Rapport du projet groupe G32

2015-2016

– Pluss, la plateforme multi-réseau sociaux –

Student:	Iffrig-Bourfa Dylan
E-Mail:	dylan.iffrigbourfa@etu.enseeiht.fr

Student:	Ihamouine Raphael
E-Mail:	raphael.ihamouine@etu.enseeiht.fr

Student:	Roudaut Axel
E-Mail:	Axel,roudaut@etu.enseeiht.fr

Student:	Bourhim Ilyas
E-Mail:	Ilyas,bourhim@etu.enseeiht.fr

Student:	Pidutti Alexandre
E-Mail:	Alexandre,pidutti@etu.enseeiht.fr

Student:	Madoui Yassin
E-Mail:	Yassin,madoui@etu.enseeiht.fr

Student:	Moutaouakil Soufiane
E-Mail:	Soufiane,moutaouakil@etu.enseeiht.fr

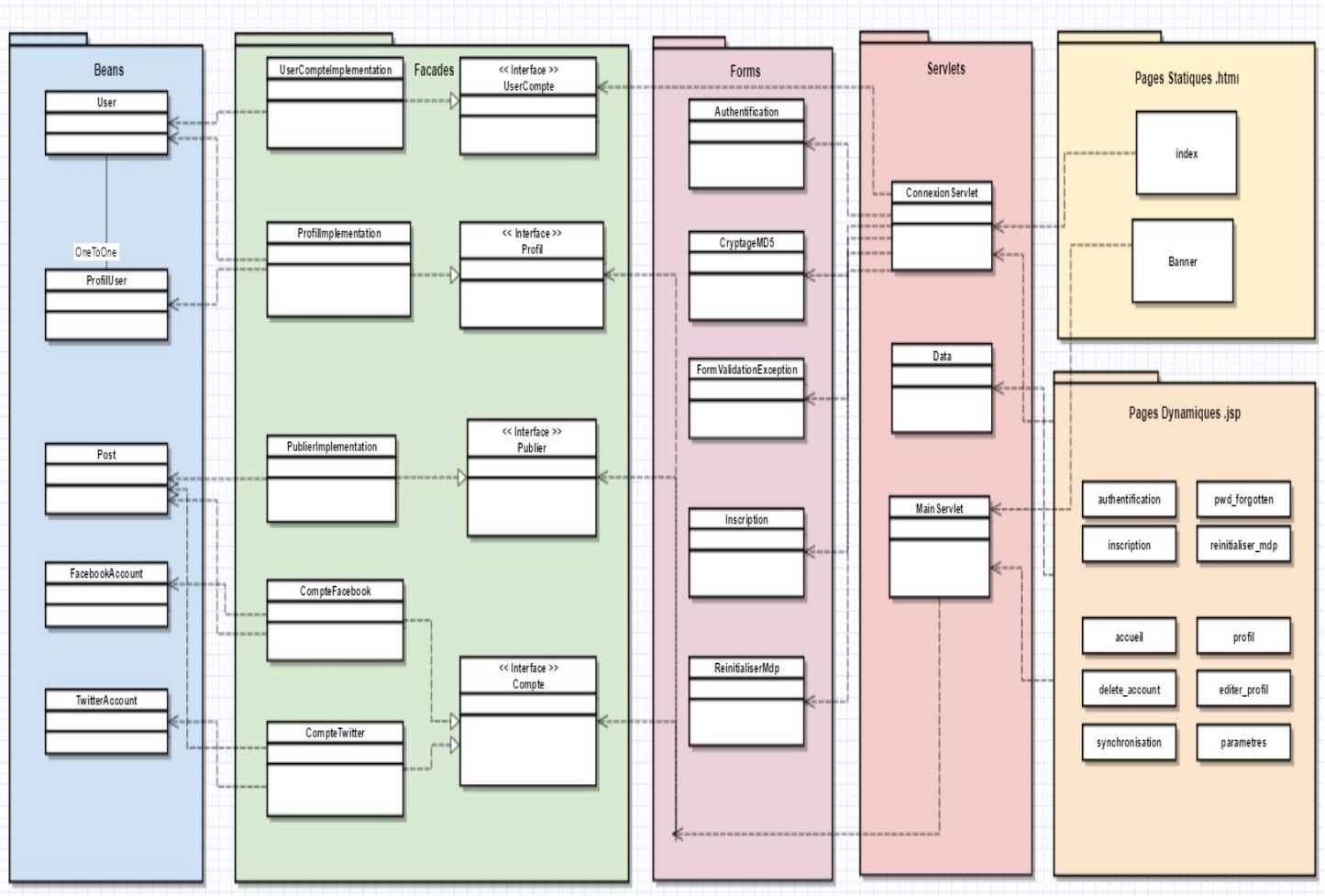
Table of Contents

Introduction

L'objectif de cette application Web est de centraliser l'ensemble des réseaux sociaux d'un utilisateur (premièrement Facebook puis Twitter) sur une seule et même application : Pluss. Et pouvoir à partir de cette application Web, publier sur l'ensemble des réseaux sociaux voulus de l'utilisateur d'un seul clic de souris. La fonctionnalité secondaire est de récupérer l'ensemble des flux voulus des divers réseaux sociaux de l'utilisateurs sur la page d'accueil du site Web Pluss.

Conception et modelisation du probleme

1) Une architecture MVC



Fig, 1: diagramme de classe générale

Nous avons assez naturellement suivi un modèle Modele Vue Controleur ici le package *servlet* correspond au controleur, la *facade* avec le *form* correspondent au modele et pour finir les *jsp* et *html* sont les élements de la vue.

Ce diagramme peut se décomposer en 3 sous-parties selon 3 fonctionnalitées majeurs du site: la partie inscription/connexion, la partie synchronisation avec les réseaux-sociaux et la partie édition de son profil perso Pluss.

2) Les principaux choix réalisés

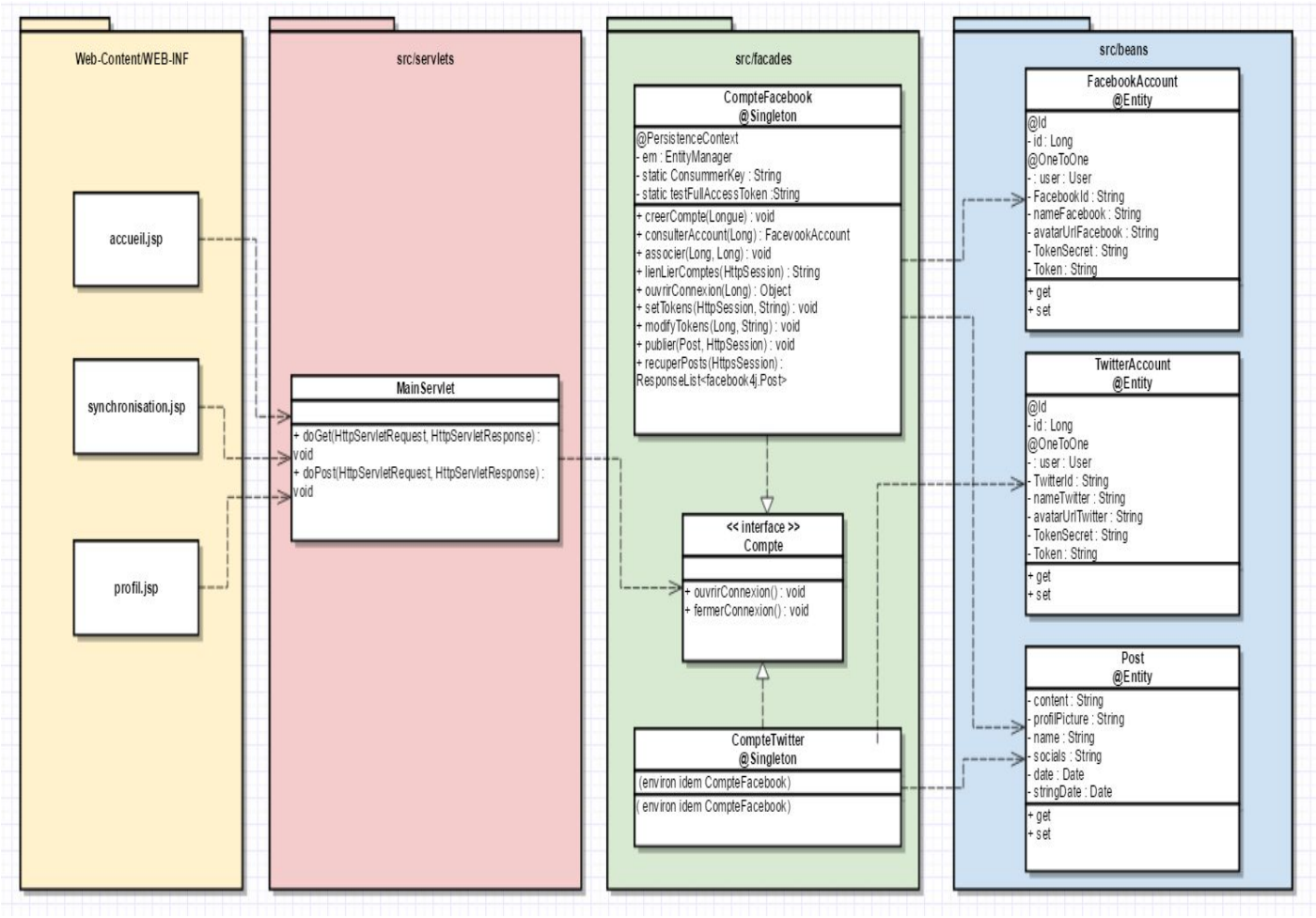


Fig. 2: diagramme de classe de la synchronisation des comptes

Dans cette partie de l'application, nous avons le lien avec les différents comptes de l'utilisateur. Chaque utilisateur a un compte Facebook, et un compte Twitter. Des lors d'une premiere synchronisation ses informations liées à son compte son enregistrés dans la base de donnée et à chaque connexion à pluss, la connexion au différente API sera automatiquement effectuée. Les classes CompteFacebook et CompteTwitter sont sensiblement identique, leurs méthodes permettent d'effectuer des opérations de base (ouvrir une connexion, recuperer le flux, encoder les données selon le bon format…).

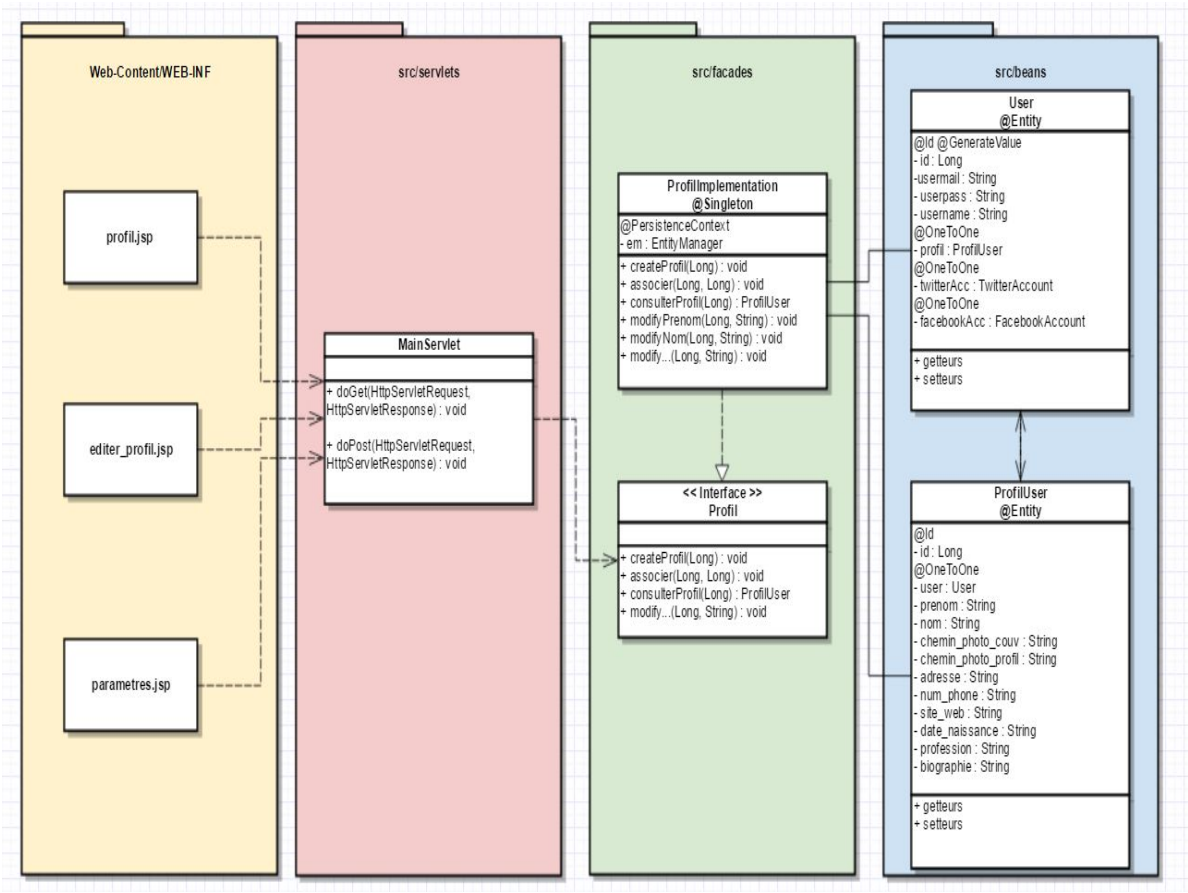


Fig. 3: diagramme de classe du profil utilisateur

Une fois inscrit ou connecté au site pluss, on peut personaliser sa page profil en cliquant sur le bouton "
"mettre à jour" ainsi un entity beans "vide" se créé avant d'acceder à editer_profil.jsp. On peut alors modifier chacun des champs disponibles (nom, prenom, adresse, etc...) ainsi que ses photos de profil et couverture.
Pour celà, on consulte la base de donnée (avec em.find) puis on la modifie (grace au setters). Les nouvelles données s'affichent alors dynamiquement après avoir clique sur les boutons "mettre à jour" pour les textes ou "Editer" pour les photos.

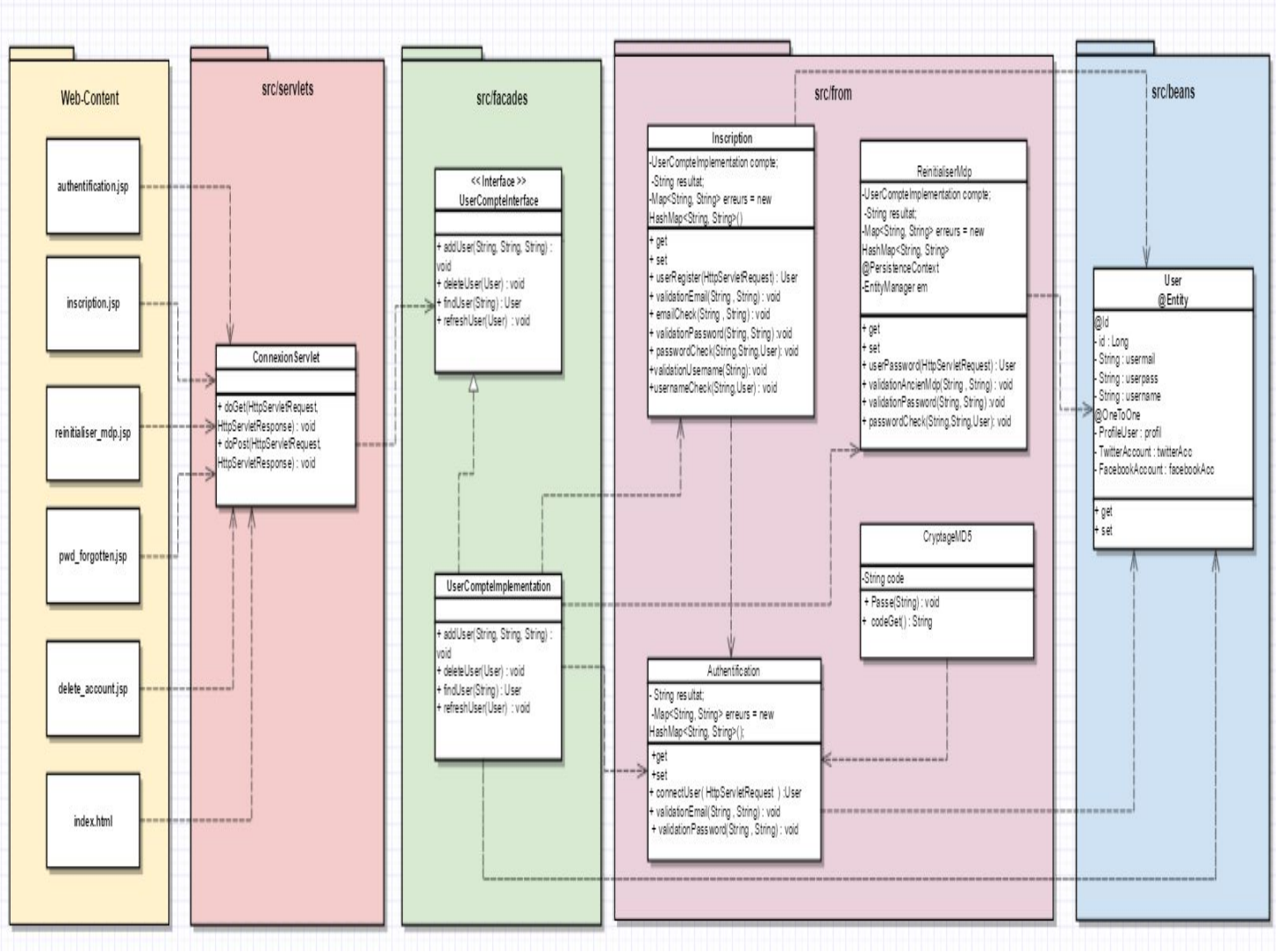
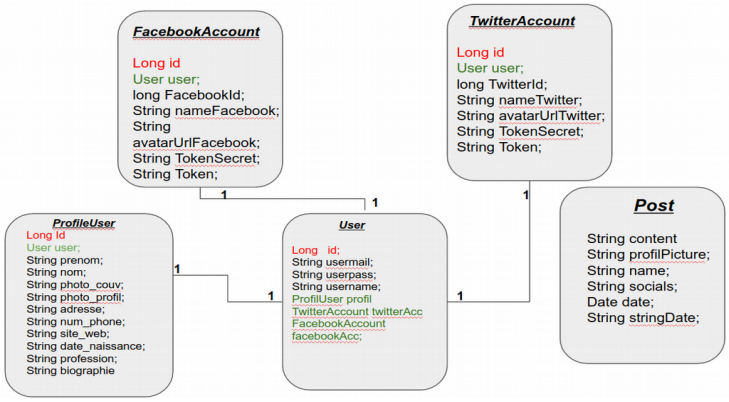


Fig. 4: diagramme de classe de l'inscription

Lors de la connexion à pluss l'utilisateur n'a pas tout de suite acces au pannel il doit d'abord se connecter ou creer un compte si il n'en a pas. Il aura la necessité de fournir des données tel que son adresse email, un pseudo, et un mot de passe d'au moins 6 carracteres. Dès lors que son compte est créer il peut alors se connecter et profiter de la plateforme. On obtient donc ce shéma relationnel de base de donnée



Technologies utilisées

1) Un Front end adaptable et malleable

La partie design et conception du site a été réalisé dans les premières semaines du projet, nous avons tout d'abord réalisé des maquettes sur des logiciels de DAO (dessin assisté par ordinateur) suite à ces travaux l'idée retenue à donc été un design simple, responsif et léger.

Les normes Material Design proposés par google répondaient directement à ces attentes là. Nous nous sommes donc logiquement tourné vers un framework HTML/CSS "Material Design Lite". La prise en main étant aisée et rapide, nous avons pu de manière assez rapide obtenir un résultat proche de nos maquettes.

Durant le développement des petites modifications ont été entreprises pour des soucis d'ergonomie. En effet nous avons poussé l'idée encore plus loin en décidant de rendre le site entièrement responsive c'est à dire, chercher avoir une expérience d'utilisation optimale sur n'importe quel support (Ordinateur, Smartphone, tablette…) Il se trouve que le material design se prêtait bien à cette contrainte, pour s'y faire nous avons définit des règles CSS propres à chaque type de support. Un petit surplus de travail qui en valait la chandelle.

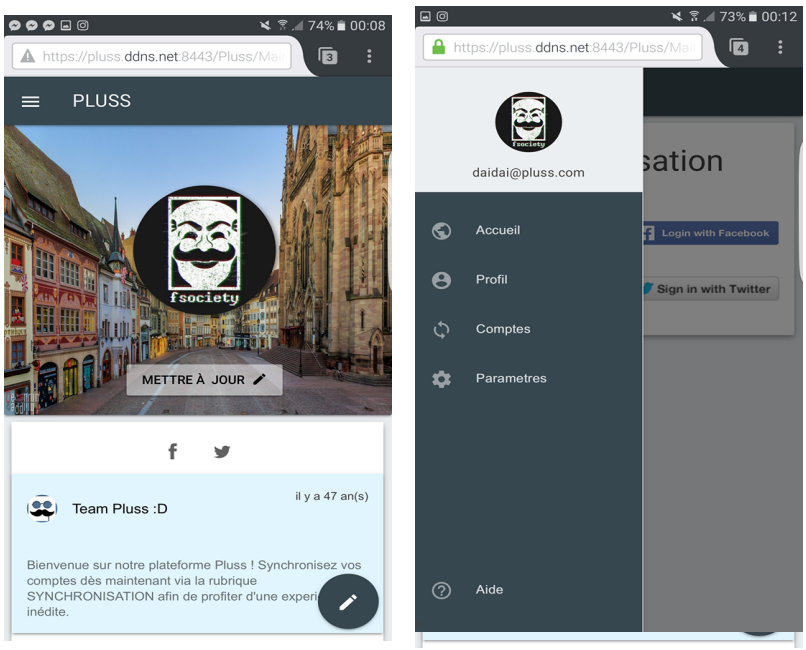


Fig. 5: rendu sur mobile

Nous avons également décidé, par nécessité ou par soucis du détails, d'intégrer des technologies annexes tels que le Javascript, l'idée est de réduire le nombre d'appel et opération du serveur en effectuant des pré contrôles, et la structuration des posts côté client. Nous pouvons citer par exemple les contrôles lors de l'inscription/connexion. On va vérifier que les adresses email correspondent bien avec le format standard d'une adresse email, on va vérifier également que le mot de passe fait bien plus de 5 caractères… Ces contrôles sont bien sur également re vérifié coté serveur. L'utilisation du javascript nous a également permis d'avoir des petites animations de styles l'ouverture du modal de publication se fait à l'aide de fonctions JS.

Enfin les différentes technologies associés tel que Ajax et JQuery nous ont permises d'afficher les différents posts de manière asynchrone (gestion du flux de l'utilisateur). Le serveur génère en réalité de simples fichiers JSON* composés des posts récupérés via les API des réseaux sociaux. Et c'est donc ensuite l'Ajax et le JQuery qui vont permettre, à partir du fichier JSON, de générer le code HTML afin d'afficher les posts sur l'interface de l'utilisateur. Note : ici nous avons préféré utiliser JSON à XML pour des questions de simplicité et performance cependant les concepts derrières sont exactement les mêmes, et ce choix à été volontairement effectué de cette façon pour permettre d'appliquer les notions vu en cours

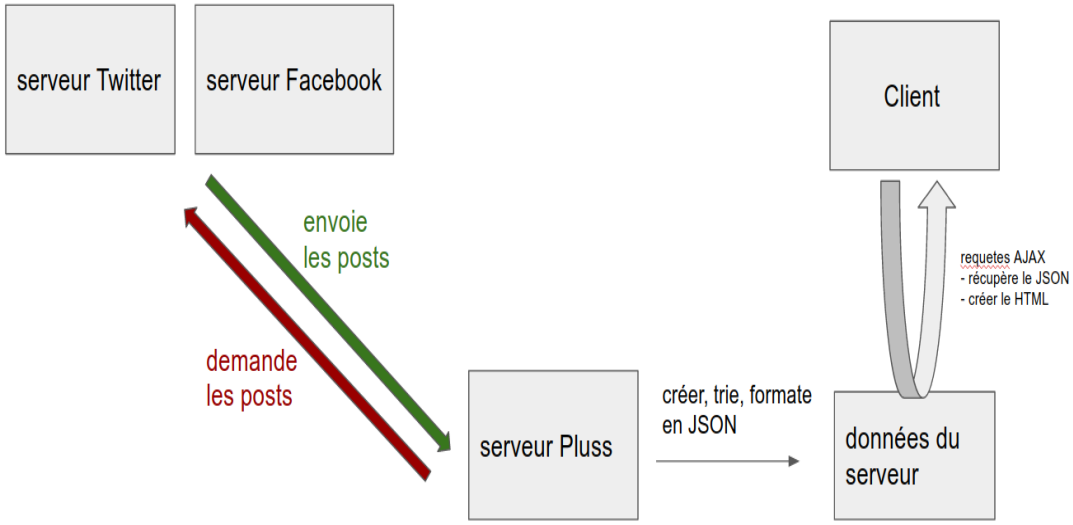


Fig. 6: représentation de la recuperation de flux et l'affichage des posts

2) Un BackEnd structurée propre à JEE

Durant notre projet nous avons été amené à utiliser de nombreuses librairies annexes, nous pouvons citer twitter4J et facebook4j qui permettent de faire l'interaction avec les API REST de twitter et facebook. Ces librairies proposent à peu près assez de méthodes pour effectuer toute les requêtes que les API proposent. Elles permettent, entre autres, d'ouvrir des connexions à l'API, de récupérer et poster un flux, le tout directement en Java.

La connection à des réseaux sociaux depuis une application tierce comme Pluss se fait via la technologie Oauth. Lors de la synchronisation twitter/facebook va demander à l'utilisateur si il accepte d'autoriser l'application pluss à obtenir des droits sur son compte. En acceptant cela le serveur va alors générer un couple de Token qui permettent d'ouvrir des connexions à l'API avec les droits choisis. Suivant le réseau social nous auront plus ou moins de difficulté à obtenir des droits sur les comptes, cela est détaillé dans la suite.

Fig. 5: rubrique de synchronisation aux réseaux sociaux

En ce qui concerne le stockage des données des utilisateurs nous avons choisi l'algorithme de hachage MD5 qui se veut (jusqu'à preuve du contraire) actuellement irréversible. Les posts récupérés sont également stockés dans un fichier avec un nom spécifique et unique propre à chaque utilisateur, créé par un jeu de cryptage MD5 également.

Enfin une grande partie du travail a été également de gérer la récupération des posts. Nous avons implémenté des fonctions de tri et de classification par date qui permettent de classer l'ensemble des posts (tout réseaux sociaux confondus) via un même critère. Il reste donc plus qu'à appliquer les différents filtres sur les attributs du post pour choisir ce que l'on veut afficher du côté de l'utilisateur.

```
202    public String JsonEncode(ArrayList<beans.Post> listPosts){
203        JSONArray obj = new JSONArray();
204        for(beans.Post p : listPosts){
205            try {
206                JSONObject postJ = new JSONObject();
207                postJ.put("profilPicture", p.getProfilPicture());
208                postJ.put("name", p.getName());
209                postJ.put("content", p.getContent());
210                postJ.put("time", p.getStringDate());
211                postJ.put("socials",p.getSocials());
212                obj.put(postJ);
213            } catch (JSONException e) {
214                e.printStackTrace();
215            }
216        }
217        JSONObject full = new JSONObject();
218        try {
219            full.put("posts",obj);
220        } catch (JSONException e) {
221            e.printStackTrace();
222        }
223
224        String jsonText = full.toString();
225        System.out.println(jsonText);
226        return jsonText;
227    }
228
229
230
231     public void trierPost(ArrayList<beans.Post> listPosts) {
232        Collections.sort(listPosts, new Comparator<Post>() {
233            public int compare(Post m1, Post m2) {
234                return m1.getDate().compareTo(m2.getDate());
235            }
236        });
237    }
238
```

Fig. 7: Fonctions JsonEncode et trierPost

Principales contraintes

1) Les problemes rencontrés, les solutions envisagées

Initialement nous avons opté de manière intuitive pour plusieurs servlets à la place d'une servlet unique pour la connexion, l'inscription, le mot de passe perdu, etc. Il nous a été conseillé de passer sur une servlet unique, et nous avons du faire un peu plus que du copier-coller pour que ça fonctionne. Les problèmes rencontrés ont majoritairement été du a une méconnaissance des technologies utilisées. Par exemple, le sendRedirect qui se fait une fois l'accolade fermante atteinte et uniquement pour des pages statiques. Contre le getRequestDispatcher qui s'execute automatiquement, avec des pages dynamiques aussi, il nous a donc fallu comprendre la nécessité de mettre un return après l'appel de cette méthode.

Le second type de problème est lié entièrement au fait que nous utilisons des API. Nous dépendons de ces API, donc des contraintes de sécurités premièrement mais aussi et surtout de toute modification de conditions d'accès au API. En cas de changement des API nous devrions certainement revoir notre code.

Enfin toujours lié à ces API nous avons eu le problème d'accès au API. Toutes n'étant pas proposées en Java (existence d'une documentation bien plus dense en PHP par exemple), il nous a fallu utiliser et nous familiariser avec les bibliothèques 4j (twitter4j et facebook4j). Toujours dans cette lignée, les conditions d'accès et d'utilisations des données d'un compte d'un utilisateur Facebook ou Twitter sont bien sur soumises à conditions, il nous a donc fallu nous dépêtrer seul au milieu de toutes ces conditions (Notamment par respect de la vie privée des utilisateurs Facebook ou Twitter, car nous aurions pu parser directement le code HTML Facebook/twitter et récupérer les mot de passes et identifiants des utilisateurs.).

Par exemple, Twitter rend obligatoire une connexion sécurisée (ssl donc en https) ce qui nous a coûté quelques problèmes surtout pour configurer notre Jboss en https. Facebook quant à lui, avec une volonté de protection de vie privée, exige une vérification du code (une vidéo permettant de contrôler l'usage de l'API avec réponse sous deux semaines) avant d'autoriser une accès à des options supplémentaires.

Ensuite du côté mise en page statique nous avons eu un petit temps de familiarisation avec les technologies. Sachant notamment que notre application se veut responsive et utilise du JavaScript. D'autres petits soucis ont été rencontrés, notamment du côté de la BDD, dont l'utilisation n'est pas forcément plus simple qu'un modèle DAO, notamment pour tout ce qui est rafraîchissement de la BDD, des combines permettant de set l'id de l'User pour qu'il puisse être récupérer, etc. Enfin pour faire le lien avec la partie suivante des améliorations possibles, nombreuses étaient pensées et prévues à la création du projet. On pourrait par exemple penser à un serveur mail permettant de vérifier l'authenticité, utile à l'inscription et la perte du mot de passe par exemple. Mais en réalité, le nombre de personne effective dans le groupe étant réduit, cette fonctionnalité n'a pas pu être implémentée en temps et en heure.

2) Les améliorations possibles

Les améliorations dans notre code sont multiples et variées, comme dit précédemment. On peut par exemple inclure un système de messagerie privée, permettant aux utilisateurs Pluss de communiquer entre eux. Enfin idée présente à la genèse du projet, on aurait voulu intégrer plus de réseaux sociaux ; LinkedIn, Instagram, Google+, etc.

Nous avons aussi une partie entamée mais non finie qui est l'ensemble des filtres de pages, la délimitation du contenu privé/public du site a été faite de manière provisoire. Nous aurions voulu crée une servlet spéciale, une classe contenant une liste de toutes les pages à accès restreint par exemple. De manière à toujours garder un projet malléable et structuré, ainsi l'ajout de n'importe quelle page web à notre site Web pourra être classée dans la partie à accès restreint ou public simplement en mettant le .jsp ou .html dans le bon dossier ou la bonne liste.

Une dernière idée aurait été d'installer un serveur Xmpp pour créer une messagerie instantanée interne, facilitant ainsi la discussions entre les différents utilisateurs de Pluss et donc rendant possible la discussion entre un utilisateur Twitter et un utilisateur Facebook.

Batterie de tests réalisés

Nous avons bien-sûr tout d'abord utilisé une méthode dure de débugage, au travers de la console. En affichant dans le terminal des éléments lors de l'exécution des méthodes. Comme par exemple pour comprendre comment utiliser le merge() ou le refresh() lors d'un changement de paramètres. Et voir ce changement effectif dans la BDD.

Enfin il a été voulu et il aurait été intéressant de réaliser un ensemble de batterie de test au moyen de Junit, pour deux principales raisons :

- Vérifier l'execution des méthodes premières au site et invisible autrement : récupération d'erreurs à l'inscription, etc.

- Permettre à l'ensemble des personnes codant l'application d'avoir une idée de la structure globale de l'application et de l'appel des différentes méthodes. L'ordre d'appel ou la nécessité d'appeler une méthode avant une autre (par soucis de logique) étant primordial, les tests permettent à cette personne de se référer à un modèle.

Conclusion

L'objectif principale de l'application web à été réalisé avec succes, nous pouvons publier à partir d'une plateforme sur plusieurs réseaux sociaux. Nous avons meme pu aller plus loins en recuperant les flux des différents réseaux et en les affichants de maniere dynamique. Il reste néanmoins quelques tests à réaliser pour garantir une certaine fiabilité. L'amélioration de la robustesse de notre application est à envisager. De nombreuses features peuvent également etre ajouter par la suite. Nous débordons d'imagination pour faire évoluer le projet.

Retours personnels

Ihamouine Raphaël

“Cette application web était un projet intéressant, dynamique et très motivant dans sa globalité, on a eu la possibilité de mêler notre intérêt propre pour un service désiré, qui plus est se voulant moderne et réellement utile, à un apprentissage et une mise en pratique concrète de technologies de programmation diverses et variées (Du HTML au JAVA en passant par du JSON par exemple). Enfin être mêlé à une équipe de sept personnes permet de nous mettre au sein de conditions de travail réelles, formatrices et de plus guidées par un module de gestion de Projet permettant d’avoir des retours sur la mise en place d’un tel projet au sein d’une entreprise.”

Madoui Yassin

“ Le sujet choisi était très intéressant car aucune application existante ne proposait un tel service . Grâce à ce projet, nous avons pu developper de nouvelles connaissances sur les applis web. L'ambiance du groupe m'a aussi permis de m'améliorer car mes collègues ont pu m'apprendre des techniques qu'ils maitrisaient mieux que moi. Le résultat du travail est très satisfaisant même si tout n'est pas fonctionnel.”

Roudaut Axel

“ Tout d'abord, le choix du sujet m'a semblé très bien, je suis content que nos professeurs aient accepter ce projet de plateforme multi-réseaux sociaux. Par ailleurs, le retard des enseignements sur les technologies web par rapport à nos avancements en developpement fut un inconvénient majeur de cette formation au web par un projet long. En effet, les cours de html puis jee/jpa arrivaient à chaque fois plus de 2 semaines après le debut du developpement de notre site avec ces technos. De ce fait, on a du à plusieurs reprise revenir sur nos pas pour nous adapter aux consignes imposées (par rapport au serveur JBoss/Tomcat et à la base de donnée MySQL/HSQL)”

Iffrig Dylan

“ Ce projet de programmation Java EE, a pu me faire découvrir un nouvel environnement web. Mon interet pour la matiere m’a vite pousser à prendre les devant d’autant plus que le sujet a été choisi par nos propres soins. Je regrette légèrement le temps que j’ai pu perdre à debbuguer le serveur jboss, et le serveur de base de donnée. Un point positif, tout ce temps passé m’a permis de vraiment me familiariser avec jboss j’ai pu aider de nombreux camarades dans qui étaient confrontés à des problemes toujours plus surprenants. La gestion en groupe de 7 est un travail très formateur et pas toujours évident également.”

Bourhim Ilyas

“ Ce fut la premiere fois que je m’investissait dans un projet avec autant de personne dans le groupe. Il est difficile de se répartir équitablement le travail et selon les niveau de chacun certains se retrouve avec beaucoup plus de travail que d’autres. Mais en générale ce fut une bonne expérience et je suis content du résultats obtenu.”

Moutaouakil Soufiane

“ Ce projet m'a permis de se projeter dans une vraie expérience de développement web surtout qu'en parallel nous devions faire de la gestion de projet pour s'organiser ainsi de chercher à apprendre l'utilisation de langages et techniques qui ne sont pas forcément vu en cours à l'école. Personnellement, j'ai aimé participer à l'innovation de cette application web car c'était notre propre idée et non pas imposée et nous nous sommes bien amusés à la developper.De plus nous étions chacun dépendant de l'autre dans le projet ce qui nous a permis d'avoir une vue globale de l'avancement de toutes les parties de projet”

Pidutti Alexandre

“C était très intéressant de découvrir le java EE a travers ce sujet. Bien que tout nos espérances n aient pas abouti nous sommes parvenus à créer une application que nous voulions tous avoir. De plus l ambiance dans l équipe était agréable et propice au bon déroulement du projet.”