

Table des matières

REMERCIEMENTS	v
INTRODUCTION	vi
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 Concurrence	2
1.2 Conclusion du chapitre 1	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 techniques de Stample	6
2.3.1 Architecture	6
2.3.2 Description de l'architecture Backend	6
2.3.3 Configuration de départ	8
2.3.4 Gestion du travail en groupe	8
2.4 Conclusion du chapitre 2	8
3 Ideas2d	9
3.1 Introduction	9
3.2 Contexte de Scala	9
3.3 Caractéristiques de Scala	10
3.3.1 Higher order functions	10
3.3.2 Hiérarchie Scala	10
3.3.3 Collections	10

3.4	Classes & Objets	11
3.4.1	Classes	11
3.4.2	Objets Singleton	11
3.4.3	Trait	11
3.5	Pattern Matching	11
3.6	Conclusion du chapitre 3	11
4	L'authentification sur Stample	12
4.1	Introduction	12
4.1.1	Configuration du projet Stample	12
4.2	Schema Model View Controller (MVC)	13
4.2.1	Cycle de vie d'une requête	14
4.3	Le login dans l'application existante	16
4.3.1	Login V0	16
4.3.2	Critiques	16
4.3.3	Fonctionnalités manquantes	16
4.4	Première approche	17
4.4.1	Etude du modèle	17
4.4.2	Première Solution	17
4.5	Deuxième approche	17
4.5.1	Phase préliminaire	17
4.5.2	Sauvegarde de session	18
4.5.3	Customisation des templates	18
4.6	Implémentation & Intégration	18
4.7	Partie Administrateur sur Stample	18
4.8	Conclusion du chapitre 4	19
5	Notification & Commentaires	20
5.1	Introduction	20
5.2	Notifications	20
5.3	Tests d'intégration des commentaires	21
5.4	Conclusion du chapitre 5	23
6	Contexte du "STAMPLE distribué"	24
6.1	Introduction	24
6.2	PARTIE 1 : Etude des réseaux sociaux existants	26
6.2.1	Les débuts décentralisés d'internet	26
6.2.2	Motivation	27
6.3	PARTIE 2 : A propos des réseaux sociaux décentralisés	28
6.3.1	Le décentralisé sur le Net	28
6.3.2	Réseautage social décentralisé en ligne	30
6.4	Conclusion du chapitre 6	31
CONCLUSION		32

TABLE DES MATIÈRES

iii

MISE AU POINT	34
Bibliographie	35
ANNEXES	36

Table des figures

2.1	Architecture de la plateforme	6
4.1	Diagramme MVC	13
4.2	Default play packages	14
4.3	Diagramme HTTP request path	15
4.4	Admin page	19
4.5	Create secret interface	19
5.1	Diagramme to Get Notifications	21
5.2	Architecture des comments	21
5.3	Tests Code	22
5.4	Tests Code	22
6.1	Social networks today	25
6.2	Social Media	26
6.3	A framework of decentralized online social networking	31
6.4	My Stample Profil	36
6.5	My Flowdock Profil	37
6.6	My Trello Profil	37
6.7	My GitHub Profil	38
6.8	Ideas2d my personnel web page	38
6.9	Stample User Profil	39
6.10	Centralized network [This Not]	40
6.11	Destributed network [This]	40

REMERCIEMENTS

Avant de vous décrire ce que j'ai appris durant ma première expérience dans le milieu professionnel, il me semble opportun de commencer par des remerciements, à ceux qui m'ont appris durant ces trois premiers mois de stage et à la famille Story de cet accueil chaleureux durant cette période.

Je tiens également à exprimer mes vifs remerciements à mes encadreurs de stage Mrs. Henry Story, Jonathan Winandy, Edward Silhol pour l'aide déterminante qu'ils m'ont accordée, pour l'intérêt qu'ils ont apporté à mon travail et à mon apprentissage, et pour m'avoir accompagné tout au long de cette expérience avec beaucoup de patience et de pédagogie.

Je remercie aussi l'équipe Stample avec laquel j'ai travaillé, pour leur soutien et leurs conseils aussi bien dans la partie implémentation que dans la rédaction de ce rapport. Je remercie particulièrement Mr. Pierre Maret, mon tuteur de stage pédagogique, d'avoir su m'orienter et m'aiguiller à travailler avec Stample et de son soutien tout au long de cette expérience.

Enfin, je remercie tous les professeurs de l'université de Jean Monnet, et toute personne qui s'intéresse au contenu de mon rapport.

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 de Stample.

Ce stage a été une bonne opportunité, je me suis familiarisé avec l'environnement technique, un ensemble d'application Scala/PlayFrameWork et GitHub. J'ai découvert le co-working et j'ai beaucoup appris sur la 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 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 plutôt de faire un tour d'horizon des aspects techniques et humains auxquels j'ai été confrontés.

Il apparaît cohérent de commencer mon rapport de stage par une présentation de Stample et de son contexte technique. Ensuite, je décrirais l'intégration de secureSocial pour le login, l'amélioration de l'interface admin et le système de notifications. Enfin, je conclurais mon travail sur les perspectives de Stample.

Chapitre 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 (échantillon)**, utiliser le script bookmarklet pour attraper facilement le contenu multimédia du web. Permettre le Drag & drop des fichiers depuis votre ordinateur, créer des notes et articles depuis n'importe quel appareil.
- **Staple (article de base)**, gracieusement organiser votre bibliothèque de contenu personnel grâce à un design 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'œil.
- **Stamp (cachet)**, 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 constituée par les co-founders : *Edward Silhol, Henry Story et Sacha Roger*. Vous trouverez une courte présentation de chacun sur la plateforme.¹.

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

1. <https://stample.co/>

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

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

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

Moncef Ben Rajeb, développeur stagiaire Backend.

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 investisseurs "friends and family", les trois fondateurs ont réussi à lever 200 000 Euros. Leur objectif maintenant est d'avoir une première démonstration prête pour septembre.

1.1.4 Besoins

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

- Filtrage des e-mails et notifications non désirées,
- 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, ect...
- 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,

1.1.5 Concurrence

- 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...

2. <http://www.zenika.com/>

3. <http://us.viadeo.com/>

4. <http://steveblank.com/>

- 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 du chapitre 1

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 contextualisées.

Chapitre 2

Environnement du travail

2.1 Introduction

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

Dans cette partie, je vais présenter les différentes technologies envisagées par les architectes de Stample et relatives au la développement du Backend (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 : Un Framework open source facilite la création des APIs web en Scala, Java. Avec Play. Il suffit de recharger la page web pour voir les modifications de la classe Scala.
- JDK 7 : Contient les bibliothèques JAVA de base, les outils de compilation et se charge à la transfert du bytecode destiné à la JVM.
- Maven³ : Un outil logiciel libre pour la gestion et l'automatisation de production des projets logiciels Java permet la synchronisation des projets indépendants.
- OS X 10.8.4
- SBT³ 0.12.4 : "simple build tool" Un support pour compiler le code Scala.

1. <http://ideas2d.com/moncef/home.php>
2. <http://www.playframework.com/>
3. <http://www.scala-sbt.org/>

- ElasticSearch⁴ 19.4.0 : Un moteur de recherche libre vise à offrir des fonctionnalités avancées de recherche à n'importe quelle API.
- MongoDB⁵ 2.2.x : Un système de gestion de base de données orientée documents(NoSQL). Il permet de manipuler des objets structurés au format BSON (JSON binaire), sans schéma prédéterminé.

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

SecureSocial est un plugin permettant d'ajouter une interface utilisateur d'authentification pour votre application qui fonctionne avec des services basés sur OAuth1, OAuth2 et OpenID.

2.2.4 Outils de développement

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

4. <http://www.elasticsearch.org/>

5. <http://www.mongodb.org/>

2.3 Aspects techniques de Stample

2.3.1 Architecture

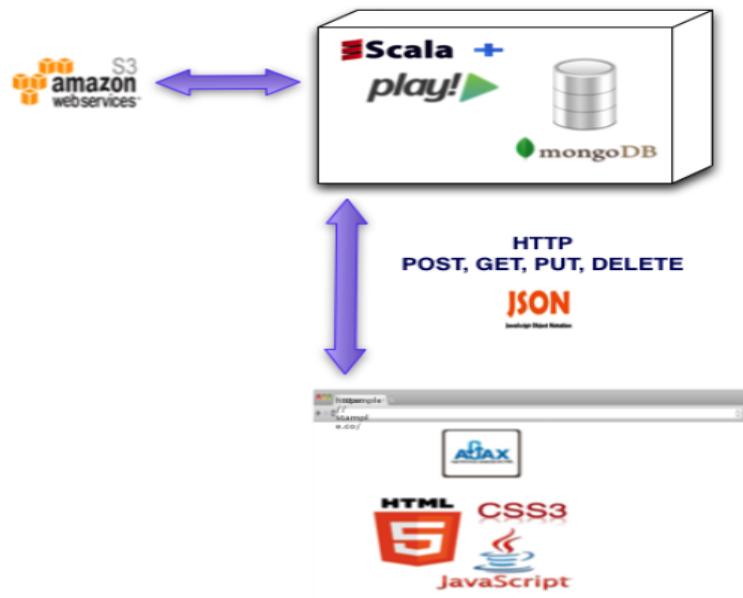


FIGURE 2.1 – Architecture de la plateforme

Commentaires :

La figure 2.1 présente une vue globale sur l'architecture de la plateforme sous forme de trois blocs. On utilise les services web de amazon S3⁶ pour stocker et extraire des données.

Le deuxième bloc représente le Backend former par "Scala + PlayFramework + Java + MongoDB", on renvoie les données sous forme de JSON au Frontend, les communications sont effectuées via des requêtes HTTP.

Le troisième bloc présente le Frontend avec tous les outils informatiques de degin "CSS3 + JavaScript + HTML5". Vous aurez plus de détaillles sur le Backend par suite.

2.3.2 Description de l'architecture Backend

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

6. <http://aws.amazon.com/fr/s3/>

Communication entre la base et l'API

On utilise le driver Casbah⁷ 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 BSON⁸, cette librairie propose un DAO (Data Access Object) générique et un model compagnon qui expose les méthodes basique des requêtes Mongo (find(), findById(), search(), ect...)

Communication entre L'API et ElasticSearch

L'idée d'ElasticSearch est d'offrir une solution de recherche à usage général, qui peut évoluer d'une seule machine à plusieurs centaines. L'indexation dans la base se fait manuellement après l'insertion ou la mise à jour d'un document MongoDB dans la base de données. L'échec d'indexation est non bloquant car il est ratrappé par un traitement spécifique d'exception.

ElasticSearch prend en entrée du JSON cela implique qu'il fonctionne particulièrement bien avec Mongo. De plus, on 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 outil d'intégration "Jenkins"⁹ qui fera tourner les tests unitaires et embarqués à chaque "push" d'une branche git pour réparer les regressions et que nous en soyons alertés.

Les collections MongoDB

Stample structure les données sous forme de différentes collections. Elles sont toutes identifiées par un ObjectId, elles regroupent toutes les informations nécessaire sur les utilisateurs (Compte utilisateurs, Stamples, ect...).

7. <http://api.mongodb.org/scala/casbah/2.0/tutorial.html>

8. <http://bsonspec.org/>

9. [http://fr.wikipedia.org/wiki/Jenkins_\(informatique\)](http://fr.wikipedia.org/wiki/Jenkins_(informatique))

2.3.3 Configuration de départ

Description de la configuration

Le projet contient de quatre répertoires séparés : *fixturedatabase* contenant les fichiers JSON et leurs sérialisation binaire encodé BSON, *restore* inclu le build de l'application et les plugins, *stample-search* contenant les classes java et la configuration nécessaire pour le lancement du projet et *stample-web* ce dernier contient les différents packages du code Front/back End.

Afin de participer au projet, j'ai installé les outils cité précédemment dans un premier temps. 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 répertoire stample-search. Enfin il est indisponible de lancer play ou sbt et de générer les fichiers idea, eclipse ou autre dans stample-web selon l'IDE envisagé. Le README.md du projet écrit avec le langage de balisages légers Markdown¹⁰ contient les détails nécessaires pour la configuration du projet.

2.3.4 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 preProd 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, on peut supprimer le projet.
5. Utiliser toujours camelCase pour nommer les branches et des noms intelligents !
6. Ajouter des explications intelligentes lors de chaque commit.

2.4 Conclusion du chapitre 2

Dans ce chapitre introductif, j'ai détaillé les outils qu'on utilise. Ensuite, j'ai présenté l'architecteure globale de la plateforme et les différentes interactions. Le prochain chapitre présente une idée générale sur le langage de programmation "Scala" du Backend Stample.

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

Chapitre 3

Ideas2d

3.1 Introduction

Pour l'intérêt de Stample, j'avais besoin d'apprendre Scala. Suite aux informations, tutoriels que j'ai suivies durant mon stage dans le but d'améliorer ma pensée fonctionnelle. J'ai décidé de développer une page web personnelle pour mettre en valeur ce que j'ai appris.

Au départ, l'idée consistait à écrire le contenu du rapport en HTML5 et l'héberger en ligne puis de l'imprimer, mais cela n'était pas possible vu que la page peut contenir des secrets professionnels.

Sur ma page web personnelle Ideas2d¹, un tutoriel intéressant pour les débutants en Scala. Ce chapitre donne une idée générale sur Scala, vous trouverez plus de détails sur ma page web.

3.2 Contexte de Scala

En 1996, James Gosling a conçu JAVA 1.0 avec Sun Microsystems. Java a remporté plusieurs victoires : le débat du « write once/run everywhere », la gestion de la mémoire, une plateforme d'entreprise, un écosystème open-source énorme.

Martin Odersky, qui n'est moins que l'inventeur du langage Scala. Il a effectué dans d'autres travaux de recherche scientifiques, langages de programmation (Pizza, GJ, Funnel langage ect...) et compilateurs. Cela lui a donné une reconnaissance dans ce domaine.

1. <http://ideas2d.com/moncef/home.php>

Ce langage est conçu à Ecole polytechnique fédérale de Lausanne en 2003, sa dernière version est Scala 2.10.

Le nom Scala vient du besoin d'un langage multi-paradigme (Programmation concurrente, Programmation fonctionnelle, Programmation Acteur, ect...). Scala peut être vu comme un métalangage.

3.3 Caractéristiques de Scala

3.3.1 Higher order functions

Ce sont des fonctions qui prennent en paramètre d'autres fonctions avec des appels récursifs dont le résultat est une nouvelle fonction. Le higher order function est très utilisé dans les programmes Scala.

3.3.2 Hiérarchie Scala

Dans Scala tout est objet, la racine présentée par la classe Any comme "Object" en JAVA. Les deux grands sous-classes sont "AnyRef", supertype de tous les types références (Scalaobject, Java classes, ect...); "AnyVal", supertype de tous types de valeur (Int, Float, String, ect...). En descendant dans la hiérarchie on trouve la class Nothing, elle hérite de tous les autres classes.

Techniquement Scala mélange la programmation orientée objet et fonctionnelle. Les deux styles de programmation ont des forces complémentaires quand il s'agit de l'évolutivité. Il tourne sur la machine virtuelle JAVA. Parmis ces points forts, il est compatible avec les librairies JAVA. Scala est statiquement et fortement typé, c'est à dire qu'on déclare une variable et on l'affecte à une chaîne de caractère. Elle sera automatiquement de type "String".

3.3.3 Collections

Il y a plusieurs concepts intéressants en Scala qu'on peut trouver dans autres langages de programmation fonctionnelle :

- **Map** : une collection pair de (clé/valeur). Toute valeur peut être récupérée en fonction de sa clé. Les clés sont uniques dans le Map, mais les valeurs n'ont pas besoin d'être unique.
- **Option** : un conteneur de zéro ou un élément de type donné. Une "Option[T]" peut être soit un "Some[T]" ou un objet "None".
- **List** : une collection contenant des éléments de même type.
- **Autres collections** : les Sets, Iterators, ect...

3.4 Classes & Objets

3.4.1 Classes

Une classe est un modèle pour les objets. Une fois que vous définissez une classe, vous pouvez créer des objets à partir du modèle de classe avec le mot-clé "new".

3.4.2 Objets Singleton

Scala est plus orientée objet que Java, parce que en Scala nous ne pouvons pas avoir des membres statiques. Au lieu de cela, Scala possède des objets uniques. Un singleton est une classe qui peut avoir qu'une seule instance 'objet'. Vous créez singleton à l'aide du mot-clé "object" au lieu de mot-clé "class". Puisque vous ne pouvez pas instancier un objet singleton, vous ne pouvez pas passer des paramètres au constructeur principal.

3.4.3 Trait

Les Traits similaire à des interface en Java sont utilisés pour définir les types d'objets en spécifiant la signature des méthodes prises en charge. Scala permet également les traits d'être partiellement implémentés, mais ils ne peut pas avoir des paramètres de constructeur. Les traits Scala, et à l'inverse des interfaces de Java, peuvent avoir des champs et implémenter des méthodes.

3.5 Pattern Matching

Pattern Matching similaire au Switch case est le deuxième élément le plus largement utilisé de Scala, après "function values". Scala fournit un grand soutien pour pattern matching pour le traitement des messages.

Un pattern match comprend une séquence d'alternatives, chacune commençant par le cas de mots clés. Chaque solution comprend un motif et une ou plusieurs expressions, qui sera évalué si le motif correspond. Un symbole flèche => sépare le motif sur les expressions.

3.6 Conclusion du chapitre 3

Dans ce chapitre, j'ai essayé de donner une idée générale sur le langage. Pour plus de détails et d'exemples vous pouvez consulter Ideas2d. La pratique et les exercices sont les seuls moyens pour avoir la spécificité de ce langage. La prochaine section se focalise sur la solution développée pour le login sur Stample.

Chapitre 4

L'authentification sur Stample

4.1 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 côté Frontend ?

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

4.1.1 Configuration du projet Stample

J'ai eu l'accé 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, ect... ont des besoins importants en ressources (mémoire,CPU). Stample m'a prêté un MacBookPro.

Pour interragir 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>

4.2 Schema Model View Controller (MVC)

Dans la plateforme Stample, les requêtes sont affectées aux contrôleurs à l'aide de règles de routage. Les contrôleurs utilisent des services et des repository du packges services/repository implémentés dans un sous package "impl" (implementation). Les views contiennent les templates html. C'est une architecture de play et généralement un design classique d'une application java que j'ai découvert.

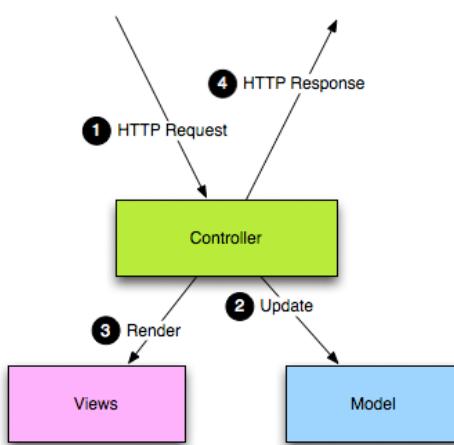


FIGURE 4.1 – Diagramme MVC

- 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, ect...). 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, ect... 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,

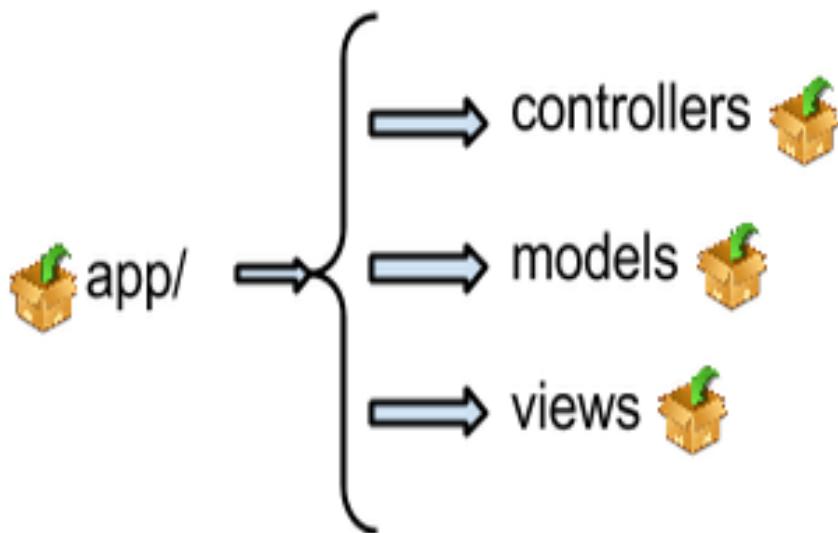


FIGURE 4.2 – Default play packages

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

4.2.1 Cycle de vie d'une requête

Le framework play est entièrement Stateless [3] 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 executée
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.

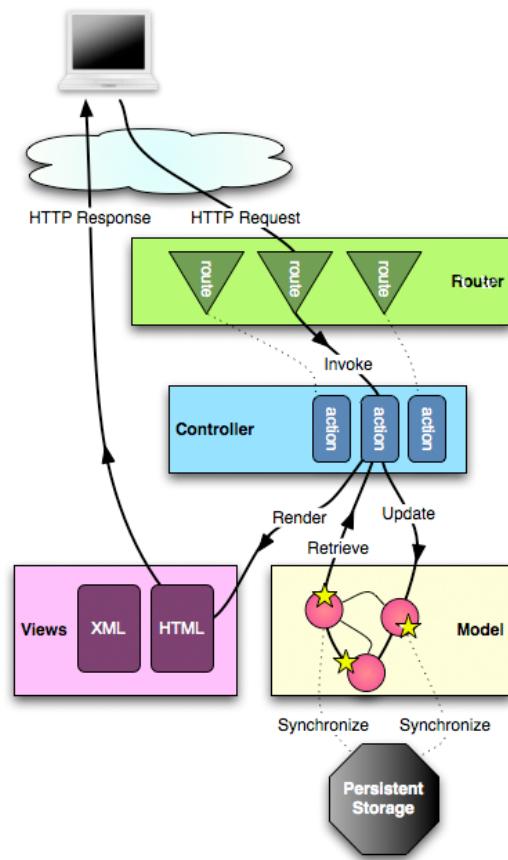


FIGURE 4.3 – Diagramme HTTP request path

4.3 Le login dans l'application existante

4.3.1 Login V0

Dans la version précédente (version V0), 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 indisponable.

4.3.2 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'ennuie de reprendre tout un formulaire pour créer un nouveau compte sur une nouvelle plateforme.

4.3.3 Fonctionnalités manquantes

Envoy des mails

Les utilisateurs de la platefome doivent être notifiés par email 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 gestions des mots de passe oublié est parmi les fonctionnalités indispensables sur une plateforme.

Activation code

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.

4.4 Première approche

4.4.1 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 trop 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 une 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, les instructions pour l'intégration 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.

4.4.2 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,
- Vérifier d'existance du hash dans la base puis faire le tri.

Solution que j'ai envisagé, elle a provoqué une approche non stable en terme d'efficacité et de consistance.

4.5 Deuxième approche

4.5.1 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 amélioration.
- Do It-Fast : Nettoyer le code et vérifier les tests.

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

4.5.2 Sauvegarde de session

Dans le fichier de configuration de SecureSocial, J'avais besoin de changer : La configuration des cookies "absoluteTimeOutInMinutes" (La durée d'authentification d'un utilisateur dans une session), l'utilisateur doit reloguer après ce timing de 720 minutes (par défaut) à 7200 minutes. La durée de session valable depuis la dernière requête "idleTimeoutInMinutes" de 30 minutes (par défaut) à 42000 minutes. Comme le sujet, l'idée est de limiter les partie amovible, donc on s'est simuler une implmentation "in-memory" de l'authentification pour commencer.

4.5.3 Customisation des templates

Après l'intégration et la stabilisation des différentes interactions 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 différents codes d'autorisations.

4.6 Implémentation & Intégration

Pour l'intégration de SecureSocial

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

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

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

4.7 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. Parmis 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, createCode).

The screenshot shows the 'STAMPLE ADMINISTRATOR PAGE'. It features a list of users with columns for ID, ADMIN, USERNAME, NAME, EMAIL, LAST CONNECTION, CREATED, and CHANGE PASSWORD. Two users are listed: '50234603845965520726' and '5044602034571210756'. Below the table are buttons for 'Reset Stample index', 'Reset User index', 'Add authorization code' (in blue), 'Manage authorization codes' (in blue), and 'List of all users'.

FIGURE 4.4 – Admin page

The screenshot shows a 'Create secret' interface. It has a large 'S' logo and the word 'Stample'. Below are buttons for 'Force Code' and 'Number Usage'. A red error message 'This field is required' is displayed above the 'Number Usage' input field. An 'Ok' button is at the bottom.

FIGURE 4.5 – Create secret interface

Depuis l'interface Admin le bouton *create add autorization code* ouvre la deuxième interface pour la création 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.

4.8 Conclusion du chapitre 4

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 enlevé, remplacé plus tard par l'authentification avec WebID.

Chapitre 5

Notification & Commentaires

5.1 Introduction

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

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

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

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

5.2 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. Vous trouverez ci-dessous un diagramme dans lequel est expliqué le début du getNotifications.

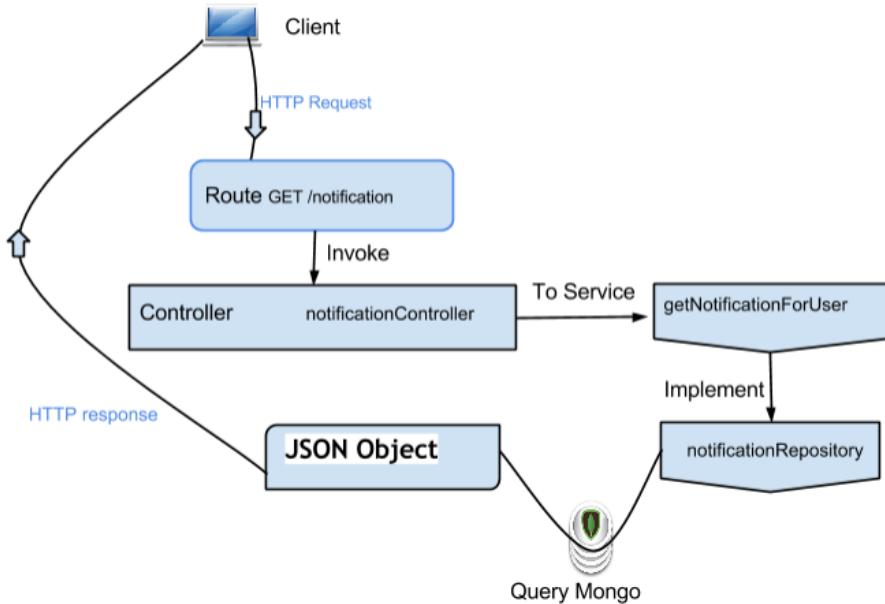


FIGURE 5.1 – Diagramme to Get Notifications

5.3 Tests d'intégration des commentaires

Dans cette partie, j'ai travaillé sur l'implémentation des tests d'intégrations et des services update, delete comment.

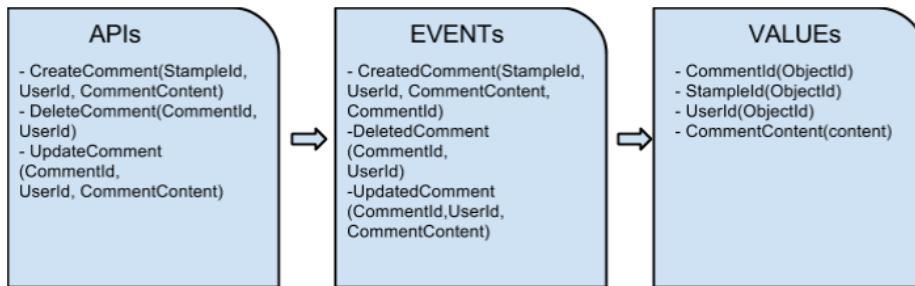
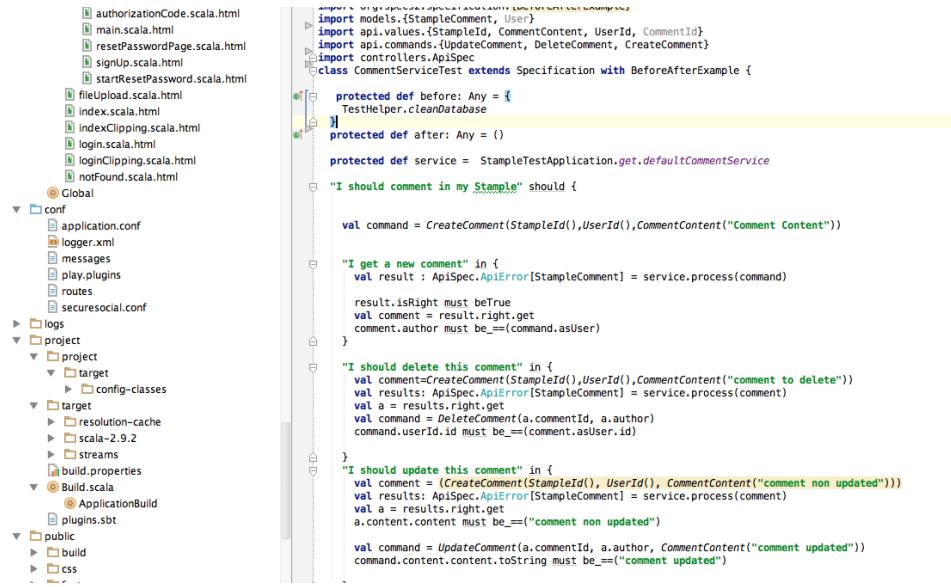


FIGURE 5.2 – Architecture des comments

Les tests unitaires, d'intégration sont écrits en Specs2 (software specification for Scala) : C'est une librairie pour écrire des spécifications des softwares exécutables. Le lancement des test se fait via sbt sur la console. 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.

Vous trouverez ci-dessous deux imprimés écran des tests d'intégration pour les commentaires ainsi que vous pouvez voir le succès des tests sur le terminal.



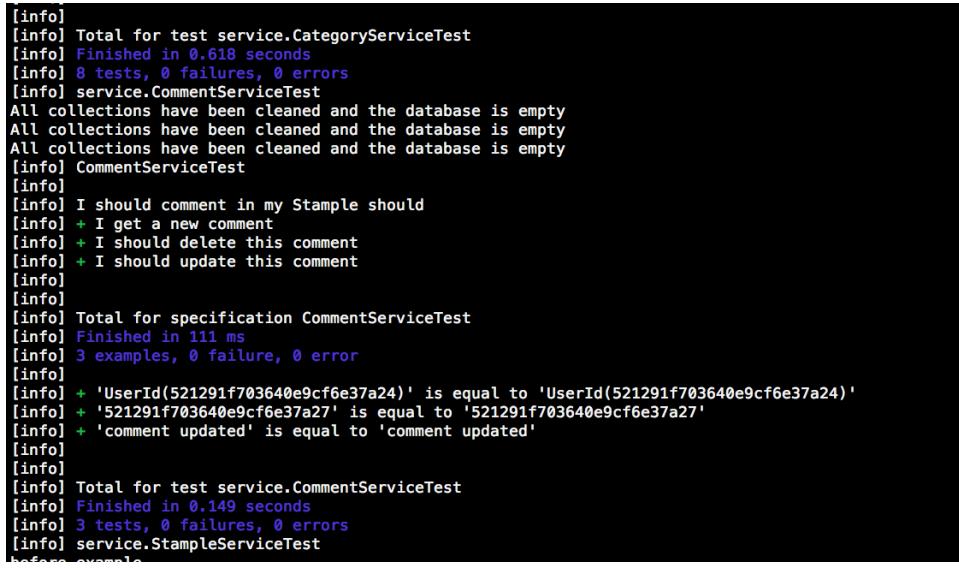
The screenshot shows a file browser interface with a tree view. The root directory is a Scala test file named `CommentServiceTest.scala`. The file contains Scala code for testing a comment service. The directory structure includes `conf`, `logs`, `project`, and `public` folders. The `project` folder contains `target`, `resolution-cache`, `streams`, `build.properties`, `Build.scala`, and `ApplicationBuild`. The `conf` folder contains `application.conf`, `logger.xml`, `messages`, `play.plugins`, `routes`, and `securesocial.conf`. The `public` folder contains `build` and `css`.

```

import org.specs2.execute.Specification
import models.{StampleComment, User}
import api.values.{StampleId, CommentContent, UserId, CommentId}
import api.commands.{UpdateComment, DeleteComment, CreateComment}
import controllers.ApiSpec
class CommentServiceTest extends Specification with BeforeAfterExample {
  protected def before: Any = {
    TestHelper.cleanDatabase
  }
  protected def after: Any = {
  }
  protected def service = StampleTestApplication.get.defaultCommentService
  "I should comment in my Stample" should {
    val command = CreateComment(StampleId(), UserId(), CommentContent("Comment Content"))
    "I get a new comment" in {
      val result: ApiSpec.ApiError[StampleComment] = service.process(command)
      result.isRight must beTrue
      val comment = result.right.get
      comment.author must be==(command.asUser)
    }
    "I should delete this comment" in {
      val comment = CreateComment(StampleId(), UserId(), CommentContent("comment to delete"))
      val results: ApiSpec.ApiError[StampleComment] = service.process(comment)
      val a = results.right.get
      val command = DeleteComment(a.commentId, a.author)
      command.userId.id must be==(comment.asUser.id)
    }
    "I should update this comment" in {
      val comment = (CreateComment(StampleId(), UserId(), CommentContent("comment non updated")))
      val results: ApiSpec.ApiError[StampleComment] = service.process(comment)
      val a = results.right.get
      a.content.content must be==(comment.nonUpdatedContent)
      val command = UpdateComment(a.commentId, a.author, CommentContent("comment updated"))
      command.content.content.toString must be==(comment.updatedContent)
    }
  }
}

```

FIGURE 5.3 – Tests Code



The screenshot shows a terminal window displaying the execution results of the Scala tests. The output is in green text on a black background. It shows the total execution time for each test, the number of examples, failures, and errors, and specific assertions for the comment service tests.

```

[info]
[info] Total for test service.CategoryServiceTest
[info] Finished in 0.618 seconds
[info] 8 tests, 0 failures, 0 errors
[info] service.CommentServiceTest
All collections have been cleaned and the database is empty
All collections have been cleaned and the database is empty
All collections have been cleaned and the database is empty
[info] CommentServiceTest
[info]
[info] I should comment in my Stample should
[info] + I get a new comment
[info] + I should delete this comment
[info] + I should update this comment
[info]
[info] Total for specification CommentServiceTest
[info] Finished in 111 ms
[info] 3 examples, 0 failure, 0 error
[info]
[info] + 'UserId(521291f703640e9cf6e37a24)' is equal to 'UserId(521291f703640e9cf6e37a24)'
[info] + '521291f703640e9cf6e37a27' is equal to '521291f703640e9cf6e37a27'
[info] + 'comment updated' is equal to 'comment updated'
[info]
[info] Total for test service.CommentServiceTest
[info] Finished in 0.149 seconds
[info] 3 tests, 0 failures, 0 errors
[info] service.StampleServiceTest
before example

```

FIGURE 5.4 – Tests Code

5.4 Conclusion du chapitre 5

Dans cette partie du Stage, j'ai découvert le système de notification en travaillant sur des sous tâches, une première approche pour le commentS-tample et j'ai écrits les tests nécessaires pour la création, la mise à jour et la suppression des commentaires.

Le chapitre suivant présentera une étude sur des alternatives relative un futur Stample en cours de construction. Des bibliothéques sont mises en ligne un projet open source "banana-rdf" et "RWW-Play", Henry Story et des membres du W3C, travaillent sur ces bibliothéques.

Chapitre 6

Contexte du "STAMPLE distribué"

6.1 Introduction

QU'EST CE QU'UN RÉSEAU SOCIAL ?

Le terme "réseau social" provient de John Arundel Barnes¹ en 1954. Les réseaux sociaux existaient bien avant Internet. En effet, un réseau social n'est rien d'autre qu'un groupe de personnes ou d'organisations reliées entre elles entretiennent. Un utilitaire social comme Facebook, GooglePlus ou autres aident les gens à communiquer de manière plus efficace avec les amis, la famille et les 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'intérêts (des multimédias sur internet ou PC).

Le principe de base est le même pour tous les réseaux sociaux : création d'un profil, inviter des amis, accepter des contacts, partager un contenu, discuter, ect... Le plus important dans ce type de réseaux est de permettre à l'internaute d'augmenter sa réputation virtuelle.

Les utilitaires sociaux Facebook, mySpace, LinkedIn, OrKut, GooglePlus, ect... gagnent chaque jour des millions d'utilisateurs, mais on commence à apercevoir quand même des utilisateurs qui désactivent leurs comptes facebook ou autres. D'après *l'article de BEGEEK* [2] sur le chute de facebook : Selon "SocialBakers", ces six derniers mois le réseau de Mark Zuckerberg aurait perdu neuf millions d'abonnés sur le sol américain.

1. http://en.wikipedia.org/wiki/John_Arundel_Barnes

Ces sites présentent trois problèmes majeurs :

Premièrement, les informations dans un site ne peuvent pas être utiliser dans les autres sites.

Deuxièmement, ces sites ne permettent pas au utilisateurs de contrôler leurs données personnelles diffusés ce qui entraîne des problèmes de confidentialité potentielles.

Finalement, les données contrôler par la firme peuvent être vendus aux commerçants pour espionner, réutiliser, ect...

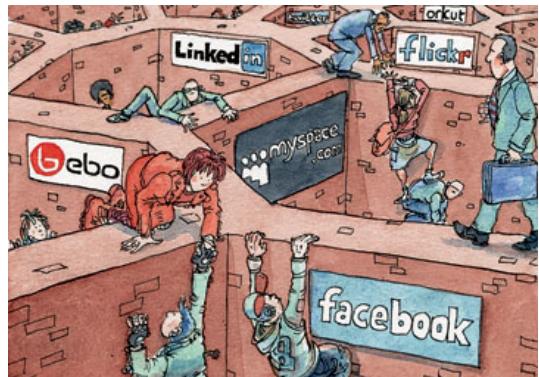


FIGURE 6.1 – Social networks today

Ces problèmes peuvent être résolue en adoptant une approche décentralisée dont 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'identités unique WEBID et l'"access contrôle" (web access contrôle). Les URIs des identificateurs permettent au framework décentralisé d'être distribué extensible, comme les utilisateurs, les applications, les données référent à leurs URIs². Le web actuel est décentralisé mais il n'est pas distribué. En effet, il forme le cercle vicieux et contourne les géants comme Google, Amazon, ect...

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

Social Media Landscape



FIGURE 6.2 – Social Media

6.2 PARTIE 1 : Etude des réseaux sociaux existants

6.2.1 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 à son propre serveur (chaque client est un serveur) c'est loin d'être une nouveauté absolu mais c'est plutôt c'est un retour aux origines d'internet.

En effet, depuis le début des "réseaux 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 diffuser 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.

6.2.2 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'ennuie de crée un compte sur une nouvelle plateforme puis re-ajouter tous leurs amis en ligne, informations, ect... 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 donnent l'impression à leurs utilisateurs qu'ils ne contrôlent pas les données, par contre ce n'est pas le cas. Par exemple Facebook donne la possibilité aux 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.

6.3 PARTIE 2 : A propos des réseaux sociaux décentralisés

6.3.1 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.

-
- Diaspora³ : 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.
 - TENT⁴ : C'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.

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 cœur 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 logique :

- metadata
 - content
 - 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 applications.

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

4. <https://tent.io/about>

- Les entités sont les utilisateurs de Tent, ils autorisent des applications, établir des relations et lire/publier des posts les entités sont définie avec leur entité URL ; https/http URL liée aux métadonnées.
- Chaque entité a un ou plusieurs serveur qui la représente.
- Les posts sont les atomics unité de contenu dans tent chaque post est un JSON, ils ont des fichiers attacher.
- Tent c'est une machine de lecture en 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*⁵ 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 y a 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 Turbulences⁶ : 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.
- Freenet⁷ : 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 ne sont pas des réseaux sociaux distribués mais ils ont un design distribué.

5. <http://thehackernews.com/>

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

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

6.3.2 Réseautage social décentralisé en ligne

Description d'un style d'architecture distribuée

Dans un cadre de réseau social distribué [1] 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 URIs, ils peuvent effectivement être stockés sur des serveurs différents.

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 URIs 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 [Re-cordon 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 d'accueil qui montre leur réseau, des activités et des photos en ligne sociaux.

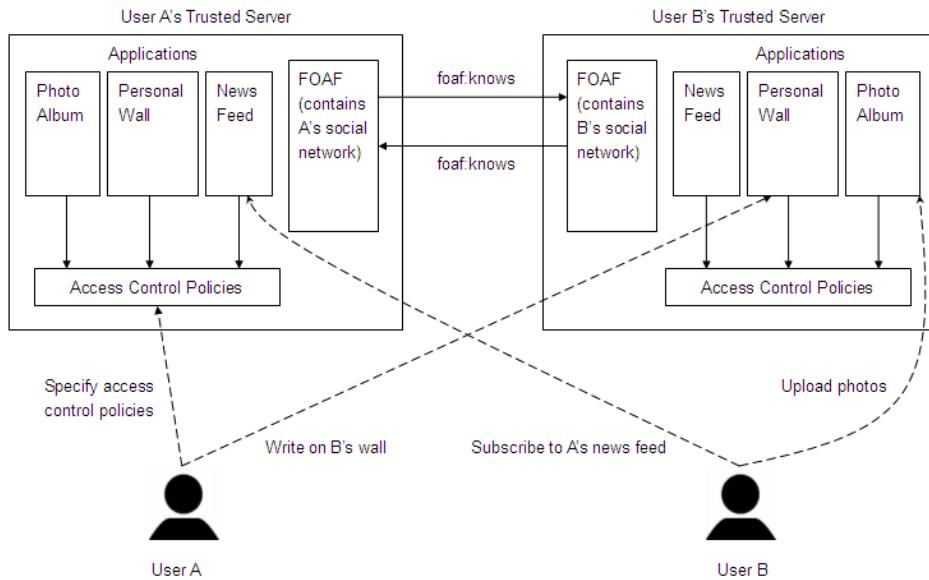


FIGURE 6.3 – A framework of decentralized online social networking

6.4 Conclusion du chapitre 6

Ces alternatives de réseaux sociaux décentralisés prendra-t-elle suffisamment pied pour constituer 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 plateforme, 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 projets expérimentent à la décentralisation appliquée aux réseaux sociaux présentent peut être une première réelle tentative, à la fois social et technique d'optimisation, des outils de réseautage social.

Le team Stample souhaitent avoir deux versions une centralisée stable en changeant l'ergonomie de l'apprentissage et en l'offrant des nouvelles fonctionnalités. Puis, un Stample décentralisé sera mis en ligne dès qu'il soit bien stabiliser et tester.

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 scalabilier. 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 crée un compte à partir d'une autres plateforme... J'ai eu des sous tâches pour les notifications, les commentaires .J'ai pu résoudre les problèmes des membres développeur coté 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*⁸, 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 des bonnes connaissance en Lisp, Ruby, Clojure, ect... Scala était le premier langage de ce genre avec lequel j'ai débuté.

J'ai développé en plus ma première page web personnelle⁹ 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 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 GitHub¹⁰, pour me proposer des suggestions personnelles ou de m'envoyer des mails depuis cette page.

8. <https://github.com/metanote>

9. <http://ideas2d.com/moncef/home.php>

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

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 Gant¹¹, ect...

Cette expérience s'est avérée très intéressante, dans le sens où j'ai pu participer à des réunions, qui m'ont fait connaitre 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, ect... et par conséquence, j'ai eu l'envie de lancer ma propre Startup plus tard.

11. <http://www.ganttproject.biz/>

MISE AU POINT

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-rdf¹².
- J'ai seulement tester le Read Write Web play rww-play¹³.

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.

12. <https://github.com/stample/banana-rdf>
13. <https://github.com/stample/rww-play>

Bibliographie

- [1] Kanghao Lu Oshani Seneviratne Tim Berners-Lee Ching-man Au Yeung, Ilaria Liccardi. Decentralization : The future of online social networking. *School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK. Decentralized Information Group, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*, pages 25,30,31, 2008.
- [2] Jennifer Larcher. Facebook : Chute d'audience et une perte de plusieurs millions d'utilisateurs. <http://www.begeek.fr/facebook-chute-daudience-et-une-perte-de-plusieurs-millions-dutilisateurs-91030>, 2013.
- [3] Sam Ruby lLeonard Richardson. *RESTful web services*. Safarir, 2007.

ANNEXES

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

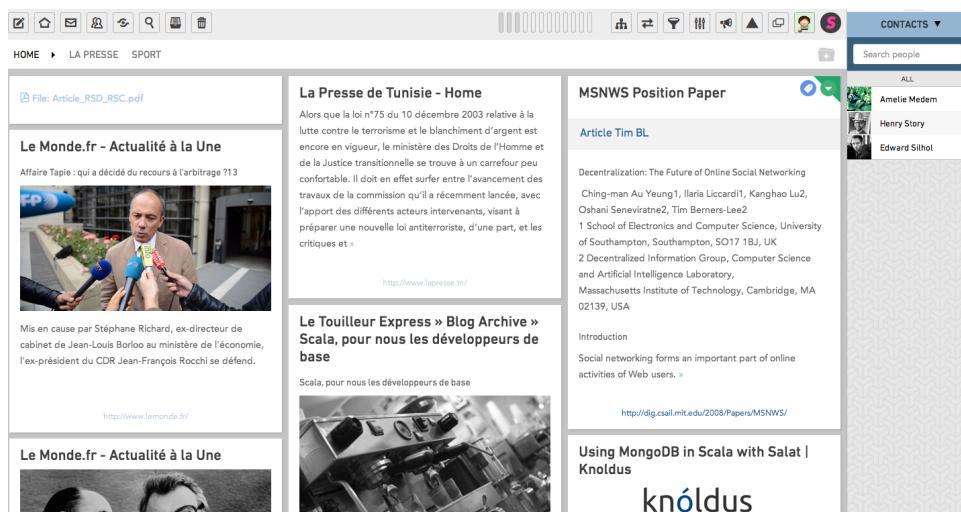


FIGURE 6.4 – My Stample Profil

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

The screenshot shows the Flowdock workspace profile interface. At the top, there are user icons for Jonathan, Edward, Amelie, Henry, mathieu, Sébastien, and Moncef. Below this is a search bar and a message input field. The main area displays a list of users and their recent activity:

- 4a6cd85** fixe date with jon metanote 2 days ago • Stample • fixPrivacy
- ab8d6eb** Minor CSS edwardsihol 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 les autres commentaires par contre j suis pas pour me trimbaler 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
- 68e078f** Merge branch 'fixPrivacy' into preProd edwardsihol 2 days ago • 11 more commits • Stample • master
- 655194e** Merge branch 'fixPrivacy' of github.com:edwardsihol/Stample into fixPrivacy 9eefef7 Minor fix edwardsihol 2 days ago • 11 more commits • Stample • master
- 68e078f** Merge branch 'fixPrivacy' into preProd 655194e Merge branch 'fixPrivacy' of github.com:edwardsihol/Stample into fixPrivacy

At the bottom, there is a message input field with 'Chat in Main...' and a 'Send' button.

FIGURE 6.5 – My Flowdock Profil

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

The screenshot shows the Trello workspace profile interface. At the top, there are sections for 'Notifications' and 'Boards'. Below this is a 'Show sidebar' button. The main area displays a board with several sections:

- BACKEND** (Done: 2, In Progress: 0, Total: 2)
- URGENT** (Add a card...)
- DOING** (Users can ask to receive emails for notifications. Add a card...)
- SECURESOCIAL** (Handle notifications (when someone added you, when receiving a Stample, etc...). Add a card...)
- NOTIFICATIONS** (Password reset: notify user in app if wrong email. Fix problem with routes when loading clipping when not logged in (remember John you told me about this). OAuth: Twitter. Add a card...)
- REMINDERS** (SPEC. Users must be reminded to a category. Add a card...)

FIGURE 6.6 – My Trello Profil

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

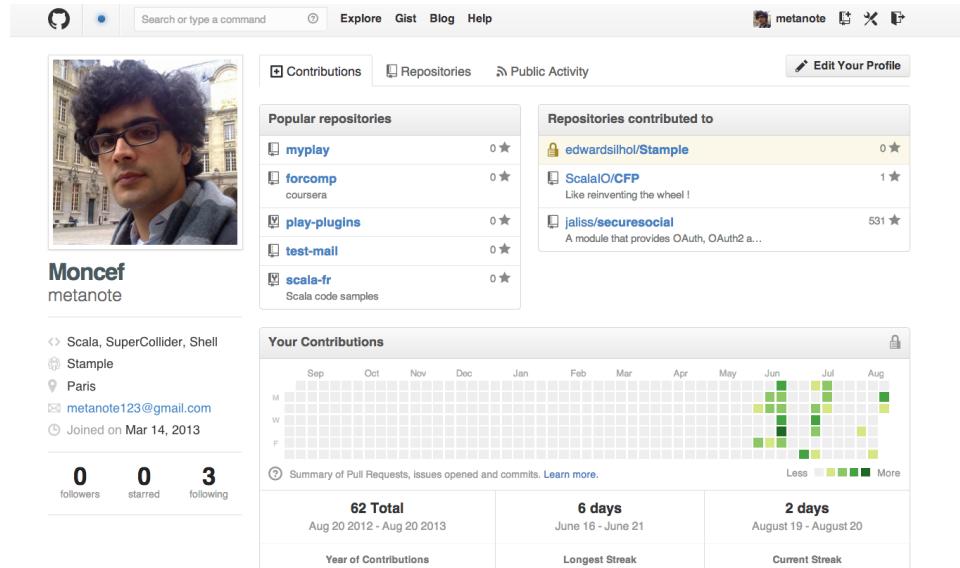


FIGURE 6.7 – My GitHub Profil

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

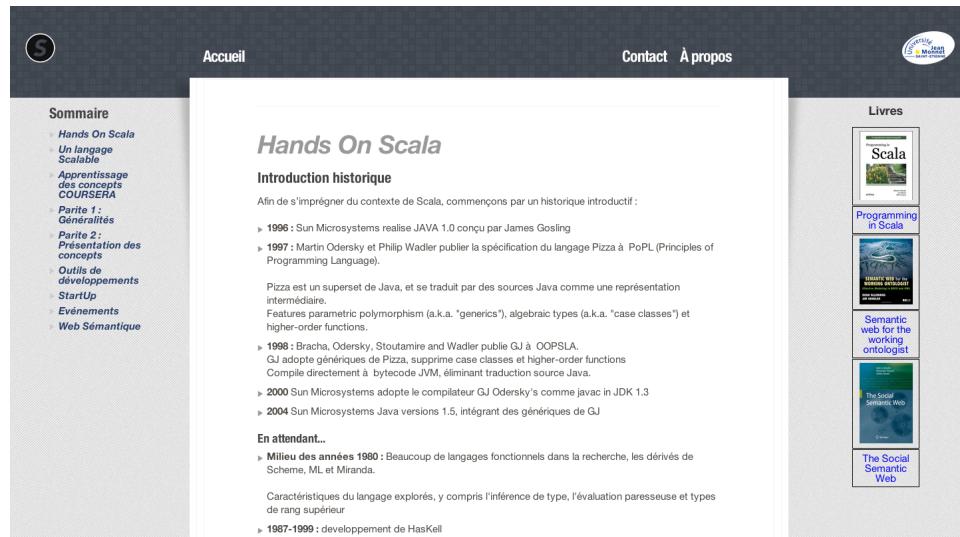


FIGURE 6.8 – Ideas2d my personnel web page

Maquette Stample User Profile.

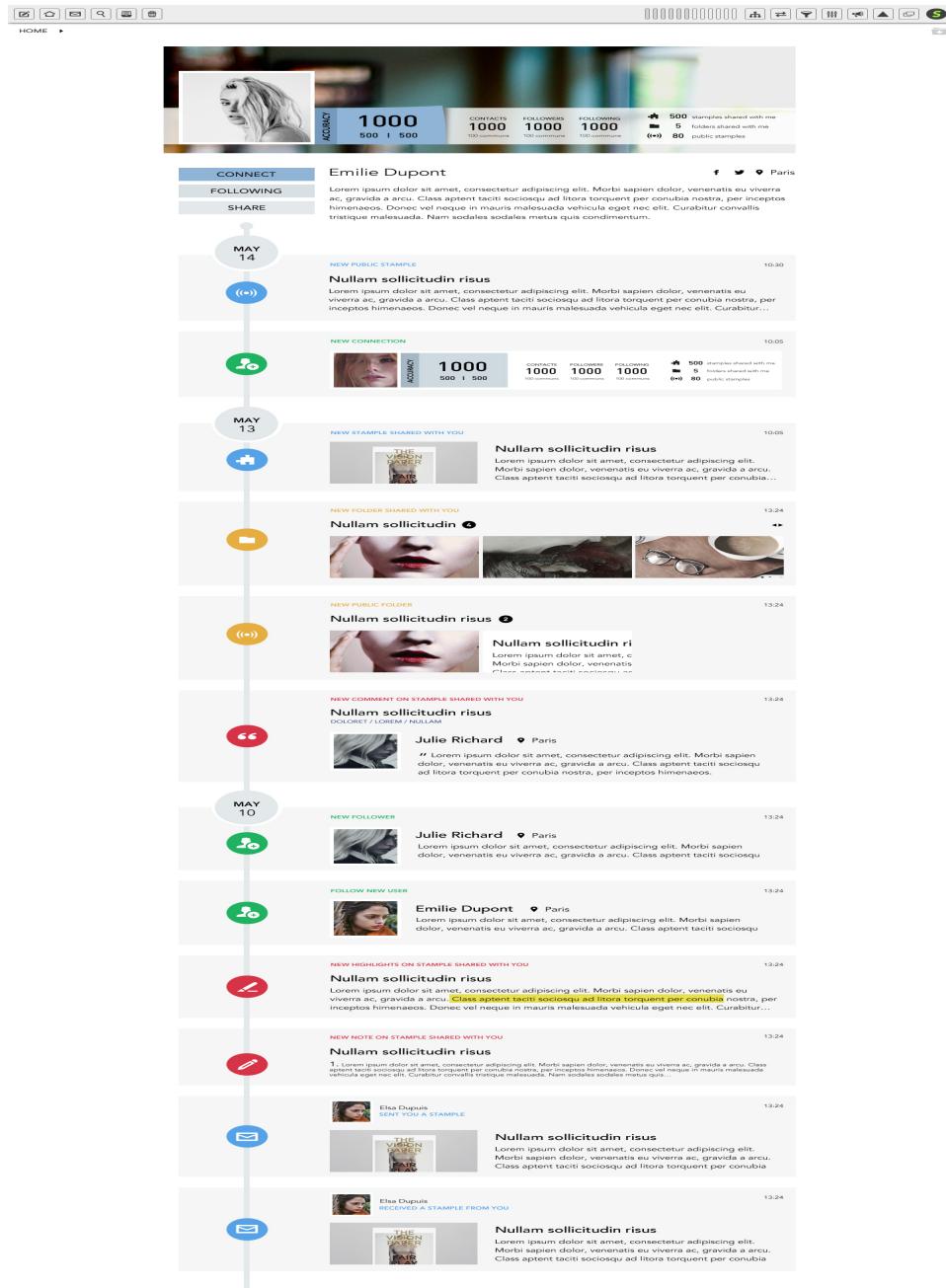


FIGURE 6.9 – Stample User Profile

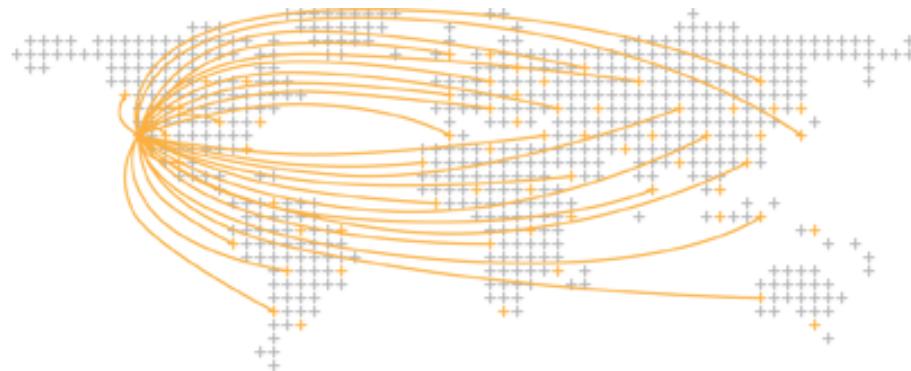


FIGURE 6.10 – Centralized network [This Not]

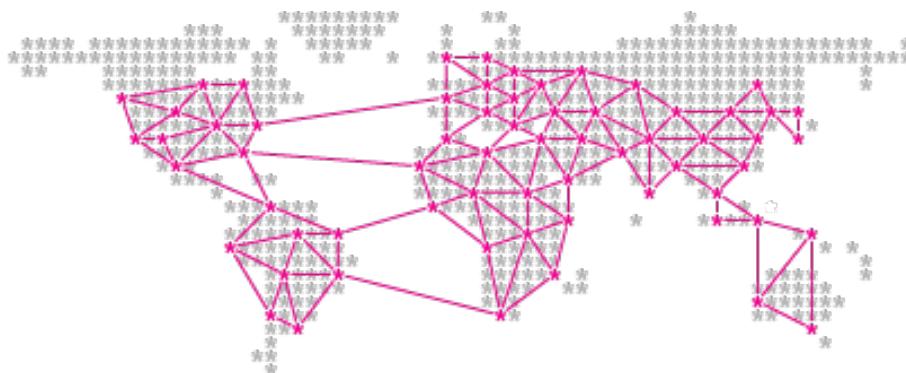


FIGURE 6.11 – Distributed network [This]