Rapport de stage

Thomas Citharel

22 août 2016

Remerciements

Merci maman merci papa

Table des matières

Remerciements **3**

Introduction **6**

Abréviations **7**

Glossaire **8**

1 Les acteurs du stage **9**

1.1 IUT 9

1.2 Framasoft 10

1.3 Site de Locaux Motiv' 13

2 Sujet du stage **14**

2.1 Présentation du sujet 14

2.2 Objectifs du stage 16

3 Analyse de l'existant **18**

3.1 Historic 18

3.2 ownCloud 19

4 Développement **21**

4.1 Organisation du travail 22

4.2 Environnement technique 23

4.3 Fonctionnalités 24

4.4 Proposition d'améliorations 28

Bilan du stage **29**

5.1 Bilan technique 29

5.2 Bilan humain 30

Webographie **31**

Annexes **32**

9.1 Les standards de gestion de calendrier 32

9.2 Liste des services de Framasoft 34

Introduction

Dans le cadre des études préparées à l'**IUT Informatique d'Orléans-La Source**, je devais effectuer un stage d'une durée de 10 semaines afin de valider le Diplôme Universitaire de Technologie. Ce stage a pour but d'acquérir une première expérience professionnelle en validant les connaissances apprises durant l'année et en exerçant des compétences au travers de la réalisation d'un projet professionnel.

Ce stage s'est pour ma part effectué du 20 juin au 27 août 2016 au sein de l'association **Framasoft** dont les buts sont la promotion, la diffusion et le développement de logiciels libres, de services libres en ligne et de la culture libre.

J'avais pour objectif de développer et mettre en place une alternative à Google Calendar nommée **Framagenda**. En accord avec mon maître de stage, je me suis basé sur l'application existante libre ownCloud dont la fonctionnalité principale est le service de stockage et de sauvegarde de fichiers mais qui propose de nombreuses applications tierces dont une faisant office de calendrier.

Abréviations

— **AGPL** : Affero General Public Licence, licence libre crée par la FSF pour les applications web utilisables comme des services.

— **CSS** : Cascading Style Sheets, feuilles de styles en cascade. Ensemble de règles qui régissent l'apparence d'éléments dans une page HTML.

— **FSF** : Free Software Foundation, une fondation promouvant l'utilisation du logiciel libre.

— **HTML** : HyperText Markup Language, un format de structuration de données employé pour représenter les pages web.

Glossaire

— **Creative Commons** : Multiples licences libres surtout employées pour les œuvres de l'esprit (textes, images, vidéos, ...)
— **iCal** : Format de fichier employé pour enregistrer les données d'un calendrier inventé en 1998.
— **empreinte** : Identifiant unique, résultat d'une fonction de hachage.
— **fédération** : Le concept de fédération consiste en la communication de plusieurs instances de l'application. Ainsi, il est possible de partager des fichiers avec des utilisateurs inscrits sur d'autres serveurs en connaissant leur identifiant de la forme *utilisateur@domaineowncloud.tld*.
— **fonction de hachage** : Fonction mathématique qui produit un identifiant unique pour une information.
— **framework** : Ensemble de composants permettant de mettre de place des applications ou des sites web plus facilement.
— **pad** : Éditeur de texte collaboratif en ligne.
— **pull request** : Requête aux mainteneurs de l'application pour que les changements dans le code soient acceptés.
— **sel** : Chaîne de texte aléatoire servant à ajouter de l'entropie aux fonctions de hachage.
— **sérialisation** : Conversion des données en mémoire sous la forme d'objets dans un format texte structuré, dans notre cas dans le métalangage de balisage XML.
— **template** : Ici, modèle de présentation des données.

Chapitre 1

Les acteurs du stage

1.1 IUT

L'**Institut Universitaire de Technologie d'Orléans - La Source** fait partie intégrante de l'Université d'Orléans - Tours. Il est composé de différents départements dont celui d'Informatique. Les domaines enseignés sont théoriques comme pratiques, et des enseignements complémentaires dont une langue vivante et des cours de Gestion et de communication professionnelle y sont également donnés.

FIGURE 1.1 – Logo de l'IUT et de l'Université d'Orléans

La formation conduisant à un Diplôme Universitaire de Technologie est d'ordinaire effectuée sur la période de deux années mais l'IUT offre également aux étudiants ayant déjà un parcours dans l'enseignement supérieur la possibilité d'effectuer cette formation en une année.

La formation se conclut par un stage obligatoire pour mettre en application les connaissances acquises au fil de l'année, exercer ses compétences et en

acquérir de nouvelles à travers la participation à un projet non-scolaire.

1.2 Framasoft

1.2.1 Présentation de la structure d'accueil

Le site internet Framasoft (pour **fra**nçais et **ma**thématiques) a été fondé par le professeur de mathématiques Alexis KAUFMANN en 2001. Il se présente à cette époque comme un annuaire de logiciels libres et gratuits comme ressources pédagogiques pour enseignants.

FIGURE 1.2 – Logo actuel de Framasoft

Le site Framasoft se développe en réseau avec l'intégration de tutoriels et du forum Framagora. Une association Loi 1901 éponyme est fondée en 2004 pour gérer le réseau et acquerra le statut d'association à but non lucratif quatre ans plus tard. L'annuaire se restreint aux uniquement aux logiciels libres en se séparant des logiciels dits *gratuiciels* la même année, et s'ouvre à d'autres systèmes d'exploitation que Windows.

Framasoft s'oriente alors vers des missions de promotion du logiciel libre auprès du grand public, et veut devenir un réseau de partage de connaissances. Deux autres axes de campagnes sont également abordés : la culture libre et les services libres. Des projets comme la *Framakey*, une clé USB contenant une sélection de logiciels libres à utiliser directement sous Windows, Linux ou MacOS X sont lancés à ce moment.

C'est en 2006 que Framasoft publie son premier Framabook, « Utilisez Thunderbird 1.5 ! » sous la licence Creative Commons. Au départ constituée unique-

ment de manuels, la collection s'étoffe quelques années plus tard avec des romans et des bandes-dessinées. C'est aujourd'hui une vingtaine de Framabooks tous sous licence libre (Creative Commons, GNU FDL, Licence Art Libre, etc) qui constituent cette collection.

Les actions de l'association, des tribunes et des traductions de prises de position en langue étrangères sont publiées dans le Framablog, qui compte aujourd'hui près de 2000 articles. Des partenariats avec l'Éducation Nationale aboutissent en 2009 et 2010 à la création d'un Framadvd de ressources pédagogiques.

À compter de l'année 2011, Framasoft commence à sortir des services web comme Framapad (un éditeur collaboratif), Framadate (un outil pour planifier des événements via sondage) ou encore Framacalc (un tableur en ligne).

À l'automne 2014 est annoncée la campagne **Dégooglisons Internet** qui a pour but de montrer au public qu'ils ne sont pas condamnés à utiliser des services qui ne respectent pas leur vie privée (plus de détails-ci dessous 1.2.2).

Aujourd'hui, Framasoft compte une trentaine de membres dont cinq salariés et deux stagiaires. Elle est une des associations francophones les plus en pointe sur le sujet des logiciels libres et du respect de la vie privée.

Côté services, on compte aujourd'hui plus d'un demi-million de visites sur l'ensemble des sites du réseau chaque mois, un nouveau sondage est créé environ chaque minute sur Framadate et 100 000 pads sont actuellement actifs sur Framapad. Il y a aujourd'hui une vingtaine de livres édités par Framasoft. Enfin, grâce à la répartition géographique de ses membres, Framasoft peut être présent sur une énorme partie des événements liés au logiciel libre dans le monde francophone.

L'association possède un budget de l'ordre de 200 000€ financés à 90% par les dons, dont la grande majeure partie provient de particuliers. Le reste est issu de la vente de produits dérivés et de rares prestations techniques.

Il est important de noter que Framasoft n'ambitionne pas de devenir le prochain géant du web. L'association reste à but non lucratif et a de plus pour volonté de décentraliser ses activités au sein de structures locales qui seraient des « hébergeurs de services de proximité ».

L'association ayant comme vocation de sensibiliser le grand public aux valeurs du logiciel libre, elle est très souvent invitée à s'exprimer lors de manifestations en rapport avec le sujet, comme celle à laquelle j'ai pu participer à Nevers le week-end du 24 au 26 Juin.

1.2.2 Dégooglisons Internet

En octobre 2014, la campagne ***Dégooglisons* Internet** est lancée par Framasoft afin de montrer au public que des solutions à base de logiciels libres sont possibles face aux services proposés par les *géants du net* (aussi nommés **GAFAM** pour **G**oogle, **A**mazon, **F**acebook, **A**pple et **M**icrosoft, mais il y en existe une multitude d'autres).

FIGURE 1.3 – Visuel de la campagne Dégooglisons

En plus de ceux proposés avant le début de la campagne (Framapad, Framadate, ...), Framasoft a ouvert une vingtaine de services en ligne et un nombre aussi important doit s'ajouter d'ici la fin de la campagne en 2018 (cf. Annexe 9.2).

Framasoft utilise donc des applications web libres existantes ou développe ses propres alternatives pour proposer à des personnes n'ayant pas forcément de compétences techniques des services viables. Ainsi, l'association a fait développer des solutions spécialement pour proposer Framapic (hébergement d'images : alternative à Imgur, Flickr), Framalink (raccourcisseur d'url : alternative à bit.ly) et Framadrop (service de transfert de fichiers : alternative à WeTransfer). De plus, elle a lancé en juin 2014 une campagne de financement participatif spécialement pour faire ajouter de nouvelles fonctionnalités au service Framapad, sous la forme d'un plugin.

1.3 Site de Locaux Motiv'

Locaux Motiv' est un tiers-lieu associatif ouvert en septembre 2011 dans le quartier de la Guillotière à Lyon. Cet espace de co-working est géré en autogestion par une association fondée en 2010 autour d'autres associations du quartier. Locaux Motiv' compte aujourd'hui une vingtaine de structures - associations et entreprises - résidentes et usagères du lieu. Framasoft a rejoint Locaux Motiv' en xxx.

Chapitre 2

Sujet du stage

2.1 Présentation du sujet

La campagne Dégooglisons Internet annonçait la sortie d'un service Framagenda en 2016. Ce service proposé par Framasoft se devait d'être une vraie alternative au service principal sur le web, Google Agenda. Elle devait essayer d'avoir un nombre de fonctionnalités équivalent ainsi qu'une apparence moderne et utilisable par le grand public.

Ainsi, des applications plus anciennes qui possédaient des fonctionnalités plus avancées ont été écartées car l'interface était non intuitive et la mauvaise qualité du code impliquait d'apporter une dette technique assez significative.

L'application Calendrier d'ownCloud permettait déjà les fonctionnalités essentielles suivantes :

Gérer ses calendriers

Lors de la création d'un calendrier l'utilisateur y associe un nom et une couleur. Ces propriétés sont éditables et un calendrier peut-être supprimé. Il est possible d'afficher tous les événements associés à un calendrier ou bien les masquer d'un simple clic.

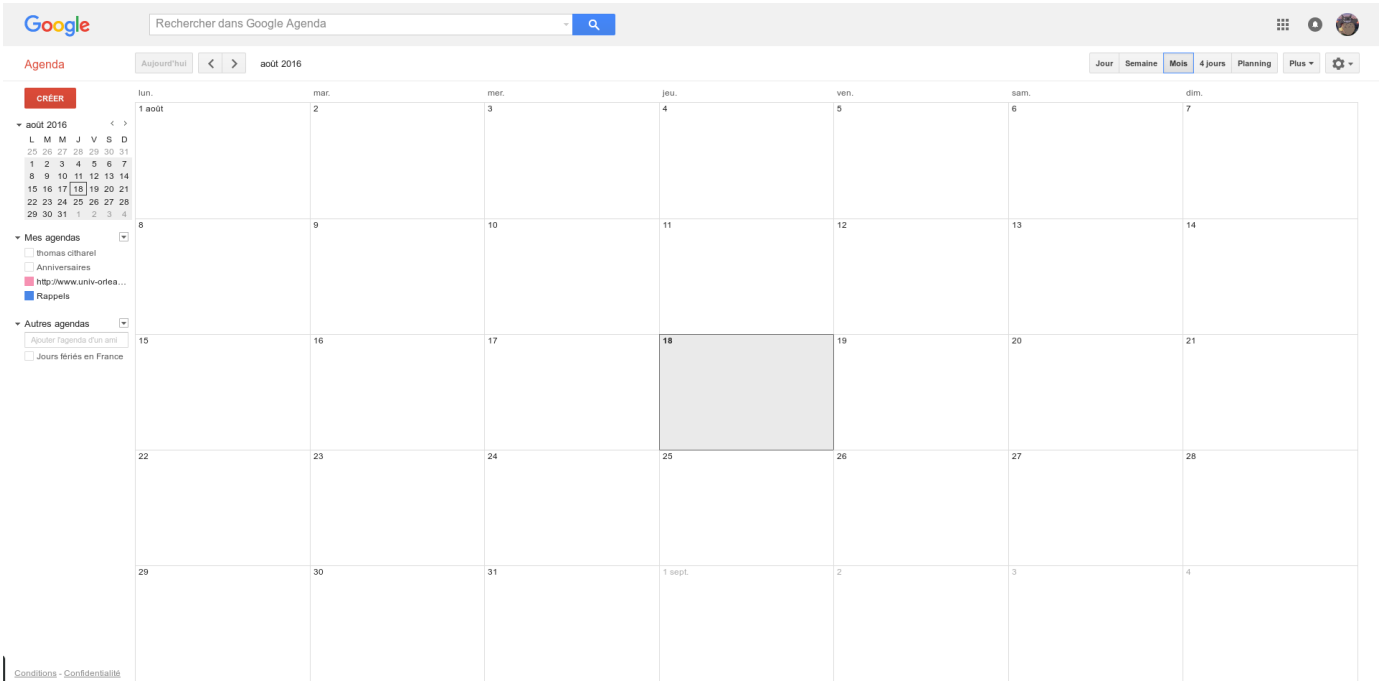

FIGURE 2.1 – Interface actuelle de Google Agenda

Gérer des événements

Les fonctionnalités de base de création, d'édition et de suppression sont présentes. Quelques propriétés simples liées à un événement sont ses dates de début et de fin, sa localisation et évidemment sa dénomination.

Il existe également des fonctionnalités plus avancées, comme pouvoir définir différentes occurrences de l'événement. Enfin, lorsque l'on précise l'adresse de courriel des participants, ils reçoivent tous automatiquement un courriel avec en pièce jointe le fichier de l'événement.

Visualisation des événements

Les événements sont visibles dans la partie centrale de l'interface. Cette vue peut permettre d'afficher le calendrier sous forme d'un jour ou d'une semaine (comme un agenda), ou bien tout le mois en entier. Un raccourci permet d'accéder directement aux événements pour aujourd'hui.

Partage de calendriers

Un utilisateur peut partager un calendrier (en lecture seule ou en écriture) avec une personne ou un groupe de personnes sur la même instance.

Import de fichiers calendriers

Il est possible d'importer des événements d'un fichier *.ics* dans un des calendriers de l'utilisateur, ou d'en créer un nouveau pour l'occasion.

Synchronisation avec des clients

L'application donne accès aux informations nécessaires pour synchroniser son calendrier avec un client par exemple sur son téléphone.

2.2 Objectifs du stage

La tâche principale était de sortir un service comme un produit fini, c'est à dire qu'il fallait mettre ajouter les fonctionnalités suivantes, mais également corriger quelques bugs bloquants de manière à ce que le code soit prêt pour l'ouverture du service.

Permettre de rendre un calendrier public et l'afficher publiquement

La première tâche est liée premièrement à l'implémentation du protocole de publication d'un calendrier à travers un plugin spécialisé dans le cœur d'ownCloud. D'autre part, il faut également une interface utilisateur dans l'application Calendrier qui permette de publier et dé-publier un calendrier. Enfin, il était nécessaire de proposer une vue « publique » accessible à tous sans être enregistré pour accéder au calendrier publié.

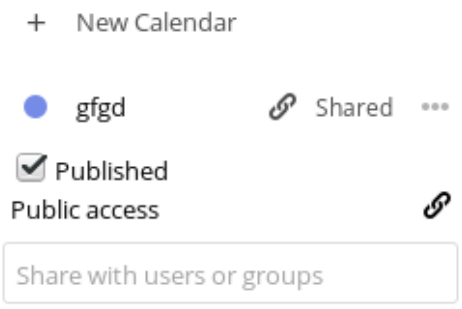

FIGURE 2.3 – Publication d'un calendrier

FIGURE 2.2 – Vue publique d'un calendrier

Permettre de s'abonner à des calendriers

NOTE : CHANGER LA CAPTURE D'ÉCRAN QUI N'EST PAS À JOUR

La seconde tâche consistait à pourvoir une fonctionnalité d'abonnement à des calendriers issus d'une source publique sur internet dans l'application Calendrier d'ownCloud. En effet, l'implémentation des abonnements selon le standard CalDAV (cf Annexe 9.1) était déjà réalisée dans le cœur d'ownCloud, il *suffisait* de réaliser les bons appels aux API et de gérer les fichiers distants du côté de l'application Calendrier.

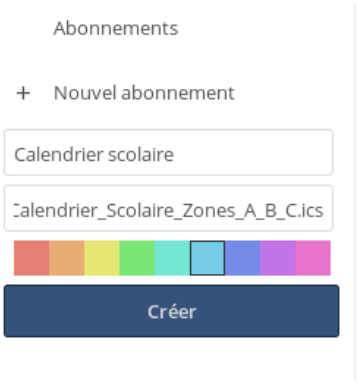

FIGURE 2.4 – Formulaire pour s'abonner à un calendrier public

Chapitre 3

Analyse de l'existant

Le fonctionnement des calendriers dans ownCloud est le suivant. Le serveur s'appuie sur la librairie SabreDAV en l'adaptant pour son utilisation particulière dans un module propre. Ce module prend en charge les requêtes suivant les standards de synchronisation de calendriers et sert les données aux clients.

L'application Calendrier fonctionne exactement comme un client de synchronisation de calendriers standard. Elle se base sur la librairie JavaScript FullCalendar et utilise le *framework* AngularJS.

3.1 Historic

FAIRE QUELQUE CHOSE AVEC CETTE PARTIE

La première semaine de mon stage a été dédiée à l'étude du fonctionnement du cœur d'ownCloud et de son intégration avec la librairie SabreDAV. J'ai dû également rechercher beaucoup d'informations sur les standards (ainsi que ce qui n'est pas standardisé!) car la synchronisation devait fonctionner avec un maximum de clients populaires.

D'autre part, j'ai également dû lire et assimiler beaucoup du code de l'application Calendrier, car elle est développée à l'aide du Framework AngularJS qui était une technologie que je ne connaissais pas particulièrement.

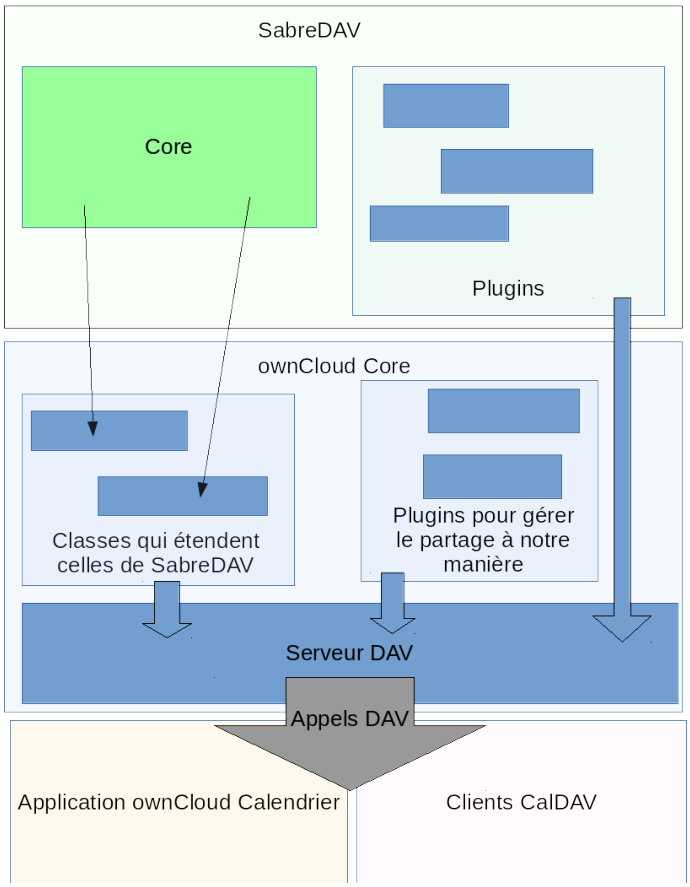

FIGURE 3.1 – Fonctionnement des calendriers dans ownCloud

3.2 ownCloud

ownCloud est une application web libre écrite dans le langage de programmation PHP. Elle a été initialement créée en 2008 comme une alternative au service Dropbox par Frank Karlitschek. L'application est sous double licence : la **licence AGPLv3** et une licence propriétaire qui s'applique à l'édition pour entreprises. ownCloud est développé par l'entreprise allemande ownCloud GmbH.

Elle consiste en un cœur (dénommé par la suite comme le *core*) qui s'occupe essentiellement de la gestion des utilisateurs (droits, groupes) et des fichiers. Les fichiers peuvent être partagés avec des utilisateurs du même serveur ou de tout utilisateur étant sur le réseau de la fédération.

ownCloud propose des API aux applications tierces afin d'obtenir les informations nécessaires sur les fichiers et les utilisateurs. Des applications comme une application de gestion de calendrier et de contacts, de client de messagerie électronique, de prise de notes ou encore de lecteur de flux RSS peuvent ainsi être intégrées à ownCloud. Ces applications sont installables par un administrateur directement à partir d'ownCloud d'un simple clic.

FIGURE 3.2 – Logo de l'application ownCloud

ownCloud peut être installé sur tout hébergement mutualisé ou serveur privé qui propose au moins une version de PHP supérieure ou égale à la version 5.4, et le site web propose également des paquets installables automatiquement sur les principales distributions à travers leurs dépôts. De plus, il est proposé à l'administrateur le choix en termes de système de bases de données entre *SQLite*, *MariaDB / MySQL* et *PostgreSQL*.

Quelques mois avant que mon stage débute, ownCloud a sorti une version 9 qui a transféré le module *DAV* (serveur responsable du calendrier et des contacts) dans le *core* et la logique de l'application Calendrier a été refaite quasiment intégralement en JavaScript à l'aide du framework *AngularJS*.

D'autre part, une partie des salariés travaillant chez ownCloud GmbH - dont le fondateur - n'étaient pas satisfaits des décisions de l'entreprise, et ont décidé de fonder leur propre société en juin 2016. Ils ont effectué un *fork* de l'application sous le nom de NextCloud. Une des différences majeures avec ownCloud est que NextCloud est licenciée uniquement avec la licence AGPL, même pour l'édition Entreprise.

Cette scission n'a heureusement pas eu trop de conséquences car il s'est avéré que les deux projets continuaient à communiquer et à échanger du code.

Chapitre 4

Développement

4.1 Organisation du travail

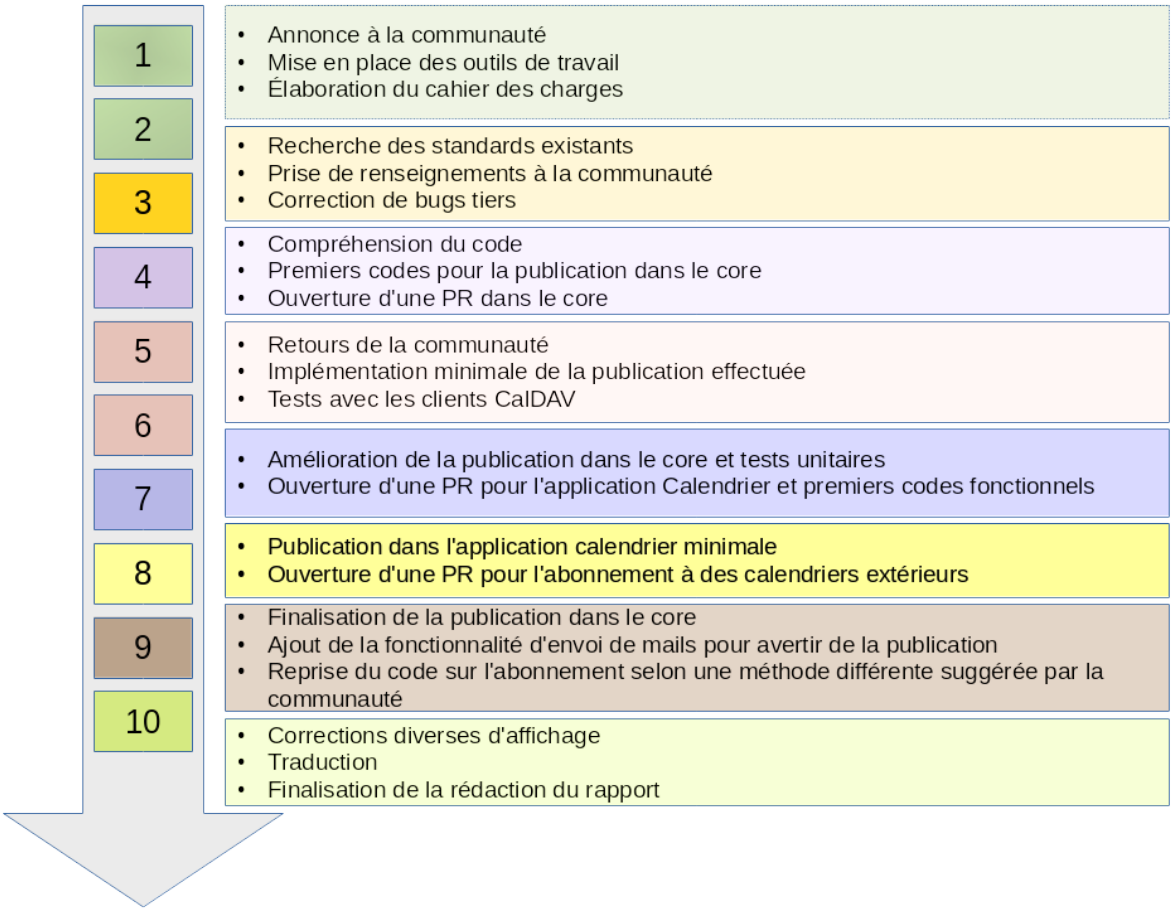

FIGURE 4.1 – Schéma du déroulement du stage

4.2 Environnement technique

Mes outils de développement au quotidien se composaient surtout de l'éditeur de texte Atom couplé à une multitude d'extensions, et des navigateurs Firefox et Chromium. L'application est servie à l'aide du serveur intégré à PHP. Pour le développement, l'application utilisait comme base de données SQLite pour sa simplicité d'utilisation mais lors de changements dans la base de données il a fallu également tester la compatibilité des modifications avec MySQL et PostgreSQL. En outre, je me suis parfois servi de l'application *Advanced REST Client* pour pouvoir communiquer plus facilement avec l'API en créant des requêtes en visualisant leur réponse.

ownCloud utilise git pour la gestion de leur code à travers la forge logicielle Github, qu'ils utilisent également pour le système de tickets. Je me suis donc abonné par courrier électronique à certaines discussions dès le départ afin de suivre certains travaux en cours.

Pour ce qui est de la mise en place de Framagenda, j'ai d'abord eu accès à une machine virtuelle afin d'effectuer des tests ne pouvant être effectués sur ma machine locale, comme l'envoi de courriers électroniques à partir d'un serveur. D'autre part, une première installation publiquement accessible avec un nom de domaine m'a aussi permis de demander à des personnes au sein de l'association de faire des tests fonctionnels et d'avoir des premiers retours.

Lors de la mise en production, une machine virtuelle a été dédiée à Framagenda, et l'administrateur système s'est occupé de la configuration de sécurité de base et de la mise en place d'un certificat HTTPS grâce au projet *Let's Encrypt*, tandis que j'ai mis en place l'application en elle-même ainsi que configuré correctement le serveur web *nginx* et le serveur de base de données *MariaDB*.

Le code de l'application installé sur les machines provenait directement de mon propre dépôt git sur la forge logicielle proposée par l'association - Framagit - et dont le code correspondait à peu près à celui proposé sur Github, les modifications particulières applicables pour Framagenda en plus.

Enfin, là où j'avais créé une branche dans git correspondant à chaque fonctionnalité développée puis publiée sur Github, je m'efforçais d'avoir une branche « produit fini » de mon côté dans laquelle les différentes branches étaient fusionnées sur Framagit.

4.3 Fonctionnalités

4.3.1 Publication d'un calendrier

Pour pouvoir publier un calendrier il fallait donc plusieurs choses : d'abord introduire dans l'application *core* la gestion de cette propriété, c'est à dire lorsqu'un client fait une requête pour interroger la ressource calendrier sur ses propriétés, il fournir celle-ci en répondant le cas échéant si le calendrier est publié. Ensuite, il nous fallait introduire un mécanisme pour pouvoir répondre à un appel d'API pour publier ou dé-publier le calendrier. Enfin, il fallait que le calendrier publié soit mis dans une collection de ressources à part, accessibles à tout le monde.

Je n'ai pas compris ce dernier aspect tout de suite. Je pensais qu'il suffisait de garder la ressource existante et de forcer l'authentification à être désactivée dans ce cas précis.

Tous ces mécanismes ont été réalisés de la même manière (avec les mêmes noms de propriétés) selon laquelle Apple utilise la fonctionnalité de publication dans son application Calendar, et le serveur de Calendrier ownCloud devrait donc être compatible avec ce client.

Il a d'abord fallu changer un peu le *backend*, c'est à dire la manière dont l'application va traiter les données pour les stocker dans la base de données pour sauvegarder ce nouveau statut du calendrier.

J'ai profité du fait que les partages de calendrier entre utilisateurs étaient stockés dans une table *dav_shares* où la colonne *access* comprenait un chiffre qui désignait le niveau d'accès au fichier.

J'ai donc rajouté un niveau d'accès qui correspondait à un partage public, ainsi qu'une nouvelle colonne pour stocker l'URL une fois qu'elle serait publiée.

En effet, l'URL publique se constitue ainsi :

`https://mondomaine.tld/index.php/apps/calendar/public-calendar/<empreinte>`

qui va chercher les données du calendrier à l'adresse

`https://mondomaine.tld/remote.php/public-calendars/<empreinte>`

Comme l'empreinte d'un calendrier est calculée à partir des informations de base du calendrier (comme son identifiant unique), si je n'avais pas cette colonne supplémentaire pour stocker l'adresse finale de la ressource, il faudrait itérer sur tous les calendriers partagés sur le serveur et vérifier que leur empreinte respective corresponde bien à celui voulu dans l'URL publique. Alors que grâce à l'introduction de cette nouvelle colonne je peux savoir directement s'il existe un calendrier pour cette URL publique et le cas échéant récupérer ses informations.

Ensuite, il a fallu écrire un plugin pour SabreDAV pour générer ces propriétés proprement et l'intégrer dans le serveur. Une classe s'occupant de la sérialisation - c'est à dire de la conversion des données dans un format texte structuré, ici XML - a également été nécessaire.

On notera que la propriété transmise - l'URL du calendrier publié - est connue dès la création du calendrier car elle est calculée avec une fonction de hachage de l'identifiant du calendrier et d'un sel (une chaîne de caractères aléatoire). Elle est transmise par la propriété *pre-publish-url* avant la publication et ensuite par la propriété *publish-url* dès lors que le calendrier est publié.

Pour prendre en charge la publication et la dépublication, notre plugin a dû intercepter les requêtes POST qui contenaient une propriété particulière. En l'occurence, si la requête s'effectuait à l'adresse d'une ressource calendrier et qu'elle contenait l'élément XML *publish-calendar* ou *unpublish-calendar*, nous traitons la requête dans un sens ou dans l'autre et empêchons le traitement de cette dernière par d'autres plugins.

Enfin, pour qu'un calendrier soit disponible publiquement, il a fallu créer un type spécifique de collection de ressources sur lesquelles un utilisateur non-authentifié peut lire les ressources.

De ce côté le serveur *core* était prêt. Dans l'application Calendrier il fallait

proposer une extension de l'interface pour permettre à l'utilisateur possédant un calendrier de pouvoir le publier et le dépublier. Ensuite, il fallait fournir à l'utilisateur le lien de l'interface web publique, qui lui fournir également d'autres informations comme les adresses de synchronisation ou le lien de téléchargement du fichier calendrier *.ics*.

Cette interface web publique devait par définition se passer d'authentification car accessible même aux utilisateurs non-inscrits. Elle devait également ne pas montrer l'intégration de l'application au système ownCloud (pour une meilleure intégration à l'intérieur de sites tiers), ne montrant que le calendrier et des informations sur celui-ci.

J'ai cherché un moment comment contourner l'intégration à ownCloud avant de me rendre compte à l'aide du code d'autres applications existantes qu'il s'agissait d'un simple paramètre à passer au *template*, hélas non documenté.

Enfin, une fonctionnalité que j'ai ajoutée plus tard dans le développement est l'envoi de messages électroniques lorsque le calendrier est publié pour notifier des personnes de sa publication. Je me suis basé sur un *template* existant et libre pour l'apparence du mail.

4.3.2 Les abonnements

Les standards prévoient la possibilité de s'abonner (ou souscrire) à des sources de calendriers externes (publiques sur internet). Cette fonctionnalité était déjà implémentée dans le *core*, mais l'interface de l'application Calendrier ne permettait pas de le faire.

J'ai donc rajouté cette fonctionnalité en étendant le modèle AngularJS correspondant à un calendrier pour en faire un modèle d'un abonnement, et rajouté dans l'interface un champ de texte pour ajouter l'adresse de cette nouvelle souscription.

Afin d'éviter de récupérer le calendrier distant chaque fois que l'on accède à l'application, le calendrier est mis en cache dans le navigateur (avec la technologie *LocalStorage*).

4.3.3 Les tests unitaires

Les tests unitaires n'étaient pas forcément requis dans le cas de mon développement, mais ils étaient obligatoires dans le cas où je voulais remonter le code *upstream* (en amont) à la communauté. ownCloud, NextCloud et la plupart des applications développent en utilisant la technique d'intégration continue, c'est-à-dire que chaque nouveau morceau de code apporté sur Github lançait l'exécution de la batterie de tests au complet.

Chaque nouvelle méthode de mon code devait par conséquent s'accompagner d'un test qui était préalablement vérifié en local. Si une partie du code que j'avais apportée n'était pas testée, alors le *code coverage*, la proportion de code couvert par les tests par rapport au code complet baissait et cela m'était automatiquement signalé. Les tests unitaires étaient réalisés à l'aide du *framework* de test *PHPUnit* pour le code PHP et de la librairie *Karma* pour le JavaScript de l'application Calendrier.

4.3.4 Développements annexes

TODO : PASSER SUR TOUS LES BUGS CORRIGÉS PAR MOI VOIR SI Y A RIEN À MENTIONNER

Mon stage consistait avant tout à implémenter de nouvelles fonctionnalités dans l'application Calendrier existante, mais il fallait également que celle-ci soit prête à être utilisée dans Framagenda sans *bugs* flagrants. Ainsi, j'ai corrigé quelques problèmes tiers dans le *core* ou l'application Calendrier. On peut citer par exemple les propriétés d'un abonnement à un calendrier qui n'étaient pas proprement sauvegardées lors d'une requête de mise à jour ou encore les événements d'un calendrier qui s'effaçaient si jamais l'éditeur était ouvert puis l'action d'édition annulée.

De plus, afin d'avoir un style de code commun pour tous les contributeurs dans les feuilles de style CSS de l'application Calendrier, j'ai proposé d'utiliser un outil de *lint* CSS et corrigé toutes les feuilles de style en accord.

Cet outil signale lorsque le code écrit ne suit pas les règles définies, par exemple l'espacement entre les blocs CSS ou l'écriture de règles désuètes.

J'ai aussi ajouté les vérifications faites par cet outil au système d'intégration continue de sorte que si quelqu'un propose une *pull request* qui ne satisfasse pas

les règles, elle soit marquée comme erronée.

4.4 Proposition d'améliorations

J'ai pu remarquer certains manques par rapport à Google Agenda, comme une vue « planning » qui affiche les événement sous forme de liste. Cette vue pourrait être ajoutée en rendant l'application Calendrier compatible avec la version 3 de la librairie FullCalendar (Figure 4.2).

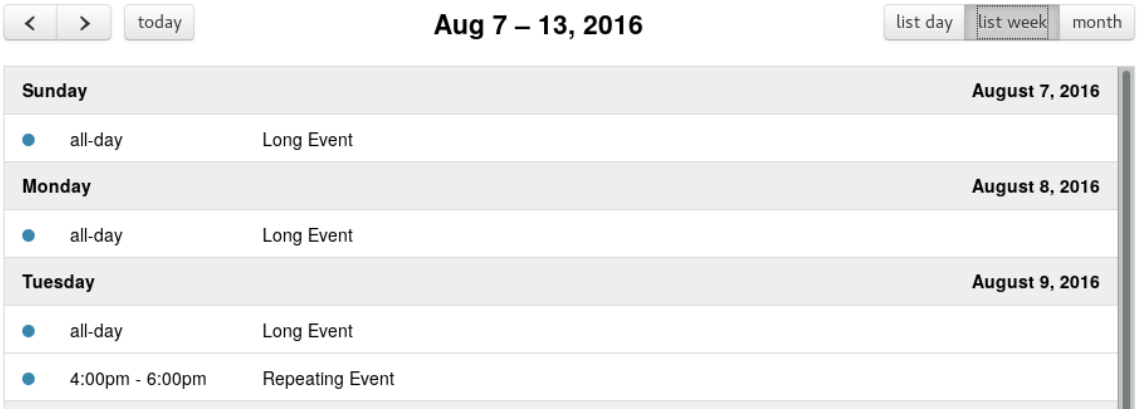

FIGURE 4.2 – Vue « planning » dans FullCalendar 3

Une autre amélioration qui pourrait être développée sans trop d'effort serait le rafraîchissement automatique du calendrier. Cette fonctionnalité serait fort utile dans le cas où un écran affiche les événements d'une organisation toute la journée.

Enfin, il serait intéressant (mais également plus complexe) d'intégrer correctement les événements issus du calendrier avec le système de notifications d'ownCloud, par exemple lors qu'un utilisateur à accès à un nouveau partage, ou bien qu'il est invité à un événement.

Bilan du stage

L'objectif de ce stage était de réaliser et mettre en place une application de gestion d'agendas. Cette application devait être intuitive et accessible à tout le monde, tout en ayant des fonctionnalités à peu près équivalentes avec Google Agenda.

5.1 Bilan technique

Pour ce qui est du niveau technique, je note avoir réussi à comprendre progressivement le fonctionnement des parties qui m'intéressaient de l'application ownCloud en un temps minimal malgré la complexité du logiciel. J'ai ensuite pu tirer avantage de mon expérience avec PHP par les projets scolaires et extrascolaires pour effectuer les modifications nécessaires sans trop de problèmes.

En revanche, j'ai eu plus de mal à comprendre les paradigmes utilisés dans la librairie SabreDAV et les standards de gestion de calendrier, d'une part à cause de la documentation plutôt moyenne ce qui m'a amené à faire beaucoup de tests par moi-même pour comprendre leur fonctionnement.

La partie de l'application en AngularJS m'a posé davantage de soucis. C'était en effet une technologie à laquelle je n'avais jamais touché et qui nécessite une certaine abstraction. Heureusement, il existe beaucoup de ressources sur AngularJS et sa courbe d'apprentissage permet de pouvoir toucher au code rapidement.

5.2 Bilan humain

La particularité de mon stage était que je n'avais pas de contact physique avec d'autres développeurs d'ownCloud. Cela n'a pas été un problème car les communications sur les tickets Github et la messagerie instantanée à travers IRC m'ont permis de poser les questions nécessaires. Les échanges se sont faits intégralement en anglais.

Les développeurs avec qui j'ai été en contact ont été sympathiques et compréhensifs, en prenant parfois le temps de m'expliquer mes erreurs, ce qui m'a agréablement surpris.

Concernant l'association Framasoft, j'ai pu apprécier leur bonne volonté lorsque j'ai lancé un appel à tester l'application en l'état.

Webographie

Annexes

9.1 Les standards de gestion de calendrier

Un serveur de calendrier doit suivre les standards **CalDAV**, qui est une extension du protocole **WebDAV** appliquée aux fichiers **iCalendar** (*.ics*). WebDAV est un protocole de gestion de fichiers sur serveurs distants conçu comme une extension du standard HTTP. Il a commencé par être standardisé par l'IETF en Février 1999.

iCalendar, pour sa part est un standard encore plus vieux (1998) qui décrit le format d'un fichier agenda. Ce format est beaucoup décrié aujourd'hui car étant assez lourd et complexe à manipuler. Heureusement, il existe de nombreuses librairies nous évitant de traiter directement ce format.

Dans notre cas, le serveur est - heureusement - géré avec le concours d'une librairie du nom de **SabreDAV** qui prend en charge CalDAV, CardDAV (pour les contacts) et WebDAV. Cette librairie fonctionne avec un système de plugins nous permettant de l'adapter pour notre utilisation dans ownCloud.

Il est important de remarquer que si les protocoles et standards internet pour la gestion de calendrier sont définis et correctement documentés, leur implémentations dans les applications clientes ou serveur sont plus qu'hasardeuses et la liste des fonctionnalités supportées peut être assez différente d'une application à l'autre.

Ainsi, c'est la société Apple qui est à la pointe des derniers standards dans son application *iCal* (désormais nommée plus simplement *Calendar*). N'ayant pas de machine Apple, je n'ai pu profiter de ce logiciel pour mes tests.

Le standard WebDAV décrit un protocole pour gérer des ressources - dans notre cas des calendriers - comme un ensemble de méthodes qui étendent celles

définies par le standard HTTP. Ces méthodes sont utilisées par des requêtes et des réponses qui structurent les données à l'aide du format XML. Par exemple pour récuperer les propriétés d'une ressource on va utiliser la méthode PROPFIND et si on veut modifier les propriétés d'une ressource la méthode PROPPATCH.

```
<d:response>
 <d:href>/remote.php/dav/calendars/tcit/test/</d:href>
 <d:propstat>
  <d:prop>
   <d:displayname>test</d:displayname>
   <d:resourcetype>
    <d:collection/>
    <cal:calendar/>
   </d:resourcetype>
   <cal:calendar-description/>
   <cal:calendar-timezone/>
   <x1:calendar-order xmlns:x1="http://apple.com/ns/ical/">0</x1:calendar-order>
   <x1:calendar-color xmlns:x1="http://apple.com/ns/ical/">#e78074</x1:calendar-color>
   <cal:supported-calendar-component-set>
    <cal:comp name="VEVENT"/>
    <cal:comp name="VTODO"/>
   </cal:supported-calendar-component-set>
   <cs:publish-url>
    <d:href>https://framagenda.org/remote.php/dav/public-calendars/cc45fbcd080add220241807b
   </cs:publish-url>
   <cs:pre-publish-url>https://framagenda.org/remote.php/dav/public-calendars/cc45fbcd080ac
   <cs:allowed-sharing-modes>
    <cs:can-be-shared/>
    <cs:can-be-published/>
   </cs:allowed-sharing-modes>
   <d:acl>
    <d:ace>
     <d:principal>
      <d:href>/remote.php/dav/principals/users/tcit/</d:href>
```

Extrait d'une requête

Le standard WebDAV décrit également ce que sont les collections de ressources et comment les droits d'accès (nommés ACL) sont gérés. Par exemple, un utilisateur ne peut accéder qu'à ses fichiers ou à ceux que d'autres utilisateurs ont partagés avec lui.

Le standard XML définit les balises appartenant à un espace de noms. En l'occurence nous avons des balises représentant les propriétés de notre ressource, elle appartiennent ainsi à autant des espaces de noms différents qu'il existe de standards différents qui s'appliquent (IETF, DAV, CalendarServer c'est à dire Apple, et ownCloud pour nos propres propriétés).

9.2 Liste des services de Framasoft

	Service Framasoft	En alternative à	Description
2013	Framapad	Google Docs	Éditeur collaboratif
	Framadate	Doodle	Planification de rendez-vous et création de sondages
	Framacalc	Google Spreadsheet	Tableur collaboratif
	Framanews	Feedly	Lecteur de flux RSS
	Framavectoriel	Pixlr	Création d'images au format SVG
	Framindmap	Bubbl.us	Création de cartes mentales
2014	Début de la campagne Dégooglisons Internet		
	Framasphère	Facebook	Réseau social
	Framabag	Pocket	Sauvegarde et lecture différée d'articles
2015	Frama.link	bit.ly	Réducteur d'URL
	Framadrive	Dropbox	Sauvegarde de fichiers
	Framagit	Github	Hébergement de code
	Framacarte	Google Maps	Cartographie
	Framabee	Google Search	Moteur de recherche
	Framapic	Imgur	Envoi d'images
	Framabin	Pastebin	Notes anonymes
	Framaboard	Trello	Gestion de projets
	Framadrop	Wetransfer	Transfert de fichiers
2016	Framateam	Slack	Communication collaborative
	Framavox	Shrtct	Prise de décisions
	Framaforms	Google Forms	Questionnaires en ligne
	Framapétition	Avaaz	Création de pétitions
	Framatalk	Skype	Visioconférence
	Framalistes	Google Groupes	Listes de diffusion
	Framagenda	Google Agenda	Agenda partagé

Les éléments grisés ne sont pas encore sortis cette année.