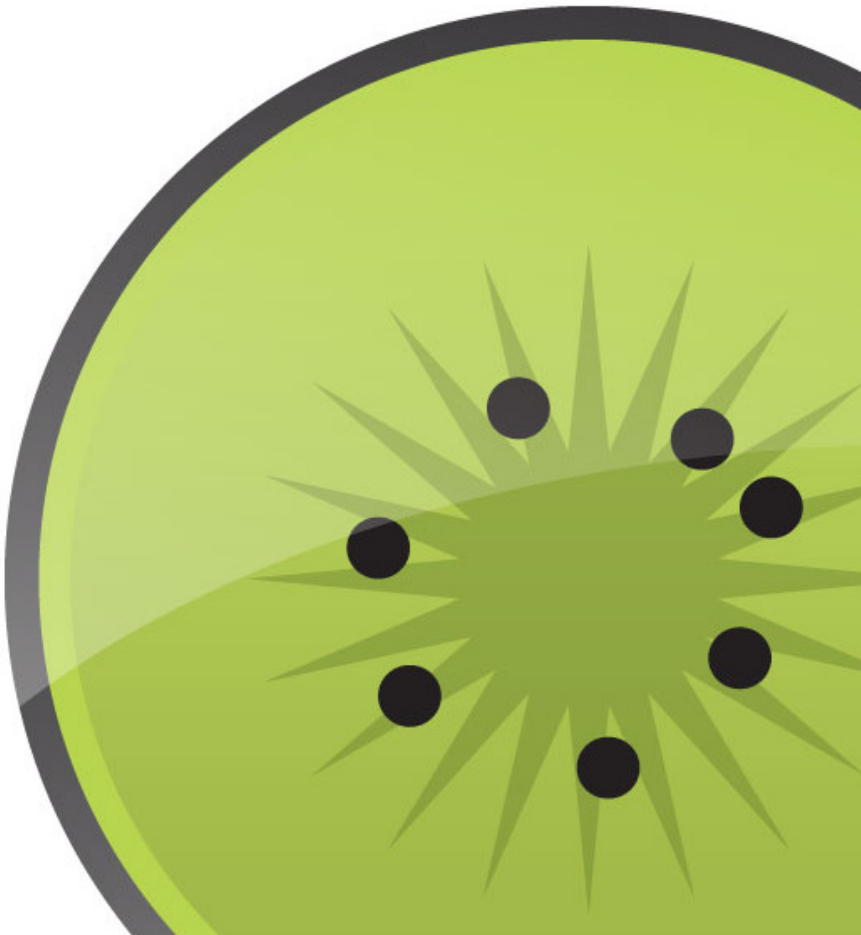
Travail de Bachelor 2012
Filière Informatique de Gestion
HES-SO Valais // www.hevs.ch

Projet : tokiwi File Manager
Etudiant : Fabien Galli
Professeur : Florian Evéquoz

tokiwi File Manager

Travail de Bachelor 2012



tokiwi

Hes·so // VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

1 PRÉFACE

Avez-vous déjà dressé une liste des communautés auxquelles vous appartenez ? Votre vie sociale s'organise autour d'une multitude de petites et grandes communautés, que ce soit votre famille, vos amis, vos collègues, vos voisins, les joueurs de votre club de sport, les membres du comité d'organisation de votre association, etc.

Vous êtes-vous déjà demandé combien de fois par mois, par semaine, ou même par jour, vous interagissiez avec les personnes de vos communautés ? Comme la plupart des gens, vous êtes probablement bombardé de mails, messages, appels téléphoniques, rendez-vous, etc.

Vous êtes-vous déjà demandé s'il existait des moyens de faciliter l'organisation de ces interactions ? Vous êtes sûrement quelqu'un à la page. Vous disposez donc d'un agenda sur votre smartphone, vous utilisez des solutions de partage de fichiers en ligne et une multitude d'autres gadgets.

Maintenant que vous avez trouvé un système d'organisation de votre vie sociale, ne serait-il pas plus simple si toutes les autres personnes de vos communautés utilisaient le même ? Est-ce que vous n'économiseriez pas du temps à communiquer et échanger de la même manière avec tout le monde ? Vous êtes quelqu'un d'intelligent, vous en êtes tellement convaincu que vous seriez même prêt à mettre en place toutes les infrastructures pour rendre cela possible.

Est-ce possible ?

Oui c'est possible. En lisant ce document, vous découvrirez comment, grâce à tokiwi et ses applications. tokiwi est la plateforme web qui permet à toutes les communautés du monde de s'organiser, communiquer et échanger.

Le sujet de ce travail de Bachelor consiste en la réalisation d'une application clé de tokiwi, le gestionnaire de fichiers. Après avoir lu ces quelques pages, vous comprendrez comment l'utiliser, comment il a été créé et pourquoi il est meilleur que tous ceux qui existent actuellement.

2 RÉSUMÉ MANAGÉRIAL

Contexte et objectifs

Le sujet de ce travail de Bachelor consiste en la réalisation d'une application tokiwi permettant la gestion de fichiers. tokiwi est une spin-off issue du programme Business eXperience de la HES-SO Valais. Elle a été fondée par 4 étudiants de la filière Informatique de Gestion en 2011. Il s'agit d'une plateforme web permettant à des communautés de s'organiser, échanger et communiquer au travers d'espaces de travail partagés. Ces communautés peuvent être des groupes d'amis, une famille, des collègues de travail, un comité d'organisation d'une association et bien d'autres encore.

La gestion et le partage de fichiers sont des aspects cruciaux du fonctionnement des communautés. Que ce soit dans un cadre ludique, afin de partager des photos, des vidéos et d'autres médias ou dans un cadre professionnel, afin de partager et maintenir à jour des documents. Il semblait donc logique que tokiwi se dote d'une application permettant une gestion simple et efficace des fichiers.

L'application

L'interface graphique de l'application s'inspire fortement des fonctionnalités proposées dans l'explorateur de fichiers Windows. Elle permet de déplacer ou copier des fichiers d'un répertoire à un autre grâce au « drag'n'drop ». Il est possible d'envoyer plusieurs fichiers à la fois vers la plateforme, soit de manière classique, soit en déposant les fichiers directement dans le navigateur web. Il est également possible de télécharger plusieurs fichiers ou répertoires d'un coup. Ceux-ci se trouvent alors dans une archive ZIP respectant leur hiérarchie d'origine. L'utilisateur peut également restaurer des fichiers qu'il aurait supprimés par inadvertance. De plus, en associant son compte Dropbox à son compte tokiwi, il peut accéder au contenu de sa Dropbox et ainsi copier des ressources du gestionnaire de fichiers tokiwi vers sa Dropbox et inversement. Finalement, l'application est conçue pour s'intégrer aux autres applications tokiwi. Elle détecte automatiquement de quel type sont les fichiers et sait quelles applications permettent d'exécuter quel type de fichier.

L'application est construite sur une architecture 3 tiers classiques. L'interface utilisateur utilise HTML 5, CSS 3 et Javascript/jQuery. La logique applicative s'articule sur des entités/contrôleurs écrits en PHP. Les informations sont persistées dans une base de données MySQL.

Conclusion

L'application développée au cours de ce travail répond aux besoins des communautés. Son interface graphique attrayante est construite sur architecture robuste et évolutive. Désormais, d'autres applications vont pouvoir venir se greffer dessus et enrichir la gamme de services offerts par tokiwi.

TABLE DES MATIÈRES

3 INTRODUCTION

3.1 CONTEXTE

Dans le cadre du cursus du Bachelor HES, il est demandé à chaque étudiant de réaliser un travail appelé « travail de Bachelor » à la fin des trois années de formation. Ce travail vaut 12 crédits ECTS, ce qui correspond à 360 heures de travail.

Le travail débute 5 semaines avant la fin du sixième semestre, durant lesquelles l'étudiant dispose de 20 heures par semaine pour le travail de Bachelor. Après les examens de module début juillet, l'étudiant dispose encore de 6 semaines avant de rendre son travail.

L'étudiant peut choisir parmi plusieurs sujets proposés par des enseignants ou des entreprises, ou alors proposer son propre sujet. Dans le cas présent, participant à la création d'une start-up, j'ai proposé un sujet qui permettrait d'accélérer sa mise sur le marché.

Généralement, le travail se décompose en trois parties distinctes. La première consiste en une recherche approfondie de la matière traitée qui se concrétise par l'établissement d'un état de l'art. La deuxième est la réalisation pratique du travail, dans le cas présent le développement de l'application. La troisième partie est la rédaction du présent rapport.

Un professeur de la HES suit l'avancement du travail et répond aux éventuelles questions de l'étudiant. M. Florian Evéquoz a assuré cette fonction pour ce travail.

3.2 PRÉSENTATION DE TOKIWI

tokiwi est une spin-off issue du programme Business eXperience de la HES-SO Valais. Elle a été fondée en 2011 par les quatre étudiants de la filière Informatique de Gestion que l'on peut voir sur la photo ci-dessous : de gauche à droite, Jérôme Treboux, Fabien Galli, Pierrick Maret, Léonard Stalder.

Figure 1 Photo des membres fondateurs de tokiwi à l'occasion des Venture Ideas 2012

Il s'agit d'une plateforme web permettant à des communautés de s'organiser, communiquer et échanger. Ces communautés peuvent être des clubs sportifs, des groupes d'amis, une famille, un comité d'organisation, un groupe de travail, une association, un parti politique et bien d'autres encore.

Le concept clé de la plateforme est d'offrir à chacun de ses utilisateurs la possibilité de créer des espaces de travail pour leurs communautés. Ils peuvent ensuite y inviter leurs membres. A l'intérieur de ces espaces de travail partagés, chaque utilisateur retrouvera les outils et les applications qui lui sont utiles à la vie de la communauté.

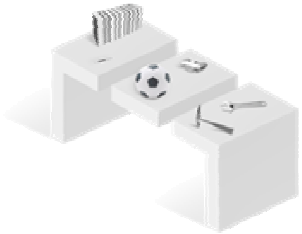


Figure 2 Représentation des espaces de travail partagés par communauté

Beaucoup de personnes ont déjà mis en place une organisation sophistiquée de leurs communautés en travaillant parallèlement sur plusieurs produits différents. Afin que ces utilisateurs puissent continuer à utiliser leurs outils favoris, tokiwi intègre une multitude de services web dans sa plateforme. tokiwi n'est pas un « n'ième » outil qui simplifie la vie, mais l'outil à partir duquel on accède à ces outils qui simplifient la vie.

Figure 3 Représentation de l'intégration de services externes dans tokiwi

L'avantage de centraliser les services web les plus populaires sur la même plateforme est qu'il est alors possible de les faire interagir entre eux. Ainsi, grâce à tokiwi, créer un album photo à partir de photos stockées sur Picasa et Flickr et associer cet album à un évènement Facebook devient un jeu d'enfant.

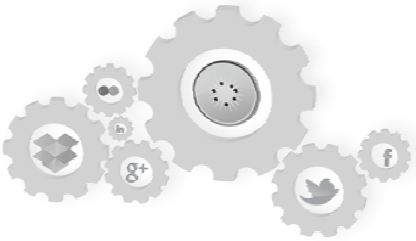


Figure 4 Représentation des interactions entre applications tokiwi et services intégrés

3.3 BESOINS FORMULÉS

tokiwi a besoin d'une application permettant aux communautés de la plateforme de stocker, utiliser et partager des fichiers. Cette application doit être accessible depuis l'espace de travail des communautés. Son fonctionnement doit être similaire à celui de l'explorateur de fichiers Windows.

De plus, elle doit permettre aux utilisateurs des communautés d'accéder à leurs fichiers stockés sur des services de stockage de fichiers en ligne tels que Dropbox ou Google Drive.

3.4 CAHIER DES CHARGES

Afin de répondre aux besoins formulés ci-dessus, j'ai dû réaliser le travail suivant :

- Développer une application compatible avec la plateforme en utilisant les mêmes technologies.
    - L'application doit correctement s'intégrer dans le graphisme actuel de la plateforme.
    - L'application doit procurer le même genre d'expérience utilisateur que l'explorateur de fichiers Windows. C'est-à-dire que des actions basiques telles que la sélection, la copie et le déplacement de fichiers puissent se faire à l'aide d'un « drag'n'drop ».
    - L'application doit fournir au minimum les fonctionnalités basiques d'un gestionnaire de fichiers. C'est-à-dire l'organisation en arbre de répertoires, la création de répertoires, la navigation dans les répertoires, la recherche de fichiers, la suppression de fichiers, l'envoi de fichiers et le téléchargement de fichiers.
- Réaliser un état de l'art des solutions présentant les mêmes fonctionnalités.
    - L'application doit intégrer la meilleure solution retenue de l'état de l'art et ainsi permettre d'interagir avec celle-ci. Concrètement l'application doit pouvoir récupérer et envoyer des fichiers depuis et vers cette solution.
    - L'application doit être conçue de manière à ce qu'il soit facilement possible d'y intégrer d'autres solutions de ce type.

3.5 MOTIVATIONS PERSONNELLES

Ce travail me tenait beaucoup à cœur car il s'agissait non pas d'un exercice factice, mais d'un cas concret dont la finalité m'impactait personnellement. Pouvoir investir autant de temps et d'énergie dans ce projet personnel a été une source de motivation durant les longues journées ensoleillées de juillet et août.

De plus, c'était une opportunité unique de me familiariser avec de nouvelles technologies comme HTML 5 et d'approfondir mes connaissances dans le langage de programmation PHP. En outre, le cahier des charges comporte de vrais challenges techniques que j'étais très enthousiaste à relever.

3.6 DÉMARCHE

J'ai utilisé les deux premières semaines pour réaliser un cahier des charges que j'ai fait valider au professeur qui me suivait. Ce cahier des charges est consultable sur le CD-ROM. Il explique entre autre que j'ai divisé le projet en 6 sprints de 2 semaines.

Durant les deux premiers sprints, j'ai réalisé la question de recherche. Il me paraissait logique de commencer par ceci car tester d'autres solutions m'a permis de noter les fonctionnalités particulièrement intéressantes pour m'en inspirer, et celles qu'il ne fallait surtout pas reproduire. De plus c'était une activité qui se prêtait mieux au contexte du début du travail, car je ne disposais que rarement de journée entière, ce qui rend plus difficile des activités comme le développement.

Ensuite j'avais pensé rédiger le rapport au fur et à mesure que je développais les fonctionnalités de l'application. Mais je me suis rapidement rendu compte que ce n'était pas un choix viable. En effet, il m'est arrivé plusieurs fois de changer des parties de l'architecture de l'application, ce qui m'a amené plusieurs fois à devoir modifier la documentation. J'ai donc décidé de réaliser le développement en un bloc entre les sprints 3 et 5, en maintenant simplement à jour la documentation technique.

J'ai donc réalisé le rapport à la fin, durant le sprint 6.

4 ETAT DE L'ART

Le contexte et les objectifs du projet étant spécifiés, nous allons voir dans ce chapitre comment le travail de recherche a été réalisé. A la fin de ce chapitre, nous comprendrons les différents arguments qui ont permis de choisir la solution de gestion de fichiers en ligne qui a été intégrée.

4.1 OBJECTIFS

Dans un premier temps, cet état de l'art a pour but de dresser une liste des solutions d'hébergement de fichiers en ligne proposant des fonctionnalités avancées pour la gestion de fichiers. Les solutions éligibles doivent au minimum correspondre aux critères suivants :

- Stockage en ligne des fichiers avec possibilité d'envoyer et recevoir les fichiers sur différents périphériques
- Gestionnaire de fichiers (gestion de dossiers, copie, déplacement, suppression, changement de nom) via une interface web
- Sécurité et partage des données
- Intégration à d'autres outils existants au travers d'une API
- Cible aussi bien des particuliers que des entreprises

Puis dans un deuxième temps, cet état de l'art a pour objectif de comparer les solutions entre elles sur les critères suivants :

- Services proposés par rapport au prix
- Popularité (nombre d'utilisateurs, tendance de la croissance)
- Possibilités et rapidité d'intégration à d'autres outils

Et finalement, de déterminer lesquels seraient intéressantes à intégrer dans tokiwi, pour autant que cela soit possible. Le prix des solutions dans la recherche ne va pas influencer le travail en soit, mais peut se révéler utile en vue d'une éventuelle commercialisation de l'application.

4.2 DÉMARCHE

1. Recherche d'un état de l'art déjà existant ou de travaux similaires
2. Recherche dans les sources d'informations suivantes avec les combinaisons de mots clés suivantes :

Source d'information	Combinaisons de mots clés
Google, bing, YAHOO!	[file manager], [web file manager], [document manager], [web document manager], [cloud file manager], [cloud document manager], [file organizer], [web file organizer], [cloud file organizer], [tool file sharing], [web tool file sharing], [file browser], [web file browser], [file explorer], [web file explorer], [online file manager], [file hosting]
WIKIPÉDIA L'encyclopédie libre	[file], [file manager], [web file manager], [cloud file manager], [file hosting], [<nom de chaque solution>]
ZDNet, Comment ça marche .net, 01net., LeJournalduNet, stackoverflow, YAHOO! ANSWERS	[file manager], [web file manager], [cloud file manager], [app file manager], [tool file manager], [file browser], [file hosting], [<nom de chaque solution>]

Tableau 1 Combinaisons de mots clés dans les sources d'information

3. Navigation sur les liens les plus pertinents afin de retenir le nom des solutions existantes
4. Recherche dans les sources d'information citées ci-dessus afin d'obtenir idéalement les informations suivantes sur chacune des solutions
    a. Nom de la solution
    b. Date de création
    c. Rang Alexa
5. Comparaison des services et des tarifications
6. Evaluation de la popularité et du potentiel de croissance
7. Evaluation du potentiel d'intégration à tokiwi et conclusions

4.3 SOLUTIONS SIMILAIRES EXISTANTES

En cherchant dans les différents moteurs de recherche, avec les mots clés cités au chapitre précédent, on se rend rapidement compte qu'il existe trois catégories de solutions proposant des fonctionnalités de gestion de fichiers :

- **Les outils de GED** (Gestion Electronique des Documents). Il s'agit d'applications servant à archiver, centraliser, classer et rendre accessibles des documents. Ce sont généralement des applications onéreuses destinées à des moyennes et grandes entreprises. Elles sont distribuées soit sous forme de logiciels à installer (serveurs et clients), soit dans le « cloud » de l'éditeur. La gamme de services proposés et le public cible ne correspondent pas aux critères de recherche.
- **Les explorateurs de fichiers** (à l'image de Windows Explorer et Finder de Mac). Il s'agit d'applications à installer sur un système d'exploitation et offrant uniquement des fonctionnalités d'interaction avec un système de fichiers existant. Ces solutions ne correspondent que partiellement aux critères de recherche puisqu'elles ne résolvent pas le problème du stockage des fichiers.
- **Les solutions d'hébergement de fichiers en ligne**. Il s'agit d'outils, généralement web, permettant de stocker des fichiers dans le « cloud » de l'éditeur, de les rendre accessibles depuis différents périphériques et de les partager (soit par échange de liens, soit par accès sécurisé classique). Ce genre de solution est destiné aussi bien à des entreprises qu'à des particuliers et offre des fonctionnalités d'exploration de fichiers (bien que souvent limitées), ce qui correspond à une partie des critères de recherche.

Deux documents sur Wikipédia permettent de constituer un état de l'art sur les solutions de type « explorateur de fichiers ». L'un explique ce qu'est un gestionnaire de fichiers *[1]* et les différentes formes qu'ils peuvent prendre. L'autre comparant différentes solutions *[2]* :

- http://en.wikipedia.org/wiki/File_manager
- http://en.wikipedia.org/wiki/Comparison_of_file_managers

J'ai également trouvé un état de l'art en deux documents sur Wikipédia concernant les solutions de type « hébergement de fichiers en ligne », structurés de la même manière que les deux précédents *[3] [4]* :

- http://en.wikipedia.org/wiki/File_hosting_service
- http://en.wikipedia.org/wiki/Comparison_of_file_hosting_services

J'ai ensuite listé les solutions correspondantes aux critères de recherche parmi les solutions listées dans la comparaison pour en dresser le tableau suivant :

Solution	Site	Première release *[4]*	Rang Alexa* *[21]*
4shared	www.4shared.com	2005	97
Box	www.box.com	2005	814
Crocko	www.crocko.com	2007	2'748
Dropbox	www.dropbox.com	2008	160
Google Drive	drive.google.com	2012	N/A
iCloud	www.icloud.com	2011	1'756
MediaFire	www.mediafire.com	2006	64
Minus	www.minus.com	2010	1'739
PutLocker	www.putlocker.com	2010	343
RapidShare	www.rapidshare.com	2002	171
sendspace	www.sendspace.com	2005	1'088
SkyDrive	www.skydrive.com	2007	N/A
Spideroak	www.spideroak.com	2007	115'786
SugarSync	www.sugarsync.com	2007	6'086
Ubuntu One	one.ubuntu.com	2009	1'162
Wuala	www.wuala.com	2008	20'171
YouSendIt	www.yousendit.com	2003	1'331

Tableau 2 Solutions de gestion de fichiers en ligne existantes

*Le rang Alexa correspond au classement du trafic global de tous les sites (+de 30 mio). Pour plus d'informations : http://www.alexa.com/help/traffic-learn-more

4.4 RÉSULTATS DES TESTS

Ne pouvant pas tester en détails chacune de ces solutions, j'ai décidé arbitrairement de ne tester que les solutions des grosses firmes, à savoir Dropbox, Google Drive, iCloud de Apple et SkyDrive de Microsoft. A cette liste j'ai encore ajouté 4shared et MediaFire car ils ont l'air d'être très utilisés et PutLocker car n'étant sorti qu'en 2010, il a déjà un trafic conséquent.

4.4.1 SERVICES PROPOSÉS PAR RAPPORT AU PRIX

Voici un graphique permettant de comparer les offres de chaque solution *[5] [6] [7] [8] [9] [10] [11]* :

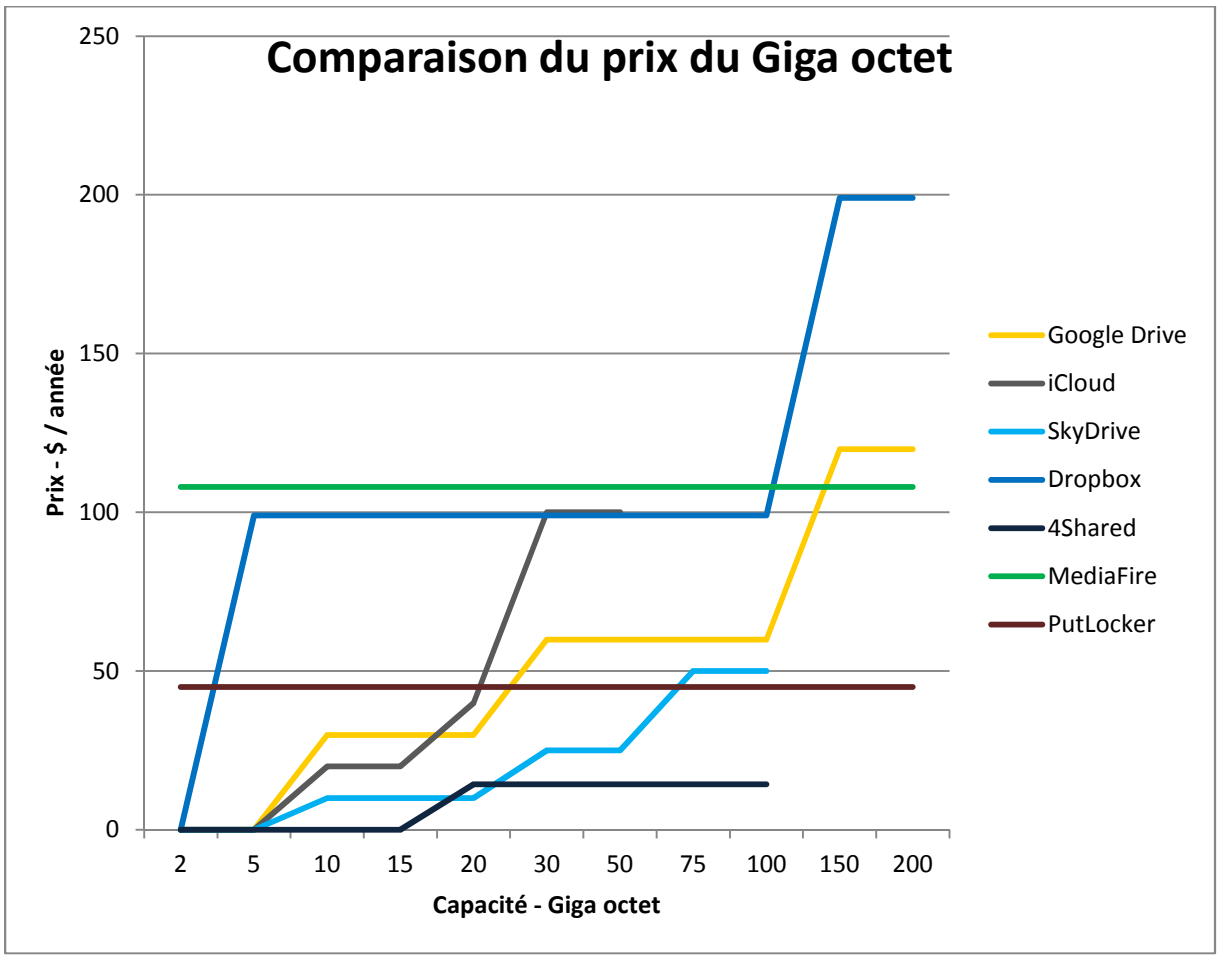


Tableau 3 Comparaison du prix du Giga octet des gestionnaires de fichiers en ligne

Le tableau suivant permet de comparer les services proposés des différentes solutions *[4]* :

Solution	Synchronisation	Versionning	App mobile	Taille max f.	Partage
Google Drive	YES	YES		10 Go	YES
iCloud	YES	NO		Infini	NO
SkyDrive	YES	YES		2 Go	YES
Dropbox	YES	YES		2 Go	YES
4Shared	YES	NO	symbian OS	5 Go	YES
MediaFire	NO	NO		4 Go	YES
PutLocker	NO	NO		5 Go	YES

Tableau 4 Comparaison des fonctionnalités des gestionnaires de fichiers en ligne

A partir du graphique et du tableau ci-dessus, nous constatons que :

- Dropbox, leader du marché, est la solution la plus chère, et de manière sensible.
- SkyDrive et 4Shared annoncent le meilleur rapport « services fournis / prix ».
- La politique de prix de MediaFire et PutLocker devient particulièrement intéressante lorsqu'il est nécessaire de stocker de gros volumes de fichiers en ligne, en revanche ces deux solutions souffrent grandement du manque de fonctionnalités annoncées.
- iCloud propose le moins bon rapport « services fournis / prix », de plus leurs services ne permettent d'obtenir que 50 Go d'espace de stockage, ce qui peut être un facteur très limitant. En revanche le service iCloud offre une multitude de services annexes à la gestion de fichiers en plus de ceux listés dans ce comparatif.
- Google Drive se positionne à mi chemin entre tous ses concurrents. Comme PutLocker et MediaFire (et Dropbox depuis peu), les services de Google Drive ne s'arrêtent pas à 100 Go, ce qui est un bon atout pour le secteur professionnel.

4.4.2 POPULARITÉ

La popularité de chaque solution est difficile à évaluer. Pour ce faire, il faudrait avoir accès au nombre d'utilisateurs, à la croissance des utilisateurs, au trafic et au nombre d'utilisateurs potentiellement atteignables au travers d'autres services de la solution. Or ces chiffres sont pour la plupart inaccessibles ou difficilement vérifiables.

De plus, iCloud et Google Drive étant des services sortis très récemment, il est impossible d'obtenir des informations offrant un recul suffisant pour évaluer concrètement leurs impacts sur le marché du stockage en ligne.

J'ai donc décidé d'établir un classement de popularité arbitraire basé sur mes propres impressions et observations :

1. Dropbox

Il est de notoriété public que Dropbox est actuellement leader du marché avec plus de 50 million d'utilisateurs *[12]*. Historiquement ce n'est pas la première solution à être apparue sur le marché, mais c'est la première à avoir rendu la synchronisation, et l'ensemble des autres services aussi simples, le propulsant donc au rang de référence du domaine. Avant de lancer iCloud, Apple avait même tenté de racheter la firme pour 800 million de dollars *[13]*.

2. Google Drive

Le classement à la deuxième place de Google Drive est quelque peu spéculatif. Il est avant tout basé sur son potentiel d'acquisition d'utilisateurs. En effet, avec environ 350 million de comptes GMail (Janvier 2012) *[14]*, 170 million de comptes G+ (Avril 2012) *[15]*, plus de 200 million de smartphones Android vendu (novembre 2011) *[16]* et la stratégie de Google visant à faire interagir tous ses services (Google Docs et Google Drive interagissant déjà ensemble), Google Drive révèle un énorme potentiel.

3. iCloud

Dans la même idée que Google, iCloud peut profiter de la quantité de personnes utilisant des appareils ou des services Apple (plus de 160 million d'iPhones *[17]* et 55 million d'iPads *[18]* vendus jusqu'en mars 2012). Avec son image de marque et ses capacités en marketing, iCloud a de bons arguments pour venir concurrencer Google Drive et Dropbox. En revanche, sur le plan fonctionnel, ils ont encore du retard à rattraper, ce qui justifie cette troisième place.

4. SkyDrive

Je classe SkyDrive à la quatrième place. Son principal atout est le prix et l'intégration dans l'environnement Windows, ce qui est particulièrement profitable aux entreprises. En revanche, il est encore loin derrière Dropbox en termes d'ergonomie et de simplicité d'utilisation, ce qui risque de beaucoup freiner son expansion sur le marché des clients privés.

5. 4Shared

4Shared est un concurrent sérieux à tous ceux cités plus haut. Sa stratégie consiste à fournir aux utilisateurs toute une gamme d'outils leur permettant d'exploiter leurs fichiers directement en ligne, ce qui à terme peut devenir un atout majeur par rapport à des concurrents comme Dropbox ou iCloud. Les prix pratiqués sont également un atout majeur de la plateforme.

6. MediaFire

Après analyse et tests, il s'avère que MediaFire s'adresse plutôt à des entreprises qu'à des particuliers. La gamme de services proposés est tout à fait adéquate à ce segment, et selon leur propre site, beaucoup d'entreprises classées au « Fortune 500 » utilisent leurs services *[19]*. Cependant, avec l'arrivée de SkyDrive sur le marché (cassant les prix au passage), il va être difficile pour MediaFire de résister longtemps sans proposer quelque chose de nouveau pouvant les démarquer.

7. PutLocker

Avec la fermeture de MegaUpload, PutLocker a vu son trafic quadruplé selon le magazine Numerama, catapultant ce service dans les grands acteurs du marché *[20]*. La grande force de PutLocker réside dans l'illimitation de l'espace de stockage mis à disposition, ce qui est un avantage certain par rapport à ses concurrents. En revanche, tant que la plateforme ne proposera pas un service de synchronisation, elle ne sera pas en mesure de concurrencer les solutions citées ci-dessus.

4.4.3 POSSIBILITÉS ET RAPIDITÉ D'INTÉGRATION

Les possibilités et la rapidité d'intégration à d'autres outils a été évaluée sur la base de deux groupes de critères distincts. Le premier est la quantité de fonctionnalités accessibles via l'API :


Lister


Télécharger


Uploader


Créer un répertoire


Supprimer


Gérer l'accès

Le second est la qualité de la documentation qui a été évalué sur la base de trois critères :

- Documentation officielle : clarté et exhaustivité des explications, facilité d'accès
- Exemples : quantité et pertinence
- Communauté : nombre de résultats obtenus dans les moteurs de recherche lors de saisies des mots clés « [nom de la solution] API » suivi alternativement des différentes fonctionnalités listée ci-dessus

Le tableau suivant permet de visualiser les résultats de chaque solution pour ces critères :

Solution	Fonctionnalités	Qualité de la documentation
Google Drive		**Doc officielle** : bien faite **Exemples** : légers **Communauté** : active
iCloud	N/A	N/A
SkyDrive		**Doc officielle** : bien faite **Exemples** : complets **Communauté** : active
Dropbox		**Doc officielle** : très bien faite **Exemples** : complets **Communauté** : active
4Shared		**Doc officielle** : bien faite **Exemples** : légers **Communauté** : limitée
MediaFire		**Doc officielle** : légère **Exemples** : légers **Communauté** : limitée
PutLocker		**Doc officielle** : bien faite **Exemples** : complets **Communauté** : inexistante

Tableau 5 Possibilités d'intégration des gestionnaires de fichiers en ligne

La richesse de la documentation et les possibilités offertes par l'API font de Dropbox la solution la plus facilement intégrable à d'autres outils. SkyDrive est derrière avec une documentation légèrement moins fournie, mais reste facilement intégrable. Les autres solutions requièrent des efforts grandement plus importants pour être intégrées.

4.5 CONCLUSION

La première chose qui m'a surprise en réalisant cette étude comparative a été le nombre de solutions différentes proposant les mêmes services. En effet, 17 solutions correspondent aux critères de recherche mentionnés, et plus de 50 si l'on enlève la nécessité d'une API *[4]*. Cette profusion témoigne de l'ampleur du marché et du besoin qu'ont les gens de gérer et partager efficacement leurs données.

Dropbox est actuellement le leader du marché. Cette place de leader est largement méritée au vue des fonctionnalités offertes, de la simplicité d'utilisation et de l'ancienneté. Néanmoins, il sera intéressant d'observer comment vont se développer Google Drive, SkyDrive et iCloud. En effet, ces trois solutions offrent des prestations quasi similaires mais beaucoup moins chers. Elles ont en outre la possibilité de faire interagir leur solution de stockage en ligne avec d'autres services qu'elles proposent.

En comparaison de ce qui précède, je fais le choix d'intégrer Dropbox dans le gestionnaire de fichiers tokiwi. Il est peu probable que Dropbox perde sa place de leader avant la fin du travail. De plus, son API est particulièrement bien conçue ce qui permettra d'intégrer cette solution beaucoup plus facilement qu'une autre.

5 IMPLÉMENTATION

Nous avons vu dans les deux chapitres précédents les enjeux de l'application à développer et les fonctionnalités proposées par différents concurrents. Nous avons également déterminé que Dropbox serait le solution intégrée au gestionnaire de fichiers tokiwi.

Nous allons maintenant aborder le cœur du sujet de ce travail de Bachelor, c'est-à-dire l'implémentation de l'application, ses fonctionnalités et son architecture.

Vous trouverez en annexe un document expliquant le développement d'applications tokiwi (« 13.5 Introduction au développement d'applications tokiwi »). De nombreux termes propres à la plateforme utilisés dans ce chapitre y sont expliqués.

Ce chapitre se décompose en trois parties :

- La première est un résumé de tout le chapitre, expliquant dans les grandes lignes comment fonctionne l'application.
- La deuxième partie explique plus en profondeur les principales fonctionnalités de l'application selon l'approche suivante :
    - Fonctionnalités : chapitre passant en revue toutes les fonctionnalités utilisateurs
    - Interface graphique : chapitre expliquant comment fonctionne l'interface graphique
    - Logique applicative : chapitre expliquant comment sont traitées les données
    - Base de données : chapitre expliquant comment sont persistées les données
    - Gestion de projet : chapitre expliquant le temps nécessaire au développement des fonctionnalités et les éventuels problèmes rencontrés
- La troisième partie est une conclusion critiquant les principaux choix techniques opérés.

5.1 ARCHITECTURE GÉNÉRALE

Voici un schéma illustrant les deux composants tokiwi qui ont été développés :

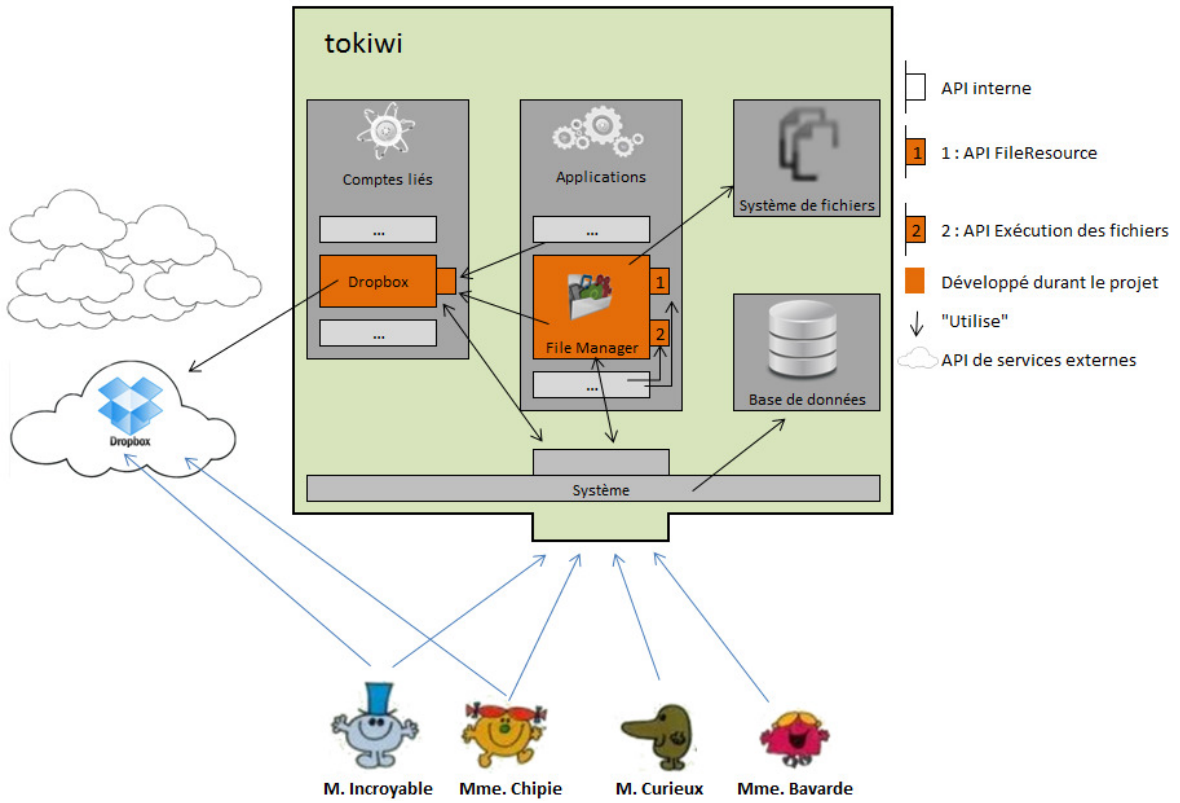


Figure 5 Schéma de l'intégration de l'application dans la plateforme

« File Manager » est le nom de l'application tokiwi qui a été développée. Les utilisateurs peuvent y accéder au travers de l'interface graphique de la plateforme. Elle s'intègre aux autres applications tokiwi grâce à deux API. La première permet d'accéder au gestionnaire de fichiers d'une communauté, et de le manipuler. La deuxième permet d'exécuter des fichiers dans le but de les modifier ou de lire leur contenu. Le gestionnaire de fichiers utilise également le système de liaison de comptes de la plateforme afin d'accéder à la Dropbox des utilisateurs.

Actuellement, les fichiers sont stockés sur le même serveur que la plateforme. A l'avenir, le service de stockage en ligne d'Amazon sera utilisé. Il faudra donc adapter l'application pour qu'elle accède à ces services.

Le système de comptes liés de la plateforme a également été développé durant ce travail de Bachelor. Il permet de standardiser l'intégration de services externes au sein de la plateforme, accélérant et uniformisant ainsi leur mise en œuvre. Le composant « Dropbox » du système de

comptes liés est une abstraction de l'API Dropbox. Toutes les applications peuvent ainsi accéder à ses fonctionnalités au travers de l'API PHP mise en place.

Le schéma ci-dessous offre une vue plus détaillée du fonctionnement de l'architecture trois tiers de l'application File Manager :

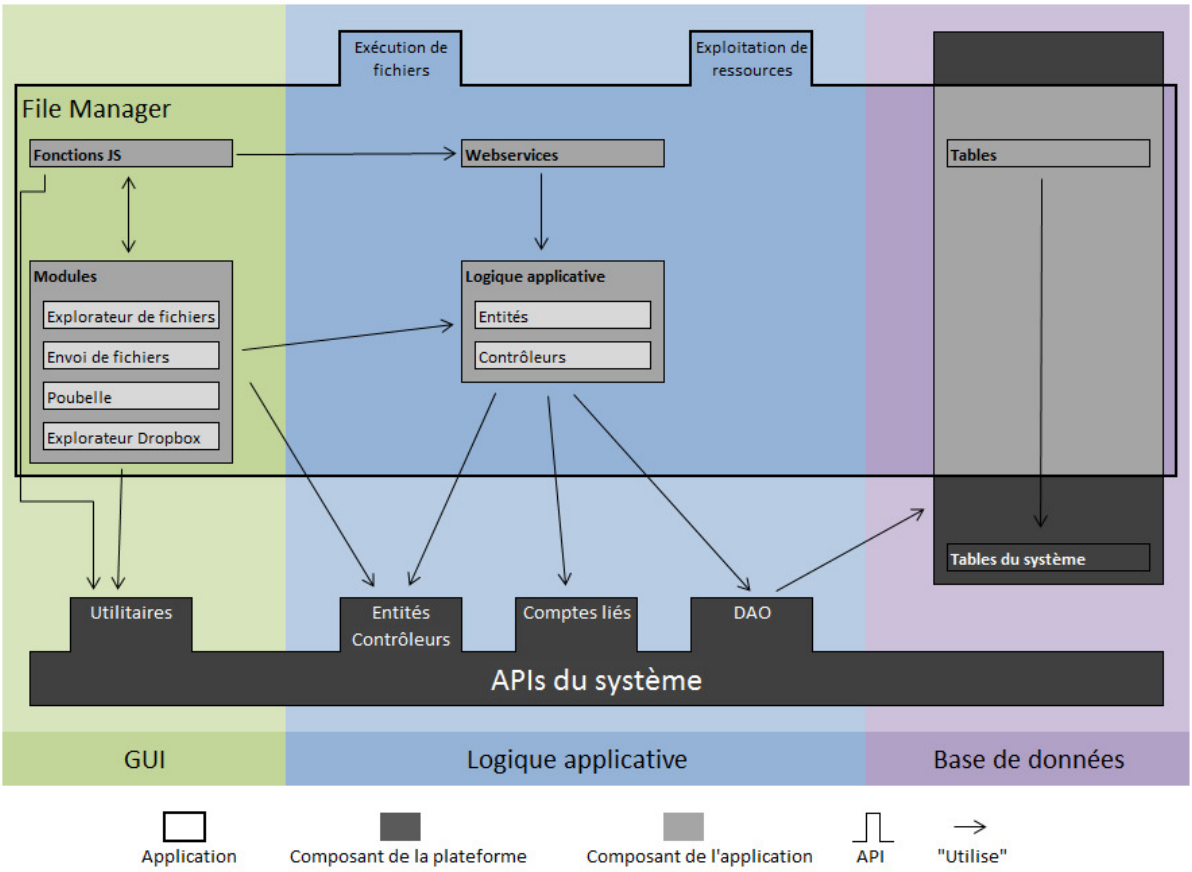


Figure 6 Schéma de l'architecture de l'application

L'interface graphique est composée de 4 modules accédant aux entités et contrôleurs de l'application pour afficher leur contenu. Toutes les actions utilisateurs sont traitées par des fonctions Javascript. Puisqu'il n'est pas possible d'accéder directement à la logique applicative depuis une fonction Javascript, on se sert de scripts PHP, ici appelé « webservices », faisant office d'interface.

Les utilitaires de l'API du système permettent d'uniformiser le fonctionnement de l'interface graphique et de réutiliser des composants, tels que des classes CSS ou des fonctions Javascript.

Comme expliqué dans le précédant schéma, la logique applicative offre deux APIs permettant à l'application d'interagir avec les autres applications de la plateforme. Les webservices déclenchent des actions dans les entités ou les contrôleurs, et permettent de retourner leurs informations au

format JSON afin d'être exploitées par les fonctions Javascript. Le format JSON a été préféré au format XML car il est moins verbeux et peut être utilisé dans des fonctions Javascript sans transformation préliminaire.

L'accès à la base de données se fait au travers du DAO de la plateforme, uniquement depuis les entités et les contrôleurs. Ainsi en utilisant uniquement le DAO de la plateforme pour accéder à la base de données et en limitant à un seul composant son utilisation, on améliore la sécurité de l'application et son indépendance à la couche de données.

L'application dispose de plusieurs tables pour persister les informations. Celles-ci interagissent avec les autres tables du système au travers de clés étrangères.

5.2 EXPLORATION DE FICHIERS

L'un des principaux objectifs de l'application était de créer une expérience utilisateur similaire à celle que l'on peut avoir en utilisant le gestionnaire de fichiers Windows. Il fallait donc qu'au travers de l'interface graphique, il soit possible de naviguer dans une arborescence de répertoires, de rechercher des ressources, de sélectionner des ressources et de les déplacer ou de les copier à l'aide d'un « drag'n'drop ».

Pour des informations plus détaillées sur le cahier des charges de cette fonctionnalité, il faut se référer aux user stories 1, 3-9, 11, 12, 21-23 et 61.

5.2.1 FONCTIONNALITÉS

L'interface utilisateur a été conçue de manière à être la plus épurée possible. Voici comment elle se décompose :

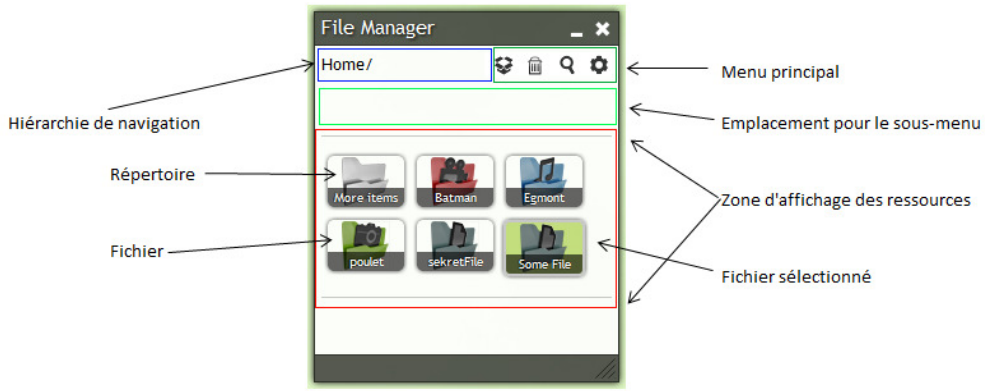


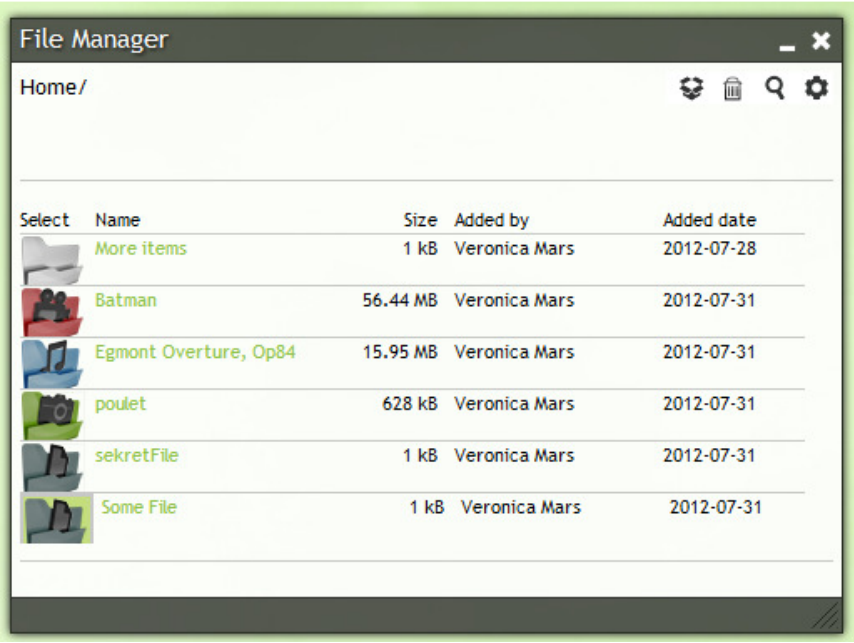
Figure 7 Conception de l'interface utilisateur de l'explorateur de fichiers

L'interface est minimale et se focalise sur ce qui est important : les ressources. Les menus principaux permettent d'accéder à la Dropbox de l'utilisateur et à la poubelle (fonctionnalités qui seront détaillées plus loin dans ce document), à la recherche et aux options de manipulation des ressources.

Avec ce genre d'interface il est donc tout à fait possible de se servir du gestionnaire de fichiers dans une petite fenêtre, ce qui s'avère très pratique dans le cadre d'utilisation de tokiwi. Ceci permet également d'ouvrir plusieurs instances du gestionnaire de fichiers pour pouvoir passer un fichier d'un répertoire à un autre par exemple.

En revanche, si l'on se situe dans une grande hiérarchie de répertoires, une partie du chemin sera tronquée par manque de place pour afficher le chemin complet.

L'utilisateur a également la possibilité d'afficher le contenu sous forme de liste. Dans cette configuration il accède à plus d'informations sur les ressources comme on peut le constater sur la copie d'écran suivante :

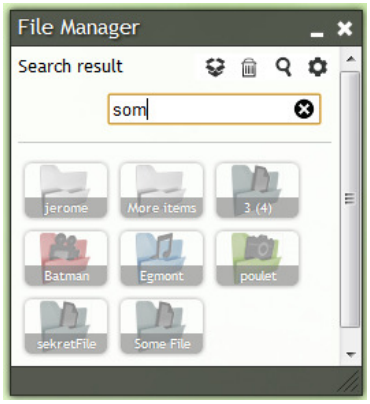


Copie d'écran 1 Explorateur de fichiers en mode liste

On constate néanmoins que l'explorateur de fichiers nécessite plus de place pour afficher toutes les informations.

Lorsque l'utilisateur effectue une recherche, la recherche s'opère sur l'ensemble des ressources contenues dans le gestionnaire de fichiers de la communauté au fur et à mesure que l'utilisateur

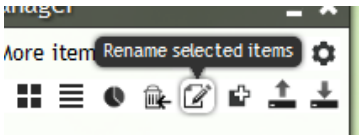
saisit des lettres. Un cadre vient griser le contenu des fichiers pour indiquer à l'utilisateur qu'un chargement est en cours, comme on peut le constater sur l'image suivante :



Copie d'écran 2 Recherche dans l'explorateur de fichiers

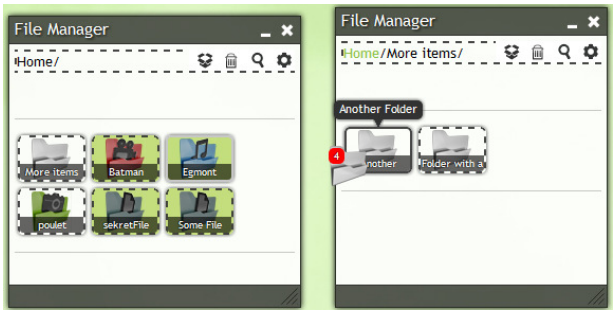
Tous les menus et toutes les ressources affichent une info-bulle contenant respectivement leur fonction et leur nom lorsqu'on passe la souris au dessus.

Copie d'écran 3 Info-bulle sur une ressource



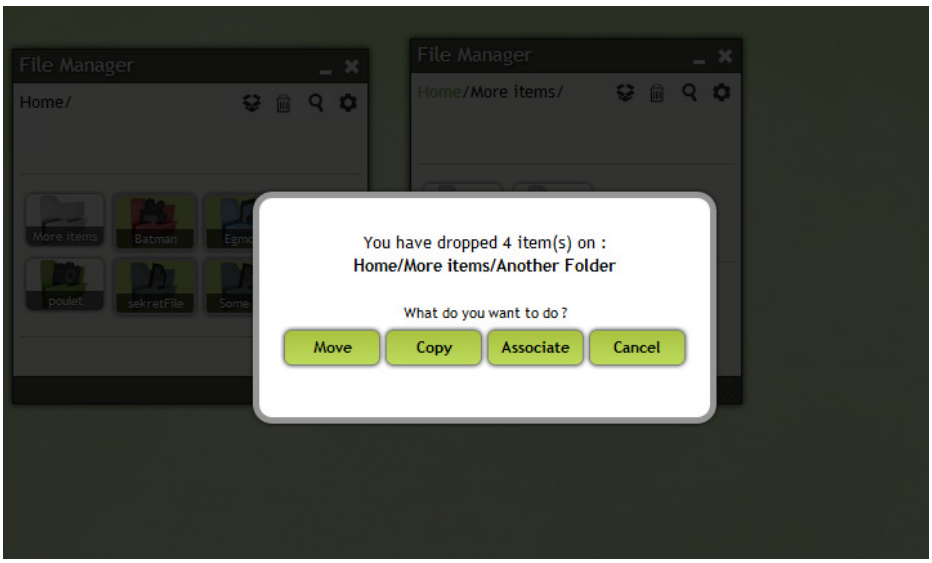
Copie d'écran 4 Info-bulle sur un menu

L'utilisateur a la possibilité de déplacer ou copier plusieurs fichiers en sélectionnant des ressources, puis en les déplaçant à l'aide de la souris sur un répertoire comme illustré dans l'image suivante :



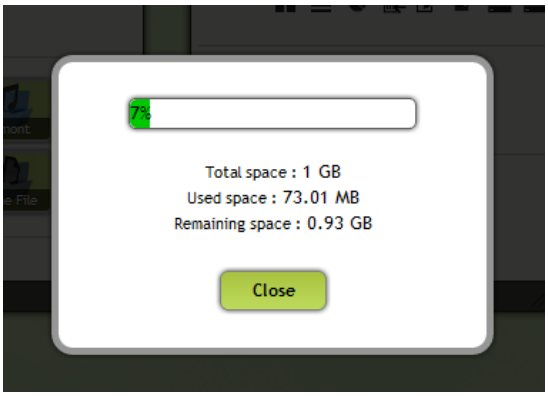
Copie d'écran 5 Illustration du "drag'n'drop"

Après quoi une pop-up apparaitra pour demander à l'utilisateur ce qu'il désire faire comme action :



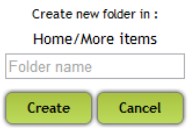
Copie d'écran 6 Pop-up permettant de déplacer, copier ou associer des ressources

L'utilisateur a la possibilité de connaitre à tout moment la quantité d'espace dont dispose la communauté :

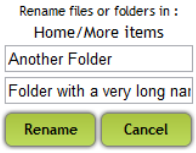


Copie d'écran 7 Utilisation de l'espace de stockage

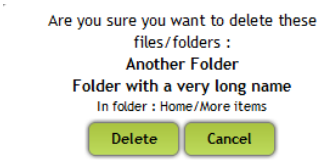
Finalement, il est possible de créer des répertoires, supprimer des ressources et renommer des ressources. Chaque formulaire s'affiche également dans une pop-up :



Copie d'écran 8 Création de répertoire



Copie d'écran 9 Renommage de ressources



Copie d'écran 10 Suppression de ressources

5.2.2 FONCTIONNEMENT DE L'INTERFACE GRAPHIQUE

Le module est composé de 7 formulaires différents. Les deux principaux sont « explorer.php » et « explorerContent.php ». Ils permettent respectivement d'afficher la hiérarchie courante et les menus puis le contenu du répertoire courant. Les autres formulaires sont affichés dans les pop-ups de suppression de ressources, de création de répertoires et de renommage de ressources.

Lorsque l'instance du module est chargée, elle va chercher dans ses paramètres le répertoire courant. Après quoi, elle appelle la méthode qui permet de récupérer son contenu puis elle le met en page. Si aucun répertoire ne figure dans les paramètres, la racine est alors affichée. Par défaut le contenu est mis en page sous forme d'icônes. Il est possible de passer en mode liste en enregistrant le paramètre correspondant dans les paramètres de l'instance du module. La navigation se fait en enregistrant le nouvel identifiant du répertoire courant dans les paramètres du module, puis en rafraichissant le contenu du module. Puisque tous les paramètres d'affichage sont stockés dans les paramètres de l'instance du module, son état est persistant. L'utilisateur peut quitter sa communauté et y revenir, il trouvera ses instances d'exploration de fichiers dans le même état.

Lorsque l'utilisateur saisit des caractères dans la barre de recherche, ces caractères sont sauvegardés dans les paramètres de l'instance du module et le contenu de l'explorateur de fichiers est mis à jour avec les résultats de la recherche. Il est possible de supprimer la recherche en cours en supprimant les caractères des paramètres de l'instance du module. Les recherches sont donc aussi persistantes.

Une fois les éléments HTML chargés, une fonction Javascript est exécutée pour mettre en place tous les évènements et les interactions sur les différents composants de l'explorateur. L'interaction jQuery « Droppable » est appliquée à la hiérarchie et à chaque ressource listée dans l'explorateur. Ces dernières se voient également appliquée l'interaction jQuery « Selectable ». De plus, elles écoutent l'évènement « selected » (issu de l'interaction Selectable). Lorsque cet évènement est capturé, l'interaction jQuery « Draggable » est appliquée à toutes les ressources sélectionnées.

La décomposition des éléments de l'interface graphique est claire. Elle est donc relativement facile à maintenir à jour. L'API mise en place dans le « backend » (voir chapitre suivant) contribue grandement à la simplification des choses.

En termes de performance et de stabilité, les interactions jQuery utilisées n'ont pas posé de problème de performance durant les tests que j'ai effectué et elles sont supportées par la grande majorité des navigateurs récents (Internet Explorer 9 inclus).

Le système de paramètres d'instance de module est largement mis à profit dans cette implémentation. Toutes les options d'affichage de l'utilisateur sont sauvées d'une session à une autre de manière transparente, ce qui améliore grandement l'expérience utilisateur.

5.2.3 LOGIQUE APPLICATIVE

Un gestionnaire de fichiers est composé de deux types d'objets, des fichiers et des répertoires. Ceux deux objets partageants de nombreuses propriétés, il était donc logique de créer un objet commun dont ils hériteraient tous les deux. Ceci nous donne donc le diagramme de classe suivant :

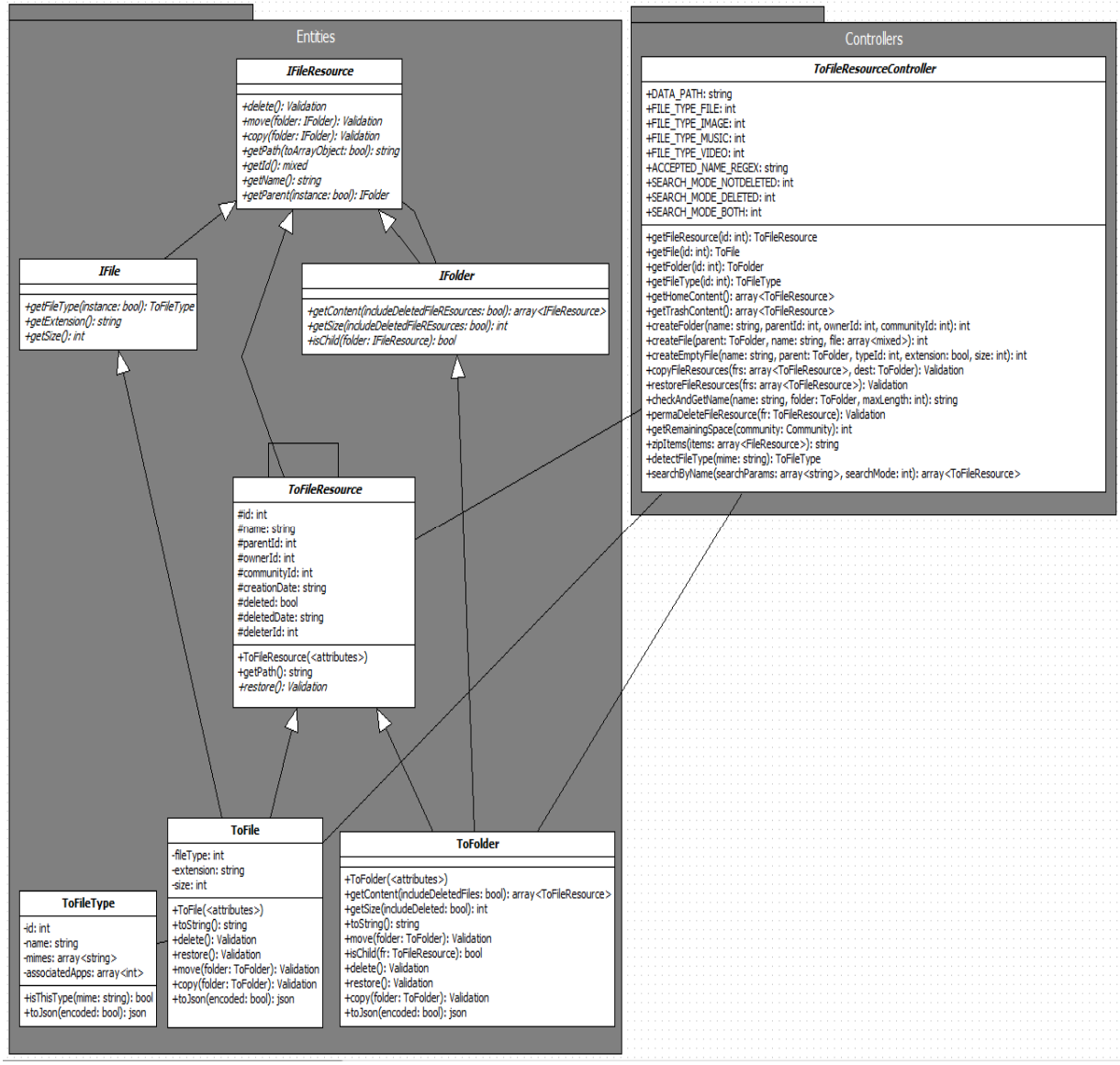


Figure 8 Diagramme de classe de l'exploration de fichiers

L'application est construite sur des entités/contrôleurs. Un contrôleur est utilisé pour gérer le cycle de vie des entités (création, instanciation et suppression) et des actions sur des collections d'entités. Les entités sont quand à elle responsables de persister les données et assurer leur cohérence.

La classe « ToFileResource » est la classe commune aux fichiers (« ToFile ») et aux répertoires (« ToFolder »). Nous noterons la présence de la classe « ToFileType » permettant de déterminer le

type de fichier (par défaut les types existants sont « Fichier », « Image », « Music » et « Video »). Cette classe prend tous son sens pour l'exécution de fichiers, sujet abordé plus loin dans ce document.

Au début du développement, j'avais défini toutes les méthodes abstraites dans la classe « ToFileResource ». Mais plus tard dans le développement je me suis heurté à deux problèmes. Le premier est que pour certaines méthodes, telles que « getSize() », la signature changeait d'une classe à une autre. En effet dans « ToFolder » il a fallu ajouter un paramètre permettant d'inclure ou non les fichiers supprimés dans la taille totale du répertoire. Dans la version 5.3 de PHP, il est encore possible d'ajouter des paramètres à une méthode héritée, mais ce ne sera plus le cas à partir de PHP 5.4. Le second problème de cette architecture est la redondance des classes lors de l'intégration de Dropbox car beaucoup d'attributs sont similaires. Avec cette architecture il aurait été impossible d'utiliser simultanément des ressources issues de Dropbox et du gestionnaire de fichiers tokiwi.

J'ai donc résolu ces deux problèmes en créant trois interfaces permettant d'établir le fonctionnement minimal des entités du gestionnaire de fichiers. La logique reste la même, une interface détermine les comportements et attributs communs aux fichiers et répertoires, puis deux autres interfaces définissent les spécificités de chacun.

L'implémentation de cette solution reste cependant partielle. Le contrôleur n'utilise pas le type de l'interface pour effectuer ses opérations, mais celui des classes concrètes. Les modifications pourraient rapidement être faites, mais la quantité de tests à effectuer pour les valider est immense, et je n'ai pas eu suffisamment de temps pour m'en occuper.

5.2.4 BASE DE DONNÉES

La base de données permettant de persister les données est relativement simple. L'unique difficulté rencontrée a été de trouver le meilleur moyen de représenter l'héritage entre les classes « ToFileResource » et « ToFile » / « ToFolder ».

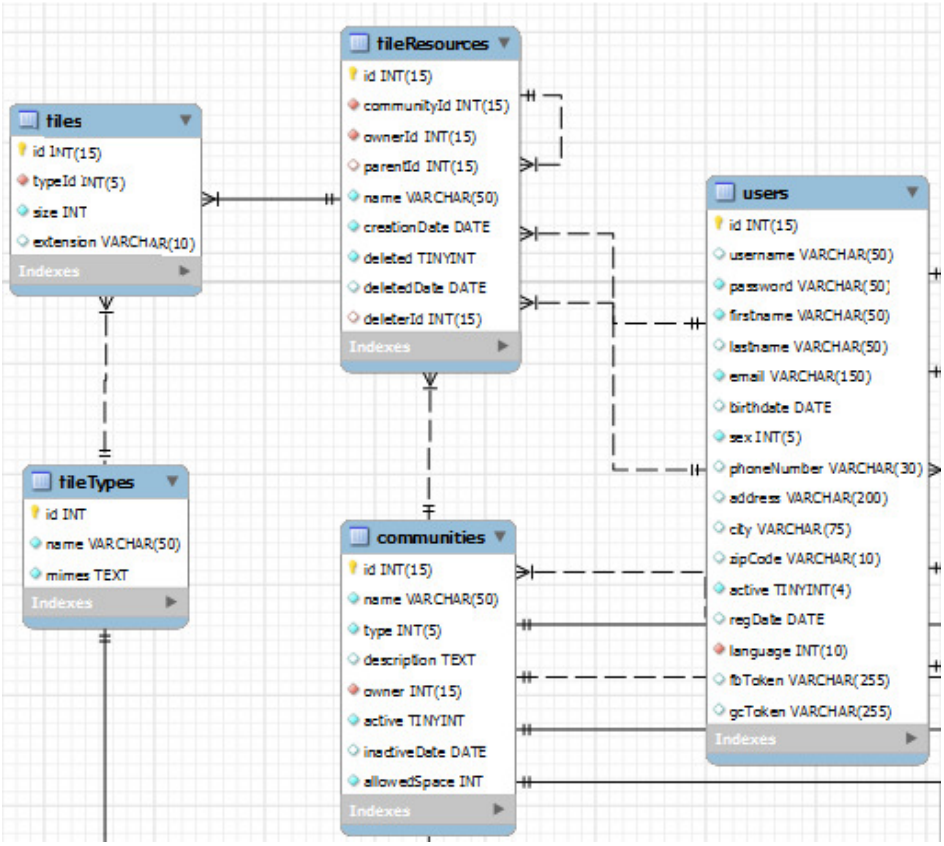


Figure 9 Modèle de la base de données pour l'exploration de fichiers

J'ai opté pour la création de deux tables. La première listant tous les attributs de la classe « ToFileResource », la deuxième ceux de la classe « ToFile », dont la clé primaire est une clé étrangère de la table correspondant à la classe « ToFileResource ». Il n'a pas été nécessaire de créer une table spécifique pour les répertoires car ceux-ci n'ont pas d'attributs supplémentaires. Ainsi, une ressource est un fichier si un enregistrement existe dans les tables « fileResources » et « files » et une ressource est un répertoire s'il n'existe qu'un enregistrement dans la table « fileResources ».

Dans la configuration actuelle, un fichier ne peut être que d'un seul type. Une amélioration possible serait de créer une table de relation entre les fichiers et les types de fichiers pour permettre à un fichier d'être de plusieurs types. Aucune user story n'allait dans ce sens donc je ne l'ai pas fait. Mais plusieurs scénarios intégrant d'autres applications seraient envisageables à l'avenir.

Nous noterons que les tables créées ne servent qu'à stocker des données utiles au fonctionnement de l'application. Une autre amélioration possible serait d'ajouter des colonnes ou même des tables permettant d'assurer un suivi de l'évolution des données. Ceci permettrait de comprendre comment les utilisateurs se servent de l'application.

5.2.5 GESTION DE PROJET

Le développement de ces fonctionnalités a été effectué principalement sur le sprint 3 et représentait au total 34 story points réalisés en 69.5 heures de travail. La charge de travail a donc correctement été estimée.

J'ai effectué le travail en trois grosses étapes. La première a été de créer toutes les fonctionnalités « backend », incluant la base de données. J'ai ensuite mis en place l'interface utilisateur. Et j'ai finalement connecté le « backend » et le « frontend » en implémentant les webservices.

Les seuls contretemps que j'ai rencontrés ont été :

- La préparation des examens de fin de module et les objectifs de Business eXperience qui ont empiété sur le temps imparti au développement (environ 2 jours).
- Un bug de la plateforme qui empêchait le chargement dynamique des scripts et des CSS que j'ai dû résoudre pour poursuivre le développement (environ une demi journée).
- Des cas de bord imprévus lors du déplacement et la copie de ressources (déplacement d'un répertoire à l'intérieur de lui-même par exemple) (environ 1 jour).

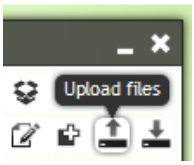
5.3 ENVOI, STOCKAGE ET TÉLÉCHARGEMENT DES FICHIERS

Afin de rendre viable l'utilisation du gestionnaire de fichiers, l'envoi de fichiers est une fonctionnalité qui doit être simple et efficace. Beaucoup d'efforts ont donc été investis dans ce sens. La principale préoccupation du stockage de fichiers était la sécurité, chose qui a largement été considérée durant le développement de ces fonctionnalités. Finalement, le téléchargement des fichiers n'était pas une préoccupation majeure car à terme le but de la plateforme est de pouvoir exploiter les fichiers directement en ligne. Cependant les fonctionnalités permettant l'exploitation en ligne des fichiers n'existent pas encore. Une multitude d'options de récupération des fichiers ont donc été mises en place.

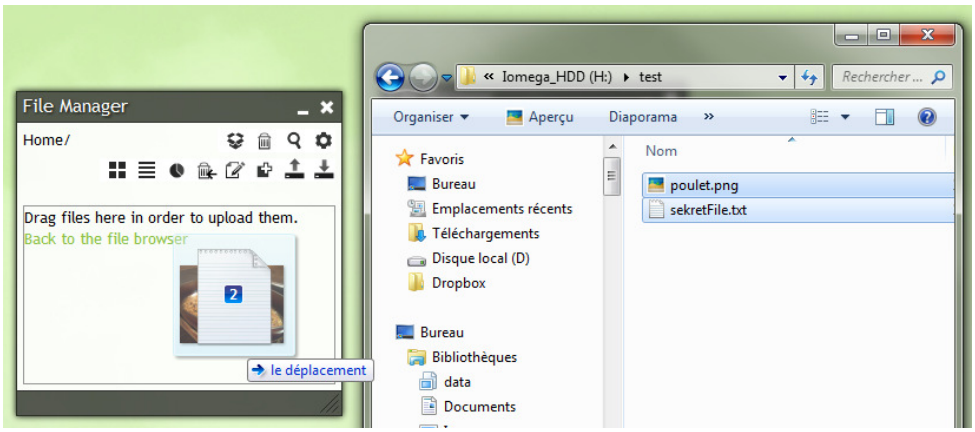
Pour des informations plus détaillées sur le cahier des charges de ces fonctionnalités, il faut se référer aux user stories 13-15 et 18-20.

5.3.1 FONCTIONNALITÉS

Il est possible d'envoyer des fichiers sur la plateforme de deux manières différentes. Soit en cliquant sur le bouton « Upload files », soit en glissant des fichiers depuis l'explorateur de fichiers Windows vers un répertoire du gestionnaire de fichiers tokiwi.

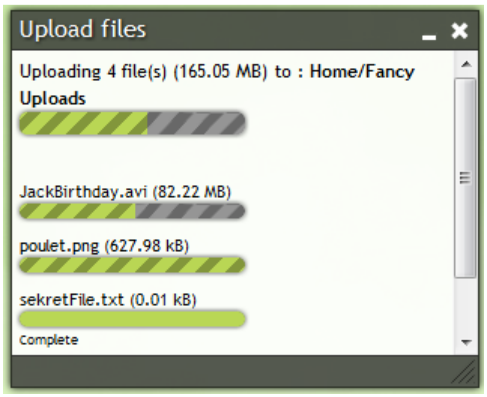


Copie d'écran 11 Envoi de fichiers classique



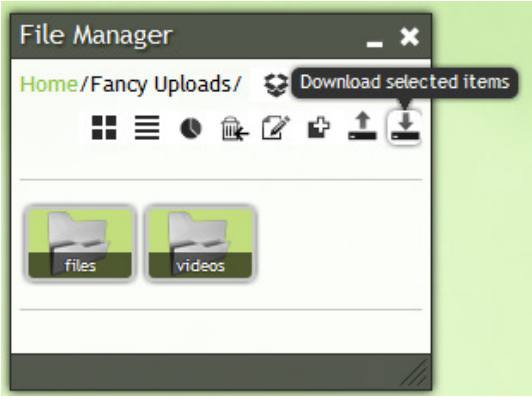
Copie d'écran 12 Envoi de fichiers "Drag'n'Drop"

Une fois les fichiers sélectionnés, un nouveau module apparait permettant de suivre l'évolution des « uploads ».



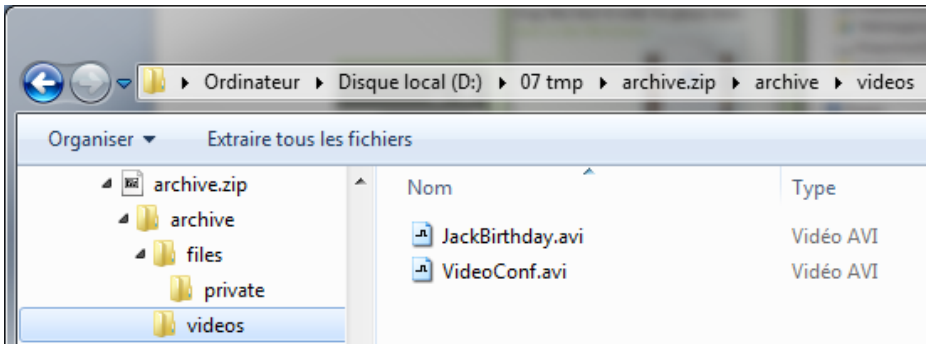
Copie d'écran 13 Progression de l'envoi de fichiers

Le téléchargement de fichiers est tout autant simple. L'utilisateur n'a qu'à sélectionner les fichiers qu'il désire télécharger et appuyer sur le bouton « Download selected items ».



Copie d'écran 14 Téléchargement de fichiers

Si seul un fichier a été sélectionné, celui-ci est envoyé tel quel. Si plusieurs fichiers ont été sélectionnés, ou qu'un répertoire a été sélectionné, ceux-ci sont placés dans une archive ZIP avant d'être envoyés. Tout le contenu des répertoires sélectionnés est inclus dans l'archive.



Copie d'écran 15 Contenu d'une archive téléchargée

5.3.2 FONCTIONNEMENT DE L'INTERFACE GRAPHIQUE

L'envoi de fichiers de manière asynchrone n'est pas quelque chose de nouveau, et il existe de nombreux plug-ins permettant de rapidement le mettre en œuvre dans un site web. Malheureusement, je n'en ai trouvé fonctionnant dans l'environnement tokiwi. Dans la plupart des cas, le problème venait d'une potentielle utilisation multiple du plug-in dans la même page, générant ainsi un comportement hasardeux de l'ensemble de la page.

Plutôt que de chercher à en adapter un, j'ai préféré développer mon propre plug-in. C'était une bonne occasion d'apprendre comment fonctionnent les objets XHR et cela m'offrait une plus grande liberté dans le rendu à l'utilisateur.

Comme présenté dans le chapitre précédent, il est possible de sélectionner des fichiers de deux manières :

- Grâce à un champ « input » de type « files ». HTML 5 a introduit l'attribut « multiple » à ce champ, rendant ainsi possible de sélectionner plusieurs fichiers à la fois. Ce champ prend l'apparence d'un bouton comme les autres menus de l'explorateur de fichiers.
- Grâce au « drag'n'drop » de l'explorateur de fichiers Windows vers celui de tokiwi. HTML 5 a également introduit 7 nouveaux évènements qui permettent au navigateur d'interagir avec le système d'exploitation du client. Dans le cas présent, l'évènement « dragover » sert à afficher la zone de dépôt des fichiers, et l'évènement « drop » sert à récupérer la liste des fichiers qui ont été déposés.

L'envoi de fichiers s'opère ensuite dans un module dédié à cet effet. Il s'exécute tout seul après que le module d'exploration de fichiers ait détecté un changement dans son champ de sélection de fichiers. Dès que le module s'exécute, il va récupérer les informations des fichiers grâce à l'API Javascript « Files » introduite avec HTML 5. Un script construit le formulaire avec les éléments suivants :

- Un bloc contenant les informations générales de l'envoi de fichiers (nombre de fichiers, taille totale, barre de progression)
- Un bloc par fichier contenant le nom du fichier, sa taille, la barre de progression et une zone pour afficher un message

Une fois le formulaire construit, un deuxième script vérifie que la taille des fichiers n'excède pas 100 Mo et que leur nom ne contienne pas de caractère invalide. Si c'est le cas, une erreur est affichée dans le bloc correspondant au fichier.

Dès que les fichiers ont été validés, un script envoie séparément chaque fichier à l'aide d'une requête HTTP asynchrone. Une fonction Javascript est associée aux évènements « progress », « timout » et « error » de l'objet XHR de chaque fichier afin de suivre l'évolution de l'envoi. En cas d'erreur, celle-ci est affichée à l'utilisateur. En cas de progression normale, les barres de progression du fichier et de l'ensemble des fichiers sont mises à jour.

Et dès que l'ensemble des fichiers ont été envoyés, tous les explorateurs de fichiers ouverts par l'utilisateur sont rafraichis.

Aucune limite d'envoi de fichiers n'a été établie. L'utilisateur peut potentiellement envoyer un très grand nombre de fichiers simultanément. Ceci a été testé localement et fonctionne très bien. Cependant dans l'environnement de production actuel de tokiwi (serveur mutualisé chez Infomaniak), seuls les 10 à 15 premiers fichiers sont envoyés. Je soupçonne Infomaniak de limiter le nombre de requête HTTP provenant de la même adresse IP dans un intervalle de temps limité (pour des raisons de sécurité et de performance). Une amélioration permettant de résoudre ce problème serait de ne pas débuter le chargement des fichiers en parallèle, mais en série.

Les fonctionnalités mises en place pour envoyer des fichiers répondent pleinement aux exigences fixées, cependant elles s'appuient lourdement sur les nouvelles fonctionnalités de HTML 5, rendant ainsi l'application incompatibles avec une multitude de navigateurs. Parmi ceux-ci il y a Internet Explorer 9, ce qui est un sérieux handicape. Pour obtenir une liste détaillée de la compatibilité des navigateurs avec HTML 5, je recommande le site suivant : http://html5test.com/.

Néanmoins, Microsoft a annoncé la sortie d'Internet Explorer 10 pour le courant de cet automne, et cette version sera compatible avec les fonctionnalités implémentées (http://html5test.com/compare/browser/ie10.html).

Le téléchargement de fichiers, coté interface graphique, est très simple. Une fonction Javascript appelle un webservice de l'application qui lui retourne une adresse permettant d'accéder au fichier demandé. Cette adresse est attribuée à une « iframe » cachée contenue dans l'explorateur de fichiers, ce qui a pour effet de déclencher le téléchargement du dit fichier par le navigateur.

Lorsqu'une beaucoup de fichiers sont sélectionnés, le webservice nécessite un certain temps (de l'ordre de 1 à 5 secondes) avant de générer le lien de téléchargement. Durant ce laps de temps, l'utilisateur pourrait croire que rien ne se passe. Pour éviter cela, un message indiquant que le téléchargement est en préparation s'affiche directement après avoir cliqué sur le bouton.

5.3.3 LOGIQUE APPLICATIVE

Dans l'environnement de production actuel de tokiwi, les fichiers sont enregistrés sur le même serveur que le serveur web. A terme, l'objectif serait d'utiliser les services de stockage en ligne proposés par Amazon. Mais étant donné que ce n'était pas encore le cas durant la réalisation de ce travail, j'ai mis en place une solution temporaire.

Lors de la création d'une communauté, un répertoire portant le nom de l'identifiant de la communauté est créé dans le répertoire « data/fs » à la racine du site. Tous les fichiers envoyés par

des utilisateurs sont ensuite stockés dans le répertoire correspondant à leur communauté. L'accès à ces fichiers est sécurisé par un fichier « .htaccess » n'autorisant que l'utilisateur technique du serveur web.

Les répertoires que les utilisateurs créent dans leurs communautés n'existent pas physiquement dans le système de fichiers, mais uniquement en base de données. Ainsi le déplacement de ressources et la création de répertoires ne sont que des opérations en base de données, ce qui améliore grandement les performances.

L'exemple ci-dessous permet de se rendre compte où sont stockées les différentes informations liées au gestionnaire de fichiers :

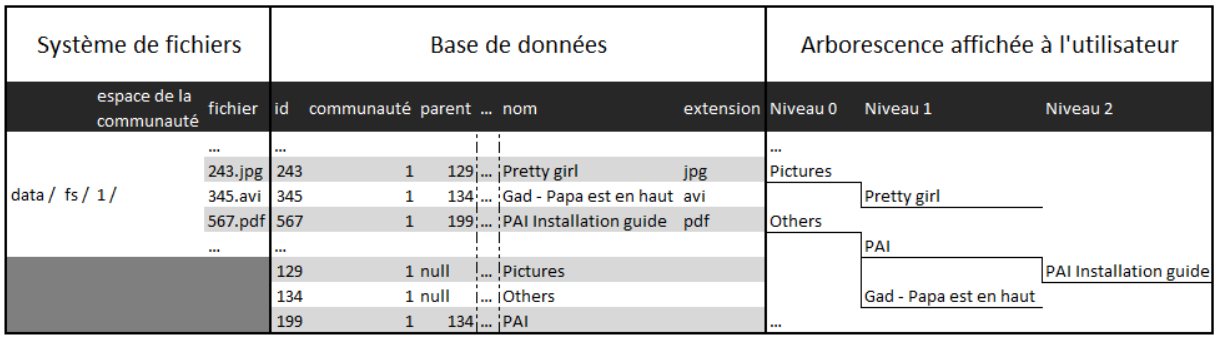


Figure 10 Exemple de ressources dans le système de fichiers et la base de données

Cette solution est parfaitement adéquate dans un environnement de test tel que c'est le cas actuellement. Mais à plus grande échelle, ce système poserait de nombreux problèmes. Premièrement, un serveur web n'est pas prévu pour faire serveur de fichiers. Une machine dédiée, avec des temps d'accès disque optimisés, serait nécessaire. De plus, aucun gestionnaire de fichiers ne peut traiter des répertoires contenant plus d'un millier de ressources sans baisse de performance. Or, il est évident que dans un avenir plus ou moins proche, tokiwi comptera plus d'un millier de communautés, ce qui posera un problème d'évolutivité (« scalability ») du système.

Lorsqu'un fichier est envoyé au serveur, l'opération de création du fichier se déroule en 3 étapes. La première consiste à vérifier que la communauté ait suffisamment d'espace disponible pour accueillir le fichier. Si ce n'est pas le cas, une erreur est envoyée au client. La deuxième étape consiste à créer un enregistrement en base de données. Pour ce faire, son nom d'origine (tronqué à 50 caractères) est utilisé et la taille qu'il fait dans les fichiers temporaires sont utilisés. La dernière étape est de déplacer le fichier des fichiers temporaires vers l'espace de stockage de la communauté, en utilisant l'identifiant généré par la base de données et son extension d'origine.

Durant la phase d'enregistrement du fichier en base de données, le système va détecter le type du fichier à partir de son type MIME et à l'aide d'une expression régulière. Si le type du fichier ne peut être détecté, il prend alors le type « Fichier » par défaut.

La détection du type de fichier se base uniquement sur le type MIME du fichier envoyé. Cette information se trouve dans les « meta data » du fichier, et peut donc potentiellement être manipulée par l'utilisateur. Malgré de nombreuses recherches, je n'ai pas trouvé de moyen plus sûr de valider le type d'un fichier. Le seul cas de figure où une mauvaise détection du type poserait problème serait lors de l'exécution du dit fichier. Cette problématique sera abordée plus loin dans le rapport.

Lorsque l'utilisateur télécharge seulement un fichier à la fois, le contenu du fichier et lu et envoyé au navigateur avec les « headers HTTP » adéquats. Si plusieurs fichiers ont été sélectionnés, ou que l'utilisateur télécharge un répertoire, une archive ZIP est créée préalablement avant d'être envoyée de la même manière.

La création de l'archive se fait à l'aide de la librairie PHP « ZipArchive », ajoutée à la version 5.2 de PHP. La méthode « zipItems » de la classe « ToFileResourceController » permet de créer une archive, de lui donner un nom temporaire, d'ajouter à l'intérieur tout le contenu sélectionné et de retourner le chemin jusqu'à l'archive créée. Le webservice permettant le téléchargement va alors envoyer les données au navigateur exactement de la même manière que pour un fichier unique, à la différence près qu'après avoir fini d'envoyer les données, l'archive est supprimée.

Ce mécanisme est relativement simple, et fonctionne bien. Cependant la création de l'archive est une opération gourmande en ressource pour le serveur. Il aurait été intéressant d'étudier d'autres manières de télécharger des données et de les « benchmarker » pour sélectionner la plus efficace. Ceci n'a pas été fait durant le projet par manque de temps.

5.3.4 BASE DE DONNÉES

Aucune modification de la base de données décrite au chapitre précédent n'a été nécessaire pour implémenter ces fonctionnalités.

5.3.5 GESTION DE PROJET

Le développement de ces fonctionnalités a été effectué principalement sur le sprint 4 et représentait au total 33 story points réalisés en 50 heures de travail. La charge de travail a donc été surévaluée. Ceci s'explique par le fait que le développement contenait beaucoup de choses que je n'avais encore

jamais faites, et qui me semblaient particulièrement compliquées. J'ai pu trouver une quantité satisfaisante de documentation ce qui m'a permis d'avancer plus rapidement.

J'ai investi beaucoup de temps dans l'apprentissage du fonctionnement des XHR et de certaines nouvelles fonctionnalités d'HTML 5. J'ai également eu beaucoup de peine à afficher de manière stable les barres de progression. En effet, par la suite j'ai découvert de nombreuses anomalies qui m'ont pris plusieurs heures à résoudre et qui ne sont pas comptabilisées ici. En revanche, je n'ai subi aucune perturbation extérieure à mon travail. J'ai pu avancer de manière régulière.

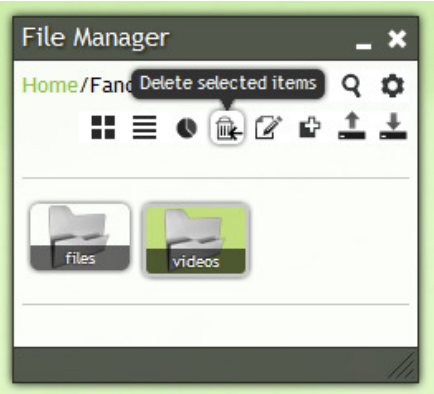
5.4 SUPPRESSION DES FICHIERS

Plusieurs personnes accèdent au même gestionnaire de fichiers au travers de leurs communautés communes. Plusieurs personnes peuvent donc potentiellement supprimer des ressources. Il était alors intéressant d'offrir un mécanisme de « poubelle » avant la suppression définitive des ressources. L'objectif étant de pouvoir restaurer ces ressources en cas de mauvaise manipulation.

Pour des informations plus détaillées sur le cahier des charges de ces fonctionnalités, il faut se référer aux user stories 10, 24, 25 et 60.

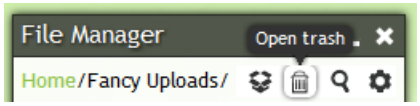
5.4.1 FONCTIONNALITÉS

La suppression de ressources se fait simplement en sélectionnant des ressources dans l'explorateur de fichiers et en appuyant sur le bouton « Delete selected items ». Si un répertoire est sélectionné, tout son contenu sera également supprimé.

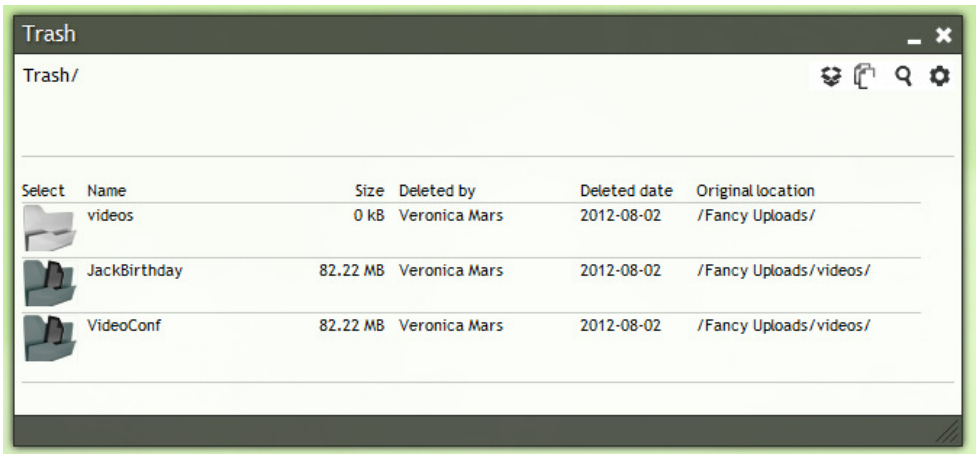


Copie d'écran 16 Suppression de ressources

Il est possible de consulter les fichiers supprimés depuis la poubelle. Celle-ci est accessible depuis le bouton « Open trash ».



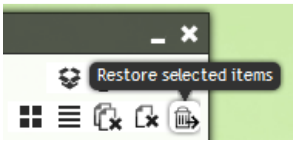
Copie d'écran 17 Ouverture de la poubelle



Copie d'écran 18 Contenu de la poubelle

En mode liste, il est possible de voir qui a supprimé les fichiers, à quelle date et où est-ce qu'ils se trouvaient à l'origine. L'affichage en mode icônes et la recherche sont également possibles.

L'utilisateur a la possibilité de supprimer définitivement les fichiers qu'il sélectionne ou l'intégralité du contenu de la poubelle. Il peut également restaurer à leur emplacement d'origine des fichiers et répertoires. Si l'emplacement d'origine a également été supprimé, les ressources sont alors placées à la racine du gestionnaire de fichiers.



Copie d'écran 19 Restauration de ressources

Les fichiers supprimés depuis plus d'une semaine sont automatiquement supprimés de la poubelle de façon définitive.

5.4.2 FONCTIONNEMENT DE L'INTERFACE GRAPHIQUE

L'interface graphique fonctionne de manière similaire à celle de l'explorateur de fichiers. La première différence est que les ressources peuvent uniquement être sélectionnées. Les interactions jQuery « Draggable » et « Droppable » sont désactivées dans le but d'empêcher des opérations qui n'auraient pas de sens, comme supprimer un fichier en le déplaçant dans un répertoire supprimé.

Il n'y a pas de navigation dans la poubelle. Toutes les ressources sont affichées à la racine, également dans le but d'empêcher des manipulations illogiques.

5.4.3 LOGIQUE APPLICATIVE

L'opération de suppression d'un fichier depuis l'explorateur de fichiers consiste à mettre à jour les valeurs « deleted », « deletedDate » et « deleterId » en base de données. Dans le cas des fichiers, il s'agit simplement d'une requête SQL. Dans celui des répertoires, la même requête est exécutée, après quoi on va récupérer le contenu du répertoire et appeler la méthode de suppression sur chaque élément. Il s'agit donc d'une méthode récursive.

Les fichiers restent physiquement dans l'espace de stockage de la communauté mais n'apparaissent plus dans l'explorateur de fichiers et ne sont plus comptabilisés dans l'espace de stockage utilisé.

Etant donné que la restauration de ressources est propre au gestionnaire de fichiers tokiwi, la méthode permettant de le faire n'a pas été ajoutée dans l'interface « IFileResource » mais dans la classe abstraite « ToFileResource ».

L'implémentation de la méthode se fait dans les classes concrètes « ToFile » et « ToFolder ». Dans le cas de « ToFile », la méthode va d'abord vérifier que son répertoire parent existe encore, si ce n'est pas le cas elle appelle la méthode qui va déplacer le fichier à la racine, puis elle va mettre à jour les valeurs « deleted », « deletedDate » et « deleterId » en base de données. Dans le cas de « ToFolder », les mêmes opérations sont effectuées, après quoi la méthode va récupérer tout son contenu et appeler la méthode « restore » sur chaque élément.

Lorsque j'ai implémenté la limitation de l'espace de stockage, et que j'ai ajouté une vérification de l'espace disponible dans les méthodes « restore » de « ToFile » et « ToFolder », je me suis rendu compte que les performances chutaient drastiquement. Ce fut le même constat pour la copie de fichiers.

La méthode permettant d'obtenir l'espace de stockage restant est une méthode récursive qui va additionner la taille de tous les fichiers et répertoires depuis la racine. Après quelques analyses, je me suis rendu compte qu'avec une centaine de fichiers, sur ma machine de développement, la méthode prenait entre 0.5 et 1 seconde pour s'exécuter. Sachant que la méthode « restore » sur un répertoire est également récursive, j'obtenais des performances carrément catastrophiques lors de la restauration de plus d'une dizaine de ressources à la fois.

J'ai donc implémenté une méthode dans le contrôleur des ressources qui permet de calculer en une fois l'espace nécessaire à la restauration des fichiers sélectionnés et l'espace restant. Celui-ci restaure autant de fichiers que possible et retourne une erreur indiquant les fichiers qui n'ont pu être restaurés par manque de place. De plus, j'ai modifié la méthode qui calculait l'espace de stockage restant de manière à ce qu'elle ne le fasse plus récursivement, mais à l'aide d'une requête SQL. Les performances ont été améliorées de manière exponentielle, ce qui a résolu le problème.

5.4.4 BASE DE DONNÉES

Les modifications en base de données consiste en l'ajout de 3 champs dans la table « fileResources » :

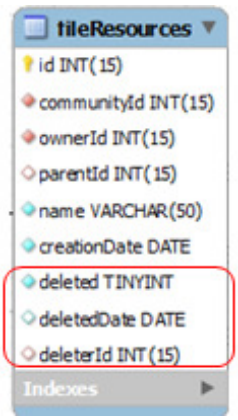


Figure 11 Champs de la table fileResources permettant la suppression

Ces valeurs permettent de savoir qui a supprimé quel fichier et à quelle date. En revanche, aucune modification en base de données n'a été effectuée pour garder une trace des restaurations. Ceci ne faisait pas partie du cahier des charges et me semblait apporter peu de valeur ajoutée, bien que ce soit une amélioration possible du système.

5.4.5 GESTION DE PROJET

Le développement de ces fonctionnalités a été effectué principalement sur le sprint 5 et représentait au total 6 story points réalisés en 15 heures de travail. La charge de travail a donc été correctement évaluée. Le temps passé à résoudre les problèmes de performances ne sont cependant pas comptabilisées dans cette user story.

5.5 EXÉCUTION DES FICHIERS

L'exécution des fichiers permet de les exploiter directement sur la plateforme, à partir d'autres applications installées dans tokiwi. C'est l'un des meilleurs atouts de la plateforme et du gestionnaire de fichiers par rapport à ses concurrents. Cette fonctionnalité n'était initialement pas prévue dans le cahier des charges, mais au vu de son excellent ratio « valeur ajoutée / story points », j'ai pris la liberté de l'ajouter.

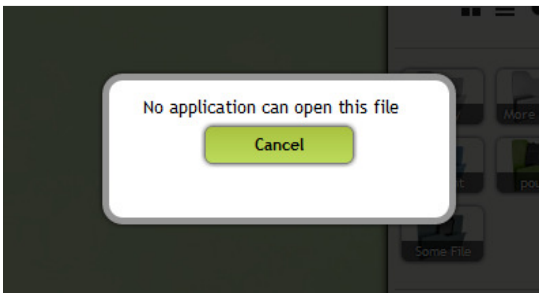
Pour des informations plus détaillées sur le cahier des charges de ces fonctionnalités, il faut se référer à la user story 59.

5.5.1 FONCTIONNALITÉS

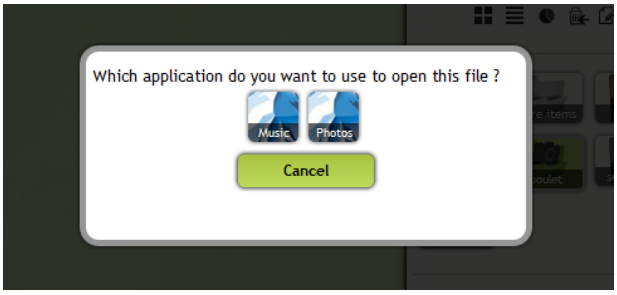
Pour exécuter un fichier, il suffit d'ouvrir l'explorateur de fichiers et de cliquer sur le nom du fichier.

Copie d'écran 20 Exécution d'un fichier

Après avoir cliqué sur le nom du fichier, une pop-up s'affiche indiquant qu'aucune application ne peut ouvrir ce fichier si c'est le cas. Si une seule application peut ouvrir le fichier, l'application s'ouvre avec le fichier. Si plusieurs applications peuvent exécuter le fichier, une pop-up s'affiche permettant à l'utilisateur de choisir avec laquelle il désire exécuter le fichier.



Copie d'écran 21 Pop-up indiquant qu'aucune application ne peut exécuter un fichier



Copie d'écran 22 Pop-up permettant à l'utilisateur de choisir l'application pour exécuter un fichier

A la suite de quoi l'application choisie s'ouvre et peut utiliser le fichier stocké dans le gestionnaire de fichiers.

Copie d'écran 23 Espace de travail avec deux applications exploitant des fichiers

5.5.2 FONCTIONNEMENT DE L'INTERFACE GRAPHIQUE

Le fonctionnement de l'interface graphique est des plus simples. Une fonction Javascript va interroger un webservice lorsque l'utilisateur clique sur le nom d'une ressource. Ce webservice lui retourne alors un JSON contenant l'identifiant, le nom, et le logo de chaque application capable d'exécuter le fichier sélectionné.

A partir de ce JSON, la fonction Javascript va soit directement exécuter le fichier avec l'application adéquate s'il y en a qu'une, soit afficher une pop-up permettant à l'utilisateur de choisir parmi une liste d'applications.

L'utilisateur n'a qu'à cliquer sur l'application de son choix, ce qui aura pour effet d'appeler la fonction Javascript exécutant une application avec l'identifiant du fichier en paramètre. L'identifiant du fichier est donc enregistré dans les paramètres de l'instance du module de l'application sous le nom de « frId ». Après quoi il est de la responsabilité de l'application d'exploiter ce paramètre.

Ce mécanisme est d'une simplicité enfantine, ce qui le rend particulièrement efficace. Une amélioration intéressante cependant pourrait être d'offrir la possibilité à l'utilisateur d'enregistrer quel type de fichier s'ouvre avec quelle application, ce qui lui éviterait l'étape de la pop-up avant d'accéder à son fichier.

5.5.3 LOGIQUE APPLICATIVE

Comme expliqué précédemment, chaque fichier obtient un type lors de son processus de création. Le type est une entité à la structure suivante :

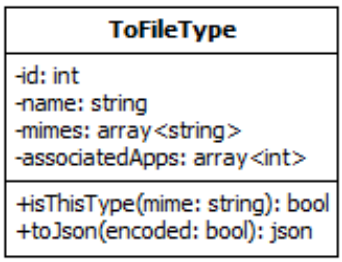


Figure 12 Diagramme de la classe ToFileType

Le type tokiwi est basé sur un ou plusieurs types MIME. Par exemple, le type « Image » correspond aux types MIME :

- image/gif
- image/jpeg
- image/pjpeg
- image/png
- image/x-png
- image/tiff
- image/vnd.microsoft.icon

Il est donc possible d'enregistrer dynamiquement de nouveaux types en base de données, en leur attribuant un identifiant, un nom et des types MIME. Ces types pourront ensuit être utilisé dans le gestionnaire de fichiers.

On constate également que l'entité possède l'attribut « associatedApps ». Il s'agit des applications capables d'exploiter ce type tokiwi de fichiers. Il est également possible d'associer dynamiquement des applications à des types tokiwi.

Ces deux opérations s'effectuent à l'installation de l'application. Aucune fonction de l'API du gestionnaire de fichiers ne permet de modifier un type tokiwi ou d'associer une nouvelle application. Ces fonctionnalités seraient développées en parallèle du SDK tokiwi.

Ainsi, par exemple, si Adobe décidait de développer une version de Photoshop dans tokiwi, au moment de publier leur application, ils pourraient effectuer les opérations suivantes :

- Associer le type tokiwi de fichier « Image » à l'application « Photoshop »
- Créer un type tokiwi de fichier « Photoshop file »
    - Valider ce nouveau type avec les types MIME suivant :
        - image/photoshop
        - image/x-photoshop
        - image/psd
        - image/vnd.adobe.photoshop
        - application/photoshop
        - application/psd
    - Associer l'application « Photoshop » au type tokiwi fraîchement créé

Après quoi, il serait possible d'ouvrir toutes les images et les fichiers « PSD » du gestionnaire de fichiers dans l'application « Photoshop ».

Cette fonctionnalité permet une intégration du gestionnaire de fichiers quasiment illimitée aux autres fonctionnalités de la plateforme, moyennant très peu d'effort. C'est, selon moi, l'un des meilleurs atouts de l'application.

5.5.4 BASE DE DONNÉES

Cette fonctionnalité repose sur 4 tables :

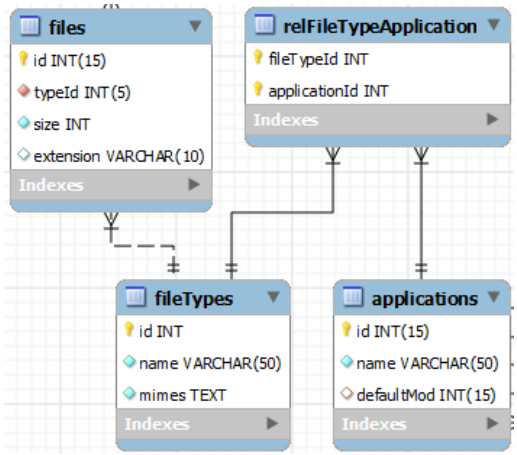
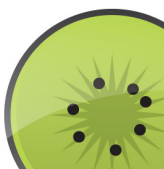


Figure 13 Modèle de la base de données permettant l'exécution de fichiers

La table « relFileTypeApplication » a été ajoutée au modèle de base afin de permettre des relations « n : n » entre les applications et les types de fichiers. Une amélioration possible de cette table serait d'ajouter une colonne « modId » afin de spécifier avec quel module de l'application le fichier doit s'ouvrir. Ceci nécessiterait également une adaptation de la logique applicative.

Nous noterons également qu'un fichier ne peut être que d'un seul type tokiwi. Une amélioration possible de cette structure serait d'autoriser les fichiers à en être de plusieurs. Un système où les fichiers aurait un type primaire, puis une multitude de types secondaires serait à la fois élégant et permettrait de débloquer plusieurs scénarios d'utilisation. En revanche cela nécessiterait beaucoup de travail pour adapter la logique applicative.

Les types MIME associés à un type tokiwi sont enregistrés dans une seule colonne, au format JSON. J'ai fait ce choix pour simplifier les choses. L'autre option aurait été de créer une table répertoriant tous les types MIME existants, puis de créer une deuxième table permettant de lier un ou plusieurs types MIME à un type tokiwi.



Cette structure aurait probablement été plus « propre », mais la complexité du modèle aurait augmenté. De plus, les performances aurait sensiblement diminué (il aurait fallu traverser deux tables supplémentaires pour accéder à la même information). Et surtout, il aurait été fastidieux de maintenir à jour une table répertoriant tous les types MIME existants.

Les deux seuls inconvénients du système mis en place ici sont la redondance des données et le potentiel risque de stocker des valeurs erronées. Compte tenu du nombre d'enregistrements que devrait atteindre la table « fileTypes » et de l'impact qu'auraient des valeurs erronées, je pense avoir fait le bon choix.

5.5.5 GESTION DE PROJET

Le développement de ces fonctionnalités a été effectué principalement sur le sprint 5 et représentait au total 6 story points réalisés en 8.5 heures de travail. La charge de travail a donc été surévaluée. J'avais anticipé des imprévus techniques qui ne se sont pas produits.

J'ai réalisé de deux applications pour tester le mécanisme. Je les ai développées avec des fonctionnalités plus que basiques, ce qui m'a fait gagner beaucoup de temps.

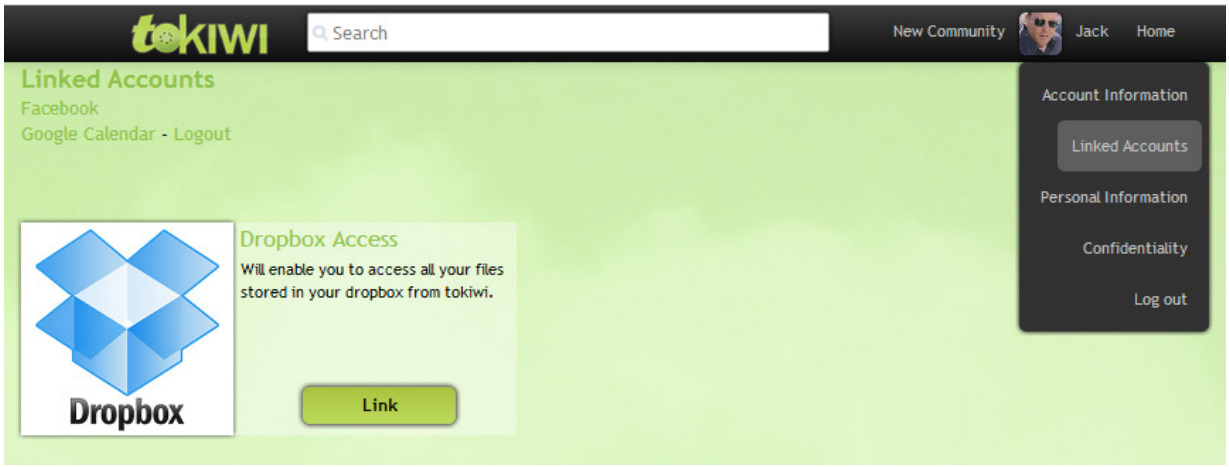
5.6 INTÉGRATION DE DROPBOX

L'un des USP de la plateforme est l'intégration de services existants dans ses propres services. L'intégration de Dropbox dans le gestionnaire de fichiers était donc l'une des fonctionnalités les plus importantes de l'application. L'objectif était d'offrir à l'utilisateur la possibilité d'accéder à tout le contenu de sa Dropbox depuis le gestionnaire de fichiers tokiwi, et de pouvoir récupérer les fichiers de son choix.

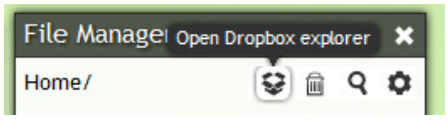
Pour des informations plus détaillées sur le cahier des charges de ces fonctionnalités, il faut se référer aux user stories 28-34.

5.6.1 FONCTIONNALITÉS

La première étape consiste à associer son compte Dropbox à son compte tokiwi. Pour ce faire, l'utilisateur a deux possibilités. Soit il va dans le menu « Linked Accounts » depuis le « Dashboard », soit il exécute le module « Dropbox » depuis l'explorateur de fichiers.



Copie d'écran 24 Page de liaison de comptes



Copie d'écran 25 Ouverture de Dropbox

Copie d'écran 26 Liaison du compte Dropbox dans l'application

En cliquant sur le bouton « Link », une page web s'affiche lui demandant de se connecter à Dropbox et d'accepter que tokiwi accède à ses fichiers.

Copie d'écran 27 Page web Dropbox permettant à l'utilisateur d'autoriser tokiwi

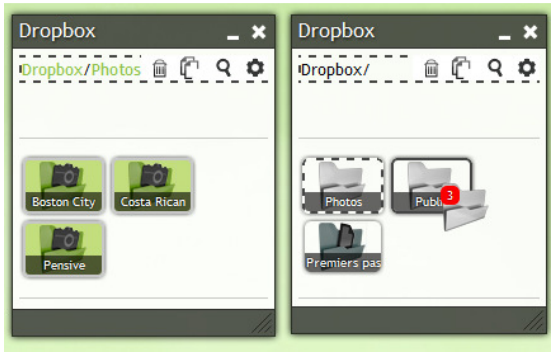
Après avoir cliqué sur le bouton « Autoriser », l'utilisateur sera redirigé sur la plateforme à la page des comptes liés. Ce qui lui permet de constater qu'il est possible à tout moment de dissocier son compte Dropbox de son compte tokiwi grâce au bouton « Unlink ».

Copie d'écran 28 Bouton permettant de supprimer un compte lié

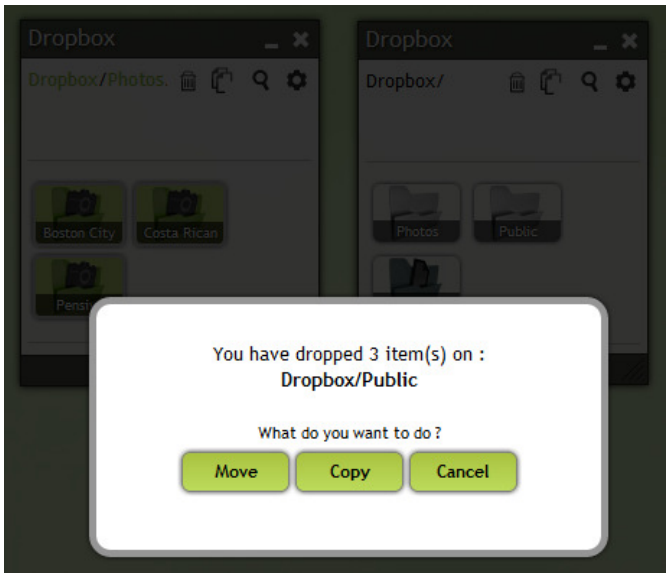
L'utilisateur peut désormais accéder à tous les fichiers contenus dans sa Dropbox. Les ressources qui y sont stockées sont présentées de la même manière. Les mêmes options d'affichage sont disponibles que dans l'explorateur de fichiers tokiwi. De plus, l'utilisateur peut rechercher des fichiers dans sa Dropbox, supprimer des ressources et créer des répertoires, le tout depuis tokiwi.

Copie d'écran 29 Explorateur de fichiers Dropbox

Toujours dans sa Dropbox, l'utilisateur peut également déplacer ou copier des ressources d'un répertoire à un autre.

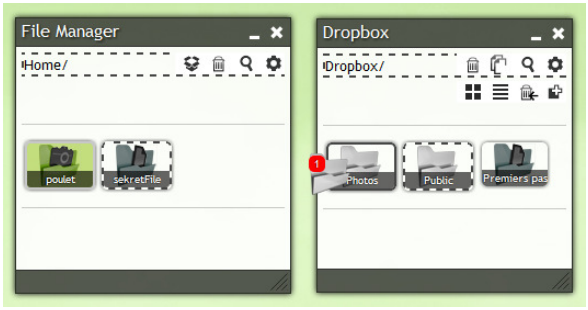


Copie d'écran 30 Illustration de la copie et du déplacement de ressources Dropbox

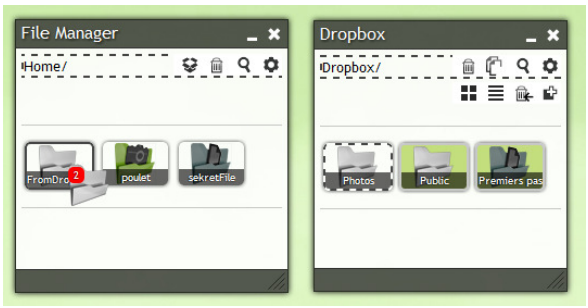


Copie d'écran 31 Pop-up permettant à l'utilisateur de déplacer ou copier une ressource Dropbox

Il peut également prendre des ressources de son gestionnaire de fichier tokiwi et les copier dans sa Dropbox, et inversement.



Copie d'écran 32 Copie de fichiers depuis le gestionnaire de fichiers tokiwi vers celui de Dropbox



Copie d'écran 33 Copie de fichiers depuis le gestionnaire de fichiers Dropbox vers celui de tokiwi

Lorsqu'un répertoire est copié, dans un sens ou dans l'autre, tout son contenu est également copié en conservant la structure du gestionnaire de fichier source.

5.6.2 FONCTIONNEMENT DE L'INTERFACE GRAPHIQUE

La page de liaison de comptes dans le « dashboard » de la plateforme était un vaste chantier utilisé pour tester la liaison de compte avec Facebook. J'ai donc investi un peu de temps pour soigner l'affichage des comptes associables à tokiwi.

Chaque compte est affiché dans une vignette organisée de la manière suivante :

Figure 14 Conception des vignettes de liaison de compte

La vignette est générée à partir d'une fonction Javascript qui récupère toutes les informations d'un compte grâce à un webservice. Il est donc possible d'utiliser cette fonction dans une application du « workspace ». C'est ce qui a été fait dans le module « Dropbox » du gestionnaire de fichiers. Le module vérifie si l'utilisateur a associé son compte Dropbox, et si ce n'est pas le cas cette fonction Javascript est appelée.

Si le compte associable n'a pas enregistré de logo, le logo suivant est affiché par défaut :

Figure 15 Logo par défaut des comptes liés

La page de liaison de comptes est prévue pour afficher trois vignettes par ligne, et autant de lignes que nécessaire. Aucun mécanisme de pagination ou de recherche n'a été mis en place pour le moment étant donné que Dropbox est actuellement le seul compte associable.

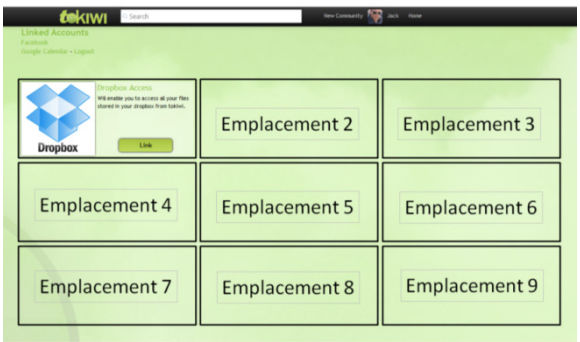


Figure 16 Affichage des vignettes de liaison de compte

L'interface graphique de l'application fonctionne exactement de la même manière que celle de l'explorateur de fichiers. Cependant il ne s'agit pas du même formulaire. En effet, la source de données n'est pas la même, et le format des données retournées par celle-ci est incompatible avec le formulaire de l'explorateur de fichiers.

De plus les fonctions Javascript permettant d'exécuter les actions de l'interface utilisateur ne sont pas les mêmes. Le même mécanisme est utilisé pour la navigation, la recherche et les autres actions, mais les webservices appelés sont différents.

Il est évident qu'avec autant de similitudes avec l'explorateur de fichiers, il serait intéressant de faire en sorte que les formulaires d'affichage des ressources soient les mêmes. Ceci pourrait relativement facilement être mis en place coté interface utilisateur, mais requerrait un travail conséquent coté logique applicative, comme abordé dans le chapitre suivant.

5.6.3 LOGIQUE APPLICATIVE

En cherchant sur internet, j'ai trouvé plusieurs librairies PHP permettant d'utiliser l'API de Dropbox. Bien que plusieurs d'entre elles répondaient aux besoins du cahier des charges, j'ai décidé d'implémenter ma propre couche d'abstraction.

En effet, comme la plupart des API, celle de Dropbox utilise une authentification OAuth. Sachant qu'à l'avenir d'autres services seraient intégrés à tokiwi, il m'a semblé pertinent d'implémenter une couche d'abstraction suffisamment large pour fonctionner avec d'autres services, et pas uniquement Dropbox. De plus, je n'avais aucune connaissance de la norme OAuth, et j'étais particulièrement intéressé de comprendre plus en profondeur comment l'implémenter.

En me renseignant sur le site officiel de PHP, j'ai découvert l'extension PECL OAuth. En général je n'aime pas utiliser des extensions qui ne font pas partie de la librairie PHP par défaut. Mais toujours selon le site officiel de PHP, cette extension serait le standard OAuth pour la version 5.4 de PHP. J'ai donc pris la décision d'installer l'extension et de me servir de cette librairie comme base de travail. L'autre option aurait été d'utiliser la librairie CURL, mais cela aurait engendré de nombreux développements supplémentaires.

L'utilisation d'OAuth pour interroger une API se déroule en quatre étapes. La première consiste à obtenir une clé permettant d'identifier le fournisseur et le consommateur du service. La deuxième consiste à associer un utilisateur du fournisseur à cette même clé, en demandant à l'utilisateur de valider l'opération chez le fournisseur. Dès lors, dans une troisième étape le consommateur peut

réclamer une nouvelle clé qui cette fois-ci identifie le fournisseur, le consommateur et l'utilisateur du fournisseur (afin que de son côté, le consommateur puisse associer cette clé à son propre utilisateur correspondant). Puis dans un quatrième temps, le consommateur peut interroger l'API du fournisseur en lui spécifiant toujours cette clé d'identification. Il est donc vital que le consommateur conserve précieusement cette clé. Le schéma ci-dessous permet de mieux visualiser le processus :

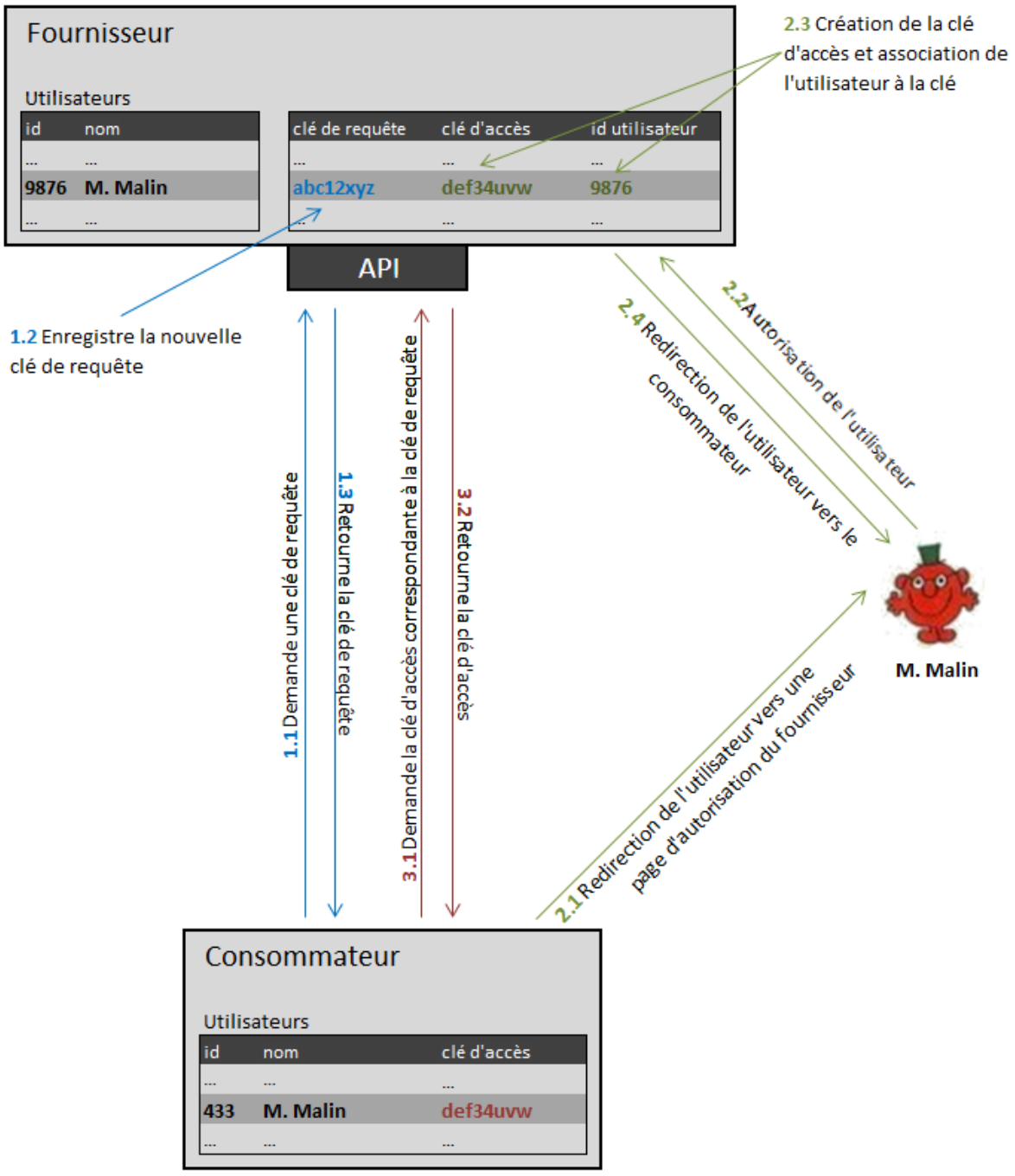


Figure 17 Processus d'obtention d'une clé d'accès OAuth

Dans l'optique de simplifier ce processus pour les prochaines intégrations de services de la plateforme, j'ai créé deux nouvelles entités. La première, « LinkableAccount », représente un compte associable, il s'agit d'une classe abstraite. La seconde, « LinkedAccount », représente un compte lié entre un utilisateur tokiwi et un utilisateur d'un fournisseur de service :

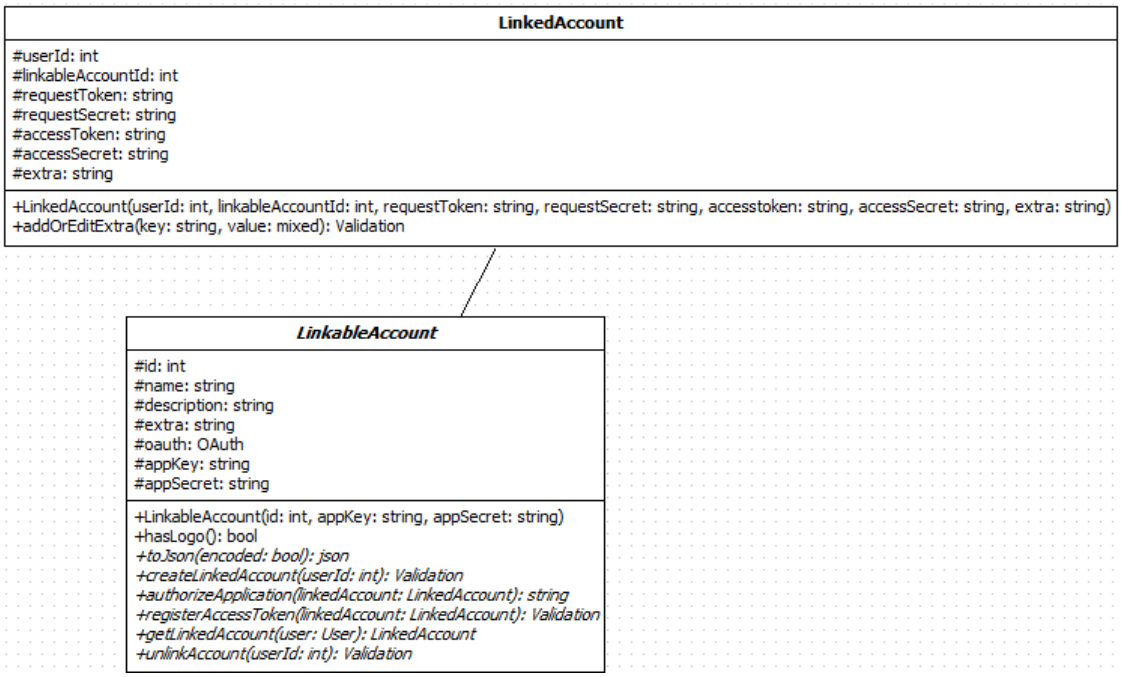


Figure 18 Diagramme de classe du système de liaison de comptes

L'idée est que pour chaque compte joignable, la classe « LinkableAccount » soit étendue par une classe implémentant ses méthodes abstraites. Les méthodes abstraites permettent de :

1. Fabriquer une nouvelle instance de l'entité « LinkedAccount » correspondant au compte qui sera joint, et récupérer la clé de requête chez le fournisseur.
2. Rediriger l'utilisateur pour autoriser tokiwi à accéder à son compte
3. Enregistrer la clé d'accès après l'autorisation
4. Désassocier le compte

Puis, dans une deuxième classe qui étend « LinkedAccount », implémenter des méthodes permettant d'accéder aux fonctions de l'API du fournisseur.

Ce système peut paraitre compliqué au premier abord, mais il facilite, accélère et standardise l'intégration de services dans la plateforme. De plus il fournit un moyen propre de stocker les clés d'accès aux comptes associés, ce qui n'était pas le cas auparavant. Il permet également d'automatiser l'affichage de comptes liés dans la partie « frontend ».

Dans un avenir plus hypothétique, ce système pourrait rendre envisageable le développement de comptes associables dans le futur SDK de la plateforme. Ainsi, si LinkedIn, par exemple, développait une application pour tokiwi, ils pourraient également développer le compte associable qui permet aux utilisateurs d'accéder aux informations de leur compte LinkedIn. Ce même compte associable développé par LinkedIn pourrait ensuite être utilisé par d'autres applications. L'avantage est bien entendu l'accélération du développement de nouvelles applications, mais surtout une démultiplication des fonctionnalités potentielles des dites applications.

Après avoir standardisé l'intégration de comptes dans la plateforme, il m'a fallu implémenter l'intégration de Dropbox. J'ai suivi le processus expliqué ci-dessus, et créé une entité implémentant le processus de liaison de compte (« DropboxAccessAccount ») et une représentant un compte Dropbox associé (« DropboxAccessLinkedAccount »). Cette dernière implémente différentes méthodes permettant d'interroger l'API de Dropbox :

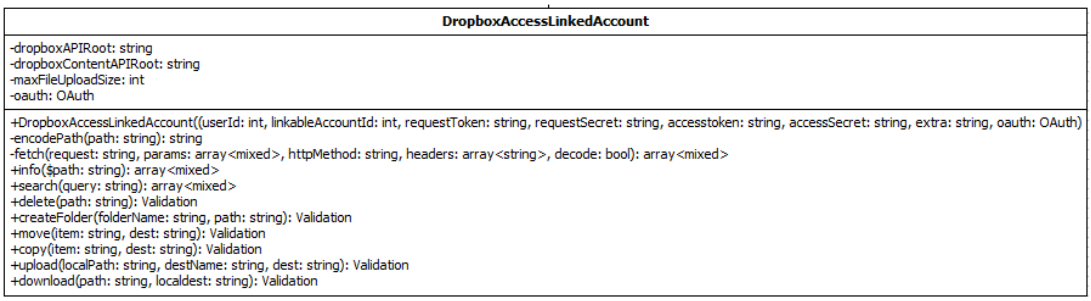


Figure 19 Diagramme de la classe DropboxAccessLinkedAccount

En instanciant cette entité, n'importe quelle application tokiwi peut donc accéder au compte Dropbox d'un utilisateur ayant associé son compte, et profiter des méthodes de l'entité accédant à l'API Dropbox. Par exemple, il serait très rapidement possible de développer un bouton « Sauver cet album dans ma Dropbox » dans une application d'album photos.

La dernière étape de l'intégration de Dropbox dans le gestionnaire de fichiers a été de créer un contrôleur qui manipule le compte associé et permet de retourner des données exploitables par l'interface graphique :

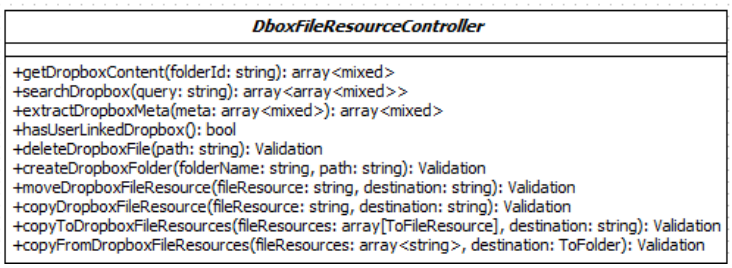


Figure 20 Diagramme de la classe DboxFileResourceController

Je suis très peu satisfait de cette manière de faire. Il aurait été préférable de créer de nouvelles entités implémentant les interfaces « IFileResource », « IFile » et « IFolder », et de garder la même logique que celle qui avait été établie avec les entités « ToFileResource », « ToFile » et « ToFolder » du gestionnaire de fichiers tokiwi. Ceci aurait grandement facilité le développement et la maintenance de l'interface graphique.

Cela n'a pas été fait par manque de temps. C'est clairement une amélioration nécessaire de cette fonctionnalité. Néanmoins, l'intégration de Dropbox fonctionne telle quelle, et l'objectif fixé dans le cahier des charges est donc atteint.

5.6.4 BASE DE DONNÉES

Pour mettre en place la standardisation de comptes associés, il a fallu créer deux nouvelles tables. « linkableAccounts » répertorie tous les comptes associables et « relUserLinkableAccount » enregistre toutes les associations de comptes des utilisateurs :

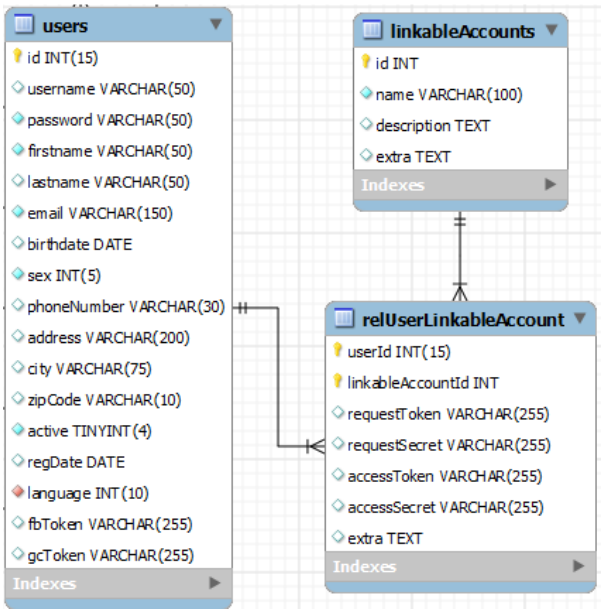


Figure 21 Modèle de la base de données du système de liaison de comptes

Grâce à ce modèle, la création de nouveaux comptes associables ne requiert plus de modification de la structure de la base de données. Elle engendre simplement un enregistrement supplémentaire dans la table « linkableAccounts ».

Les colonnes « extra » des deux tables permettent de stocker des informations au format JSON au cas où des données supplémentaires seraient nécessaires ou utiles à l'utilisation d'un compte associable.

Une amélioration possible de ce modèle serait de supprimer les colonnes « requestToken » et « requestSecret », qui ne sont utiles que durant la phase d'obtention de la clé d'accès. Elles ont été créées car j'avais mal compris le fonctionnement d'une authentification OAuth lors ma première implémentation du système.

5.6.5 GESTION DE PROJET

Le développement de ces fonctionnalités a été effectué principalement sur le sprint 5 et représentait au total 30 story points réalisés en 48.5 heures de travail. La charge de travail a donc été surévaluée.

Etant donné qu'il s'agissait de fonctionnalités complètement nouvelles pour moi, j'ai donc été (excessivement) prudent sur les prévisions du temps nécessaires à leur implémentation.

De manière générale, j'ai perdu beaucoup de temps à chercher de la documentation sur le fonctionnement de OAuth avec PHP. La documentation est très limitée. J'ai également perdu du temps pour trouver un moyen de tester localement les fonctionnalités. En effet, le serveur web de ma machine de développement n'était pas configuré pour gérer des requêtes HTTPS, j'ai donc du trouver un moyen de remédier à ce problème.

5.7 CONCLUSION DES FONCTIONNALITÉS IMPLÉMENTÉES

D'un point de vue client, je pense avoir été au de-là des exigences fixées dans le cahier des charges. En effet, toutes les fonctionnalités ont été implémentées et sont fonctionnelles. A ça, j'ai en plus ajouté le système d'exécution des fichiers qui n'était initialement pas prévu. De plus, en intégrant Dropbox à l'application, j'ai également créé un système de liaison de compte quasiment automatique. Ce fut un investissement en temps qui n'aurait pas nécessairement été utile à l'application en soit, mais qui sera largement payant lors de futures intégrations de services.

N'excellant pas dans les arts graphiques, je suis néanmoins satisfait du résultat obtenu avec l'interface graphique dans son ensemble. Il reste encore probablement beaucoup d'améliorations possibles en termes d'ergonomie, mais j'ai le sentiment que l'application est acceptablement intuitive en l'état et les retours utilisateurs obtenus jusqu'ici confirment cette impression.

L'architecture de l'application est solide et évolutive. C'est à mon avis l'un de ses meilleurs atouts. En effet, chaque composant peut être modifié indépendamment des autres et toutes les meilleures pratiques tokiwi ont été rigoureusement appliquées. De plus la documentation des différentes API a été méticuleusement faite, ce qui rend son utilisation d'autant plus simple.

Je me permets donc de conclure ce chapitre en affirmant qu'il s'agit d'un travail de qualité et abouti. J'émettrais peut être juste quelques réserves quant à son utilisation dans un environnement de production. En effet, aucun mécanisme industriel de tests unitaires, de performances et de montée en charge n'ont été mis en place. Par conséquent il serait prétentieux d'affirmer que l'application n'a aucun dysfonctionnement. Ce sujet n'a pas été abordé dans ce rapport car cette problématique est valable pour la plateforme dans son ensemble et nécessiterait la mise en place d'un système incluant également les applications dans le processus de qualité.

6 SCÉNARIOS D'UTILISATION

Nous avons vu dans le chapitre précédent quelles étaient les fonctionnalités de l'application et comment elles ont été implémentées. Nous allons maintenant les tester au travers d'un scénario étape par étape, et ainsi découvrir concrètement comment en tirer profit.

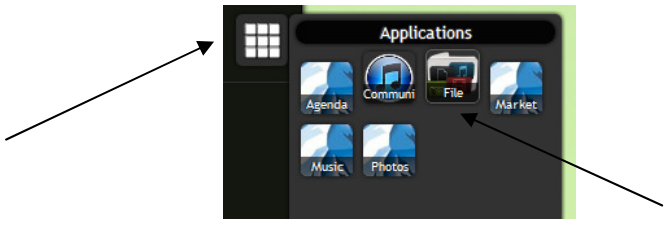
Puis dans un deuxième temps, nous verrons un scénario d'utilisation d'un utilisateur fictif. Celui-ci décrit un problème qu'il a résolu grâce à une application tokiwi utilisant le gestionnaire de fichiers mais n'existant pas encore. D'autres scénarios de ce genre se trouvent dans l'annexe « 13.4 Scénarios d'utilisation pour d'autres applications ».

6.1 LE TOURNOI ANNUEL DU VBC JORAT-MÉZIÈRES

Vous êtes Jack Bauer, fier membre du club de volley de Mézières. Le club organise un tournoi chaque année et vous faites partie du comité d'organisation. Pour cette édition, vos tâches étaient de préparer la liste des prix, le planning des tournus et le flyer. De lourdes responsabilités que vous avez assumées avec brio. Bravo.

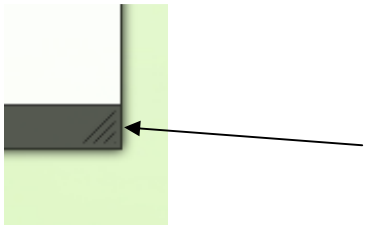
Maintenant vous allez mettre ces documents à disposition des autres membres du comité :

1. Vous ouvrez votre navigateur préféré, c'est-à-dire Google Chrome ou Firefox.
2. Vous vous rendez à l'adresse suivante : www.tokiwi.com/dev/envs/mezieres.
3. Vous ne vous souvenez plus de votre mot de passe, mais puisque vous n'êtes pas quelqu'un de très à cheval sur la sécurité, vous vous rappelez l'avoir enregistré dans le fichier « login.txt » qui se trouve dans le répertoire « VBC Jorat-Mézières » du CD fourni avec ce document.
4. Vous vous connectez puis accéder à la communauté « VBC Jorat-Mézières » en cliquant dessus.
5. Vous voulez maintenant accéder au gestionnaire de fichiers de la communauté. Vous cliquez donc sur l'explorateur d'applications (gros bouton blanc à gauche de l'interface) et lancez l'application « File ».



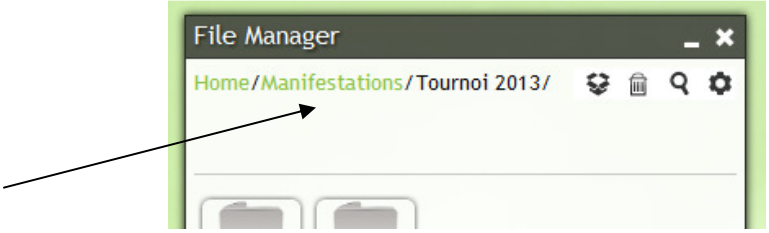
6. L'explorateur de fichiers s'ouvre, et vous vous apercevez que quelqu'un a ajouté une photo, « L'équipe ». Curieux, vous cliquez sur son nom.

7. Vous décidez de l'ouvrir avec l'application « Photos ».
8. Vous agrandissez un peu la fenêtre qui s'ouvre avec le bouton en bas à droite et vous admirez les joueurs du club quelques secondes avant de fermer la fenêtre.

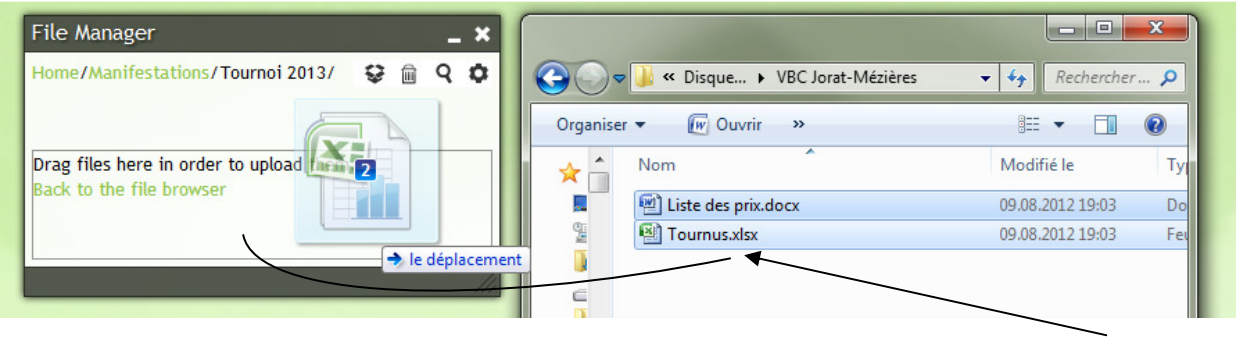


9. Vous voulez maintenant envoyer la liste des prix et le planning des tournus dans le répertoire « tournoi 2013 », mais vous ne vous rappelez plus où il se trouve. Vous vous inquiétez un bref instant pour votre mémoire puis vous cliquez sur la loupe pour faire une recherche.
10. Vous tapez les lettres « tour » dans la barre qui s'est affichée, puis vous cliquez sur le nom du répertoire « Tournoi 2013 ».

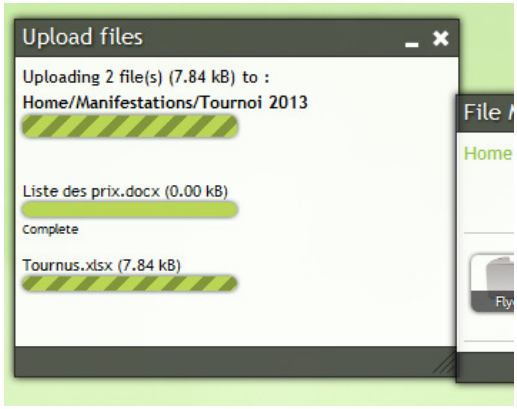
11. Il s'affiche et contre toute attente, vous vous apercevez qu'il se trouve dans le répertoire « Manifestations ».



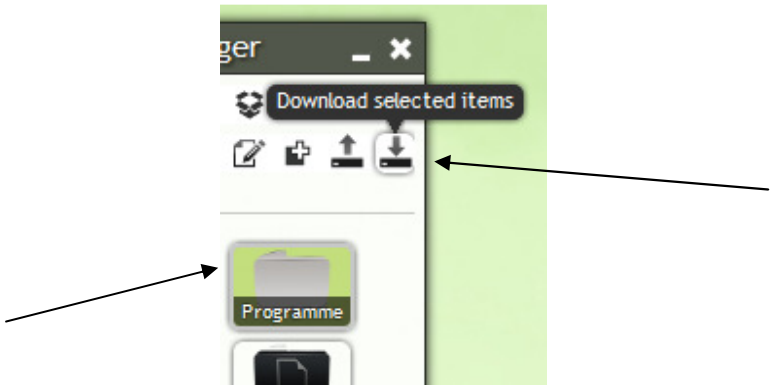
12. Vous ouvrez maintenant votre explorateur de fichiers Windows, et naviguez jusqu'au répertoire « VBC Jorat-Mézières » du CD fourni avec ce document. Vous sélectionnez les fichiers « Liste des prix.docx » et « Tournus.xlsx » puis vous les glissez dans le gestionnaire de fichiers tokiwi.



13. Un nouveau module s'ouvre, affichant l'état du téléchargement. Vous attendez la fin, puis fermez le module. Vos deux fichiers apparaissent dans l'explorateur de fichiers tokiwi.

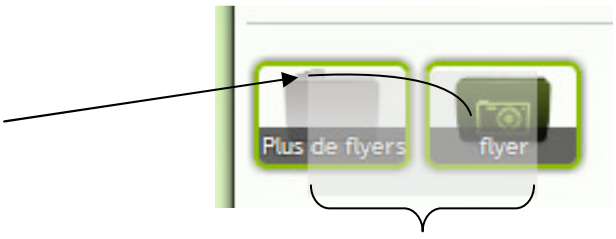


14. Vous constatez maintenant que quelqu'un a créé le répertoire « Programme des matchs ». Vous le sélectionnez et cliquez sur le bouton de téléchargement.

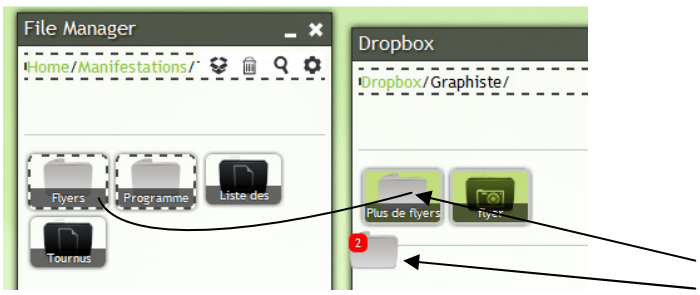


15. Il s'est correctement téléchargé, et pendant que vous jetez un œil au programme, le graphiste que vous avez engagé pour les flyers vous appelle. Il les a fini et les a mis à disposition dans le répertoire Dropbox que vous avez en commun.
16. Vous lancez donc l'application Dropbox et débutez la procédure d'association de compte en cliquant sur le bouton « Link »

17. Une page Dropbox s'affiche et vous demande de vous connecter. Bien sûr vous ne vous rappelez plus de votre mot de passe, et aller consulter votre fameux fichier « login.txt ».
18. Après vous être connecté, vous autorisez l'application « tokiwi Access » à accéder à votre compte.
19. L'association de compte est terminée, vous cliquez sur le bouton « Home » et retournez dans votre communauté préférée (VBC Jorat-Mézières donc).
20. Le module Dropbox affiche maintenant la racine de votre Dropbox, vous allez dans le répertoire « Graphiste » en cliquant sur son nom.
21. Vous sélectionnez le répertoire « Plus de flyers » et le fichier « flyer.png » en laissant appuyant le bouton gauche de votre souris sur « Plus de flyers » et en la déplaçant sur « flyer ». Un lasso gris apparait, et le fond des icônes devient vert au passage de la souris.



22. Vous les glissez ensuite dans le répertoire « Flyers » du gestionnaire de fichiers tokiwi.



23. Vous choisissez l'option « copy » et attendez un bref instant. A la fin de l'opération vous allez dans le répertoire « Flyers » en cliquant sur son nom et vérifiez qu'ils ont bien été copiés.

Après avoir autant travaillé, vous décidez de faire une pause. Vous fermez votre navigateur et aller boire un café.

6.2 ALBUM PHOTOS

Le scénario suivant est un témoignage fictif d'un utilisateur tokiwi se servant de l'application « toPic ». Cette application utilise les fonctionnalités de File Manager. La personne nous décrit le problème qu'elle résout, et comment elle y parvient.

« Hello, je m'appelle Annie et j'ai 29 ans. J'ai une grande passion dans la vie, c'est les voyages. Mes amies, Véronique, Sandra et Rachel partagent cette passion. Du coup, nous partons chaque année visiter une nouvelle ville ensemble.

En vacances, nous sommes de vraies paparazzis, nous avons toutes un super appareil photo et nous nous en servons pour photographier des monuments, nos chambres d'hôtel, nos repas, les beaux mecs dans la rue…

Quand nous rentrons de vacances, nous avons tout le temps le même problème ; le partage des photos. Sandra et Véronique les mettent sur Facebook, faut pas demander à Rachel d'allumer un ordinateur, elle y comprend rien, quant à moi j'aime pas du tout mettre mes photos sur des sites internet où je ne maitrise pas qui en a l'accès.

Depuis peu, nous avons créé une communauté tokiwi. Ceci nous a bien facilité l'organisation de nos diverses activités. Nous avons résolu le problème d'échange de photos grâce au gestionnaire de fichiers de la communauté. Même Rachel arrive à y envoyer ses photos, c'est dire ! (Bon… son copain doit toujours lui montrer comment on transfert les photos de l'appareil à l'ordinateur, mais la suite elle gère.)

En bidouillant un peu, Sandra nous a installé l'application « toPic », pour créer des albums photo dans la communauté. C'est magique ce truc ! Il suffit de créer un album, d'ouvrir le gestionnaire de fichiers, et de glisser des photos d'une fenêtre à l'autre. On peut ensuite modifier l'ordre dans lequel elles apparaissent, dire où elles ont été prises, qui est sur la photo, … Nous avons créé un album par ville visitée, et j'adore passer des soirées entière à regarder et commenter nos photos ! »

7 CONCLUSION

L'objectif de ce travail était de réaliser une application tokiwi permettant de gérer les fichiers d'une communauté et qui soit capable d'interagir avec la solution d'hébergement de fichiers en ligne la plus pertinente.

Le choix de la solution à intégrer s'est fait à la suite d'un état de l'art. Dropbox a été retenue, principalement pour sa popularité et la qualité de son API. Lors de la phase de développement, toutes les fonctionnalités prévues dans le cahier des charges ont été implémentées dans le temps imparti.

L'architecture de l'application est robuste et évolutive. Elle s'intègre parfaitement dans l'environnement tokiwi et d'autres services d'hébergement de fichiers en ligne pourraient facilement être intégrés. L'interface graphique est quant à elle attrayante mais pourrait encore être améliorée au niveau de l'ergonomie.

Selon Michel Cachin, l'une des neuf personnes qui ont testé l'application et dont les feedbacks sont disponibles à l'annexe « 13.3 Retours utilisateurs », l'application répond aux besoins des utilisateurs de tokiwi : « *[…] Le concept est très intéressant parce qu'il permet de partager facilement des informations ou médias dans un cercle restreint et maitrisé. Cela permet, par exemple, d'éviter d'utiliser des outils peu fiables quant à la protection des données, ici tokiwi offre un réel sentiment de sécurité. […]* »

Mais File Manager n'est que la première brique d'un grand édifice qui se construit petit à petit. Elle débloque de nombreuses problématiques qu'éprouvent les communautés, et donc d'autres applications tokiwi vont venir se greffer dessus pour étoffer l'offre de base.

Pour conclure, je suis personnellement content de la manière dont ce travail s'est déroulé. J'ai bénéficié d'une très grande liberté, que ce soit dans la façon de procéder ou dans les choix techniques et technologiques opérés. J'ai eu l'occasion de mettre en pratique différents concepts vus en cours durant mes trois ans de formation, et d'en expérimenter de nouveaux. Ce fut très enrichissant.

8 REMERCIEMENTS

Je tiens à remercier chaleureusement les personnes suivantes pour l'aide et le support qu'elles m'ont apportées durant ce travail.

Pour m'avoir encadré tout au long du travail et répondu à mes diverses questions :

- **M. Florian Evéquoz**

Pour m'avoir permis de faire le travail sur ce sujet :

- **M. Blaise Crettol**
- **M. Antoine Perruchoud**
- **M. Jonas Vonlanthen**
- **M. Florian Evéquoz**

Pour avoir relu le document et testé l'application :

- **Mme. Verena Galli**
- **Mme. Anne Galli**
- **Mme. Estelle Pagani**
- **Mme. Jacqueline Bovy**
- **Mme. Tamara Monney**
- **Mme. Noémie Allemann**
- **Mme. Carina Yerly**
- **M. Patrick Bovy**
- **M. Michel Cachin**

9 AUTHENTICITÉ DU TRAVAIL

Je déclare, par ces quelques lignes, que j'ai effectué l'intégralité de ce travail de Bachelor seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées.

Fabien Galli

10 GLOSSAIRE

Agile : Méthodologie de gestion de projet apparue dans les années 90 dans le but de corriger les problèmes liés à la gestion de projet en cascade.

AJAX (Asynchronous Javascript XML) : Terme couramment utilisé pour désigner un fonctionnement asynchrone d'une page web.

API (Application Programming Interface) : Groupe de fonctions mises à disposition par un éditeur tiers.

CURL (Client URL Request Library) : Ensemble de fonctions permettant d'accéder au contenu d'une ressource accessible via une URL.

CSS (Cascading Style Sheet) : Littéralement « feuille de style en cascade ». Instructions permettant d'indiquer au navigateur comment doivent s'afficher les différents éléments HTML d'une page web.

DAO (Data Access Object) : Interface permettant d'interroger une base de données.

Drag'n'Drop : Littéralement « Glisser et Déposer ». Action effectuée par l'utilisateur permettant de déposer un objet de l'interface graphique sur un autre.

HTML 5 (Hyper Text Markup Language 5) : Langage de programmation permettant de mettre en forme du contenu graphique dans un navigateur. Le 5 signifie qu'il s'agit de la 5ème version, introduisant une multitude de nouvelles fonctionnalités.

Javascript : Langage de programmation interprété s'exécutant dans le navigateur du client.

jQuery : librairie Javascript riche en fonctionnalités, notamment graphique, et open source.

JSON (Javascript Object Notation) : Format spécifique de stockage de données. Par rapport à un format alternatif comme XML, il présente les avantages d'être directement exploitable en Javascript et d'être plus léger.

Type MIME (Multipurpose Internet Mail Extensions) : permet d'identifier un format de données.

MySQL : Système de gestion de base de données gratuit appartenant à la société Oracle.

OAuth : Norme définissant comment doivent communiquer un fournisseur de services web avec ses consommateurs au travers d'une API.

Extension PECL (PHP Extension Community Library) : Extension issue de la communauté de développeurs PHP. Ces extensions ne font pas partie de la l'API standard de PHP.

PHP (Hypertext Preprocessor) : Langage de programmation interprété s'exécutant coté serveur.

REGEX (Regular Expression) : Littéralement « Expression Régulière ». Masque codé permettant de valider ou modifier une chaine de caractères.

Sprint : Itération d'un projet géré selon une méthodologie Agile. Dans le contexte de ce travail, correspond à une période de 2 semaines.

SQL (Structured Query Language) : langage permettant d'interroger une base de données.

Story Point : Manière d'évaluer la charge de travail d'une user story. Selon les projets, la valeur en heure d'un story point peut varier. Dans le contexte de ce travail, correspond à 2 à 3 heures de travail.

User Story : Littéralement « Histoire d'utilisateur ». Il s'agit d'une description haut niveau d'une fonctionnalité d'un système utilisant les termes du futur utilisateur.

Webservice : Fonction accessible au travers d'une requête HTTP permettant de déclencher une action ou retourner des données.

XHR (XmlHttpRequest) : Objet Javascript permettant d'effectuer une requête asynchrone à un serveur.

11 TABLE DES ILLSUTRATIONS

11.1 TABLEAUX

11.2 FIGURES

11.3 COPIES D'ÉCRAN

12 SOURCES

[1] Wikipédia, « File manager »

http://en.wikipedia.org/wiki/File_manager

Consulté le 28.05.2012, utilisé à la page : 13

[2] Wikipédia, « Comparison of file managers »

http://en.wikipedia.org/wiki/Comparison_of_file_managers

Consulté le 28.05.2012, utilisé à la page : 13

[3] Wikipédia, « File hosting service »

http://en.wikipedia.org/wiki/File_hosting_service

Consulté le 28.05.2012, utilisé à la page : 13

[4] Wikipédia, « Comparison of file hosting services »

http://en.wikipedia.org/wiki/Comparison_of_file_hosting_services

Consulté le 28.05.2012, utilisé aux pages : 13, 14, 16, 20

[5] Google, « Storage plan pricing »

http://support.google.com/drive/bin/answer.py?hl=en&answer=2375123

Consulté le 31.05.2012, utilisé à la page : 15

[6] iCloud, « Storage Upgrade Options »

http://www.apple.com/icloud/includes/lightbox-storage.html

Consulté le 31.05.2012, utilisé à la page : 15

[7] Microsoft, « Compare »

http://windows.microsoft.com/en-us/skydrive/compare

Consulté le 31.05.2012, utilisé à la page : 15

[8] Dropbox, « Pricing »,

https://www.dropbox.com/pricing

Consulté le 31.05.2012, utilisé à la page : 15

[9] 4shared, « Obtenez un compte Premium 4shared »

http://www.4shared.com/premium.jsp

Consulté le 31.05.2012, utilisé à la page : 15

[10] MediaFire, « Professional »
http://www.mediafire.com/tour/professional/
Consulté le 31.05.2012, utilisé à la page : 15

[11] PutLocker, « Upgrade Your Account Today »
http://www.putlocker.com/gopro.php
Consulté le 31.05.2012, utilisé à la page : 15

[12] Dropbox, « A propos de Dropbox »
https://www.dropbox.com/about
Consulté le 31.05.2012, utilisé à la page : 17

[13] Guillaume Belfiore, « Apple aurait tenté de racheter Dropbox », publié le 15 septembre 2011, Clubic
http://www.clubic.com/univers-mac/actualite-446826-apple-racheter-dropbox-idisk-stockage.html
Consulté le 11.06.2012, utilisé à la page : 17

[14] Answers, « How many Gmail accounts in the World? »
http://wiki.answers.com/Q/How_many_Gmail_accounts_in_the_world
Consulté le 11.06.2012, utilisé à la page : 17

[15] Mathieu Gerard, publié le 12 avril 2012, « Google peut-il encore lutter contre Facebook ? », Le Plus, Le nouvel Observateur
http://leplus.nouvelobs.com/contribution/522548-google-peut-il-encore-lutter-contre-facebook.html
Consulté le 11.06.2012, utilisé à la page : 17

[16] Horace Dediu, publié le 21 décembre 2011, « How many Android phones have been activated? », Asymco
http://www.asymco.com/2011/12/21/how-many-android-phones-have-been-activated/
Consulté le 11.06.2012, utilisé à la page : 17

[17] Wikipédia, « iPhone »
http://en.wikipedia.org/wiki/IPhone
Consulté le 11.06.2012, utilisé à la page : 17

[18] Ardjuna Seghers, publié le 7 mars 2012, « 55 Million iPads Sold to Date », Trusted Reviews
http://www.trustedreviews.com/news/55-million-ipads-sold-to-date
Consulté le 11.06.2012, utilisé à la page : 17

[19] MediaFire, « Business »

http://www.mediafire.com/tour/business/

Consulté le 11.06.2012, utilisé à la page : 18

[20] Guillaume Champeau, publié le 12 mars 2012, « Putlocker grand gagnant de la fermeture de MegaUpload », Numerama

http://www.numerama.com/magazine/21984-putlocker-grand-gagnant-de-la-fermeture-de-megaupload.html

Consulté le 11.06.2012, utilisé à la page : 18

[21] Alexa, « Discover success. »

http://www.alexa.com/

Consulté le 30.05.2012, utilisé à la page : 14

13 ANNEXES

13.1 LIVRABLES

Tous les livrables se trouvent sur le CD-ROM fourni avec le présent rapport, ce dernier contient :

- Développement
  - sources_app : code source de l'application
  - sources_account : code source du compte lié Dropbox
  - scripts : scripts d'installation
  - install.txt : procédure d'installation
  - docs
    - class_diagram.jpg : Diagramme de classes
    - database_model.png : Modèle physique de données
- Gestion de projet :
  - Product Backlog.pdf
  - Release Roadmap-Velocity.pdf
  - Sprint Backlogs.pdf
  - Sprint Burndown.pdf
  - Suivi des heures.pdf
- tokiwi File Manager.pdf : le présent rapport
- Cahier des charges initial.pdf : le cahier des charges réalisé durant les 2 premières semaines du travail
- VBC Jorat-Mézières : ressources pour pouvoir effectuer le scénario de test

13.2 GESTION DE PROJET

13.2.1 DÉMARCHE

Le projet a été réalisé selon une méthodologie Agile. Durant les deux premières semaines, j'ai réalisé le cahier des charges du travail et celui-ci contenait le Product Backlog de l'application. Pour réaliser cet artéfact, j'ai procéder de la manière suivante :

- Décomposition du travail en « Epic »
- Décomposition des « Epic » en User Stories
- Meeting avec les autres fondateurs de tokiwi pour estimer la valeur ajoutée (Business Value) de chaque User Story
- Evaluation de la charge de travail en story points pour chaque user story, et établissement des dépendances
- Planification des users stories pour les 6 sprints en fonction de la charge, des dépendances et de la valeur ajoutée

Durant les sprints, à chaque fois que je commençais une nouvelle user story, je la décomposais en tâches précises. Lorsqu'une tâche était accomplie, je reportais le nombre d'heure effectuée dessus et la date à laquelle elle était terminée. Si toutes les tâches d'une user story étaient terminées, je reportais son nombre de story points dans le sprint burndown.

A la fin de chaque sprint, je reportais les user stories complétées dans le product backlog, et adaptais ma planification.

En parallèle de ceci, j'ai tenu un journal de travail où j'ai noté précisément le temps passé sur chacune des activités, planifiées ou imprévues.

13.2.2 RÉSULTATS

Voici quelques chiffres clés des résultats de cette gestion de projet :

- 350 à 400 heures de charge de travail estimée (165 story points)
- 406.5 heures de travail (au moment d'écrire ces lignes)
    - Prévu/imprévu
        - 21.5 heures (5%) pour la préparation du travail
        - 307 heures (76%) travaillées sur des activités planifiées
        - 78 heures (19%) travaillées sur des activités imprévues

- Répartition par activité
  - 30.5 heures (7%) travaillées sur la gestion de projet
  - 44 heures (11%) travaillées sur la recherche
  - 254 heures (63%) travaillées sur le développement
  - 78 heures (19%) travaillées sur le rapport

Les deux graphiques suivant permettent de se rendre comment ont été réparties les story points au fil des sprints sachant qu'un story point correspond à entre 2 et 3 heures de travail, et un sprint 2 semaines :

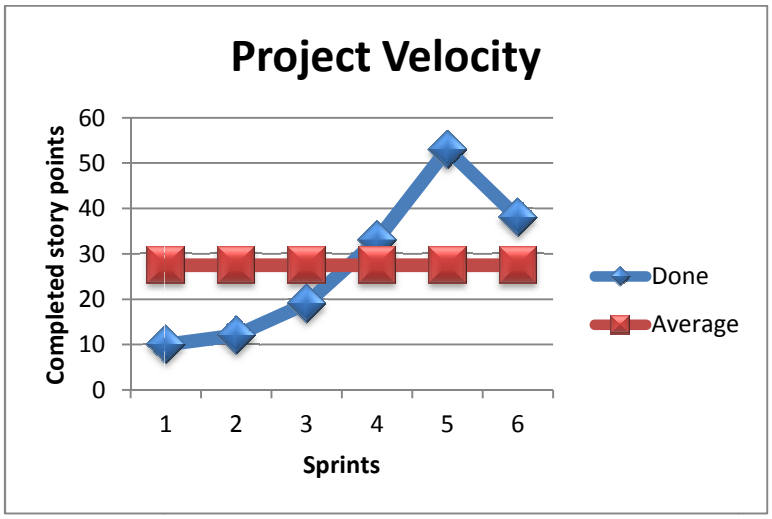


Figure 22 Project Velocity

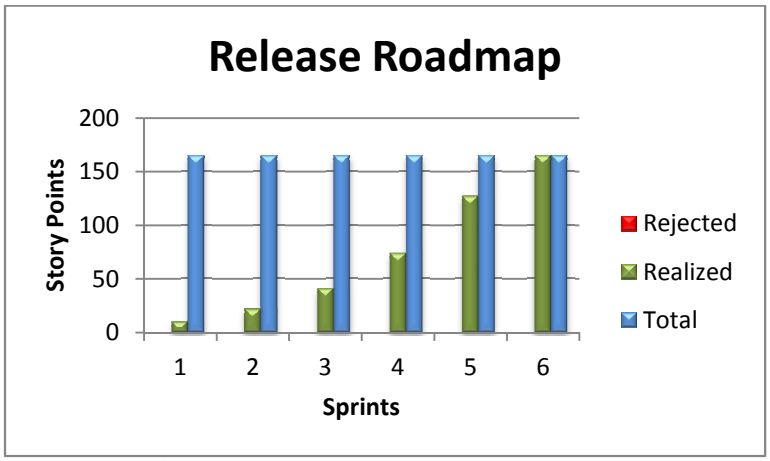


Figure 23 Release Roadmap

Pour rappel, durant les deux premiers sprints nous avions encore des cours en parallèle. Durant le cinquième sprint je ne suis quasiment pas sorti de chez moi pour rattraper le retard accumulé au troisième sprint.

13.3 RETOURS UTILISATEURS

A la fin du 5ème sprint, j'ai créé un environnement de test pour que des utilisateurs puissent aller essayer l'application. Ceci dans le but de déceler des dysfonctionnements dans l'application, et obtenir des retours utilisateurs permettant d'améliorer l'application.

Au total, neuf personnes ont testé l'application sur une durée de 10 jours. Je n'ai malheureusement pas eu suffisamment de temps pour consolider ces retours et en faire une synthèse exploitable. Cependant vous trouverez ci-dessous des extraits de conversation que j'ai eue avec les différents testeurs :

« J'aime beaucoup la mise en page et le design. »
Tamara Monney, 35 ans, informaticienne

« L'anglais c'est du charabia pour moi… Il faut une version française de l'application ».
Jacqueline Bovy, 55 ans, employée de commerce

« Pas mal, pas mal. Dropbox fonctionne, et je me réjoui de pouvoir ouvrir les fichiers Excel et Word lorsque les applications seront installées. En tout cas, je suis déjà séduit par l'interface ! »
Patrick Bovy, 58 ans, expert comptable

« Je ne trouve pas l'interface très intuitive. Mais je trouve intéressant le concept, je pourrais facilement partager mes photos de vacances avec mes amis tout en maitrisant leur accès. »
Anne Galli, 26 ans, comptable

« Le concept est très intéressant parce qu'il permet de partager facilement des informations ou médias dans un cercle restreint et maitrisé. Cela permet, par exemple, d'éviter d'utiliser des outils peu fiables quant à la protection des données, ici tokiwi offre un réel sentiment de sécurité. De plus, même si la communauté est sur internet, elle peut se révéler très locale et ciblée (club de sport ou jeunesse du coin). Cela donne ainsi une autre perspective des outils internet qui se veulent très, voir trop, ouverts et qui cherchent des liens dans tous les sens. Avec tokiwi on peut juste utiliser la technologie pour faciliter le lien déjà existant entre certaines personnes. »
Michel Cachin, 29 ans, architecte

« J'aime beaucoup l'application, et je pense qu'elle peut vraiment être utile à beaucoup de gens. J'attends avec impatience la version mobile. »
Verena Galli, 32 ans, employée de commerce

13.4 SCÉNARIOS D'UTILISATION POUR D'AUTRES APPLICATIONS

File Manager est la première application de la plateforme. Elle peut interagir avec les applications qui vont être développées dans les semaines à venir. Cette annexe liste des descriptions sommaires de futures potentielles applications tokiwi (sous la forme de scénarios d'utilisation) qui interagiront avec l'application développée durant ce travail.

13.4.1 « PLAYMED », LE LECTEUR MULTIÉMDIA

Scénario 1 :

« Hello, je m'appelle Mathieu, j'ai 31 ans et je suis père d'une petite Isabelle, 4 ans, et d'un petit Léo, 18 mois. Ma femme et moi avons acheté une caméra à la naissance d'Isabelle, depuis nous nous sommes transformés en véritable équipe de production de films hollywoodiens.

Annie, ma sœur, a également attrapé le virus avec ses propres enfants. Nos parents n'arrêtent pas de nous demander après les vidéos que nous tournons, et je dois avouer que j'aime bien aussi regarder les vidéos de famille faites par ma sœur.

Jusqu'à présent, nous nous contentions de stocker nos vidéos dans le gestionnaire de fichiers de notre communauté « Famille » sur tokiwi. Nous pouvions ainsi récupérer les vidéos, mais il fallait toujours les télécharger avant de pouvoir les regarder.

Mais depuis que nous avons installé l'application « Playmed » dans notre communauté, nous pouvons regarder directement en ligne ces vidéos, depuis n'importe quel ordinateur. Il suffit d'ouvrir le gestionnaire de fichier, de cliquer sur la vidéo et c'est parti ! »

Scénario 2 :

« Salut, je m'appelle Frédéric, j'ai 24 ans et j'aime beaucoup la musique. Je n'aime pas un style particulier, j'écoute la radio et lorsqu'un morceau me plait, je vais l'acheter sur iTunes. Depuis plusieurs années, j'ai accumulé beaucoup de musique sur mon ordinateur, que je m'efforce de synchroniser avec mon laptop et mon téléphone mobile.

Lorsque je suis en voiture ou que je vais courir, j'écoute toujours de la musique. Et à chaque fois, c'est le même cirque ; je ne sais pas ce que j'ai envie d'écouter, entre les morceaux que j'ai déjà trop entendu, et ceux auxquels je ne pense pas. Je me dis à chaque fois que dès que j'aurai 15 minutes de libre, je les consacrerai à la création de playlists à écouter dans différentes circonstances. Sauf que ces 15 minutes ne se sont jamais libérées.

Sylvain, un bon copain à moi, écoute beaucoup de musique aussi. A chaque fois qu'on fait des soirées, c'est lui le DJ et il passe que des bonnes playlists. L'autre jour, en buvant une bière après le boulot, j'expliquais à Sylvain et Grég, un autre copain, que je m'énervais à chaque fois je voulais écouter de la musique en voiture. Grég a bien rigolé, et a avoué avoir le même problème.

Sylvain a alors eu une idée qui a révolutionné mes trajets en voitures et mes courses à pieds. Il a créée une communauté sur tokiwi dans laquelle il nous a invité Grég et moi. Nous avons tous mis nos morceaux de musique dans le gestionnaire de fichiers et Sylvain nous a concocté de nombreuses playlists dans le lecteur multimédia de la communauté. Elles sont classées par genre, donc quand je vais courir il me suffit maintenant d'aller dans cette communauté et de choisir une playlist stimulante, et lorsque je conduis, j'évite justement ce genre de playlists là… »

Scénario 3 :

« Salut, je m'appelle Clément, j'ai 27 ans et je suis chroniqueur sur Couleur3. Je prête ma voix dans plusieurs rubriques de la radio, et je fais également partie de l'équipe qui sélectionne les repérages.

Nous mettons tous les podcasts de nos émissions sur le site de la radio, mais celui-ci ne permet pas à nos auditeurs de pouvoir communiquer avec nous. Il en va de même pour les repérages.

J'ai alors décidé de créer une communauté tokiwi pour couleur3 et nos auditeurs. Tous les podcasts y sont librement accessibles dans le gestionnaire de fichiers de la communauté ou en écoute directement depuis l'application « Playmed ». Les auditeurs peuvent ainsi les commenter et en discuter avec nous ou entre eux.

De plus, j'ai eu une idée géniale qui nous a permis de découvrir plein de nouveaux groupes pour nos repérages. J'ai créé un répertoire sécurisé dans le gestionnaire de fichiers de la communauté dans lequel des auditeurs ou des artistes peuvent déposer des fichiers audio. Ces fichiers audio apparaissent automatiquement dans la playlist « Nouveaux repérages » de l'application « Playmed ». Les auditeurs peuvent alors évaluer chacun des morceaux qui s'y trouvent. Au bout de deux semaines dans la playlist, si le morceau n'a pas atteint les 50 votes positifs, je le retire de la playlist, sinon je l'ajoute à la liste des repérages et prépare une petite présentation du groupe.

Depuis que nous avons mis en place cette communauté, nous récoltons beaucoup plus de retours des utilisateurs, ce qui nous permet d'améliorer considérablement nos émissions et nous avons rendu célèbres plusieurs groupes régionaux. »

13.4.2 « TOPIC », L'ALBUM PHOTOS

Scénario 1 :

« Hello, je m'appelle Annie, je suis la sœur de Mathieu. J'ai une grande passion dans la vie, c'est les voyages. Mes amies, Véronique, Sandra et Rachel partagent cette passion. Du coup, nous partons chaque année visiter une nouvelle ville ensemble.

En vacances, nous sommes de vraies paparazzis, nous avons toutes un super appareil photo et nous nous en servons pour photographier des monuments, nos chambres d'hôtel, nos repas, les beaux mecs dans la rue…

Quand nous rentrons de vacances, nous avons tout le temps le même problème ; le partage des photos. Sandra et Véronique les mettent sur Facebook, faut pas demander à Rachel d'allumer un ordinateur, elle y comprend rien, quant à moi j'aime pas du tout mettre mes photos sur des sites internet où je ne maitrise pas qui en a l'accès.

Depuis peu, nous avons créé une communauté tokiwi. Ceci nous a bien facilité l'organisation de nos diverses activités. Nous avons résolu le problème d'échange de photos grâce au gestionnaire de fichiers de la communauté. Même Rachel arrive à y envoyer ses photos, c'est dire ! (Bon… son copain doit toujours lui montrer comment on transfert les photos de l'appareil à l'ordinateur, mais la suite elle gère.)

En bidouillant un peu, Sandra nous a installé une application d'album photo dans la communauté, « toPic ». C'est magique ce truc ! Il suffit de créer un album, d'ouvrir le gestionnaire de fichiers, et de glisser des photos d'une fenêtre à l'autre. On peut ensuite modifier l'ordre dans lequel elles apparaissent, dire où elles ont été prises, qui est sur la photo, … Nous avons créé un album par ville visitée, et j'adore passer des soirées entière à regarder et commenter nos photos ! »

Scénario 2 :

« Hey, je m'appelle Annabelle, j'ai 39 ans et je suis responsable marketing pour le Caprices Festival. Cette année, nous avons décidé de promouvoir le festival avec une communauté tokiwi. Cette plateforme est super, il y a plein d'outils pour communiquer avec les gens et partager du contenu.

Nous avons d'ailleurs fait plusieurs albums photos qui mettent en scène les préparations du festival et les différentes soirées. C'est pratique parce que nos 11 photographes prennent les photos, les envoie dans le gestionnaire de fichiers tokiwi et ensuite l'équipe marketing les retouche à l'aide

l'application « toPic » avant de les publier. Ça marche bien, mais ce n'est pas très innovant comme système, et les photographes coûtent de plus en plus cher.

Aujourd'hui, les téléphones mobiles sont équipés d'appareils photo quasiment professionnels, donc nous nous sommes dit « Pourquoi ne pas engager les festivaliers comme photographes ? ».

Nous avons donc lancé un concours. Les gens peuvent envoyer des photos du festival vers un répertoire sécurisé du gestionnaire de fichier de la communauté directement depuis leur mobile. Les photos apparaissent dans l'album de la soirée correspondante et sont donc visibles par tous les membres de la communauté. Ils peuvent alors voter pour les meilleures photos. A la fin du festival, les photographes qui ont obtenu le plus de votes gagnent des entrées gratuites, des abonnements de ski dans la station et un chèque de CHF 3'000.- pour le premier.

Cette idée a eu un franc succès, beaucoup de gens ont joué le jeu et nous avons obtenu de nombreuses photos de qualité. De plus, la communauté a été des plus actives durant l'ensemble du festival. »

13.4.3 « MEETING TOOLKIT », LE GESTIONNAIRE DE RENDEZ-VOUS

Scénario 1 :

« Bonjour, je m'appelle François, j'ai 42 ans et je suis membre du conseil d'administration dans deux entreprises et une start-up. J'ai un agenda bien rempli, je brasse beaucoup de documents et je vous avoue que j'ai parfois du mal à m'y retrouver.

J'ai découvert tokiwi grâce à un collègue de travail. Depuis j'ai instauré le système dans mes trois conseils d'administration et çeci s'est avéré très pratique pour échanger nos documents et coordonner nos agendas. Depuis, je respire un peu plus, mais je n'ai pas franchement gagné beaucoup de temps dans la préparation des meetings et leur suivi.

Vous n'imaginez pas ma joie lorsque tokiwi a sorti l'application « Meeting toolkit ». L'application permet de créer des meetings, ceux-ci sont automatiquement ajoutés dans l'agenda. Il est possible de lier les meetings entre eux s'ils ont des liens et d'y associer des documents comme des rapports financiers ou des documents techniques. Elle dispose également d'un éditeur d'ordre du jour, de prise de notes et de procès verbaux. La personne qui rédige le procès verbal peut accéder à toutes les notes prises par les différentes personnes présentes.

Depuis que nous utilisons cette application, les ordres du jour et les procès verbaux s'écrivent presque tout seul, et il est devenu beaucoup plus facile de retrouver les documents du gestionnaire de fichiers. Le gain de temps et d'efficacité est colossal. »

13.4.4 « SCHOOL TOOLKIT », LE GESTIONNAIRE DE COURS

Scénario 1 :

« Bonjour, je m'appelle Clara, j'ai 32 ans et je suis professeure à la HES-SO Valais depuis 2 ans déjà. J'enseigne la gestion de projet à des étudiants de la filière Tourisme. Je dispense mes cours à l'aide de présentations PowerPoint, et les étudiants doivent réaliser de nombreux travaux et documents à des échéances bien établies.

Je vous avoue que je n'aime pas vraiment le programme interne à l'école qui sert à donner le cours. Il permet de mettre à disposition des documents, mais il n'est pas très pratique ni « user friendly ». De plus, il ne dispose d'aucun moyen de communication et à l'approche des examens je me fais toujours bombarder de mails, souvent pour les mêmes questions.

Cette année j'ai décidé de donner mon cours à l'aide d'une communauté tokiwi. Et je dois dire que c'est un autre monde. Avec l'application « School toolkit », je peux créer un cours, le décomposer en plusieurs parties, planifier les leçons et fixer des échéances et des examens, le tout étant reporté dans l'agenda de la communauté. Et grâce au gestionnaire de fichiers de la communauté, il est possible d'associer le support de cours à une partie du cours et des répertoires de dépôts de travaux à des échéances et des examens.

De plus, l'application dispose d'une fonctionnalité qui permet aux étudiants de poser des questions en rapport à un document, une échéance, un examen ou une partie du cours. Les questions et les réponses étant visibles par toute la communauté, il n'y a donc plus de questions redondantes.

Je peux également corriger les travaux des étudiants directement depuis l'application. Chaque travail rendu par un étudiant peut être commenté, et il est possible de limiter l'accès à ces commentaires. Mieux que ça, je peux préparer des commentaires, les sauver, et ensuite en associer plusieurs à des travaux.

Depuis que j'utilise tokiwi, je peux beaucoup plus facilement préparer mes cours, les étudiants peuvent poser des questions directement en rapport au support de cours et je gagne un temps fou en corrections de travaux. »

Scénario 2 :

« Tchô, moi c'est Nicolas, j'ai 21 ans. Je viens de passer en deuxième année de Bachelor dans la filière Tourisme. J'ai choisi cette voie parce que je suis quelqu'un de sociable et j'aime beaucoup organiser des évènements en tout genre avec mes copains.

On n'a pas beaucoup de cours, mais il y a quand même beaucoup de travaux à rendre et d'examens à préparer. La plupart des travaux sont des travaux de groupe, et c'est à chaque fois une sacrée pagaille pour réussir à se coordonner sur qui fait quoi. On s'en sort toujours, mais dieu sait si on sue le soir avant de rendre le travail. Pour les préparations d'examens, en général chacun fait un résumé sur une matière et la semaine avant l'examen c'est un peu la foire aux résumés dans la salle d'étude.

Au deuxième semestre, on a décidé d'utiliser l'application « School toolkit » pour voir si c'était possible de moins se prendre la tête avec les travaux de groupe. L'application est super, elle permet de décomposer le travail en plusieurs tâches, de les ordonner, les planifier et les attribuer. Donc ce qu'on fait pour les travaux de groupe, c'est qu'on se voit tous ensemble une fois, on fait la table des matières du document, on se répartit les chapitres entre nous et on entre tout ça dans l'application. Après chacun bosse un peu de son coté, si on a besoin d'informations d'un autre, il suffit d'aller voir les pièces jointes dans ses tâches. Et à la fin, on se revoit une fois tous pour mettre ensemble les différents travaux. C'est franchement pratique pour s'échanger des documents et savoir à quoi on en est.

L'application dispose aussi d'un outil de prises de notes. Ca c'est génial, car je déteste prendre des notes, mais je peux accéder à toutes les notes prises par les autres étudiants de la classe depuis le gestionnaire de fichiers de la communauté. On utilise ces notes en général pour faire nos résumés pour les examens qu'on met aussi dans le gestionnaire de fichiers. Après chacun complète les résumés avec ce qu'il juge utile et à la fin on a des résumés très bien foutus.

En tout cas je ne regrette pas d'utiliser cette application tokiwi, c'est hyper bien pour s'organiser pour les cours et en plus il y a plein d'autres applications très utiles, comme les albums photos de nos soirées et le chat ou la bataille navale pour pendant les cours… »

13.5 INTRODUCTION AU DÉVELOPPEMENT D'APPLICATIONS TOKIWI

13.5.1 TECHNOLOGIES UTILISÉES

tokiwi utilise les technologies suivantes :

- **HTML 5** : pour mettre en forme le contenu.
- **CSS 3** : pour l'esthétisme et la mise en page du contenu.
- **Javascript**, **jQuery 1.7 et jQuery UI 1.8** : pour toutes les interactions et effets graphiques coté client. La plupart des requêtes sont effectuées de manière asynchrone et la librairie jQuery est largement utilisée dans ce but.
- **PHP 5.3** : pour toute la couche logique.
- **MySQL 5.5** : pour le stockage des données.

Les deux principaux avantages de ces technologies sont qu'elles sont gratuites et qu'elles bénéficient d'une large communauté de développeur, rendant ainsi aisé l'accès à la documentation, accélérant le développement et diminuant les coûts.

De plus elles ne nécessitent pas l'installation d'un quelconque plug-in coté client. Un navigateur récent et une connexion internet sont les seuls pré-requis pour l'utilisation de la plateforme.

En revanche, certaines machines peuvent ne pas être compatibles avec ces technologies (les anciens navigateurs et les navigateurs dont les règles de sécurité désactivent Javascript par exemple). HTML 5 est en cours de normalisation, et à l'heure actuelle, seules les versions récentes de Firefox (10+), Google Chrome (14+) et Opéra (11+) ne sont pleinement compatibles avec les fonctionnalités d'HTML 5 utilisées dans la plateforme.

jQuery est également une librairie particulièrement critiquée pour sa lourdeur et ses problèmes de performances. Il existe des librairies plus efficaces pour accomplir le même cahier des charges, mais peu d'entre eux sont aussi bien documentés et aussi évolutifs que jQuery. De nombreux efforts sont effectués par les développeurs de ce framework pour corriger les problèmes connus, et je suis convaincu que son utilisation est un bon choix technique.

13.5.2 ARCHITECTURE DU SYSTÈME D'APPLICATIONS

Une application tokiwi est composée de modules, communément appelés « mods », représentant des points d'entrée de l'application. Chaque module est exécuté dans une fenêtre dédiée qui peut être déplacée et redimensionnée par l'utilisateur. De plus, plusieurs instances du même module peuvent s'exécuter dans le même espace de travail. Ces modules servent ensuite à afficher les différents formulaires de l'application et à interagir avec l'utilisateur.

Outre les modules, une application est également composée des éléments suivants :

- **Classes PHP** : permettant de gérer la logique applicative de l'application.
- **Fonctions Javascript** : permettant de traiter les interactions avec l'utilisateur et appelant les services appropriés de l'application.
- **Scripts PHP** : communément appelés « webservices » dans le cadre de la plateforme, permettant de faire le lien entre la couche logique et les fonctions javascripts.
- **Thèmes** : permettant de définir les règles d'affichage des formulaires et les sets d'images à utiliser.

Le schéma suivant récapitule la composition des applications tokiwi :

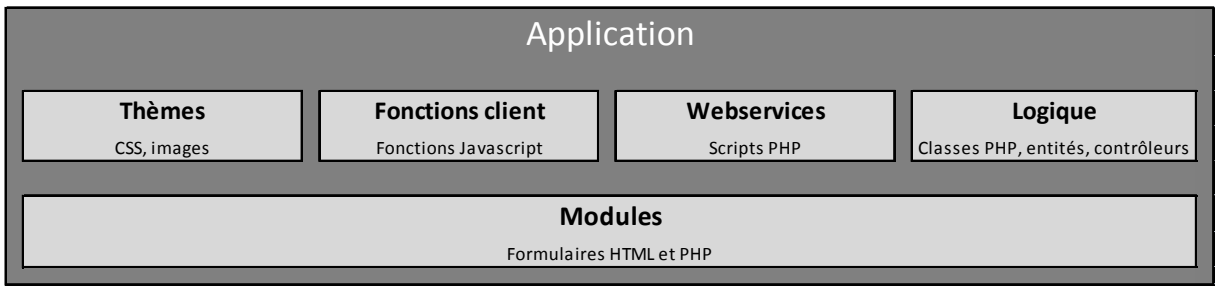


Figure 24 Composition des applications tokiwi

13.5.3 PROCESSUS D'EXÉCUTION D'UNE APPLICATION

Chaque fois qu'un module d'une application est exécuté, une instance de ce module est créée. Cette instance va ensuite être associée à un ou plusieurs utilisateurs qui vont pouvoir interagir avec.

Ces instances de module peuvent obtenir des paramètres dynamiques au cours de leur cycle de vie. Les paramètres sont automatiquement sauvegardés en base de données. Par exemple, un module qui permettrait d'afficher le détail d'un produit pourrait prendre l'identifiant du produit en paramètre.

Les dimensions, la position et l'état (minimisé ou maximisé) sont dépendant de l'utilisateur. Ces paramètres sont également automatiquement sauvegardés en base de données à chaque modification.

Le schéma suivant est un exemple d'utilisation d'une application d'e-commerce :

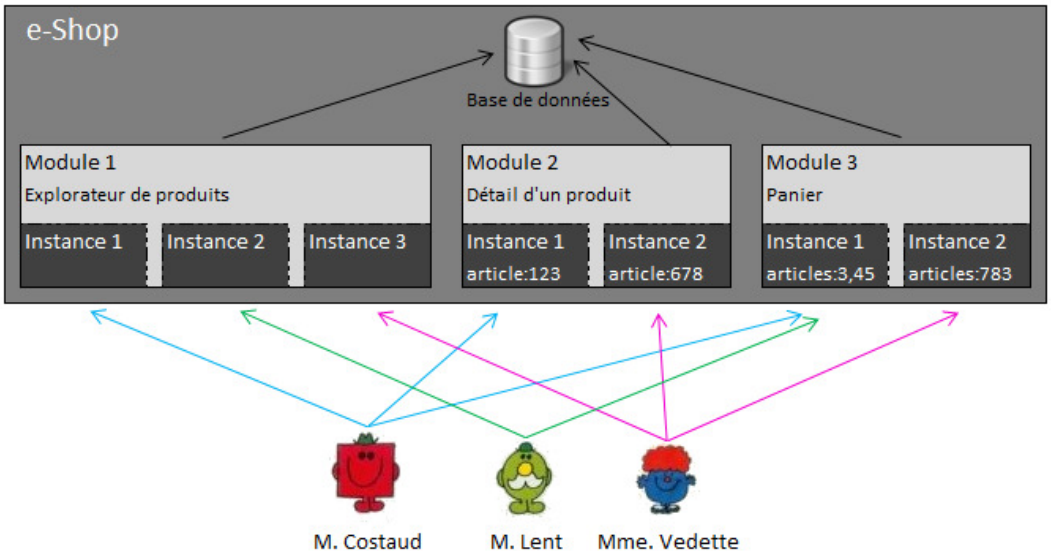


Figure 25 Exemple d'une application e-Shop

Dans cet exemple, on constate que M. Costaud et M. Lent partage le même panier (ils utilisent la même instance du module 3) mais effectuent leurs achats à partir d'une instance du module 1 différente. On constate également que les modules 2 et 3 peuvent obtenir des paramètres, en l'occurrence le numéro d'article dont il faut afficher le détail pour le module 2 et la liste des articles ajoutés au panier pour le module 3. Ces paramètres sont partagés par tous les utilisateurs de l'instance du module.

13.5.4 UTILISATION DE L'API DU SYSTÈME

La plateforme met à disposition des applications une API PHP qui permet d'accéder à toutes les entités du système, notamment les utilisateurs et les communautés. L'accès à la base de données se fait par un DAO très simple à utiliser. Malheureusement la plateforme n'étant pas encore terminée, la documentation de cette API n'existe pas encore.

Outre l'API PHP, il existe également une API Javascript qui fournit une multitude de fonctions utilitaires particulièrement pratiques. Par exemple, l'exécution d'une application se fait en effectuant un simple appel à la fonction « loadMod(<numéro du module>) ». Des fonctions servant à afficher

des pop-ups ou des info-bulles sont également présentes. Ces fonctions utilitaires permettent d'accélérer le développement et d'uniformiser les éléments graphiques de l'interface.

Finalement, il est également possible de se servir des feuilles de style de la plateforme pour intégrer également visuellement les applications dans le système. Il existe des styles pour les titres, les boutons, les champs de saisies, les icônes, et bien d'autres encore.

13.5.5 CRÉATION D'UNE APPLICATION

La création d'une application tokiwi se déroule en trois étapes :

1. Enregistrement des identifiants de l'application dans la base de données :
    a. Chaque application possède un identifiant numérique unique et un nom
    b. Chaque module d'une application possède un identifiant numérique unique, un nom et des paramètres optionnels (dimensions par défaut par exemple)
    c. Tous les identifiants inférieurs à 1000 sont réservés pour les applications dites « systèmes ». Les applications systèmes sont installées par défaut dans toutes les communautés et n'apparaissent pas dans le magasin d'applications.
2. Développement de l'application
    a. L'ensemble des fichiers nécessaires à l'exécution de l'application doivent être contenus dans un répertoire ayant l'identifiant de l'application comme nom
    b. Les fichiers suivants doivent obligatoirement figurer dans ce répertoire :
        i. **jsLib/eventsHandler.js** : contient la librairie Javascript de l'application, elle est chargée automatiquement par le système au lancement de l'application.
        ii. **themes/styles/appearance.css** : contient toutes les propriétés visuelles de l'application. Cette feuille de style est également chargée automatiquement au lancement de l'application.
        iii. **themes/styles/layout.css** : contient toutes les propriétés de mise en page de l'application. Cette feuille de style est également chargée automatiquement au lancement de l'application.
        iv. **mods/<*identifiants*>/index.php** : chaque module de l'application doit être contenu dans un répertoire portant son identifiant comme nom qui est ensuite placé dans le répertoire « mods ». Le point de lancement du module est le fichier « index.php », ce dernier peut ensuite appeler d'autres formulaires.

c. Lorsque le système exécute un module, il appelle le fichier « index.php » correspondant après avoir initialisé les variables suivantes (qui peuvent donc directement être utilisées dans le module) :
    i. **$user** : instance de la classe « User » correspondant à l'utilisateur actuellement connecté.
    ii. **$community** : instance de la classe « Community » correspondant à la communauté dans laquelle s'exécute le module.
    iii. **$application** : instance de la classe « Application » correspondant à l'application à laquelle appartient le module exécuté.
    iv. **$mod** : instance de la classe « Mod » correspondant au module actuellement exécuté.
    v. **$modInstance** : instance de la classe « ModInstance » correspondant à l'instance unique du module actuellement exécuté.
    vi. **$params** : tableau associatif contenant tous les paramètres enregistrés de l'instance du module.
d. Plusieurs instances d'un module peuvent s'exécuter dans la même page, ainsi pour éviter des collisions d'identifiants, chaque :
    i. Attribut « id » des balises HTML doit être préfixé par l'identifiant de l'instance du module.
    ii. Nom de classe CSS doit être préfixé par l'identifiant de l'application.
    iii. Fonction Javascript doit être préfixée par l'identifiant de l'application.

3. Déploiement de l'application
    a. Une fois l'application développée et testée, il faut copier le répertoire de l'application dans le répertoire « applications » du système.
    b. Créer les tables de l'application en base de données

13.6 COMPLÉMENTS TECHNIQUES

13.6.1 FONCTIONNEMENT DES INTERACTIONS GRAPHIQUES

Les tableaux suivants permettent de visualiser à quelles classes HTML sont appliquées les interactions jQuery « Draggable », « Droppable » et « Selectable » et quelles actions sont associées aux événements qu'elles génèrent :

File Manager

Shared Folders and Files	Interaction		
Classes	Type	Target	Action
associableItem			
_3_fileResourceItem	selectable		save selected
			add draggable interaction
	draggable		

Folders	Interaction		
Classes	Type	Target	Action
_3_folderItem	droppable	associableItem	moveCopyAssociateResource (la fonction identifie l'action adéquate elle-même)
		_3_fileResourceItem	
		_3_dropboxFileResourceItem	

Files	Interaction		
Classes	Type	Target	Action
_3_fileItem	droppable	associableItem	associate
		_3_fileResourceItem	

Trash

Shared Folders and Files	Interaction		
Classes	Type	Target	Action
_3_trashFileResourceItem	selectable		save selected

Folders	Interaction		
Classes	Type	Target	Action
_3_trashFolderItem			

Files	Interaction		
Classes	Type	Target	Action
_3_trashFileItem			

Dropbox

Shared Folders and Files	Interaction		
Classes	Type	Target	Action
_3_dropboxFileResourceItem	selectable		save selected
	draggable		

Folders	Interaction		
Classes	Type	Target	Action
_3_dropboxFolderItem	droppable	_3_fileResourceItem	moveCopyResource (la fonction identifie l'action adéquate elle-même)
		_3_dropboxFileResourceItem	

Files	Interaction		
Classes	Type	Target	Action
_3_dropboxFileItem			

Figure 26 Référence des interactions jQuery

13.6.2 SET D'ICÔNES

La liste des icônes utilisées pour les menus et les ressources se trouve dans le tableau ci-dessous :

Nom	Icône	Libellé	Description
folderFileIcon.png		-	Représente un répertoire
fileFileIcon.png		-	Représente un fichier de type inconnu
imageFileIcon.png		-	Représente un fichier de type image
musicFileIcon.png		-	Représente un fichier de type audio
videoFileIcon.png		-	Représente un fichier de type vidéo
logo.png		-	Logo de l'application
settingsIcon.png		Settings	Ouvre la liste déroulante des menus
searchIcon.png		Search	Affiche la barre de recherche
trashIcon.png		Open Trash	Ouvre la poubelle
dropboxIcon.png		Open Dropbox explorer	Ouvre l'explorateur de fichiers dropbox
fileManagerIcon.png		Open File Manager	Ouvre l'explorateur de fichiers tokiwi
downloadIcon.png		Download selected items	Télécharge les fichiers sélectionnés
uploadIcon.png		Upload files	Upload des fichiers à partir de l'ordinateur
newFolderIcon.png		New folder	Créé un nouveau répertoire
renameIcon.png		Rename selected items	Renomme les fichiers sélectionnés
deleteIcon.png		Delete selected items	Supprime les fichiers sélectionnés
restoreIcon.png		Restore selected items	Sort de la poubelle les fichiers sélectionnés
diskUsageIcon.png		Disk Usage	Affiche des statistiques d'utilisation du disque
permaDeleteSelectedIcon.png		Delete permanently selected items	Supprime de la poubelle les fichiers sélectionnés
permaDeleteAllIcon.png		Delete permanently all items	Supprime tous les fichiers de la poubelle
removeIcon.png		Remove search result	Supprime le résultat d'une recherche
iconItemsIcon.png		Show items as icons	Affiche les fichiers sous forme d'icônes
listItemsIcon.png		Show items as a list	Affiche les fichiers sous forme de liste

Tableau 6 Icônes de l'application

Les 6 premières icônes ont été réalisées par le graphiste de la plateforme. Les autres sont des icônes que j'ai trouvées sur le site : http://www.iconfinder.com et que j'ai ensuite adaptées au contexte. Je me suis assuré qu'elles soient toutes libres de droit.

13.6.3 DIAGRAMME DE CLASSES

Voici le diagramme de classes complet des fonctionnalités développées :

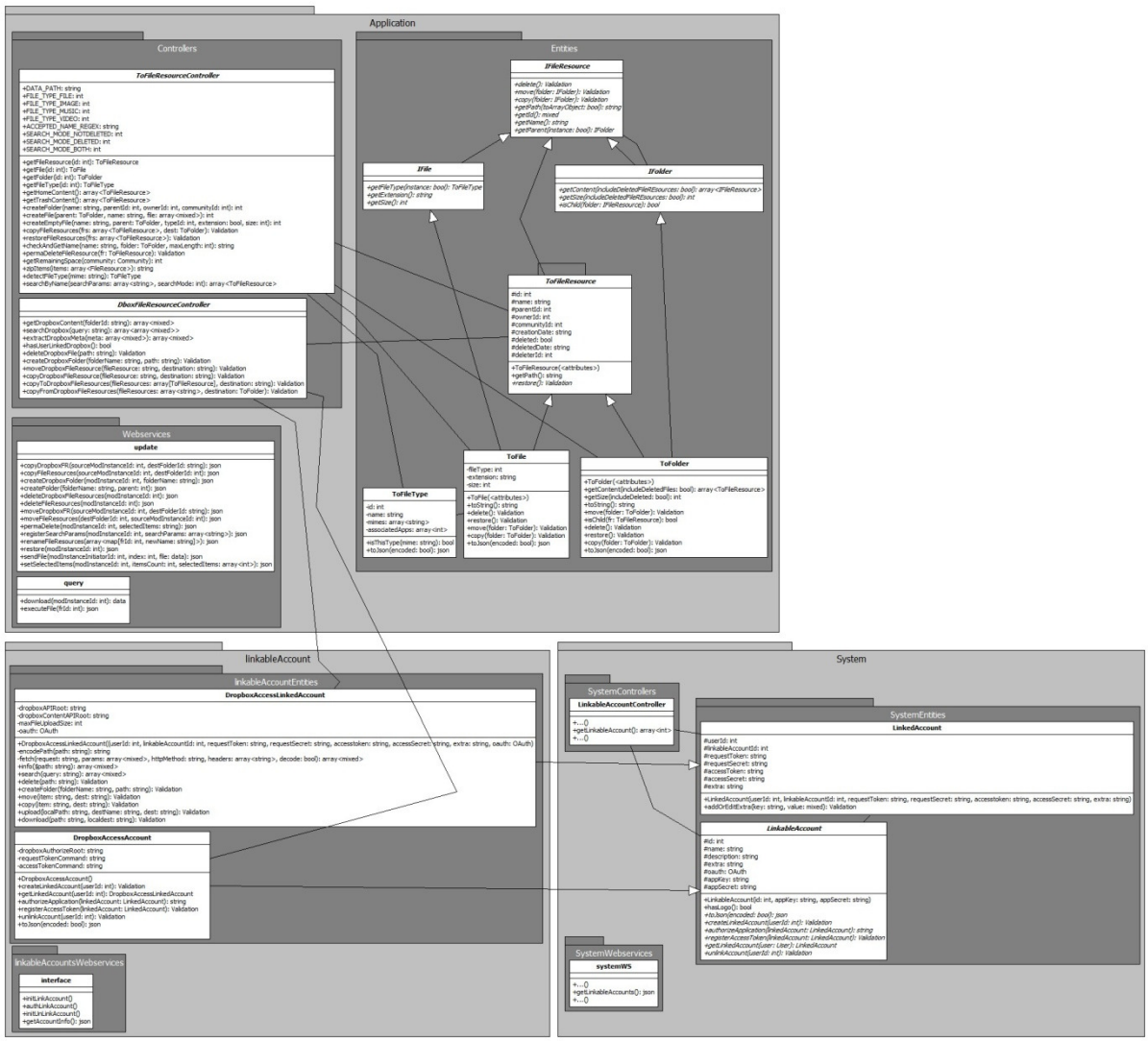


Figure 27 Diagramme de classes général

Il se décompose en trois blocs principaux (gris clairs) qui correspondent à l'emplacement des classes (Applications, comptes liés, système). A l'intérieur, on retrouve chaque fois trois sous-blocs (gris foncés) qui correspondent à la fonction logique des classes (contrôleurs, entités, webservices). Les classes apparaissent dans les blocs blancs.

13.6.4 MODÈLE PHYSIQUE DE LA BASE DE DONNÉES

Le schéma suivant permet de visualiser le modèle physique de la base de données :

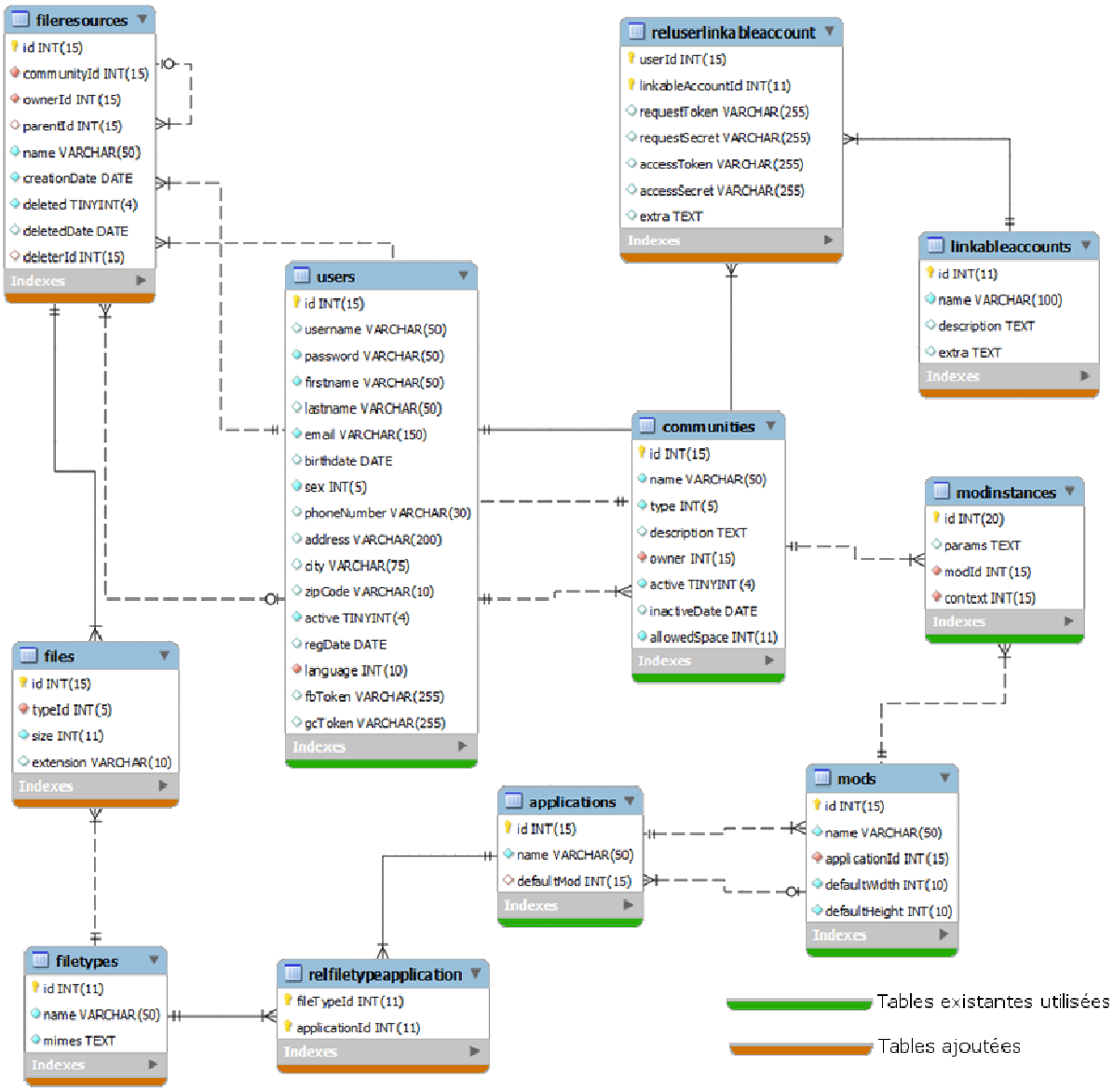
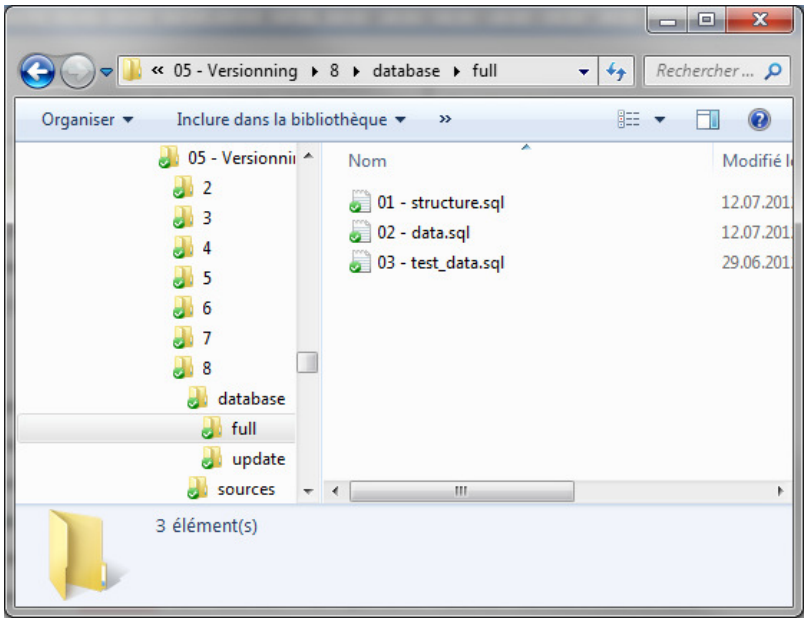


Figure 28 Modèle physique de données

13.6.5 VERSIONNING

Etant donné que la plateforme est encore en construction, j'ai également du y faire de nombreuses modifications en développant l'application File Manager. Je n'ai donc pas fait de versionning de l'application, mais j'ai inclus les différentes versions de l'application dans le versionning de la plateforme.

Celui-ci est très simple. Toutes les versions de tokiwi se trouvent dans un « dépôt » (repository). A chaque nouvelle livraison de fonctionnalités (ce qui généralement correspond à la fin d'un sprint), le numéro de version est incrémenté de 1. Si la livraison ne contient pas de nouvelles fonctionnalités, mais uniquement de corrections, le numéro de version est alors incrémenté d'une décimale. Le dépôt contient des répertoires portant le numéro de version comme nom. Chaque version contient deux répertoires nommés « database » et « sources ». Le répertoire « source » contient tout le code source de la plateforme correspondant à la version. Le répertoire « database » contient tous les scripts permettant de créer la base de données et ceux permettant de la mettre à jour à partir de la version précédente.



Copie d'écran 34 Illustration du versionning

Grâce à ce système, il est possible de passer d'une version à une autre en un rien de temps. La création de nouveaux environnements est très rapide. Il serait possible d'améliorer ce système en forçant la création de scripts permettant de rétrograder d'une version la base de données. Ainsi, les « roll-backs » deviendraient tout autant simple à réaliser.

Durant ce travail, tokiwi est passé de la version 6.0 à la version 9.1.

13.6.6 OUTILS DE DÉVELOPPEMENT

Les outils suivants ont été utilisés pour mener à bien le développement de l'application :

- **Eclipse Indigo** 3.7.0 avec les modules PHP, Web developers et GIT : comme éditeur de code pour les différents composants de l'application. Parmi les différents éditeurs de code que j'ai eu l'occasion de tester, celui-ci est le moins mauvais pour le développement PHP, mais il est encore loin de disposer des mêmes fonctionnalités que pour des développements Java par exemple.
- **XAMPP** 1.7.7 avec Apache 2.2.21 et PHP 5.3.8 : comment environnement de développement. XAMPP est une solution clé en main très facile à mettre en œuvre.
- **MySQL Workbench** 5.2.36 : pour administrer la base de données et maintenir à jour la documentation de la base de données. Il s'agit d'un outil très puissant doté de nombreuses fonctionnalités rivalisant avec ses concurrents payants.
- **Git for Windows** 1.7.8 : pour livrer le code sur le serveur GIT de tokiwi.
- **GIMP** 2.8.0 : pour réaliser les différentes icônes de l'interface graphique. Il s'agit d'une alternative viable à photoshop pour des designers amateurs de mon niveau.
- **StarUML** 5.0 : pour réaliser les différents diagrammes de la documentation. Il s'agit d'un outil un peu « vieillot », mais qui permet de facilement réaliser n'importe quel diagramme UML.
- **PasswordSafe** 3.27 : pour sauver les informations des différents comptes utilisés au cours du travail.
- **Microsoft Excel** 2007 : pour toute la gestion de projet. Il existe probablement des outils plus adaptés, mais je n'ai pas pris le temps d'en chercher.
- **Google Chrome** 21, **Mozilla Firefox** 13 et 14, **Internet Explorer** 9 : pour tester et débugger les différentes fonctionnalités imlémentées.

Bien que tous ces outils soient gratuits, à l'exception de Microsoft Excel, j'en suis pleinement satisfait. Si je devais entamer un nouveau projet du même type, il est fort probable que je m'en servirai à nouveau.