

Contents

Introduction	iii
1 Présentation de la Plateforme Stample	1
1.1 Présentation de Stample	1
1.1.1 Stample en quelques mots	1
1.1.2 Effectif	1
1.1.3 Levée de fonds	2
1.1.4 Besoins	2
1.1.5 Concurrency	3
1.2 Conclusion	3
2 Environnement du travail	4
2.1 Introduction	4
2.2 Working tools	4
2.2.1 Environnement de Développement	4
2.2.2 Langages de Programmation	5
2.2.3 Plugins	5
2.2.4 Outils de développement	5
2.3 Aspects technique de Stample	5
2.3.1 Architecture	5
2.3.2 Description de l'architecture globale	6
2.3.3 Configuration de départ	6
2.4 Conclusion	7
3 L'authentification sur Stample	8
3.1 Introduction	8
3.1.1 Configuration du projet Stample	8
3.2 Schéma Model View Controller (MVC)	10
3.2.1 Cycle de vie d'une requête	11
3.3 Présentation du contexte Business	13
3.3.1 Login V0	13
3.3.2 Critiques	13
3.3.3 Fonctionnalités manquantes	13

3.4	Première approche	14
3.4.1	Etude du modèle	14
3.4.2	Première Solution	14
3.5	Remise à Zéro	14
3.5.1	Phase préliminaire	14
3.5.2	Implémentation & Intégration	15
3.5.3	Keep user Logged In SecureSocial :Authenticator Cookie	15
3.5.4	Customisation des templates	15
3.5.5	Partie Administrateur sur Stample	15
3.6	Conclusion	16
4	Notification& Commentaire	
	17	
4.1	Introdutction	17
4.2	Notifications	17
4.3	Commentaires	18
4.4	Conclusion	20
5	Contexte des Réseaux sociaux Centralisés et décentralisés	
	21	
5.1	Introduction	21
5.2	PARTIE 1 : Etude des réseaux sociaux existant	23
5.2.1	Les débuts décentralisés d'internet	23
5.2.2	Motivation	24
5.3	PARTIE 2 : A propos des réseaux sociales décentralisé	25
5.3.1	Le décentralisé sur le Net	25
5.3.2	Réseautage social décentralisée en ligne	27
5.4	Alternatives	29
6	Conclusion	30
	Critiques	32
	Annexe	33

Introduction

Au cours de ma première année de master Web Intelligence à l'université de Jean Monnet Saint-Etienne, nous devions effectuer un stage d'une durée minimale de trois mois à compter du 18 mars.

Ce rapport présente le travail que j'ai effectué lors de mon stage au sein du Stample.

Ce stage a été une bonne opportunité, je me suis familiarisé avec l'environnement technique, un ensemble d'application Scala/PlayFrameWork et GuitHub. J'ai découvert le coworkinge et j'ai beaucoup appris sur philosophie du web.

Stample s'est avéré un projet intéressant et très enrichissant pour une expérience professionnelle. Grâce à ce stage, j'ai travaillé sur des différentes fonctionnalités d'un réseau social.

Le but de ce rapport n'est pas de faire uniquement une présentation exhaustive de tous les aspects techniques que j'ai pu apprendre ou approfondir de manière synthétique et claire, mais aussi de faire un tour d'horizon des aspects techniques et humains auxquels j'ai été confrontés.

Il apparait cohérent de commencer mon rapport de stage par une présentation de Stample, ensuite de présenter le contexte technique. Il s'agira après de décrire l'intégration de secureSocial pour le Login et l'amélioration de l'interface admin, puis de développer le système de notifications et commentaires pour la plateforme, finalement je conclurais mon travail en ajoutant des perspectives.

Chapter 1

Présentation de la Plateforme Stample

1.1 Présentation de Stample

Stample, plateforme de réseau social, redonne à chacun le contrôle exclusif de ses informations et améliore l'ergonomie de l'apprentissage, du travail individuel et de la collaboration.

1.1.1 Stample en quelques mots

- **Sample**, attraper facilement du contenu du web avec notre puissant bookmarklet. Drag & drop des fichiers depuis votre ordinateur, créer des notes et articles de n'importe quel appareil.
- **Staple**, Gracieusement organiser votre bibliothèque de contenu personnel grâce à notre architecture d'arborescence similaire à un système de fichier. Options visuelles conçues avec soin des informations dont vous avez besoin et accessibles en un coup d'oeil.
- **Stamp**, résumer, mettre en évidence et annoter tous vos contenus. Partager parfaitement du contenu et des métadonnées (des catégories) avec certains membres de votre réseau.

1.1.2 Effectif

L'équipe Stample est conçu par les Co-founders : *Edward Silhol, Henry Story et Sacha Roger* une courte présentation disponible sur le site de la plateforme¹.

Amélie Medem chercheur, Phd en computer science. Elle travaille en temps complet sur le FrontEnd de Stample.

¹<https://stample.co/>

Sébastien Lober développeur ingénieur Backend, il travaille actuellement chez Zenika² et en temps partiel pour Stample.

Jonathan Winandy, développeur ingénieur Backend, il travaille chez Viadeo³ et en temps partiel sur Stample Plateforme.

Matthieu gayon, développeur Frontend il travaille en temps partiel sur la Plateforme.

Moncef Ben Rajeb, développeur stagiaire Backend.

Ce sont les développeurs actuelle de Stample mais il y avait d'autres telque *Francesco Piccoli*, développeur Backend qui a travaillé sur ce projet...

1.1.3 Levée de fonds

D'après Steve Blank⁴ « Il y'a une différence fondamentale entre une entreprise établie et une Startup : une Startup cherche un business model alors qu'une entreprise, elle, a déjà un business model »

Suite à des réunions avec des inverstisseurs "friends and family", les trois fondateurs ont réussi à lever 200 mille euro. Leur objectif maintenant est d'avoir une première démo pour continuer qui sera probablement en mois de septembre.

1.1.4 Besoins

Il y a un besoin croissant d'un outil de partage sécurisé des connaissances numérique. Les gens perdent des heures chaque semaine en raison de la complexité croissante de leur vie numérique:

- Filtrage des e-mails et notifications non désirés,
- Créations et mises à jour de leurs profils sur de trop nombreux services isolés les uns des autres,
- Récupération de mots de passe perdus ou volés...
- L'information pertinente est de plus en plus difficile à extraire du déluge de données,
- La grande segmentation des outils rend l'organisation et la collaboration frustrante.
- Améliorer l'ergonomie de l'apprentissage et du travail par la contextualisation avancée de l'information,

²<http://www.zenika.com/>

³<http://us.viadeo.com/>

⁴<http://steveblank.com/>

1.1.5 Concurrency

- Stockage en ligne: Dropbox, Box, Google Drive, etc...
- Réseaux sociaux: Facebook, Tumblr, Pinterest, Instagram, etc...
- Aggrégateurs et plateformes de blogs: WordPress, Twitter, Reddit, Scoop.it, Paper.li, Flipboard, Zite, Jolicloud, etc...
- Réseaux sociaux professionnels: LinkedIn, Quora, Yammer, Podio, etc...
- Outils de note et d'organisation d'informations: Evernote, SpringPad, Clipboard, Pearltrees, Kippt, Google Keep, etc...
- Outils de partage des connaissances: Mendeley, Kno, Scribd, SlideShare, Issuu

1.2 Conclusion

Stample est conçu pour s'adapter aux gestes et aux usages quotidiens des gens. Les utilisateurs de la plateforme pourront chacun construire leur réseau de connaissances personnelles, en évaluant leurs sources et les contenus qu'ils partagent. Stample utilise la mécanique du jeu pour encourager les gens à partager des informations hautement qualifiées et contextualisés.

Chapter 2

Environnement du travail

2.1 Introduction

Certes, le succès ou l'échec d'un projet informatique dépend du choix des technologies utilisés. Ce choix dérive essentiellement des objectifs à attendre et des contraintes d'accompagnement qui doivent être prises en considération.

Dans cette partie, je vais présenter les différentes technologies envisageables par les architectes de Stampile et relatives à la développement du Backend la partie sur la quel j'ai travaillé et le FrontEnd. Vous trouverez plus de détails sur l'environnement du travail sur ma page¹ web.

2.2 Working tools

2.2.1 Environnement de Développement

- PlayFramework² 2.1.0
- JDK 7
- Maven3
- OS X 10.8.4
- SBT³ 0.12.4
- ElasticSearch⁴ 19.4.0
- MongoDB⁵ 2.2.x

¹<http://ideas2d.com/moncef/home.php>

²<http://www.playframework.com/>

³<http://www.scala-sbt.org/>

⁴<http://www.elasticsearch.org/>

⁵<http://www.mongodb.org/>

2.2.2 Langages de Programmation

- BackEnd : SCALA , JAVA, JSON
- FrontEnd : HTML5, JavaScript, JQuery, CSS3, JSON, Ajax.

2.2.3 Plugins

- BackEnd : SecureSocial, Salat,
- FrontEnd :Backbonejs

2.2.4 Outils de développement

- IDE : ItelliJ IDEA 12.1.14 Community Edition;
- Base de donnée : Mongodb NoSQL data base;
- outils de compilation automatique : sbt.

2.3 Aspects technique de Stample

2.3.1 Architecture

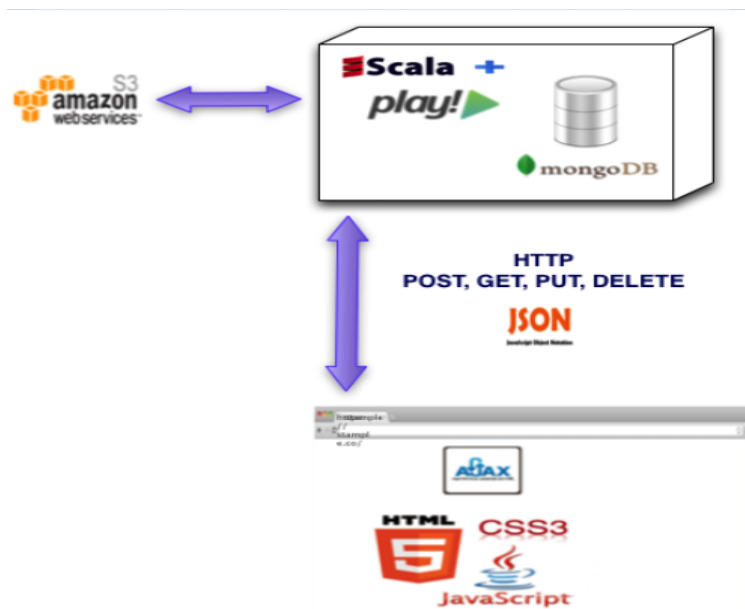


Figure 2.1: Architecture de la plateforme

2.3.2 Description de l'architecture globale

Les requêtes HTTP sont faites en JSON. Le JSON est désérialisé dans des objets métiers (Scala case class) pour être manipulés au sein de l'application.

Communication entre la base et l'API

On utilise le driver Casbah MongoDB en Scala et une librairie "Salat" qui permet une bonne intégration de Casbah dans l'application. Salat permet de sérialiser les objets métiers (case class) en BSON⁶, cette librairie propose un DAO (Data Access Object) générique un model companion qui expose les méthodes basique des requêtes Mongo (find(), findById(), search()...)

Communication entre L'API et ElasticSearch

L'indexation dans la base se fait manuellement après l'insertion, la mise à jour d'un document MongoDB dans la base de donnée. L'échec d'indexation est non bloquant car il est rattrapé par un Job.

ElasticSearch prend en entrée du JSON cela implique qu'il fonctionne particulièrement bien avec Mongo. De plus, utilise salat aussi pour produire le JSON d'un objet métier (case class). Donc nous avons le même document dans MongoDB et ElasticSearch. Cela permet de rendre un document similaire à Backbone pour la recherche. On utilise un client JAVA pour ElasticSearch.

SBT

Le build est réalisé par SBT, il faudrait mettre en place un Jenkins qui fera tourner les tests unitaires et embarqués à chaque push pour réparer les regressions et que nous en soyons alertés.

Les collections MongoDB

Sample structure les données sous forme de différents collections. Ils sont tous identifiés par un ObjectId, elle regroupe toutes les informations nécessaires sur les utilisateurs (Compte utilisateurs, Stamps ...).

2.3.3 Configuration de départ

Description de la configuration

Le projet contient de quatre répertoires séparés : *fixturedatabase* contenant les fichiers JSON et leur sérialisation binaire encodée BSON, *restore* inclut le build de l'application et les plugins, *sample-search* contenant les classes java

⁶<http://bsonspec.org/>

et la configuration nécessaire pour le lancement du projet et *sample-web* ce dernier contient les différents packages du code Front/back End.

Pour l'interaction avec le projet, j'ai installé les outils cités auparavant. Ensuite, j'ai configuré ma machine avec les paths nécessaires pour lancer sbt, play, mongo et elasticSearch. Puis, j'ai lancé le mvn dans le repertoire sample-search. Enfin il est indispensable de lancer play ou sbt et de générer les fichier idea, eclipse ou autre dans sample-web selon l'IDE envisagé. Le README.md du projet écrit avec le langage de balisages légers Markdown⁷ contient les détails nécessaires pour la configuration du projet.

⁷<http://fr.wikipedia.org/wiki/Markdown>

Gestion du travail en groupe

Les règles du travail en groupe sur le projet Git :

1. NE PAS travailler directement sur la branche master. Ce n'est que pour la production.
2. La branche de développement principale est "preProd". A partir de cette branche on commence une nouvelle branche.
3. Lors du développement sur une branche, merger régulièrement pre-Prod en elle, pour s'assurer d'être à jour avec le travail des autres développeurs.
4. Lorsque la branche est stable et a été testé par le chef de projet, il va tester, merger dans preProd, puis on peut supprimer cette branche.
5. Utiliser toujours camelCase pour nommer les branches et des noms intelligents!
6. Ajouter des explications intelligents lors de chaque commit.

2.4 Conclusion

Dans ce chapitre introductif, j'ai détaillé les outils qu'on utilise. Ensuite, j'ai présenté l'architecture globale de la plateforme et les différentes interactions. La prochaine section se focalise sur la solution développée pour le login sur Stample.

Chapter 3

L'authentification sur Stample

3.1 Introduction

Dans une première partie suite à la tâche "Login sur Stample" assignée dans *Trello*¹ avec Jonathan Winandy, j'ai commencé à lire, comprendre le code du backend Stample et à voir l'architecture du projet.

Problématique, Comment faire pour intégrer une couche horizontal sans tout casser coté Frontend ?

Au début, j'ai implémenté une solution basique, relativement fonctionnel mais non achevé suite aux complexités liées à la compatibilité avec le code existant. Ensuite, remise à zéro en appliquant les méthodes d'ingénierie nous avons réussi à finaliser la tâche en moins de temps.

3.1.1 Configuration du projet Stample

J'ai eu l'accès au code source du projet privé sur le compte Git d'Edward avec son autorisation. Au début, j'ai utilisé un outil windows pour la gestion de mes projets Git d'apprentissage. Ensuite, j'ai changé mon PC parce que les outils SBT, play, mongo.... ont des besoins importants en ressources (mémoire, CPU). Stample m'a prêté un MacBookPro.

Pour interagir avec Stample, j'ai consolidé mes connaissances dans le terminal-land (tmux², zsh³, bash, git et SBT).

¹<https://trello.com/>

²<http://tmux.sourceforge.net/>

³<https://github.com/robbyrussell/oh-my-zsh>

3.2 Schéma Model View Controller (MVC)

En effet, dans la plateforme Stample il s'agit des requêtes affectées à des contrôleurs à l'aide de règles de routage, ces derniers impliquent des services dans un package services et sont implémentés dans un sous package "impl" (implementation) puis de même pour les repository et des views qui contiennent les templates. C'est une architecture de play et généralement un design classique d'une application java que j'ai découvert.

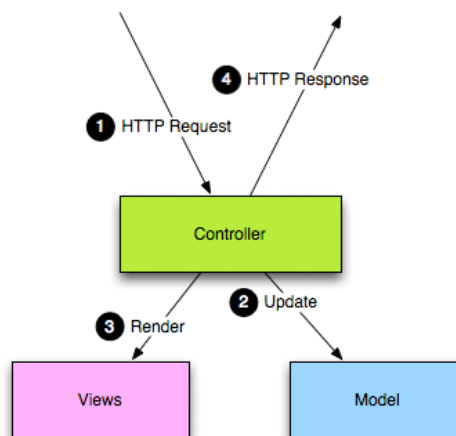


Figure 3.1: Diagramme MVC

- **Model** : Le modèle est la représentation spécifique au domaine de l'information sur laquelle l'application fonctionne. La logique de domaine ajoute «sens» aux données brutes (par exemple : le calcul des totaux, taxes de l'utilisateur, les frais d'expédition pour un panier ...). La plupart des applications utilisent un mécanisme de stockage persistant comme une base de données pour stocker des données. MVC ne mentionne pas spécifiquement la couche d'accès aux données, car il est entendu d'être en dessous, ou encapsulé par le modèle.
- **View** : La vue rend le modèle dans une forme appropriée pour les interactions, en général une interface utilisateur. Plusieurs views peuvent exister pour un modèle unique, à des fins différentes. Dans un format de préférence "Web" (HTML, XML ou JSON) en fonction de la négociation avec le navigateur et des capacités du contrôleur.
- **Controller** : Le contrôleur répond aux événements (généralement des actions de l'utilisateur) et les traite, et peut également invoquer des changements sur le modèle. Dans une application Web, les événements sont généralement des requêtes HTTP: un contrôleur écoute les

requêtes HTTP, extrait les données pertinentes de la «événement», telles que les paramètres de chaîne de requête, demander des têtes ... Et applique les modifications sur les objets du modèle sous-jacent.

Dans une application play ces trois couches sont définies dans un répertoire app, chacun dans un package,

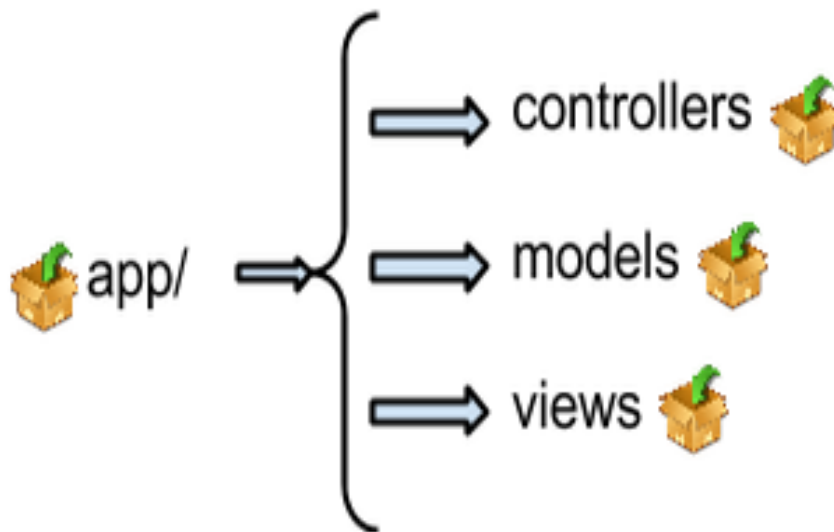


Figure 3.2: Default play packages

Ainsi que d'autres packages qu'on a rajouté comme event, api, services, repository, plugins, templates...

3.2.1 Cycle de vie d'une requête

Le framework play est entièrement Stateless[1] et orientée Request/Response. Tout les requête HTTP suivent le même path.

1. Une requête HTTP reçue par le framework.
2. Le Router Component essaie de trouver la route la plus spécifique en mesure d'accepter cette demande. Pour invoker la méthode d'action correspondante.
3. Le code d'application est exécutée
4. Si une vue complexe doit être générée, un fichier template est rendu.
5. Le résultat de la méthode d'action (Code de réponse HTTP, Content) s'écrit alors comme une réponse HTTP.

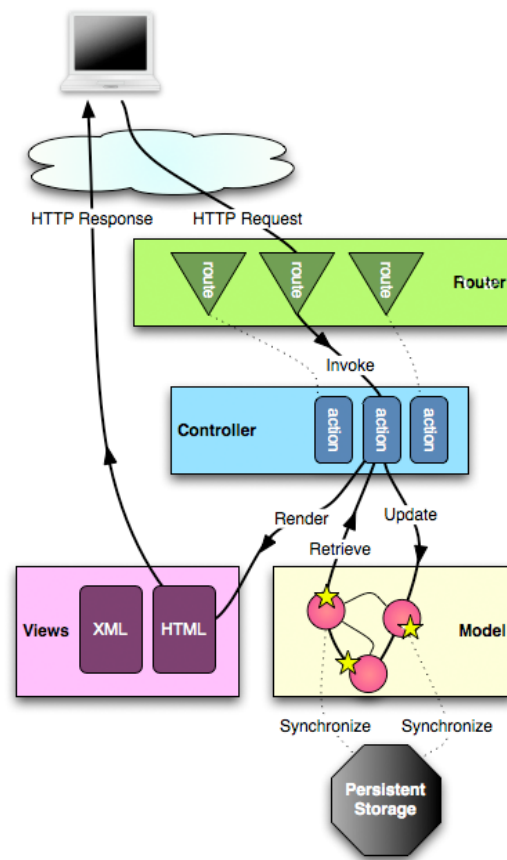


Figure 3.3: Diagramme HTTP request path

3.3 Présentation du contexte Business

3.3.1 Login V0

Dans la version précédente, il s'agit d'un formulaire classique à remplir pour créer un nouveau compte utilisateur qui sera stocker avec les coordonnées nécessaires (firstname, fullname, username, password, secretCode). Cela est une première approche, comme la plateforme est en cours de construction des nouveaux besoins apparaissent au fur et à mesure. Pour plusieurs raisons : Satisfaire la clientèle (utilisateurs de Stample), faciliter la perception de l'interaction et offrir des nouvelles fonctionnalités classique mais indisponible.

3.3.2 Critiques

Cette approche pour la gestion de la création d'un compte utilisateur sur Stample doit être refondues. Il nous manque l'envoi des emails d'information lors des différents étapes d'inscription notamment le reset du mot de passe, ainsi que la possibilité de créer un compte à partir d'une autres plateformes en mesure que l'utilisateur s'ennui de reprendre tout un formulaire pour créer un nouveau compte sur une nouvelle plateforme.

3.3.3 Fonctionnalités manquantes

Envoi des mails

Les utilisateurs de la plateforme doivent être notifier par mail lors de l'inscription, l'activation du compte et le changement du mot de passe. Les différents emails nécessaires sont signupEmail, welcomeEmail, alreadyRegisteredEmail, unknownNotice, passwordResetEmail, passwordChangeNotice.

Authentification avec d'autres plateformes

Plusieurs sites web offrent à leurs utilisateurs la possibilité d'utiliser gmail, facebook, twitter ou autres pour le signup. Cette fonctionnalités facilite l'inscription et rendre cette tâche rapide pour certains utilisateurs.

Rest mot de passe

Plusieurs utilisateurs, changent leurs mot de passe d'un site à un autre ce qui provoque souvent l'oubli du password. La gestion des mots de passe oublié est parmi les fonctionnalités indispensables sur une plateforme.

AuthCode

La version Beta de Stample est protégée avec un code d'activation, c'est une pratique habituelle pour sécuriser une plateforme mise en ligne et en

cours de construction.

3.4 Première approche

3.4.1 Etude du modèle

Après quelques jours pour faire des considérations de conceptions et de mieux comprendre SecureSocial, j'ai réalisé que la mise en œuvre des méthodes n'était pas difficile à comprendre. C'est bien la conception de la logique dans un service backend qui compte. SecureSocial offre des APIs Scala et Java, on utilise ce module Scala pour le Backend de la plateforme. Les services d'authentifications utilisent la catégorie des services de premiers plans comme Google, Twitter, Facebook ect..., Il offre aussi un mécanisme de username/password avec les fonctionnalités Signup, Login, Rest Password.

Ce module est compatible avec les versions de Play 2.1.x, 2.0.x et 1.x, il est relativement rapide à intégrer dans une application en suivant les instructions sur *le site*⁴. SecureSocial est extensible, il est basé sur un modèle de design qui vous permet d'ajouter des nouveaux services d'authentifications.

3.4.2 Première Solution

Les étapes d'implémentation:

- J'ai ajouté une nouvelle table dans la base de donnée Stample pour mettre le hash (email, random number, time),
- Envoyer le mail avec un lien `url?hash=$hash`,
- Enfin vérifier l'existence du hash dans la base puis faire le tri.

3.5 Remise à Zéro

3.5.1 Phase préliminaire

D'après Jonathan Winandy, pour réussir il faut passer par les étapes suivantes :

- Do It : Commencer par poser le problème et puis écrire du code qui répond à ce besoin.
- Do It-Right : Se débarrasser du code inutile, faire des tests et des améliorations.
- Do It-Fast : Nettoyer le code et vérifier les tests.

⁴<http://securesocial.ws/guide/getting-started.html>

3.5.2 Implémentation & Intégration

Pour l'intégration de SecureSocial

$$DoIt = \begin{cases} \text{La vérification de compilation, intégration basique de secure Social dans Stample.} \\ \text{Une écriture basique dans la mémoire et l'implémentation de UserService.} \\ \text{Working Wiring : Authentification avec email.} \end{cases}$$

$$DoIt - Righ = \begin{cases} \text{Ajout du template signUp email.} \\ \text{Résoudre les problèmes de Token et de Memory.} \end{cases}$$

L'étape Do It-Fast n'est pas encore faite mais elle pourrait être mise dans le planing des tâches plus tard.

3.5.3 Keep user Logged In SecureSocial :Authenticator Cookie

Dans le fichier de configuration de SecureSocial il y avait des changements à faire comme la configuration des cookies comme absoluteTimeoutInMinutes(La durée d'authentification d'un utilisateur dans une session. Après cela l'utilisateur doit relogger (par défaut à 720 minutes - 12 heures). idleTimeoutInMinutes(La durée de session valable depuis la dernière requête (par défaut 30)). Comme le sujet, l'idée est de limiter les partie amovible, donc on s'est simulé une implimentation in-memory de l'authentificaion pour commencer.

3.5.4 Customisation des templates

Après l'intégration et la stabilisation des différentes interractions avec le module, j'ai travaillé pour refactoriser les différentes views pour s'adapter avec ce module (le signup, resetPasswordPage, authorisationCode, startResetPassword) et un main global. L'activationCode c'est un code secret pour protéger la version Beta de Stample(autoriser l'accé qu'à certain utilisateur), il y avait un activationCode écrit dans le code de la plateforme que nous avons enlevé pour mettre dans l'interface admin la possibilité de générer des diffirents codes d'autorisations.

3.5.5 Partie Administrateur sur Stample

Stample c'est un réseau social en cours de construction il y avait plusieurs bugs et amélioration à ajouter sur la plateforme durant mon stage. Parmi ces amélioration l'ajout dans l'interface administrateur de l'API la possibilité de générer/écrire un code secret pour les utilisateurs lors de l'inscription avec un nombre d'usage pour chaque code. Cela nécessite l'ajout des routes pour gérer et créer les codes d'authentification dans l'admin controller et les templates. J'ai implémenté les différentes templates (authorisationCodes,

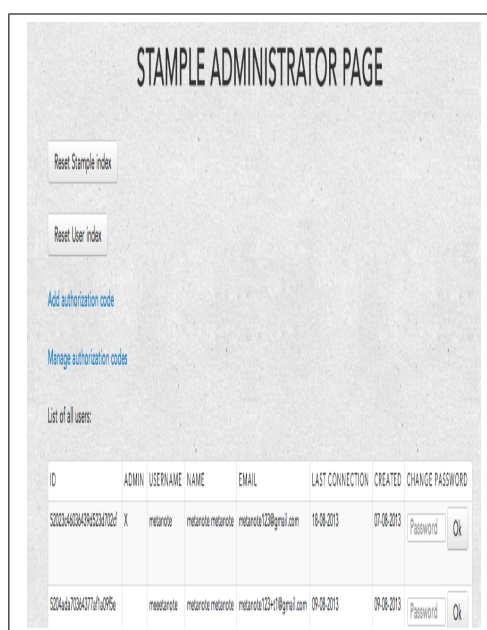


Figure 3.4: Admin page

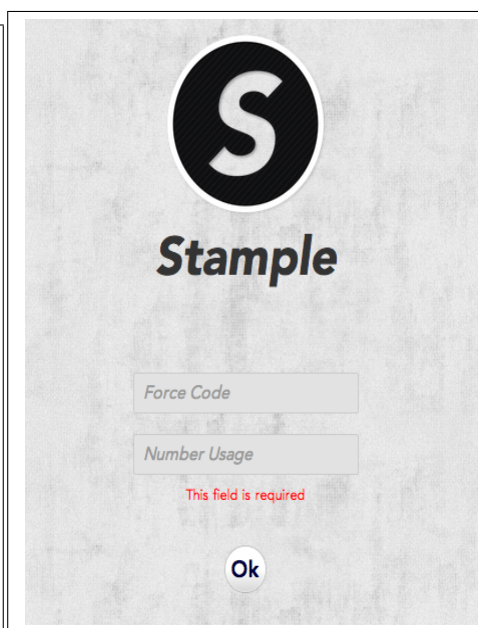


Figure 3.5: Create secret interface

createCode). Depuis l'interface Admin le bouton create add autorization code ouvre la deuxième interface pour la creation d'un code secret aléatoire avec un le nombre d'usage qui est un champ obligatoire ou bien en choisissant un code d'autorisation. Le bouton manage autorisation codes permet d'afficher les codes d'autorisation enregistrer et le nombre d'usage restant.

3.6 Conclusion

Cette partie d'authentification sur Stample m'a beaucoup apporté en écrivant du code/des templates et en fixant des bugs. Sur Backend, l'interaction des controllers, les models et les views avec les différentes classes et traits existant m'avait permis de comprendre le langage en ajoutant du code compatible.

Cette partie du développement m'a permis de reconnaître le plugin SecureSocial qui peut servir dans d'autres projets et d'améliorer mes connaissances. Ce plugin sera peut-être enlever plus tard et remplacer par l'authentification avec WebID.

Chapter 4

Notification& Commentaire

4.1 Introductction

Dans une autre phase d'implémentation de la plateforme, comme autres réseaux sociaux il est indispensable de notifier l'utilisateur dans le cas du partage d'un stample, une catégorie, le demande d'ajout...

Stample offre la possibilité d'organiser le contenu diffusé sur la plateforme sous forme des catégories, le partage de ces derniers c'est une nouvelle fonctionnalité qu'on trouve sur dropbox, sauf que pour Stample les catégories sont organisées sous forme d'une arborescence similaire à un système de fichier.

Le stample ou autrement l'article c'est un contenu qui peut être écrit en précisant le titre, le contenu, l'auteur, la source.... Peut être aussi générer par des darg & drop en utilisant le clipper depuis un autre site.

Ces options nécessitent d'informer les utilisateurs de la plateforme en option avec l'email et sous forme d'une alert sur le site.

Les commentaires sont aussi une fonctionnalité intéressante pour interagir avec le contenu d'un stample.

4.2 Notifications

Le système de notification représente dans la base mongo par une collection `notification_inbox` cette dernière elle se compose d'un ID, notification et isRead. On utilise des services : `addNotificationsForUser`, `getNotificationForUser`, `deleteNotification`, `readNotification`.

4.3 Commentaires

J'ai travaillé dans cette partie sur l'implémentation des tests d'intégrations et des services update, delete comment. Les tests sont écritent en Specs2 (software specification for Scala) : c'est une librairie pour écrire des spécifications des softwares executables et lancer sur la console avec sbt "it:test". Vous trouverez par suite deux captures d'écran une pour le lancement du test sbt et l'autre du code. Il faut ajouter les dependencies du plugin dans le fichier ApplicationBuilt du projet.

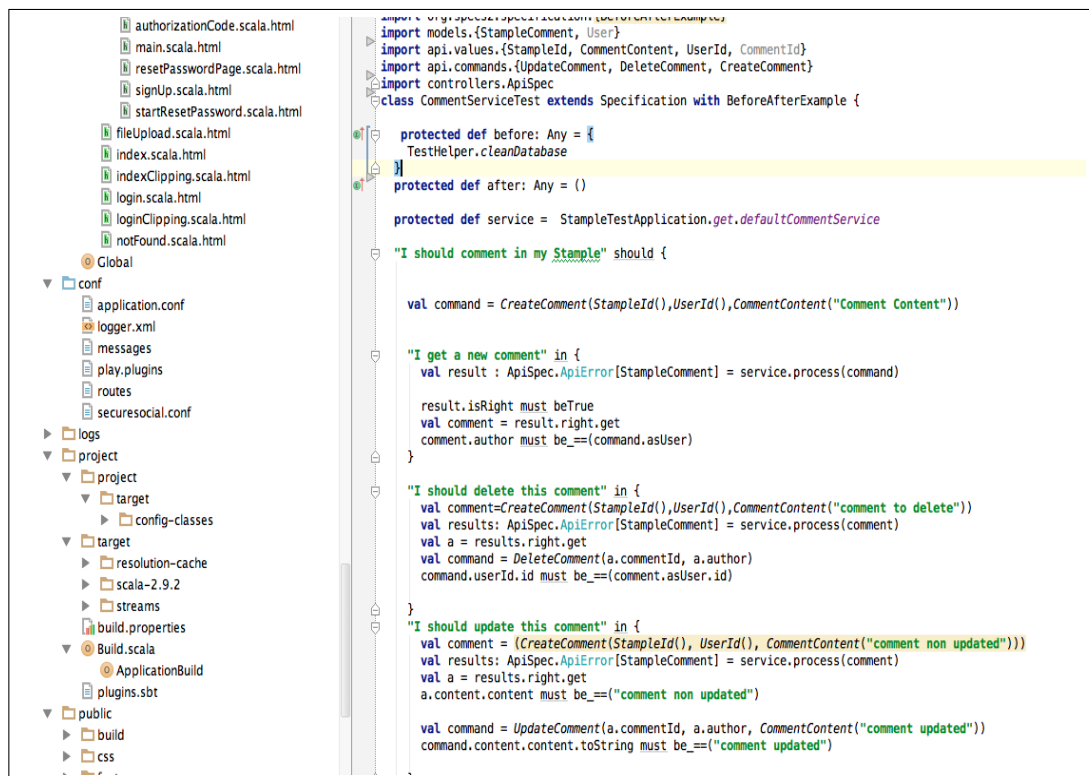


Figure 4.1: Tests code

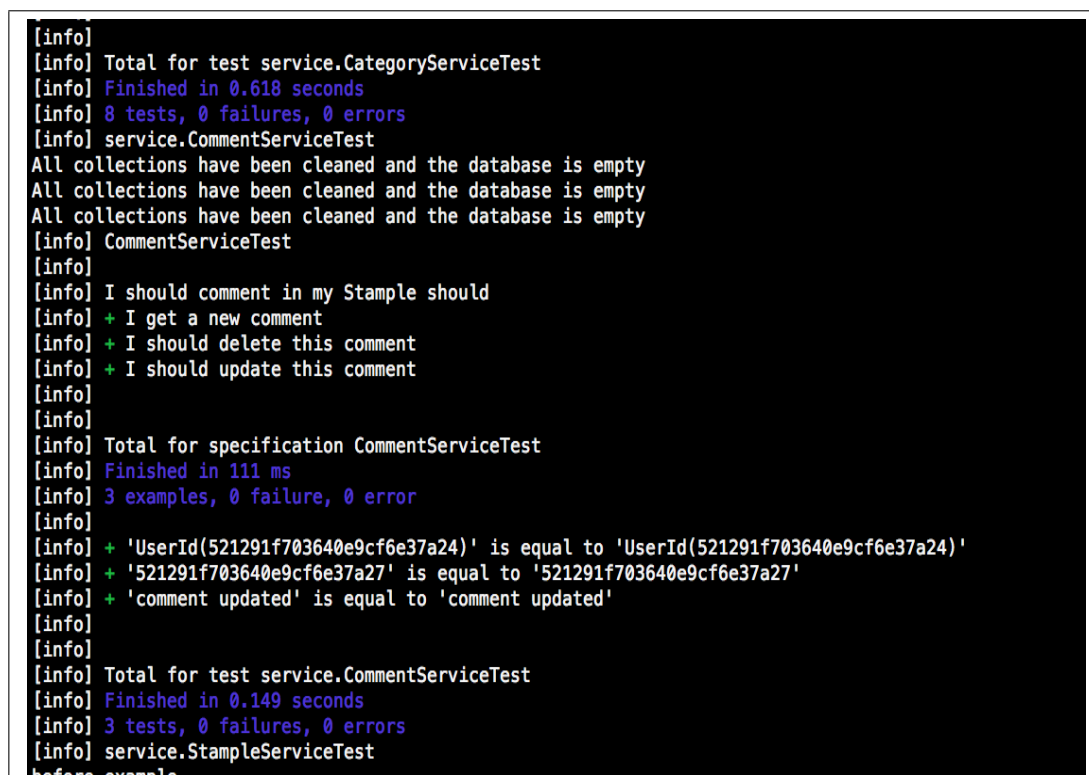


Figure 4.2: Terminal lance tests

4.4 Conclusion

Dans cette partie du Stage, j'ai découvert le système de notification en travaillant sur des sous tâches, une première approche pour le commentStample et j'ai écrit les tests nécessaires pour la création, la mise à jour et la suppression des commentaires. Le chapitre postérieur sera des alternatives sur les réseaux sociaux centralisées et distribuées relative un future Stample en cours de construction. Des bibliothèques sont misent en ligne un projet open source "banana-rdf" et "RWW-Play", Henry Story, travaille sur ces bibliothèques pour les adaptées au plateforme Stample.

Chapter 5

Contexte des Réseaux sociaux Centralisés et décentralisés

5.1 Introduction

Tout d'abord *QU'EST CE QU'UN RÉSEAU SOCIAL ?*

Le terme "réseau social" provient de John Arundel Barnes en 1954. Les réseaux sociaux existaient bien avant Internet. Un réseau social n'est en effet rien d'autre qu'un groupe de personnes ou d'organisations reliées entre elles entretiennent. Un utilitaire social comme Facebook, Google+ ou autres aident les gens à communiquer de manière plus efficace avec leurs amis, leur famille et leurs collègues. Aujourd'hui le réseau social c'est une application internet dédiée à la communication avec ses connaissances, à la rencontre de nouvelles personnes, à la construction de son réseau professionnel, à la partage des données ou le sauvegarde des centres d'intérêt (des multimédias sur internet ou PC).

Le principe de base est le même pour tous le réseau sociaux : Création d'un profile, Inviter des amis, Accepter des contacts, Partager un contenu, Discuter... Le plus important dans ce type de réseau est de permettre à l'internaute d'augmenter ça réputation virtuelle.

Les utilitaires sociaux Facebook, mySpace, LinkedIn, OrKut, Google+ ... gagnent chaque jour des millions d'utilisateurs, mais on commence à apercevoir quand même des utilisateurs qui désactive leurs comptes facebook ou autres. Comme *l'article de BEGEEK* ¹sur le chute de facebook : Selon SocialBakers, ces 6 derniers mois le réseau de Mark Zuckerberg aurait perdu 9 millions d'abonnés sur le sol américain.

¹<http://www.begeek.fr/facebook-chute-daudience-et-une-perte-de-plusieurs-millions-dutilisateurs-91030> publié le 29 avril 2013 à 19h04

Ces sites présentent trois problèmes majeurs: Le premier c'est que les informations dans un site ne peuvent pas être utilisées dans les autres sites. Deuxièmement ces sites ne permettent pas aux utilisateurs de contrôler leurs données personnelles diffusées ce qui entraîne des problèmes de confidentialité potentiels. Finalement les données contrôlées par la firme possédante des données au commerçant et puis c'est dernier il les revende au gouvernement dont il dépend, pour espionner, réutiliser ...

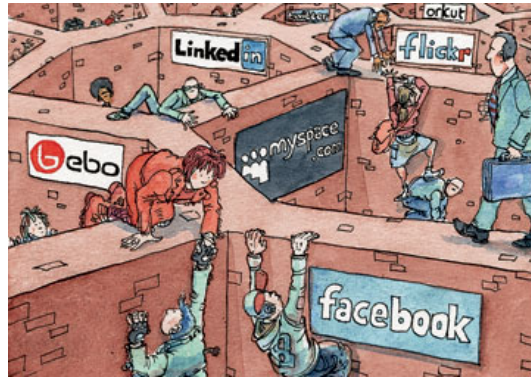


Figure 5.1: Problem with today's social networks

Ces problèmes peuvent être résolus en adoptant une approche décentralisée pour les réseaux sociaux, avec cette approche les utilisateurs n'ont pas à être limités par un service social. Cette approche peut fournir le même niveau, acquérir un nombre bel et bien important d'utilisateurs tout en leur accordant plus de contrôle vis-à-vis leurs informations personnelles et maintenir mieux leur gestion.

Le réseau social distribué est basé sur les technologies ouvertes : Linked Data, les ontologies du web sémantique, les systèmes d'identité unique WE-BID et le contrôle d'accès (web access contrôle). Les URIs des identifiants permettent au framework décentralisé d'être distribué extensible, comme les utilisateurs, les applications, les données réfèrent à leurs URIs². Le web actuel est décentralisé mais il n'est pas distribué, en effet il forme le cercle vicieux contournant autour des géants comme Google, Amazon ...

²<http://dig.csail.mit.edu/2008/Papers/MSNWS/>

Social Media Landscape



Figure 5.2: Social Media

5.2 PARTIE 1 : Etude des réseaux sociaux existant

5.2.1 Les débuts décentralisés d'internet

La modélisation d'un réseau social décentralisé qui élimine la dualité entre le fournisseur de service et l'utilisateur, le modèle client/serveur, remplacer par une situation où chaque utilisateur a son propre serveur (chaque client est un serveur) c'est loin d'être une nouveauté absolue mais c'est plutôt c'est un retour aux origines d'internet. En effet depuis le début du "réseau des réseaux" le principe de décentralisation a été à la base de transmission et de communication qui y circule. Pourtant en 1990 l'introduction du web a conduit progressivement à un modèle client/serveur; les services diffusés sur internet les plus répandus (les réseaux sociaux (Twitter), les services de stockage des données numériques (dropbox)...) sont conçus à partir d'un modèle technique dans lesquels l'utilisateur final demande une donnée, information ou service à un puissant centre de serveurs qui stockent les informations et gèrent le trafic sur le réseau donc même sur internet si le trafic fonctionne avec le principe de distribution généralisée actuellement il est concentré autour de serveurs qui autorisent l'accès au contenu. Pourtant la conception d'un réseau distribué de façon que la communication/les échanges auront

lieu entre des noeuds jouant un rôle symétrique dans le système c'est une alternative possible qui est le plus à même d'assurer la durabilité du réseau Internet.

5.2.2 Motivation

Les services des réseaux sociaux existant sont centralisé la compagnie qui donne le service à tout le contrôle de l'information ce n'est pas facile à l'utilisateur de réutiliser ces propres données inclus son réseau social, le contenu multimédia dans d'autres plateforme jusqu'à maintenant il n'y a pas un mécanisme permettent de transporter les données d'un utilisateur d'une plateforme à une autre. Essentiellement les gens s'ennui de crée un compte sur une nouvelle plateforme puis re-ajouter tous leurs amis on ligne, informations ... Et puis la présentation de leurs informations est souvent dépendante du design du service social qu'ils utilisent.

De plus les utilisateurs doivent accepter les conditions générales d'utilisation de ces réseaux sociaux quand ils les utilisent, ainsi qu'ils peuvent accepter l'utilisation de leurs données. En outre, très souvent, les utilisateurs doivent explicitement opt-out de certaines applications si elles sont plus conscients de la confidentialité de leurs données. Cependant, avec un contrôle décentralisé, pas un seul service est le seul accès aux données et la capacité d'appliquer les décisions arbitraires.

Les réseaux sociaux populaires comme facebook, googleplus et autres donne l'impression au utilisateurs qu'ils ne contrôlent pas leurs données, par contre ce n'est pas le cas. Par exemple facebook donne la possibilité au utilisateurs de désactiver leurs compte, mais ce n'est pas possible d'écraser tous les données et les informations personnelles misent sur le serveur du site.

5.3 PARTIE 2 : A propos des réseaux sociaux décentralisé

5.3.1 Le décentralisé sur le Net

Un nombre de projet actuellement relèvent le défi à créer le réseau social décentralisé qui puisse ériger à compétiteur crédible et fiable de Facebook.

- Diaspora ³ :

Diaspora c'est le premier réseau social décentralisé qui a eu un très grand écho dans les médias histoire ou différentes ordinateurs indépendant dits "graines" sont amenés directement entre eux tout en abritant leurs propre profil.

- NoseRub ⁴ : C'est un protocole distribué sous licence MIT (Massachusetts Institute of Technology) il a été créé par Dirk Olbertz, permettant aux utilisateurs de réseau de garder les information de leur profile sur leur propre terminaux, et à leurs terminaux d'interagir et de se synchroniser automatiquement.

- TENT ⁵:

Tent est un protocole de communication distribués. Tent peut être utilisé comme un emplacement de stockage personnel des données, un seul signe sur le service, et / ou un réseau social distribué. N'importe qui peut héberger leur propre serveur de tent. En plus d'accueillir des tents, Tent.is donne quelques applications de base pour aider les utilisateurs à démarrer, une application d'administration du serveur et une application de micro-blogging.

Both the protocol / open source staff and Tent.is are built and maintained by the same team which is half American, half Canadian... N'importe quelle personne peut créer un serveur Tent. Il n'est pas centralisé pour limiter les autorités des développeurs ou les utilisateurs.

Les posts sont au coeur de tent protocole, Chaque morceau de données dans la tent, à partir d'un message d'état de configuration de l'application, est stockée dans un post.

Les Posts sont du JSON consiste de trois composants logic :

- metadata
- content

³<https://diasporafoundation.org/about>

⁴<http://en.wikipedia.org/wiki/Noserub>

⁵<https://tent.io/about>

- binary attachment
- 1) Tent vous permet de sauvegarder vos données dans une place dont vous avez le contrôle.
- 2) Vous pouvez choisir un fournisseur d'hébergement ou lancer votre propre serveur.
- 3) Si vous souhaitez déplacer les hosts vous aurez vos données et relations.
 - tent utilise https et JSON pour transporter les posts entre les serveurs et les apps.
 - les entités sont les utilisateurs de tant il autorisent apps, établi des relations et lire/publier des posts les entités sont définie avec leur entité URL; https/http URL liée les métadata
 - chaque entité a 1 ou plusieurs serveur qui la représente.
 - posts sont les atomics unité de contenu dans tent chaque post est un JSON, ils ont des fichiers attacher.
 - APPS fourni des UI (user interface) pour tent.
 - tent c'est une machine readable JSON api les apps doivent être autorisée avec oauth2
 - relationShips : les entités établissent des relations lorsque il envoient des messages qui mentionnent d'autres entités.

Les fonctionnalités trouvées aussi sur les sites d'information communautaires comme *hacker news*⁶ peuvent être reproduite avec tent, mais nécessairement architecturée d'une manière différente. les messages et les commentaires sont situé sur le serveur tent au lieu d'être regroupé sur un site unique ou base de donnée.

Les données sont stockées dans tent comme messages. Messages, comme les fichiers, sont tapés. Il ya un petit nombre de types de poste prévues par le protocole que les serveurs utilisent des tentes. Les développeurs sont libres de créer de nouveaux types de poste pour le contenu / stockage de données.

- En france Turbulences⁷ :
Propose une solution technologique open-source, utilisant pour une variété d'acteurs institutionnels et de secteur privé afin d'assembler et de lancer leur service de réseau social, intégré au services en ligne existant au travers de protocoles et de standards libres.

On peut cité aussi

⁶<http://thehackernews.com/>

⁷<http://ticmigrations.fr/fr/etat-de-lart/projets/116-turbulence>

- Freenet ⁸ : C'est un réseau informatique anonyme et distribué construit sur l'Internet. Il vise à permettre une liberté d'expression et d'information totale fondée sur la sécurité de l'anonymat, et permet donc à chacun de lire comme de publier du contenu. Il offre la plupart des services actuels d'Internet (courriel, Web, etc.).

C'est deux derniers réseaux ne sont pas des réseaux sociaux distribué mais ils font du distribué.

5.3.2 Réseautage social décentralisée en ligne

Description d'un style d'architecture distribuée

Dans un cadre de réseau social distribué et ouvert, un utilisateur n'a pas besoin d'adhérer à un service particulier de réseautage social tels que Facebook ou MySpace. Au lieu de cela, l'utilisateur choisit un serveur qui il a confiance pour héberger ses propres données telles que son FOAF (Friend-Of-A-Friend) [fichier, son journal d'activité et ses albums photos. Étant donné que nous nous référons à ces fichiers avec leurs URI, ils peuvent effectivement être stockés sur des serveurs différents⁹.

En utilisant FOAF dans un cadre de mise en réseau social décentralisé, le WEBID d'un utilisateur peut être utilisé comme un point d'accès à ses données. Les autres utilisateurs qui veulent accéder au réseau de l'utilisateur sociaux (liste d'amis), son statut, ses photos, ou d'écrire sur son tableau de message personnel, seront versés à son dossier de FOAF et obtenir les URI correspondant. En stockant les données dans un serveur de confiance choisie par l'utilisateur, les utilisateurs ont plus de contrôle sur les données. Il ya aussi une option pour créer des politiques de contrôle d'accès à grains fins beaucoup en utilisant des langages de la politique comme de l'air [Kagal et al. 2008] pour aider à limiter l'accès à ses données ou des applications. Contrairement à un site centralisé de réseautage social, les utilisateurs devront s'authentifier sur des serveurs différents quand ils veulent accéder aux données confidentielles de leurs amis. Cela peut être fait en utilisant par exemple le protocole OpenID [Recordon et Fitzpatrick 2006] en permettant aux utilisateurs de créer des identités en ligne qui utilisent des protocoles existants comme URI, HTTP, SSL et Diffie-Hellma etc ou en utilisant FOAF des utilisateurs + certificats SSL . Un tel cadre décentralisé permet également une personnalisation plus importante des applications et des interfaces. Par exemple, les utilisateurs peuvent créer leur propre page

⁸<http://fr.wikipedia.org/wiki/Freenet>

⁹<http://dig.csail.mit.edu/2008/Papers/MSNWS/>

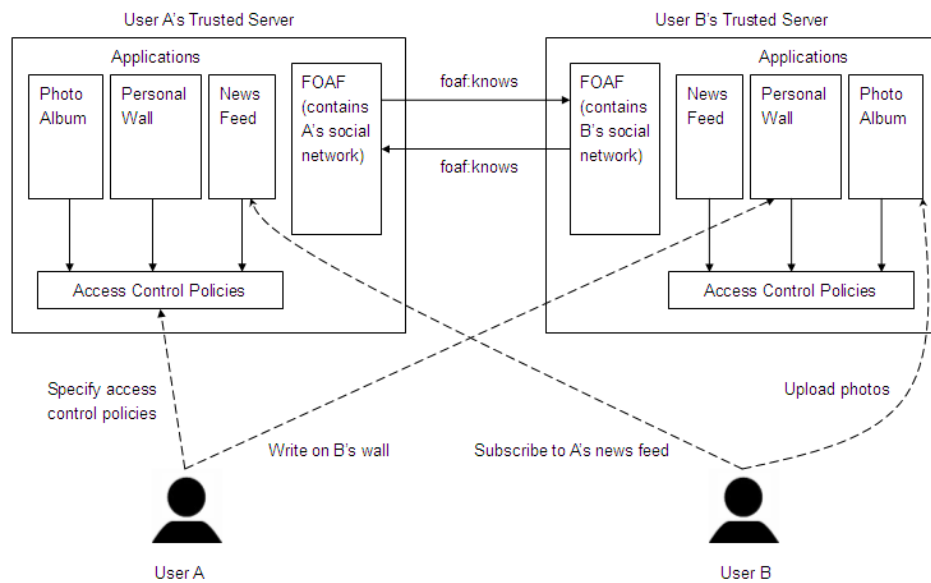


Figure 5.3: A framework of decentralized online social networking

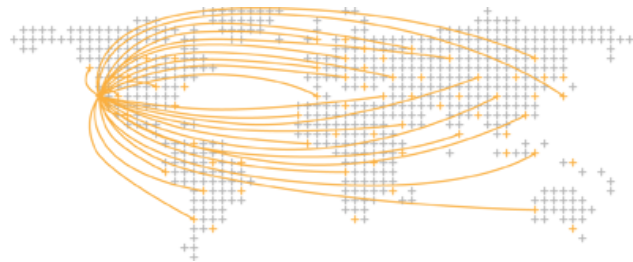
d'accueil qui montre leur réseau, des activités et des photos en ligne sociaux.

5.4 Alternatives

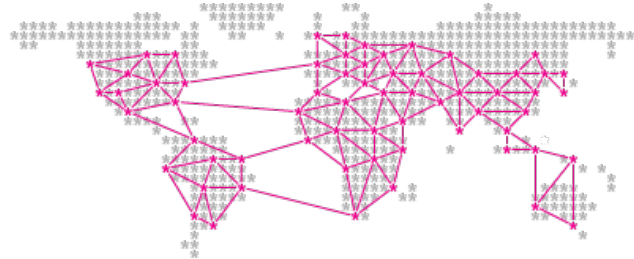
Ces alternatives de réseaux sociaux décentralisé prendra-t-elle suffisamment pied pour constitué un véritable défi pour le géant Facebook ?

Le point d'interrogation principale concerne sans doute la réceptivité des utilisateurs à la possibilité de migrer non seulement sur une nouvelle plate-forme, mais aussi sur une application dans la prise en main, la facilité d'utilisation et les bénéfices sont peut être moins immédiats "la gestion en autonomie de son propre 'petit' serveur". Les différents projet expérimentent à la décentralisation appliqué aux réseaux sociaux présentent peut être une premier réelle tentative, à la fois social et technique d'optimisation, des outils de réseautage social.

Le team Stampile souhaitent avoir deux versions une centralisé stable en changeant l'ergonomie de l'apprentissage de l'utilisateur et en l'offrent des nouvelles fonctionnalités et puis quand on réussit à stabiliser la version décentralisé nous allons le lancer au grand publique.



(a) Centralized network [This Not]



(b) Destributed network [This]

Figure 5.4: Networks

Chapter 6

Conclusion

Suite à ce stage, j'arrive à la conclusion que travailler dans le domaine des réseaux sociaux est passionnant. C'est un domaine en évolution qui mènera toujours à de nouvelles problématiques très intéressantes. Stample offre à ces futur utilisateurs un réseau social de qualité en ajoutant des fonctionnalités manquantes et améliorant l'ergonomie de l'apprentissage.

Ce stage m'a permis de consolider mes connaissances techniques et ma scalabilité. Durant lequel, j'ai travaillé pour améliorer le système de login en ajoutant plus d'options le reset du mot de passe, la possibilité de créer un compte à partir d'une autre plateforme... J'ai eu des sous-tâches pour les notifications, les commentaires. J'ai pu résoudre les problèmes des membres développeur côté frontend.

En dehors de Stample, j'ai enrichi mon Scala en suivant les cours sur Coursera et en faisant des exercices. Ces derniers sont disponibles *sur mon compte GitHub*¹, pour deux raisons :

- La première d'apprendre les différents outils du langage.
- La deuxième pour améliorer mon Scala en s'adaptant à la programmation fonctionnelle.

Cela n'était pas évident pour moi, habitué à la pensée impérative, en outre je n'avais pas de bonnes connaissances en Lisp, Ruby, Clojure... Scala était le premier langage de son genre avec lequel j'ai débuté.

J'ai développé en plus ma première page web personnelle² en HTML5, ce projet à part entière est mis en ligne sur mon compte GitHub, permettant ainsi à mes professeurs, mes collègues et mes amis de naviguer

¹<https://github.com/metanote>

²<http://ideas2d.com/moncef/home.php>

facilement pour découvrir un bon tutoriel Scala et de profiter de mes index disponibles en ligne. De plus, il est possible d'accéder aux sources, sur mon compte GitHub³, pour me proposer des suggestions personnelles ou de m'envoyer des mails depuis cette page.

Pour la gestion du projet, j'ai découvert deux outils en ligne très efficaces :

- Flowdock qui ne permet pas seulement d'avoir un système de "chat" en ligne avec les membres de l'équipe mais aussi de voir l'avancement du projet git instantanément et qui s'intègre aussi trello.

- Ce dernier c'est un outil permettant d'attribuer des tâches aux différents membres de l'équipe, de suivre l'historique, l'avancement d'un projet en ajoutant des cartes aux développeurs Frontend ou Backend comme (doing, done and later) et d'assigner chacun à ces cartes. Personnellement, je trouve cet outil plus efficace qu'un gestionnaire de tâches classiques comme Gant⁴...

Cette expérience s'est avérée très intéressante, dans le sens où j'ai pu participer à des réunions, qui m'ont fait connaître les contraintes et les avantages du travail en groupe. En dehors de ces bénéfices plus ou moins attendus, j'ai appris un vocabulaire de Startup : Lever de fonds, Investisseurs ... et par conséquent j'ai eu l'envie de lancer ma propre Startup plus tard.

³<https://github.com/metanote/Summary-report>

⁴<http://www.ganttproject.biz/>

Critiques

Ce Stage était un premier pas relativement dure dans la vie professionnelle pour des multiples raisons :

- Apprendre Scala et avancer dans le sujet simultanément.
- Ecrire du code dans un Backend existant.
- Connaitre les plugins du Backend comme Salat...
- Utiliser une base de donnée NoSQL.
- Je n'ai pas pu écrire aucune ligne de code dans le Linked Data Platform `banana-rdf`⁵.
- J'ai seulement tester le Read Write Web play `rww-paly`⁶.

Ces deux derniers points peuvent être des nouveaux objectifs au cours de mon deuxième année master vu que les deux bibliothèques open source sont disponible sur GitHub.

⁵<https://github.com/stample/banana-rdf>

⁶<https://github.com/stample/rww-play>

Annexe

<https://stample.co/login> my Stample Profile.

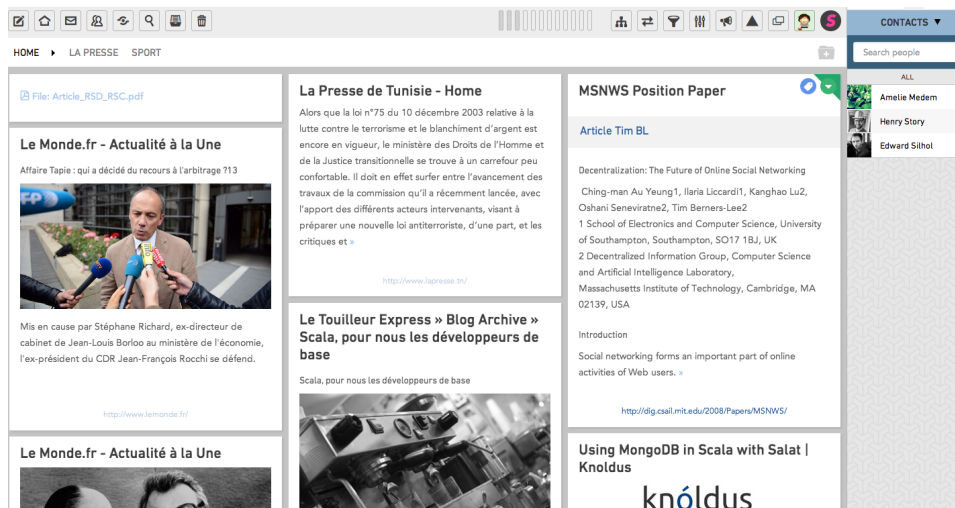


Figure 6.1

<https://www.flowdock.com/> my Flowdock workspace Profile.

Main

@Jonathan @Edward @Amalia @Henry News and Miscelton @matthieu @Sébastien

Moncal

> Inbox

Type to search ...

4a5cd05

fixe date with jon
metanote 2 days ago • [Stample](#) • [fixPrivacy](#)

ab8d0eb

Minor CSS
edwardsilhol 2 days ago • [Stample](#) • [master](#)

Commented commit 7d8872d

a priori c'est juste un quickfix en attendant de créer la nouvelle case class mais si on a le temps autant partir sur la nouvelle case class directement
slorber 2 days ago • [Stample](#)

Commented commit 7d8872d

Je suis d'accord avec tes autres commentaires par contre j'ai pas pour me trimballer avec des ObjectId partout en entrée des services, ça couple tout à Mongo, après on peut créer notre propre classe Id s
slorber 2 days ago • [Stample](#)

68e070f

Merge branch 'fixPrivacy' into preProd

655194e Merge branch 'fixPrivacy' of github.com:edwardsilhol/Stample into fixPrivacy

9eeef7e

Minor fix
edwardsilhol 2 days ago • 11 more commits • [Stample](#) • [master](#)

68e070f

Merge branch 'fixPrivacy' into preProd

655194e Merge branch 'fixPrivacy' of github.com:edwardsilhol/Stample into fixPrivacy

>

};

Sébastien pour les 2 users?

Edward ouo

oui

Sébastien cool

Edward ;)

Jonathan @Sébastien , il faudrait peut-être ne pas avoir le même filtre pour ES et le Front

Sébastien oui c'est fort probable mais bon en attendant ça marche

mais dans un sens je pense que si car le ES doit retourner des json qui sont directement servis au front

la question c'est est-ce que on renvoie creationDate au front

Edward non il faudrait pas

Sébastien sachant que le timestamp ES est basé dessus

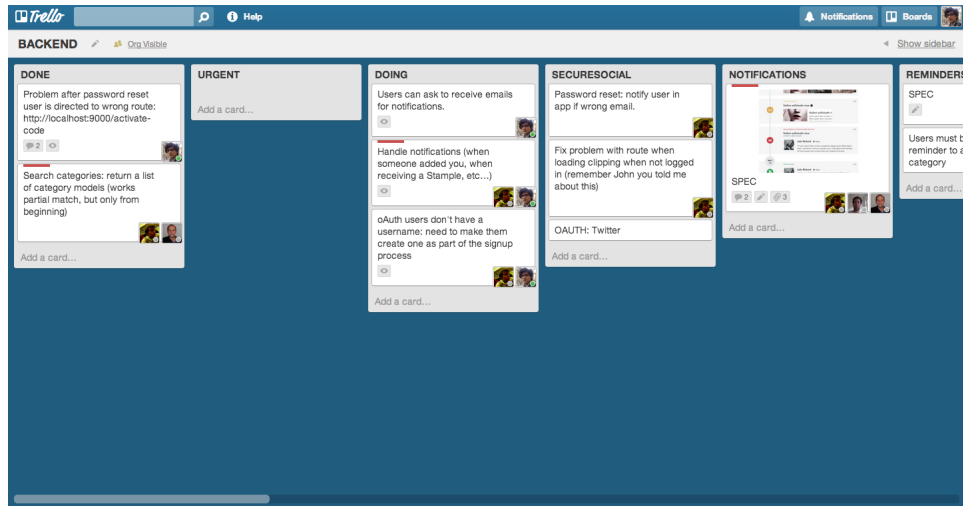
Edward mais c'est pas un drame haha

Sébastien @Edward , bah la temporisation on le fait maintenant

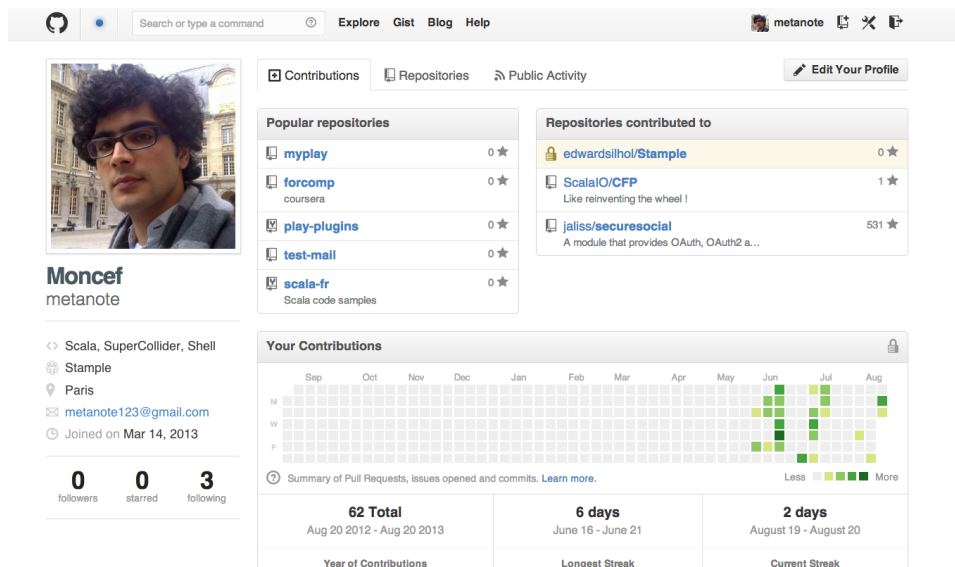
Chat in Main...

Send

<https://stample.co/login> my Trello workspace Profile.



<https://github.com/metanote> my GuitHub workspace Profile.



<http://ideas2d.com/moncef/home.php> my Personnel web Page.

Sommaire

- [Hands On Scala](#)
- [Un langage Scalable](#)
- [Apprentissage des concepts COURSERA](#)
- [Parité 1 : Généralités](#)
- [Parité 2 : Présentation des concepts](#)
- [Outils de développements StartUp](#)
- [Événements](#)
- [Web Sémantique](#)

Hands On Scala

Introduction historique

Afin de s'impregnier du contexte de Scala, commençons par un historique introductif :

- **1996** : Sun Microsystems réalise JAVA 1.0 conçu par James Gosling
- **1997** : Martin Odersky et Philip Wadler publient la spécification du langage Pizza à PoPL (Principles of Programming Language).

Pizza est un superset de Java, et se traduit par des sources Java comme une représentation intermédiaire. Features parametric polymorphism (a.k.a. "generics"), algebraic types (a.k.a. "case classes") et higher-order functions.

- **1998** : Bracha, Odersky, Stoutamire and Wadler publie GJ à OOPSLA. GJ adopte génériques de Pizza, supprime case classes et higher-order functions. Compile directement à bytecode JVM, éliminant traduction source Java.
- **2000** Sun Microsystems adopte le compilateur GJ Odersky's comme javac in JDK 1.3
- **2004** Sun Microsystems Java versions 1.5, intégrant des génériques de GJ

En attendant...

- **Milieu des années 1980** : Beaucoup de langages fonctionnels dans la recherche, les dérivés de Scheme, ML et Miranda.

Caractéristiques du langage explorés, y compris l'inférence de type, l'évaluation paresseuse et types de rang supérieur

- **1987-1999** : développement de Haskell

Livres

- [Scala](#)
- [Programming in Scala](#)
- [Semantic web for the working ontologist](#)
- [The Social Semantic Web](#)