heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Nestlé

Twitting Cocoa Plants

Proposé par Nestlé

Auteur	**: Christian Masson**
Filière	**: Informatique**
Orientation	**: Logicielle**
Lieu et date	**: Yverdon-les Bains, le 26.07.2012**
Professeur responsable	**: Stéphan Robert**

Maison Cailler
DEPUIS 1819
VALLÉE DE BROC, SUISSE

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Remerciements

J'aimerai remercier Maison Cailler pour leur proposition de projet ainsi que l'entreprise Koubachi pour l'aide qu'ils ont apporté et leur disponibilité.

Merci à Sandy Vibert, Yvan Da Silva et Michel Masson pour leurs aides lors des corrections de ce rapport, David Zéni, Valentin Gaillard, Nathalie Wuethrich pour leurs idées et prêt du matériel lors du développement.

Merci aussi à Matthieu Reussner et au club de robotique pour les locaux ainsi que l'accès au serveur pour les tests.

Finalement, merci à ma famille et mes amis qui m'ont soutenu durant tout ce projet.

Résumé

« Twitting Cocoa Plants » est une application Android, développé pour Maison Cailler, qui permet de faire de la publicité. Cette application, nommée « Cocoa Tweet » utilise des capteurs ainsi qu'un téléphone Android, implanté au pied d'une plante de Cacao en Équateur.

« Cocoa Tweet » est capable de récupérer les données de capteurs ainsi qu'une photo de la plante et de poster des messages la concernant sur Twitter. Ces messages sont générés en fonctions des données reçues des capteurs.

Pour configurer cette application, un serveur web a été créé. Il permet de définir tous les paramètres de « Cocoa Tweet », les messages à Twitter ainsi que leurs conditions d'envoi.

Abstract

« Twitting Cocoa Plants » is an Android application made for advertisement which has been developed for Maison Cailler. This application, named « Cocoa Tweet », uses sensors and an Android phone, located at the foot of a cocoa tree in Ecuador.

« Cocoa Tweet » is able to recover data from sensors, a picture of the plant and post messages on Twitter. These messages are generated using the data received from all sensors.

To configure this application, a web server was created. It defines all the parameters of « Cocoa Tweet », the tweets and their conditions to be tweeted.

Table des matières

1 Introduction

Ce projet consiste à développer un système pouvant envoyer des tweets pour informer sur l'état environnemental d'une plante de cacao située dans une plantation en Équateur.

Ce système doit être réalisé en prenant en compte qu'il va être déployé dans une région humide, chaude et n'ayant comme réseau que du 3G.

Il va servir à faire de la publicité pour Maison Cailler qui est une filiale de Nestlé produisant du chocolat de haute qualité. Cela va leur permettre de montrer une étape supplémentaire de la production du chocolat, de la pousse du cacao à la création du chocolat (1).

Il a été nécessaire de trouver des composants compatibles pour réaliser ce projet. Ainsi que comprendre le fonctionnement de Twitter et d'Android. Il a finalement fallu faire un serveur pour configurer le téléphone à distance et comprendre le principe de synchronisation de Google nommé « C2DM » qui signifie « Cloud to Device Messaging ».

2 Matériel utilisé et fonctionnement

Le choix du matériel est une étape cruciale pour un tel projet.

2.1 Matériel utilisé

Le choix du téléphone s'est vite tourné vers le Sony Xperia S (2) sous Android. Cet appareil est récent et est l'un des rare à posséder une caméra 12 méga pixels, ce qui en fait un bon choix si l'on veut poursuivre ce projet avec de l'analyse d'image.

Le choix des capteurs fut plus compliqué, vu la diversité des informations nécessaires. Il est devenu limité quand la décision de trouver un ensemble de capteurs ressemblant à une station météo a été prise. Le problème avec ce type d'installation est qu'elles sont généralement sur des fréquences différentes de celles disponibles sur le téléphone et ne sont dès lors pas compatible avec ce projet. Un ensemble presque complet de capteurs a été trouvé. Il s'agit à la base d'un capteur pour surveiller l'état d'une plante, ce qui se rapproche grandement de ce projet, et qui fonctionne sur du Wifi, que notre téléphone possède.

Ce capteur est celui de Koubachi (3), une spin-off de l'école polytechnique de Zürich.

Comme le montre la Figure 1, il est capable de lire :

- L'humidité du sol
- La température
- La luminosité

Figure 1: Capteur de Koubachi. Tiré de (4)

Cependant il manque l'humidité de l'air pour compléter les mesures voulues.

Pour cette mesure il a été choisi de se fier à un site météo de Yahoo (5), ce qui permet de diminuer la charge de travail au niveau de l'entretien des appareils.

2.2 Fonctionnement

Pour pouvoir utiliser ces différents composants, il faut comprendre leurs fonctionnements.

2.2.1 Sony Xperia S

Ce téléphone fonctionne actuellement sous Android 4.0.4.

D'après les tests effectués, son autonomie est dans la moyenne actuelle des Smartphones Android. Il est capable de tenir en veille et en mode avion (tous réseaux désactivés) pendant deux semaines en ayant consommé environ 50% de la batterie. Dans le chapitre Android nous allons expliquer le fonctionnement de l'application qui tournera sur cet appareil.

2.2.2 Capteur Koubachi

Ce capteur a été créé pour aider au maintien de l'équilibre d'une plante en eau, luminosité et température. Nous avons juste besoin de ces valeurs pour les interpréter différemment.

Pour récupérer ces valeurs il faut commencer par comprendre l'architecture du capteur.

Comme montré dans la Figure 2, le capteur envoi ses informations à un serveur qui va analyser ces données pour informer l'utilisateur sur les mesures à prendre pour sa plante.

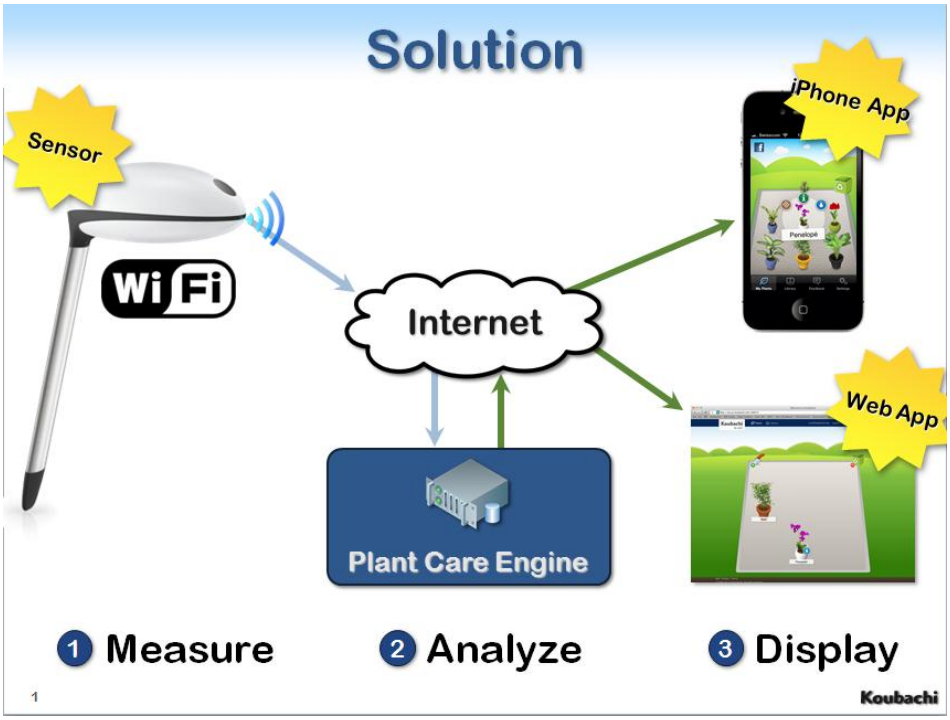

Figure 2: principe de fonctionnement du capteur, tiré de la présentation de Koubachi

Dans notre utilisation du capteur, nous allons laisser celui-ci envoyer ses informations au serveur et aller directement sur celui-ci pour récupérer les informations utiles pour notre application.

La particularité de notre infrastructure implique un système d'envoi de données qui n'est pas optimal. Comme l'on peut le voir dans la Figure 3.

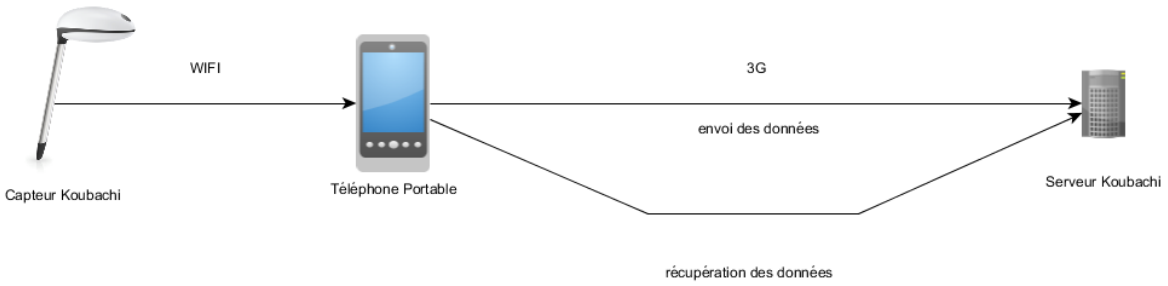

Figure 3: Schéma d'envoi et récupération des données

L'information passe à travers le téléphone pour être envoyée au serveur, puis le téléphone va chercher l'information sur le serveur. Ceci est dû à la conception du capteur, qui utilise un canal sécurisé pour la transmission de ses données. Il n'est donc pas possible de les récupérer à la volée et de les analyser directement sur le téléphone.

3 Twitter

Twitter est un système de microbloging, ce qui veut dire que l'on peut écrire au reste du monde de petits messages de 140 caractères maximum. Pour pouvoir utiliser ce système, il faut avoir un compte et s'authentifier, pour nous, en tant que développeur et inscrire l'application auprès de Twitter.

3.1 Création d'un compte utilisateur

Pour pouvoir utiliser Twitter il faut commencer par créer un compte. Pour cela il suffit d'aller sur https://twitter.com/ et de remplir la zone d'inscription comme montré sur la Figure 4 et confirmer ces informations sur la page suivante présentée à la Figure 5.

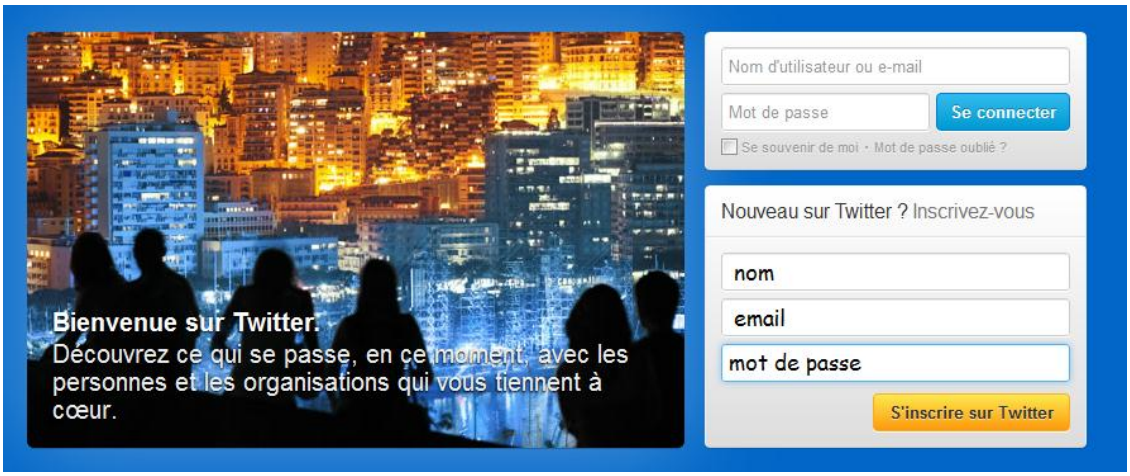

Figure 4: page de création du compte Twitter

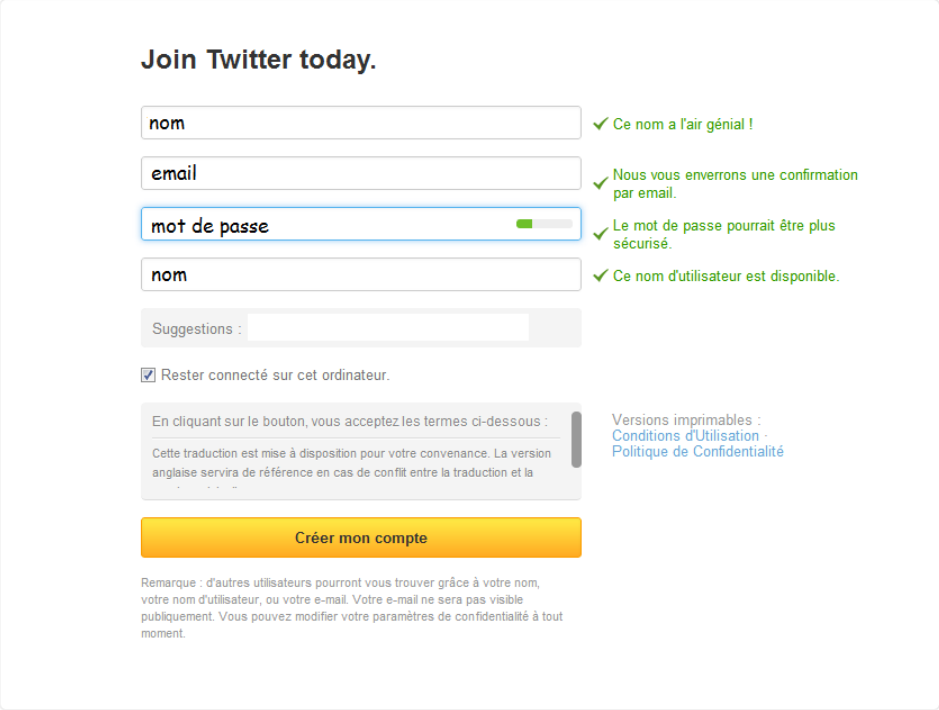

Figure 5: page de confirmation de création du compte Twitter

3.2 Création d'un compte développeur

Pour pouvoir utiliser Twitter dans une application, il faut avoir un compte développeur chez Twitter. Cela permet d'avoir un jeton d'authentification pour le développeur, qui doit être utilisé dans toutes les applications utilisant Twitter.

Pour créer un compte il suffit de se connecter sur https://dev.twitter.com/ avec le compte créé précédemment pour Twitter.

3.3 Enregistrement d'une application sur un compte développeur

Les applications voulant utiliser l'API Twitter doivent être référencées. Pour faire cela il faut l'enregistrer sur le site des développeurs de Twitter. Il faut se connecter sur https://dev.twitter.com/ et aller sur « Create an app », où il faut remplir les champs comme montré dans la Figure 6.

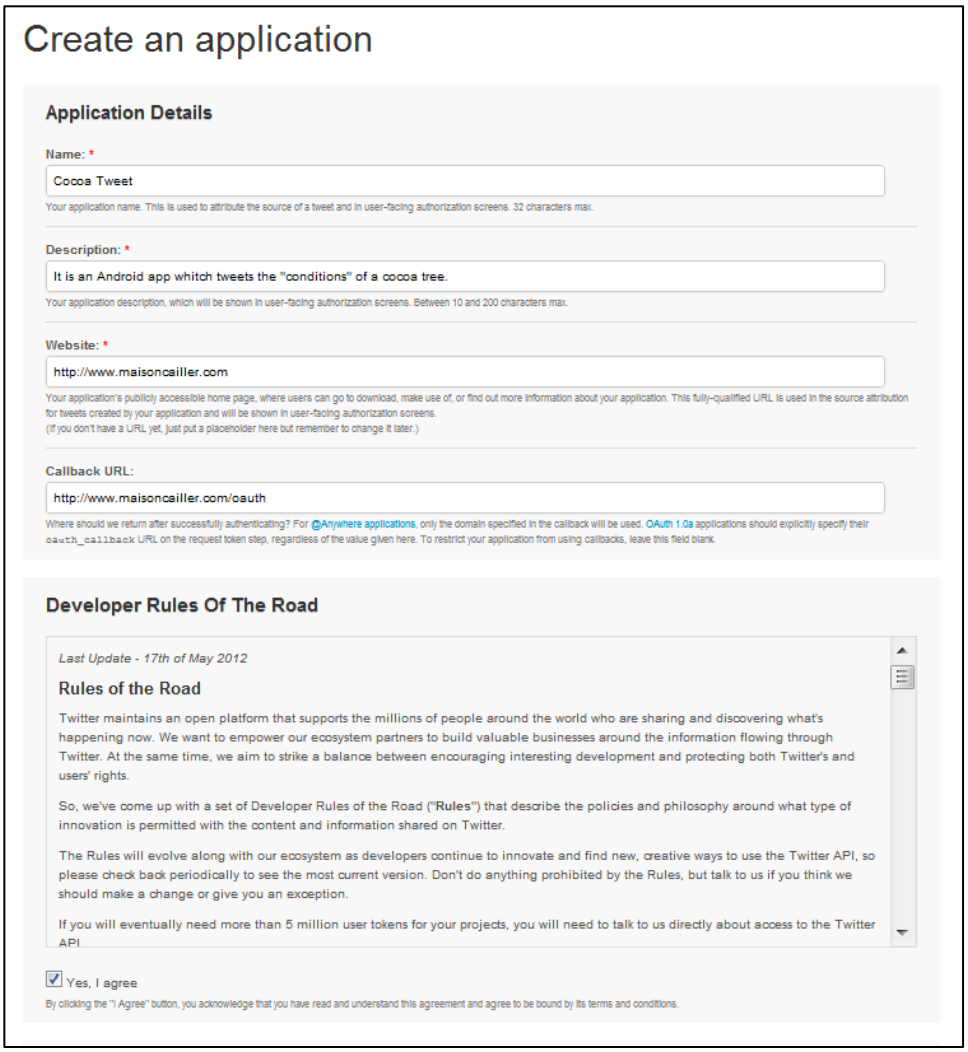

Figure 6: enregistrement d'une application chez Twitter

Ceci nous donne les jetons nécessaires au fonctionnement de l'application, présenté à la Figure 7.

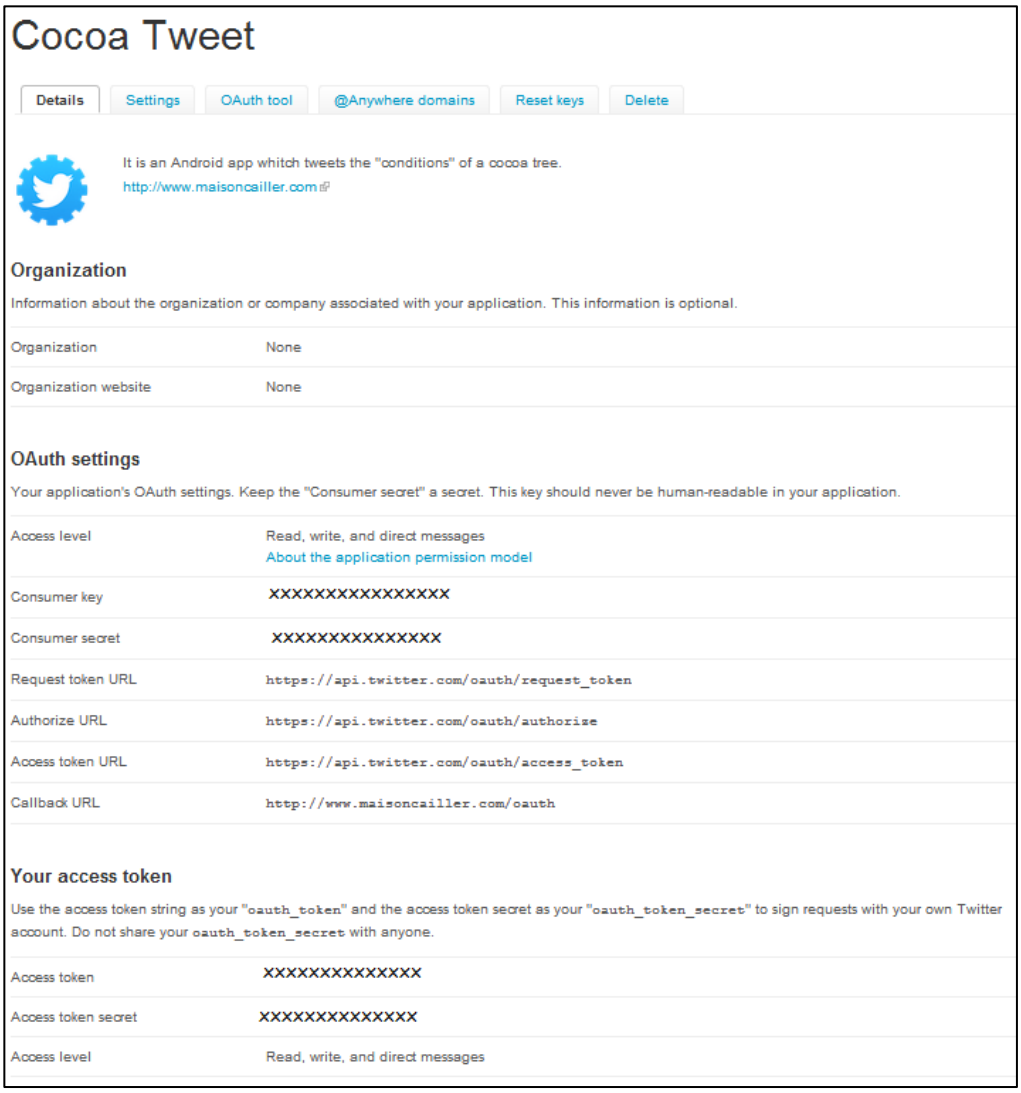

Figure 7: affichage des jetons pour l'application

Nous avons maintenant tous les jetons nécessaires pour authentifier l'application sur Twitter.

3.4 Authentification par jetons

Twitter utilise ce principe d'authentification pour que les applications tierces ne gardent pas en mémoire les noms d'utilisateurs et mots de passes de leurs utilisateurs. Ce principe est plus compliqué à implémenter car il faut respecter plusieurs étapes d'échanges d'informations entre l'application et Twitter. Il faut aussi ouvrir une page internet au milieu de l'authentification et, de ce fait, en général, sortir de l'application pour aller sur internet.

Ce système d'authentification n'est pas fait pour les téléphones mobiles, car cela implique de changer d'application deux fois pour s'authentifier. Il est toutefois possible d'inclure une interface web dans l'application, ce qui est la solution qui a été adoptée pour cette application. Ceci permet d'avoir une expérience utilisateur et un « look and feel » identique au reste de l'application.

L'authentification par jetons se passe de la manière suivante.

Pour commencer, l'application envoie une requête à Twitter avec comme paramètres la clé et le secret de l'application ainsi que l'adresse de callback pour la suite de l'authentification. Le serveur renvoie deux jetons, l'un d'eux est l'authentification et l'autre est le secret comme l'on peut le voir sur la Figure 8.

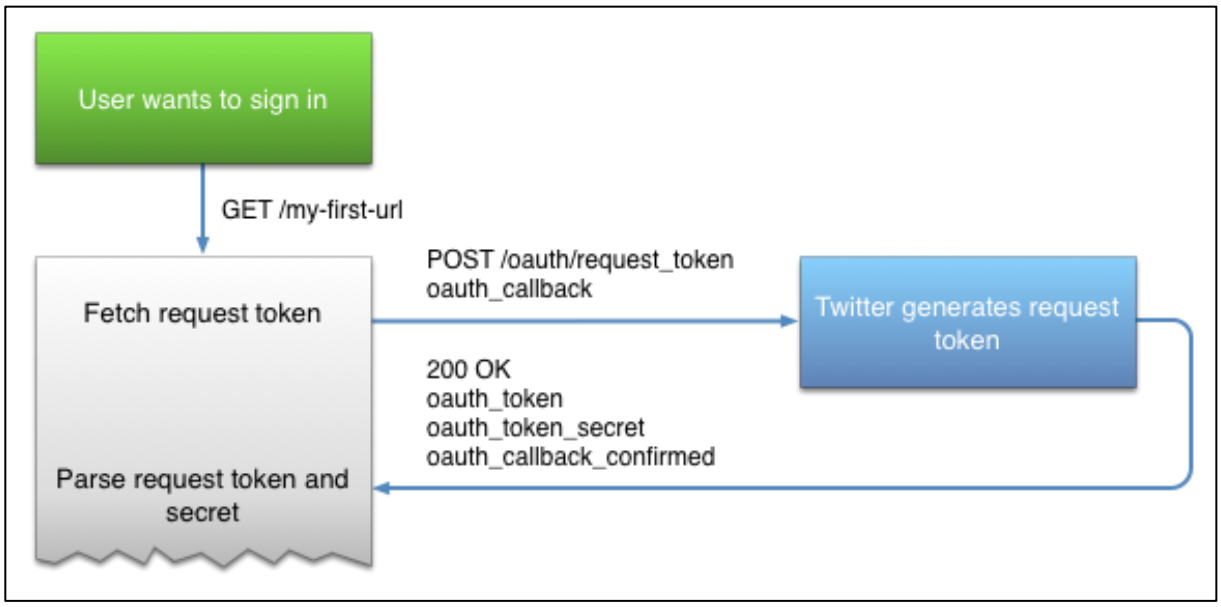

Figure 8: 1ere étape d'authentification Twitter. Tiré de (6)

La deuxième étape, représentée sur la Figure 9, consiste à permettre à l'utilisateur de se connecter sur Twitter. Pour ce faire, l'application va demander au serveur de Twitter de lui donner une URL basée sur le jeton d'authentification reçu à l'étape précédente et va afficher la page de connexion visible sur la Figure 10.

Apres la connexion de l'utilisateur, le serveur Twitter renvoie, sur le lien de call back donné au début, deux nouveaux jetons. Ces jetons sont l'un pour l'authentification et l'autre pour la vérification.

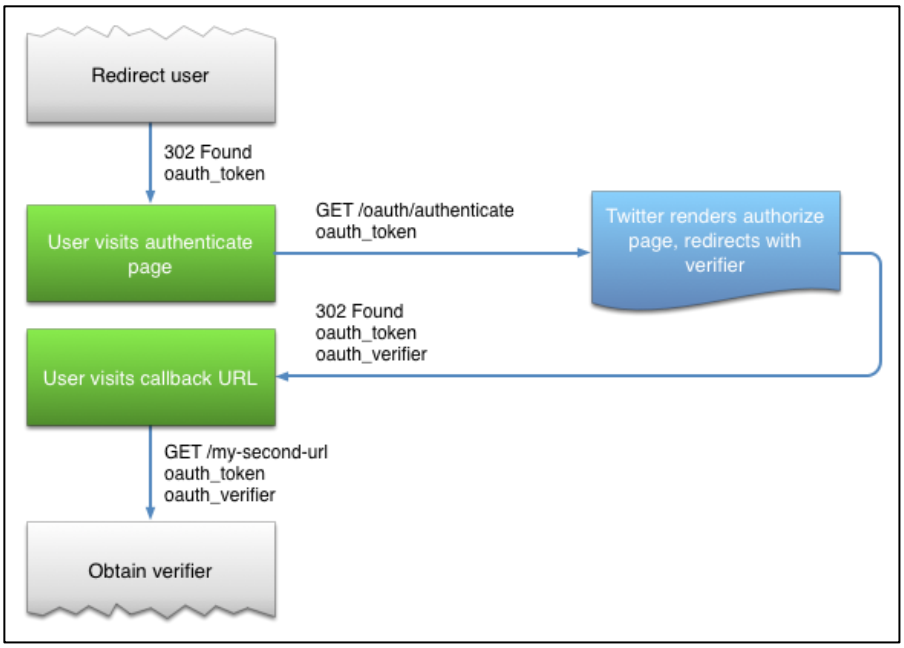

Figure 9: 2ème étape d'authentification. Tiré de (6)

Figure 10: authentification au compte Twitter depuis l'application

Comme le montre la Figure 11, la dernière étape consiste à obtenir les jetons finaux d'authentification et de secret du serveur à partir du jeton de vérification reçu à l'étape 2. Pour envoyer un tweet, il nous suffira d'utiliser ces deux jetons qui seront envoyés avec le tweet.

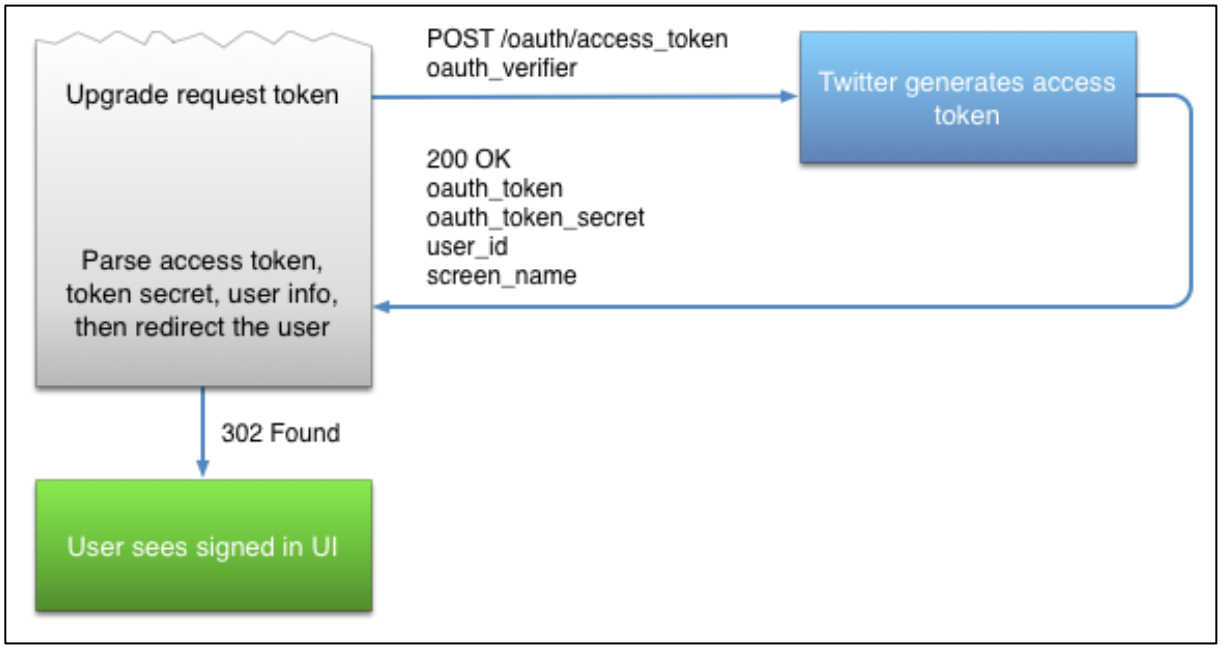

Figure 11: fin de l'authentification Twitter. Tiré de (6)

Ces deux jetons ne changent pas, sauf si l'utilisateur décide de les supprimer et refaire tout le processus d'authentification.

4 Android

Le téléphone utilisant le système d'exploitation Android, l'application demandée pour ce projet va être fait en Java pour Android, qui est une version modifiée et allégée de Java.

Pour pouvoir développer cette application il est souhaitable d'installer un environnement de développement pour simplifier la réalisation ainsi que les tests.

4.1 Eclipse

Eclipse est l'environnement de développement conseillé par Android. Ils ont aussi développé des plugins Eclipse pour simplifier certaines étapes de développement et de tests.

4.1.1 Installation de l'environnement de développement

La procédure qui suit est tirée de (7).

Eclipse peut être téléchargé sur http://www.eclipse.org/downloads/.

Une fois cette opération faite, Eclipse est une application contenue dans un dossier et ne nécessite aucune installation supplémentaire.

Il faut, ensuite télécharger le SDK sur http://developer.android.com/sdk/index.html, l'extraire et l'installer. Ce software développement kit est la base indispensable pour développer sous Android. Il permet aussi de télécharger et installer les différentes versions des plateformes utiles pour les tests et développements.

Il faut finalement ajouter le plugin « ADT » à Eclipse. Ceci se fait au travers du menu d'Eclipse. Il faut aller dans Help -> Install New Software, comme montré dans la Figure 12.

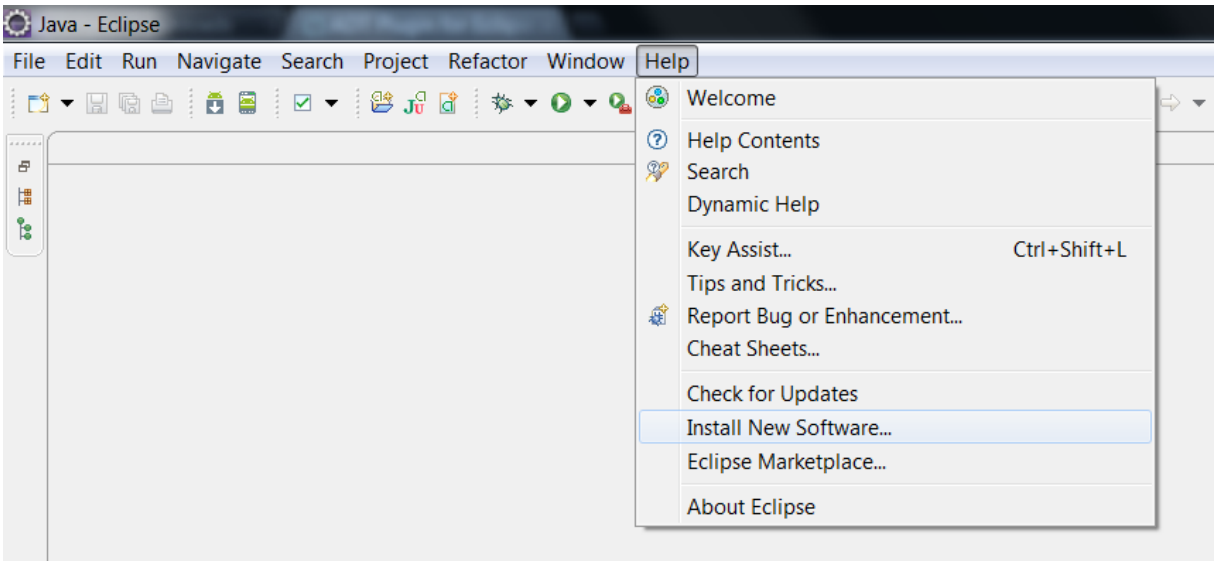

Figure 12: menu d'installation de l'ADT dans Eclipse

Il faut ensuite ajouter le lien https://dl-ssl.google.com/android/eclipse/ dans la dans la nouvelle fenêtre et sélectionner « Developer Tools », comme montré dans la Figure 13.

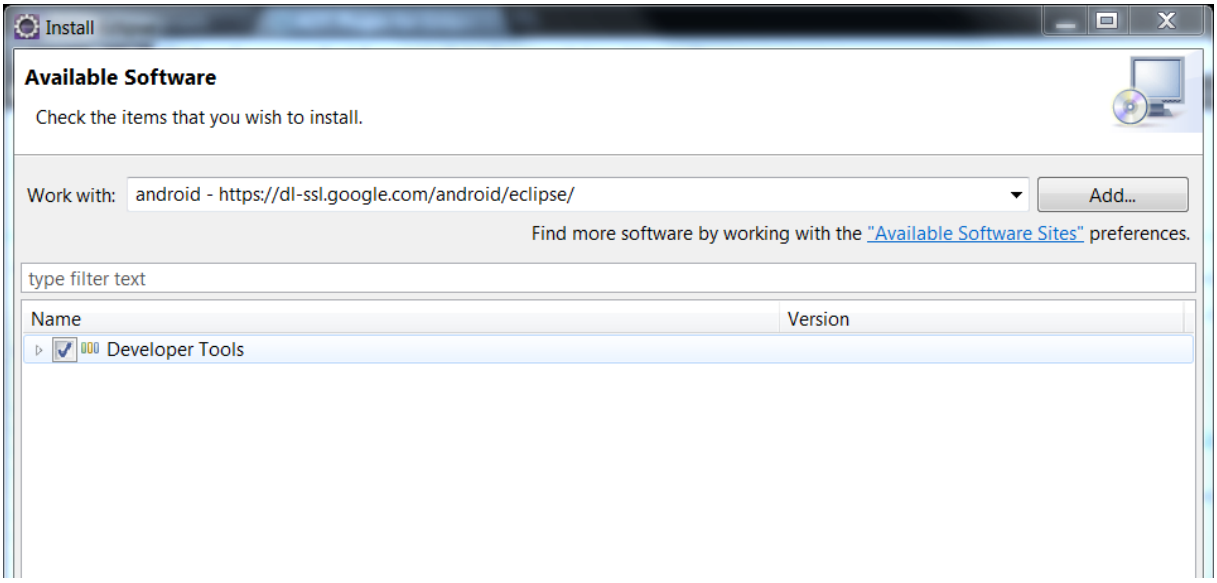

Figure 13: sélection du package ADT dans Eclipse

Ceci va installer le plugin et ajouter deux icones, visible dans la Figure 14. Ces icones permettent, pour celle de gauche, la gestion du SDK et, pour celle de droite, la gestion des environnements de test.

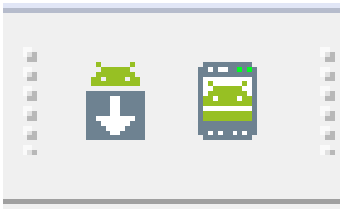

Figure 14: icones de l'ADT dans Eclipse

Pour finaliser cette installation, il faut encore indiquer dans les paramètres d'Eclipse le chemin vers le SDK installé auparavant. Ceci se fait dans le menu « Window -> Preferences », puis sous la rubrique Android.

4.2 Ce qu'il faut savoir sur la façon de coder sous Android

Avant de se lancer dans la programmation d'une telle application, il a fallu apprendre et comprendre le principe de fonctionnement du système Android. Ceci fut possible grâce à (8) ainsi qu'au site web pour les développeurs (9).

Android a développé plusieurs technologies simplifiant grandement la création des applications.

4.2.1 Les fichiers XML

Android a décidé d'utiliser au maximum les fichiers XML. Ils servent principalement à stocker les affichages, paramètres et variables de l'application. Il y a aussi un fichier XML indispensable qui est le manifeste. Celui-ci est décrit plus bas.

L'architecture de ces fichiers est faite pour pouvoir modifier simplement le comportement de l'application suivant certains facteurs. Pour les String, il est possible d'offrir simplement le support de plusieurs langues en spécifiant le nom de la langue dans le nom de chemin du fichier. Il est aussi possible de dire d'appliquer des paramètres différents suivant le téléphone ou son orientation.

Les vues peuvent aussi, en suivant la même procédure, être modifiées. Ceci permet d'adapter simplement le contenu à la situation sans devoir sans arrêt regarder dans le code dans quel sens le téléphone se trouve et allège énormément le code.

4.2.2 Les Intents

Un « Intent » est une intention. C'est grâce à ce mécanisme que l'on peut démarrer une activité ou un service. Il est possible de simplement lancer une activité ou d'attendre un retour après la fin de celle-ci.

4.2.3 Les Activités

La classe « Activity » correspond à ce que l'on voit à l'écran. Cette classe est créée la première fois qu'elle est appelée et est reprise à chaque fois que l'on veut l'afficher, comme expliqué dans la Figure 15. Il se peut que le système d'exploitation détruise l'activité s'il a besoin de ressources. Il faut donc toujours veiller à sauvegarder les changements effectué dans la vue ainsi que son état. Le fait de changer l'orientation du téléphone force aussi l'arrêt de l'activité pour la recharger avec la nouvelle vue. Ceci permet, comme vu plus haut, de prendre une autre vue si disponible, pour avoir un affichage plus agréable suivant la disposition de l'écran.

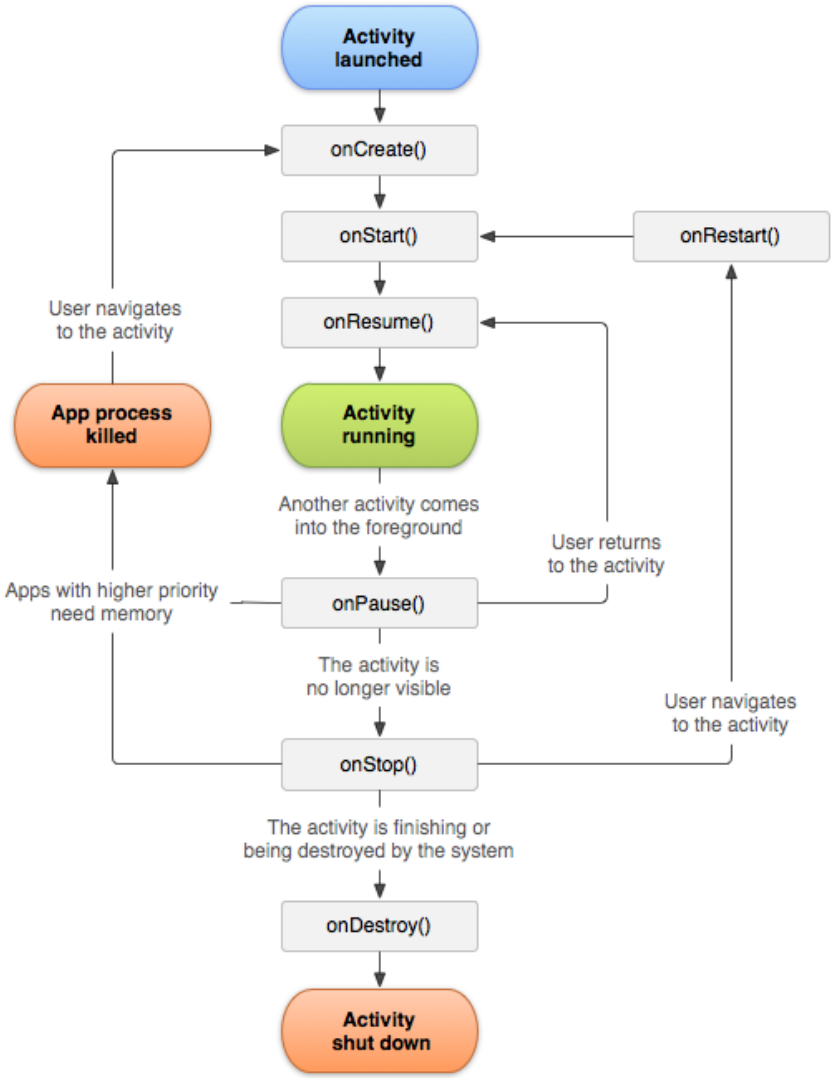

Figure 15: workflow d'une Activité, tiré de (10)

4.2.4 Les Services

La classe « Service » permet d'avoir une partie d'application qui est toujours active, même si l'application n'est pas lancée. Ceci est utile pour la synchronisation des données en arrière-plan ou permettre l'écoute de musique tout en navigant sur le Web par exemple. Le service est lancé grâce à un Intent et est actif jusqu'à ce qu'il décide de se terminer ou qu'il soit arrêté par l'application.

4.2.5 Les classes asynchrone

Les classes « Service » et « Activity » possèdent un mécanisme qui les arrête si elles sont inactives pendant plus de 5 secondes. Ce mécanisme est là pour permettre à l'utilisateur d'avoir toujours un système et un affichage réactif. Pour pouvoir faire des longues procédures comme des connexions réseau ou des transferts de fichiers, il faut utiliser des threads qui, une fois lancé, permettent de faire

ces traitement de façon asynchrone. Pour simplifier ce principe, Android propose les « AsynkTask » qui sont des threads qui ont la possibilité d'interagir avec l'affichage pour, par exemple, afficher le pourcentage de téléchargement.

4.2.6 La classe R

Cette classe est générée automatiquement et permet de faire la liaison entre les fichiers XML et le code Java. Pour faire le lien il suffit de noter « R.string.exemple » R est la classe, le « string » représente le type de donnée que l'on cherche et exemple est le nom de la variable.

4.2.7 Le manifeste

C'est le document de base d'Android. C'est ce fichier qui définit ce que l'application a le droit de faire et quelles classes peuvent être utilisées pour se connecter via des « Intents ». C'est sur ce document que l'Android Play va se baser pour répertorier l'application.

4.2.8 Les Alarmes

Une alarme est un mécanisme de réveil pour une activité. Cela permet de déterminer quand l'activité doit être démarrée. Cela permet simplement de retarder l'émission d'un « Intent ».

4.3 Tester une application

Une fois l'application créée, il faut savoir si elle s'exécute et sous quelles conditions. Pour cela Android propose deux possibilités.

Il est possible de créer simplement des environnements (téléphones, tablettes) virtuels et simplement lancer l'application dans ces environnements. Cette façon de faire a ses limites car les simulateurs n'ont pas tous les périphériques disponibles sur un appareil physique.

Il est également possible de connecter un téléphone ou une tablette à l'ordinateur pour tester directement sur l'appareil.

Etant donné que le programme développé lors de ce projet utilise la caméra, qui ne peut pas être simulé, les tests ont été directement faits sur le téléphone utilisé pour ce projet.

5 Cocoa Tweet

L'application a comme nom « Cocoa Tweet ». Elle est constituée de trois parties principales qui sont :

- La configuration complète de l'application
- L'Activité de tweet
- La mise à jour de la configuration via un serveur web

Ces trois parties fonctionnent ensemble et forment un tout.

À cause des spécifications, comme la qualité des photos, cette application a uniquement été développée pour le Sony Xperia S.

5.1 Architecture de base de l'application

Un UML complet de l'application se trouve en annexe.

Cocoa Tweet possède une activité de base qui permet la gestion de celle-ci.

La Figure 16 représente cette vue. Elle est composée de cinq boutons qui permettent la configuration et le test de différents composants.

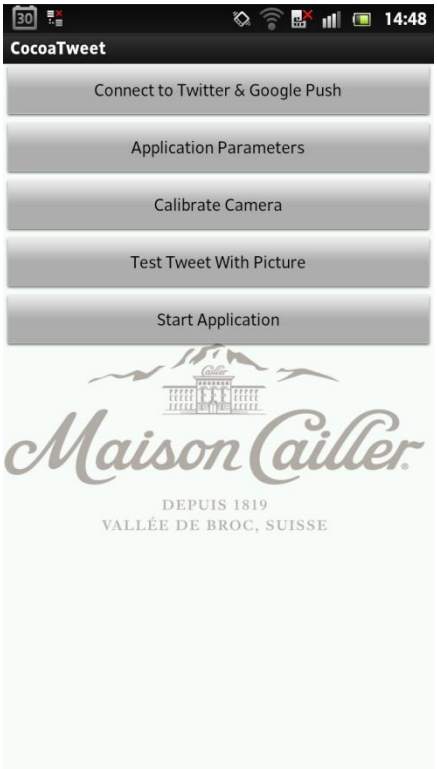

Figure 16: vue de base de l'application

Le premier Bouton permet la configuration du compte Twitter à utiliser, ainsi que le compte Google Push. Ceci ouvre une nouvelle vue, visible à la Figure 17, qui permet de faire plusieurs actions.

La première étant la connexion/déconnexion à un compte Twitter.

Elle est suivie par la possibilité d'envoyer les jetons créés pour Twitter au serveur de gestion des paramètres pour les sauvegarder en dehors du téléphone et pouvoir, au cas où, les remettre sur celui-ci.

La dernière action permet la création d'un jeton pour utiliser la fonction Google Push, décrite dans le chapitre suivant.

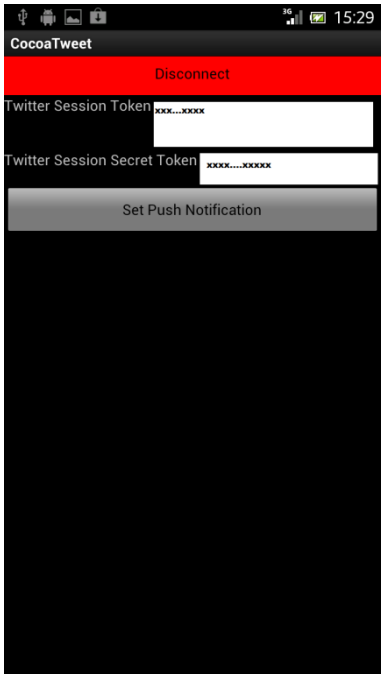

Figure 17: vue de la configuration des comptes Twitter et Google Push

Le deuxième bouton permet de gérer les paramètres de l'application comme mentionné dans la Figure 18.

Le troisième bouton permet simplement de voir l'image de la caméra pour orienter le téléphone vers la plante.

Le quatrième bouton affiche l'image de la caméra et permet d'envoyer un tweet avec la photo. Ceci est pour tester l'envoi de photo.

Le cinquième bouton permet d'activer l'application. Ceci va permettre un cours instant de synchroniser le capteur de Koubachi et va programmer une alarme pour le redémarrage de l'application à l'heure paramétrée.

5.2 Gestion de la configuration de l'application

Étant donné que le téléphone va être déposé loin de la personne l'utilisant, il faut faire en sorte que tout soit paramétrable.

Android met, pour cela, à disposition plusieurs technologies.

5.2.1 SharedPreferences

SharedPreferences est une classe d'Android permettent de stocker des préférences sous forme de clé, valeur. Ces préférences peuvent être de type :

- Boolean
- Int
- Long
- String

Pour enregistrer ces préférences, il faut soit les mettre dans le fichier « parameters.xml » dans le dossier « res/xml/ » (ce fichier va être utile dans le prochain sous-chapitre pour afficher ces préférences de façon ordonnées), soit les créer à la volée grâce à la classe d'édition comme montré dans le Code 1 (ceci est utile surtout pour les paramètres que l'utilisateur ne doit pas toucher, comme les jetons proposé par les différents systèmes d'authentification utilisé).

```
SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(this);
Editor edit=preferences.edit();
edit.putBoolean("exemple", true);
edit.apply();
```

Code 1: création d'une préférence, de type booléen avec la clé « exemple », à la volée

Pour récupérer une valeur des préférences, il suffit de de faire comme dans le Code 2. Il faut indiquer, dans la méthode « get » du bon type, la clé et une valeur par défaut si la clé n'est pas trouvée.

```
SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(this);
preferences.getBoolean("exemple", false);
```

Code 2: récupération d'une préférence

En utilisant les fichiers XML pour stoker les variables, il est possible d'implémenter un système qui permet d'être sûr de toujours accéder au bon paramètre et ce même après avoir modifié le nom de celui-ci. Ce principe est, bien évidemment, utilisé dans ce projet.

Pour réaliser cela il a fallu ajouter trois fichiers XML dont deux indispensables. L'un possédant les noms des clés pour les paramètres et l'autre avec les titres de ces paramètres. Le troisième est pour les

valeurs par défaut. Pour récupérer un paramètre, il faut, comme montré dans le Code 3, procéder en deux étapes :

1. Récupérer la valeur de la clé stockée dans « activate_app_key »
2. Utiliser la clé récupérée pour accéder au paramètre booléen

```
Boolean isActive = preferences.getBoolean(getString(R.string.activate_app_key),
false);
```

Code 3: récupération d'un paramètre via les variables de clés stockées

Ce fragment de code peut sembler long et compliqué. Il est cependant utile quand il faut modifier le nom d'un paramètre et permet de le changer à un seul endroit plus-tôt que dans tous les codes l'utilisant.

5.2.2 PreferenceActivity

Cette classe permet de faire simplement une vue permettant la gestion des paramètres de l'application.

Il suffit de lui donner le lien vers le fichier XML des préférences comme montré dans le Code 4 où « R.xml.parameters » est le chemin du fichier à utiliser comme expliqué plus haut. Ceci fabrique automatiquement la vue de la Figure 18, ce qui permet un gain de temps et de code considérable.

```
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	addPreferencesFromResource(R.xml.parameters);
}
```

Code 4: création de la vue gérant les paramètres de l'application

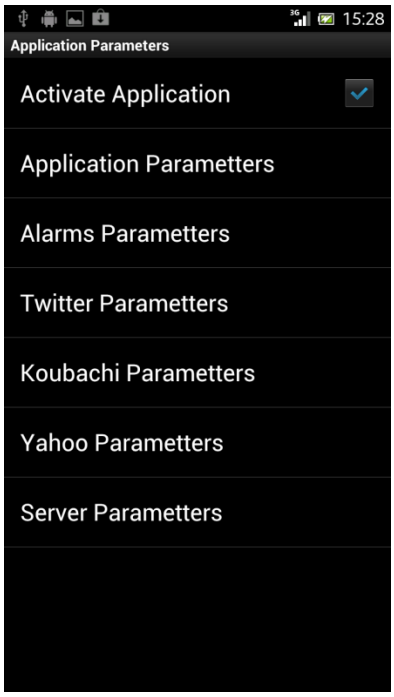

Figure 18: vue des paramètres de l'application

5.3 L'activité principale de l'application

Le but de cette activité est d'envoyer, si besoin est, un tweet avec des informations récupérées des capteurs, et ce dans des intervalles de temps régulier.

Vous trouverez en annexe le workflow de cette activité.

Pour commencer il faut activer le réseau Wifi et 3G pour permettre au capteur d'envoyer ses données à ses serveurs de traitement. Puis il faut regarder si l'on doit tweeter de la publicité. Après cela, il faut récupérer les données des capteurs ainsi que des niveaux de batteries. Il faut ensuite regarder si des tweets correspondent aux conditions des capteurs et lister ceux qui correspondent. Il ne reste plus qu'à prendre l'un de ces tweets au hasard et le mettre sur Twitter. Tout ce processus doit être fait en tenant compte des erreurs qui peuvent survenir tout au long de celui-ci et de les traiter en envoyant des e-mails d'erreurs, si besoin est. Il ne faut pas oublier de déterminer quand l'activité devra redémarrer grâce à une alarme.

Pour pouvoir débugger ou simplement savoir ce qui se passe, une vue, sous forme d'un terminal, a été créée pour l'action principale. Ceci est utile quand le téléphone est sur sa zone d'exploitation pour comprendre ce que l'application essaye de faire sans devoir le connecter à un ordinateur pour afficher les logs. La Figure 19 est un exemple de ce qui peut être affiché. On découvre ici que le message qui devait être envoyé dépasse la taille maximum de 140 caractères.

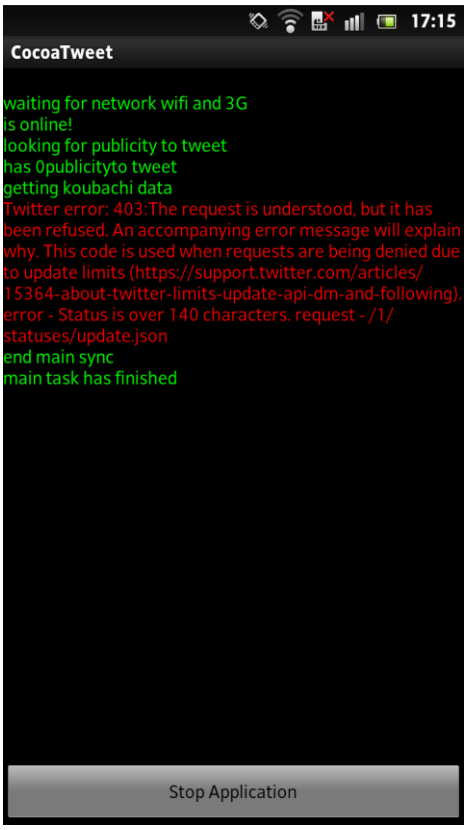

Figure 19: terminal de l'activité principale avec un message d'erreur de Twitter

5.4 La gestion des fichiers XML

Pour gérer les paramètres de l'application, il a été décidé de se baser sur des fichiers XML. Ceci permet une gestion simple et humainement compréhensible de toutes les parties paramétrables de l'application. Nous allons voir ici la structure de ces fichiers qui représentent les paramètres du téléphone, les publicités à tweeter ainsi que les trigger pour les tweets.

5.4.1 Les Paramètres

Le fichier des paramètres est formé de deux éléments, visible dans le Code 5. Il y a l'élément « parameters » qui représente un groupe de paramètres et l'élément « parameter » qui est un paramètre. Ce deuxième élément est composé de trois variables. Il y a la variable « key » indiquant le nom du paramètre à modifier, la variable « type » qui indique le type du paramètre et finalement « value » qui est la valeur du paramètre à modifier.

```
<?xml version="1.0" encoding="utf-8"?>
<parameters>

        <parameter key="activate_app" type="Boolean" Value="true" />
        <parameter key="picture_quality" type="String" Value="2" />
        <parameter key="wake_up_time_alarm" type="Integer" Value="12" />

</parameters>
```

Code 5: fichier XML des paramètres

5.4.2 Les Publicités

Au cours de la réalisation de ce projet, il a été demandé, par le fabricant du capteur utilisé, de leur faire de la publicité en contrepartie de leur implication. La publicité étant émise à fréquence régulière, il a fallu intégrer plusieurs éléments. Tout d'abord il faut compter les tweets pour savoir quand émettre de la publicité. Ceci est fait grâce à un fichier stockant le nombre de tweet envoyé. Cette méthode a pour avantage de supporter la remise d'origine des paramètres sans être influencé. Il a fallu aussi donner un moyen d'informer l'application du moment et du message à envoyer. Cela est fait grâce à un fichier XML comme visible dans le Code 6. Ce fichier contient deux éléments. Le premier nommé « publicities » représente un ensemble de publicités. Le second, « publicity » est une publicité particulière et contient deux variables. La variable « every » est la fréquence d'émission de la publicité et la variable « text » représente le message publicitaire.

```
<?xml version="1.0" encoding="utf-8"?>
<publicities>

        <publicity every="50" text="visitez Koubachi.ch pour avoir plus d'info sur le capteur" />

</publicities>
```

Code 6: fichier XML des publicités

5.4.3 Les Triggers

Ce fichier représente les déclencheurs pour choisir les messages pouvant être tweetés suivant certains paramètres.

Ce fichier est structuré de façon à être le plus paramétrable possible. Comme le montre le Code 7, il contient quatre éléments.

Le premier, nommé « triggers », représente un ensemble de déclencheurs. Il contient des éléments nommés « trigger » formant les règles pour un certain nombre de messages. Ce deuxième élément contient trois variables qui déterminent si un intervalle de temps est défini pour activer ce déclencheur, la première est « use_time_interval », qui est un booléen, qui détermine si l'on doit tenir

compte de l'intervalle. Le deuxième, nommé « start », est le début de l'intervalle et le troisième, nommé « end » est la fin de cet intervalle, cet élément contient deux autres éléments.

Le premier sous-élément est nommé « condition » et représente une condition d'activation du déclencheur. Il contient trois variables qui sont « sensor » représentant le type de capteur utilisé, «comparator » qui est la comparaison à faire et « value » qui est la valeur à comparer. Si l'on prend la première condition du Code 7 et que l'on a comme valeur pour l'humidité de l'air 14, cette condition serra fausse et ce déclencheur ne sera pas utilisé.

Le deuxième élément, nommé « tweet » est le message à envoyer. Il contient à son tour deux variables qui sont « needPicture », permettant de savoir si le message doit être accompagné d'une photo et « value » contenant le message à envoyer.

Ce message peut contenir des tags qui déterminent les informations à remplacer en cas d'envoi du message. Ceci permet d'afficher les valeurs réelles des différents capteurs.

Ces tags sont :

- #PICTURE (le lien de l'image)
- #SOILHUMIDITY (l'humidité du sol)
- #AIRHUMIDITY (l'humidité de l'air)
- #TEMPERATURE (la température)
- #LUMINOSITY (la luminosité)
- #BATTERY (le niveau de batterie du capteur)

Si une information n'est pas disponible, l'acronyme « DNA » qui veut dire « Data Not Available » est affiché.

```
<?xml version="1.0" encoding="utf-8"?>

<triggers>

        <trigger use_time_interval="false"  start="08:12"  end="19:00">

                <condition sensor="airHumidity" comparator="<=" value="12"/>
                <condition sensor="luminosity" comparator=">" value="4000"/>
                <tweet needPicture="false" value="joli tweet #AIRHUMIDITY :-)"/>
                <tweet needPicture="true" value="joli tweet #LUMINOSITY :-)"/>

        </trigger>

</triggers>
```

Code 7: fichier XML des triggers

5.5 La mise à jour des paramètres à distance

Un serveur a été développé en parallèle de l'application. Celui-ci permet la mise à jour des paramètres à distance. Il est décrit dans le chapitre « Serveur ».

5.6 Les défis technologiques et problèmes rencontrés

Comme dans tous projets, il arrive que l'on se trouve devant un problème ou un défi à relever.

5.6.1 Appareil photo du téléphone

L'appareil photo du téléphone est monté de manière à faire des photos plus larges que hautes, ce qui pose un problème mineur pour le projet. Ce problème est que l'on aimerait avoir une vue plus haute que large pour voir un maximum le tronc de la plante de cacao. Il faut donc tourner l'image prise pour pouvoir faire cela.

L'autre point à respecter est que les photos prises doivent être de la meilleure qualité possible et, de ce fait, prennent beaucoup d'espace mémoire. Une photo 12MP peut avoir une taille d'environ 4,24 Mo quand l'image est compressée dans le format « jpeg ».

L'API propose une méthode pour demander à la caméra de tourner l'image au moment de la prise de celle-ci. Le défaut de cette méthode est que l'API précise que les constructeurs de téléphones ne sont pas obligé de tourner l'image (11) mais peuvent aussi juste indiquer dans l'entête de l'image qu'il faudrait tourner celle-ci. Ce qui est le cas du téléphone utilisé.

Pour tourner l'image, il faut la charger en mémoire et lui appliquer une matrice de rotation. Ceci crée une deuxième image. Le problème est que la mémoire utilisable par l'activité, qui est d'environ 7 Mo, est plus petite que la place mémoire prise par l'image et il faut pouvoir la charger deux fois.

Une solution probable est de passer par du code en C, qui n'est pas limité de la même manière que la partie en Java. Le temps imparti, pour ce travail de Bachelor, est trop court pour pouvoir implémenter et tester cette méthode.

6 Android C2DM

Android possède une technologie pour synchroniser les téléphones avec leurs serveurs. Cette technologie se nomme « Cloud To Device Messaging » qui a comme acronyme « C2DM ». Android met cette technologie à disposition des développeurs pour leurs permettre de maintenir à jour les données de leurs applications sur les téléphones. Il est important de savoir que la méthode utilisée ici est considérée comme étant dépréciée par Google depuis le 26 juin 2012. Ceci signifie qu'il ne faut pas l'utiliser en développant de nouvelles applications. La raison à cela est que Google s'est rendu compte du potentiel de cette technologie et a décidé de l'améliorer et de la rendre plus facile à utiliser pour les développeurs. Il est, à l'heure actuelle, préférable d'utiliser le nouveau système nommé « GCM » qui veut dire « Google Cloud Messaging ». Vu le nombre important d'applications utilisant C2DM, il a été décidé de ne pas arrêter ce système avant une longue période. Toutefois aucune date butoir n'a été annoncée par Google.

6.1 Fonctionnement

Android maintient un lien permanent entre le téléphone et les serveurs de Google tant que le téléphone est connecté à un réseau. Ce lien permet d'informer le téléphone des changements à la place qu'il aille vérifier si des changements ont eu lieu. Ceci permet de drastiquement diminuer l'utilisation de la batterie. Ce système n'accepte qu'une petite quantité d'information (1Ko) et permet juste d'informer un téléphone qu'il peut aller chercher une mise à jour d'informations sur le serveur.

Comme présenté sur la Figure 20, le système est composé de trois actions principales.

Pour commencer, le serveur de l'application envoie un message aux serveurs de Google. Ce message est ensuite envoyé au téléphone qui va rechercher la mise à jour sur son serveur d'application.

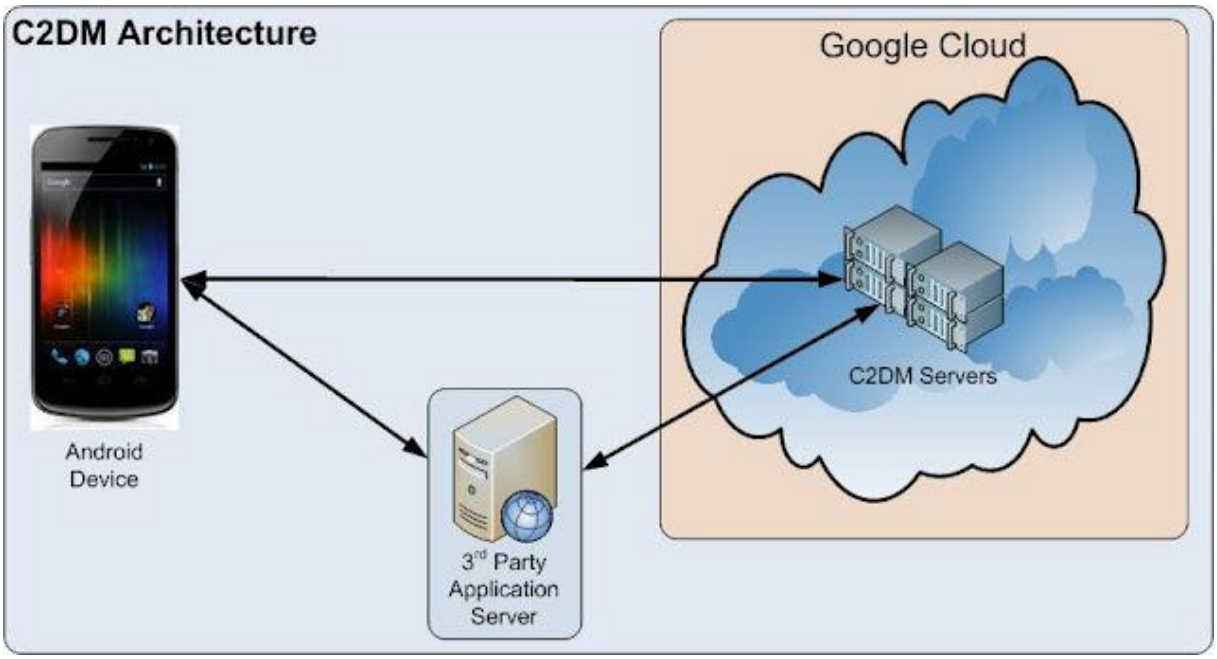

Figure 20: principe de base du fonctionnement du C2DM tiré de (12)

La Figure 21 représente l'ensemble du système avec toutes les interactions entre les différents modules.

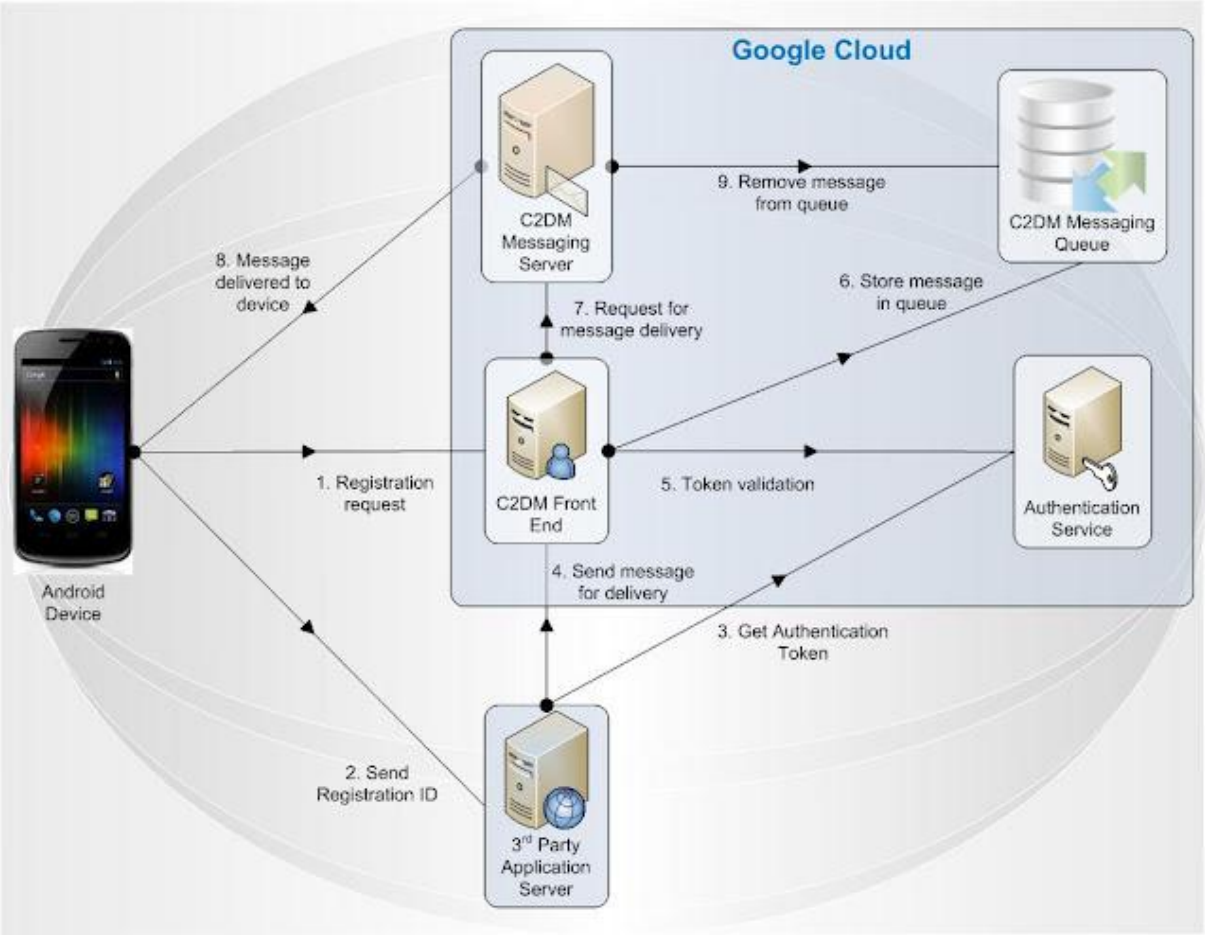

Figure 21: workflow C2DM, tiré de (12)

Voici la signification des différentes flèches de la Figure 21.

1. Le téléphone se connecte au serveur C2DM et, avec l'email du compte du serveur de l'application, demande un ID unique.
2. Cet ID est envoyé au serveur de l'application, qui la stocke précieusement.
3. Avant de pouvoir envoyer des notifications, le serveur de l'application doit s'authentifier auprès du serveur d'authentification de Google et reçoit un jeton.
4. Lors de l'envoi d'une notification, le serveur de l'application envoie un message au serveur C2DM contenant le message à transmettre (maximum 1Ko), son jeton d'authentification, l'ID du téléphone ainsi que d'autres options déterminant le comportement de l'envoi du message au téléphone en cas d'indisponibilité du téléphone et finalement la « collapse key ».
5. Le serveur C2DM vérifie l'identité du serveur d'application auprès du serveur d'authentification.
6. Le message est stocké dans une base de donnée avant l'envoi.

7. Le serveur C2DM informe le serveur d'envoi du message et des conditions d'envoi.
8. Si les conditions sont respectées, le message est envoyé.
9. Le message est supprimé de la base de données.

Les conditions d'envoi du message sont les suivantes :

- Autoriser un délai si trop de messages sont en cours d'envoi vers le téléphone.
- Définir le délai maximum avant suppression du message si le téléphone est hors-ligne.

La « collapse key » est un string déterminant le message. Il peut être considéré comme l'ID du message. Si deux messages ont les mêmes ID du destinataire, jeton du serveur de l'application et collapse key, le message déjà dans la base de donnée serra remplacé par le nouveau message. Cela permet de diminuer la quantité de messages envoyé au téléphone lors de sa reconnexion.

6.2 Utilisation

Pour fonctionner, le service doit être implémenté sur le serveur et dans l'application Android

6.2.1 Côté serveur

Le serveur utilise un module nommé node-c2dm (13) qui permet de gérer toute l'interaction entre le serveur de l'application et les serveurs de Google.

Le Code 8 montre l'envoi d'un message. Il faut commencer par configurer le module avec le compte mail du serveur. Puis se loger au serveur d'authentification qui nous renvoie un jeton que l'on peut stocker. Il faut finalement créer et envoyer le message.

```
var config = {
        user: user@gmail.com',
        password: '1234',
        source: 'com.nestle.maisoncailler.android.cocoatweet',
};
var c2dm = new C2DM(config);
c2dm.login(function(err, token){});
var message = {
        registration_id: ''+regId,
        collapse_key: ''+collapse_key_value,
        'data.payload': ''+payload
};
c2dm.send(message, function(err, messageId){});
```

Code 8 : envoi d'un message avec le module node-c2dm

6.2.2 Côté application Android

Comme montré dans le Code 9, le manifeste doit contenir les droits pour les messages ainsi que pour les recevoir.

Il faut aussi mettre en place des filtres pour intercepter les Intents envoyés des serveurs Google, que ce soit pour un enregistrement ou la réception d'un message, et les envoyer à la classe RegistrationReciever.

Le RegistrationReciever va, comme montré dans le Code 10, interpréter l'Intent reçu.

Si c'est un message, on va récupérer son payload et effectuer une action suivant celui-ci. Si le payload contient « trigger », « parameters » ou « publicity », l'application va aller chercher le fichier XML correspondant sur le serveur pour modifier les paramètres corespondant. Si le payload contient « all » tous ces fichiers XML seront chargés.

Si c'est une demande d'enregistrement, on récupère l'ID et on l'envoie au serveur.

```
<permission
android:name="com.nestle.maisoncailler.android.cocoatweet.permission.C2D_MESSAGE"
android:protectionLevel="signature"
/>

<uses-permission
android:name="com.google.android.c2dm.permission.RECEIVE"
/>

<uses-permission
android:name="com.nestle.maisoncailler.android.cocoatweet.permission.C2D_MESSAGE"
/>

…

<receiver
android:name=".RegistrationReceiver"
android:permission="com.google.android.c2dm.permission.SEND"
>

<!-- Receive the actual message -->

      <intent-filter>

            <action
            android:name="com.google.android.c2dm.intent.RECEIVE"
            />
            <category
            android:name="com.nestle.maisoncailler.android.cocoatweet"
            />
      </intent-filter>
<!-- Receive the registration id -->
      <intent-filter>
            <action
            android:name="com.google.android.c2dm.intent.REGISTRATION"
            />
            <category
            android:name="com.nestle.maisoncailler.android.cocoatweet"
            />
      </intent-filter>
</receiver>
```

Code 9 : le fichier manifeste de l'application Android

```
public void onReceive(final Context context, Intent intent) {
	preferences = PreferenceManager.getDefaultSharedPreferences(context);
	String action = intent.getAction();
	Log.d("CMN", "Registration Receiver called");
	if ("com.google.android.c2dm.intent.REGISTRATION".equals(action)) {
		Log.d("CMN", "Received registration ID");
		String registrationId=intent.getStringExtra("registration_id");
		…
	} else {
		if ("com.google.android.c2dm.intent.RECEIVE".equals(action)) {
			String payload = intent.getStringExtra("payload");
			Log.d("CMN", "Received message: " + payload);
			…
		}
	}
`
```

Code 10 : une partie de la classe BroadcastReceiver

Pour recevoir un Id du serveur de Google, il faut commencer par en faire la demande, ce qui est fait dans le Code 11. Il suffit de créer un Intent dans lequel il faut ajouter un paramètre « sender » avec le nom du compte email utilisé par le serveur.

```
Intent registrationIntent =new Intent("com.google.android.c2dm.intent.REGISTER");
registrationIntent.putExtra("app",
PendingIntent.getBroadcast(getApplicationContext(), 0, new Intent(), 0));
String email=preferences.getString(getString(R.string.c2dm_sender_email_key),
"");
registrationIntent.putExtra("sender", email);
Log.d("CMN", "send connection to push");
startService(registrationIntent);
```

Code 11 : demande d'ID au près des serveurs de Google

7 Serveur Web

Pour pouvoir paramétrer l'application sans devoir être physiquement en face du téléphone, il a été décidé d'utiliser un serveur et une interface web permettant de tout paramétrer.

Ce serveur est basé sur une nouvelle technologie nommée «node js » (14). Il est basé sur du JavaScript et a comme avantage d'être non bloquant, simple d'utilisation et léger. Ce qui en fait un candidat parfait pour ce projet.

Ce serveur est une ébauche rapide permettant la gestion de l'application et ne peut pas être considéré comme une version de mise en production sécurisée. C'est une preuve de concept.

7.1 Installation

Pour pouvoir utiliser « node js », il faut commencer par télécharger le programme et lancer l'installation.

Une fois cette étape réalisée il suffit, dans un terminal, de noter « node monserveur.js » pour démarrer le serveur.

7.2 Architecture

Le site web est formé de plusieurs pages permettant de paramétrer différentes parties de l'application. Le serveur est aussi séparé en plusieurs parties pour simplifier la gestion de celui-ci.

7.2.1 Le site web

Il contient une page principale, visible dans la Figure 22, qui est un menu des différents types de paramètres ainsi qu'un lien qui envoie au téléphone l'ordre de recharger tous les paramètres.

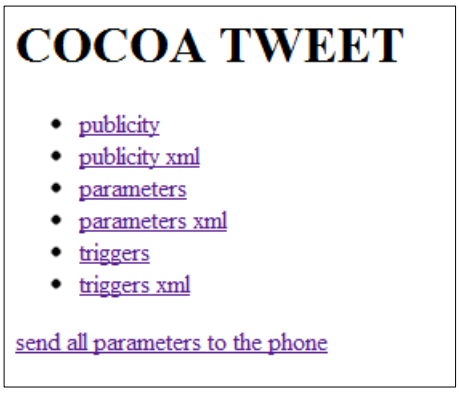

Figure 22: page d'index du site web

Toutes les pages sont visibles sous la forme d'un formulaire ou d'un fichier XML, qui est le fichier utilisé par l'application pour charger les paramètres.

7.2.2 Le serveur

Il est séparé en quatre types de fichiers différents.

7.2.2.1 Le JavaScript

Le premier type est les fichiers JavaScript, c'est la base du serveur. Ces fichiers contiennent le fonctionnement de celui –ci. Le fichier de démarrage est « startserver.js », il crée le serveur et défini le port à utiliser. Il envoie ensuite toutes les requêtes qu'il reçoit à «analyser.js » qui se charge de les analyser et de les diriger vers le bon module de traitement. Les modules de traitement fonctionnent tous de la même manière, ils servent à retourner l'information demandée par l'utilisateur ou stocker les données des formulaires. Pour retourner les informations, s'il s'agit d'un XML, celui-ci sera juste envoyé. S'il s'agit d'un formulaire, l'opération est plus complexe, le module va charger le squelette du formulaire et remplir celui-ci avec les informations stockées dans le fichier XML correspondant. Lors de la sauvegarde d'un formulaire, le module va créer le fichier XML avec tous les paramètres du formulaire.

7.2.2.2 L'HTML

Le deuxième type est les fichiers html. Il y a l'index et les formulaires. L'index est le fichier de la page principale et contient simplement un menu. Les formulaires sont plus complexes. Étant donné que la majorité des formulaires doivent être dynamiques, ils contiennent une structure permettant l'ajout et la suppression de champs grâce à des scripts JavaScript.

7.2.2.3 Le XML

Le troisième type est les fichiers XML. Ils stockent les données des formulaires et sont récupéré par le téléphone pour modifier ses paramètres. Ils sont recréés à chaque validation de formulaire.

7.2.2.4 Le TXT

Le quatrième type est le fichier texte. Il permet de stocker l'identifiant du téléphone pour lui envoyer les messages de configurations.

7.3 Utilisation

Les paramètres sont divisés en trois formulaires.

7.3.1 La publicité

Le formulaire de publicité, visible sur la Figure 23, permet d'informer l'application sur les messages de publicité à tweeter et tous les combiens de tweets ces messages doivent être envoyé.

Figure 23: formulaire de publicité

7.3.2 Les paramètres de fonctionnement

La Figure 24 montre le formulaire gérant tous les paramètres permettant le fonctionnement basique de l'application. Sauf si l'on veut changer un des comptes ou un capteur, ce formulaire est édité qu'une seul fois lors de l'installation de l'application.

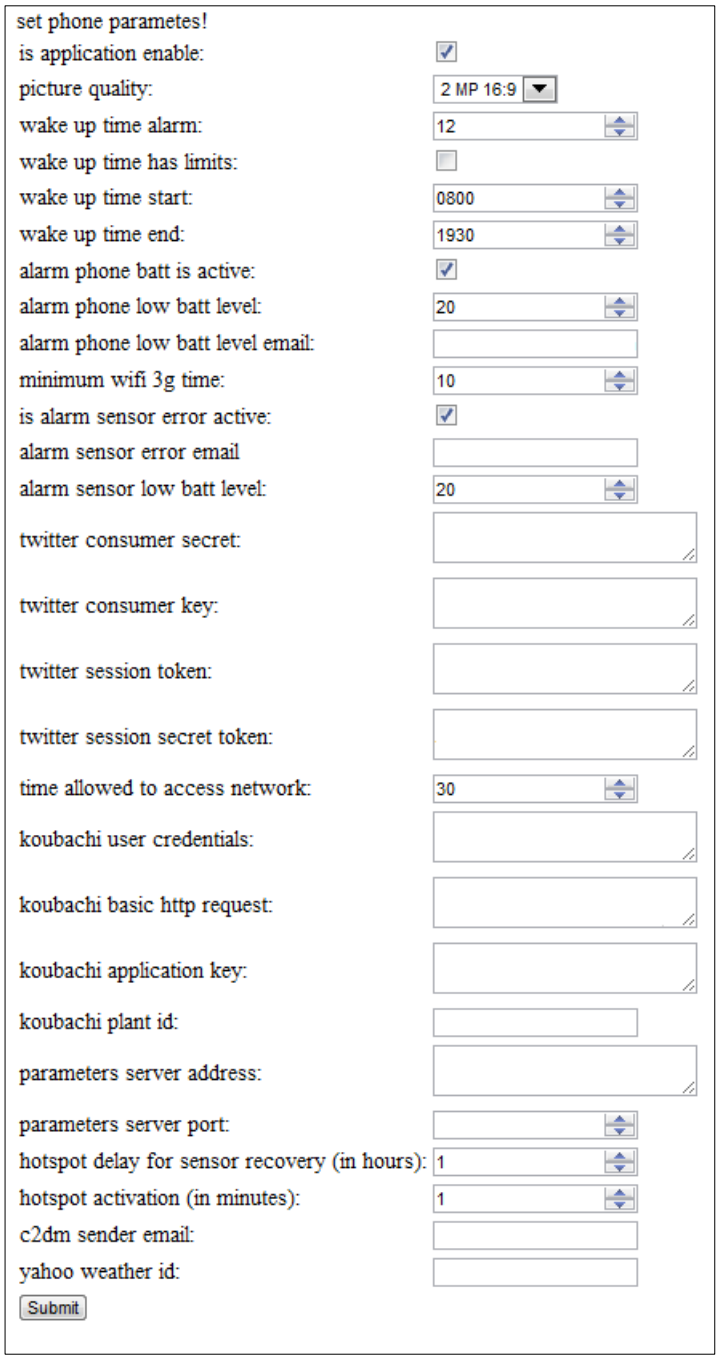

Figure 24: formulaire des paramètres de fonctionnement

Tous ces paramètres sont expliqués en détail dans le mode d'emploi en annexe.

7.3.3 Les déclencheurs

Le formulaire de la Figure 25 représente les messages ainsi que leurs conditions d'émission.

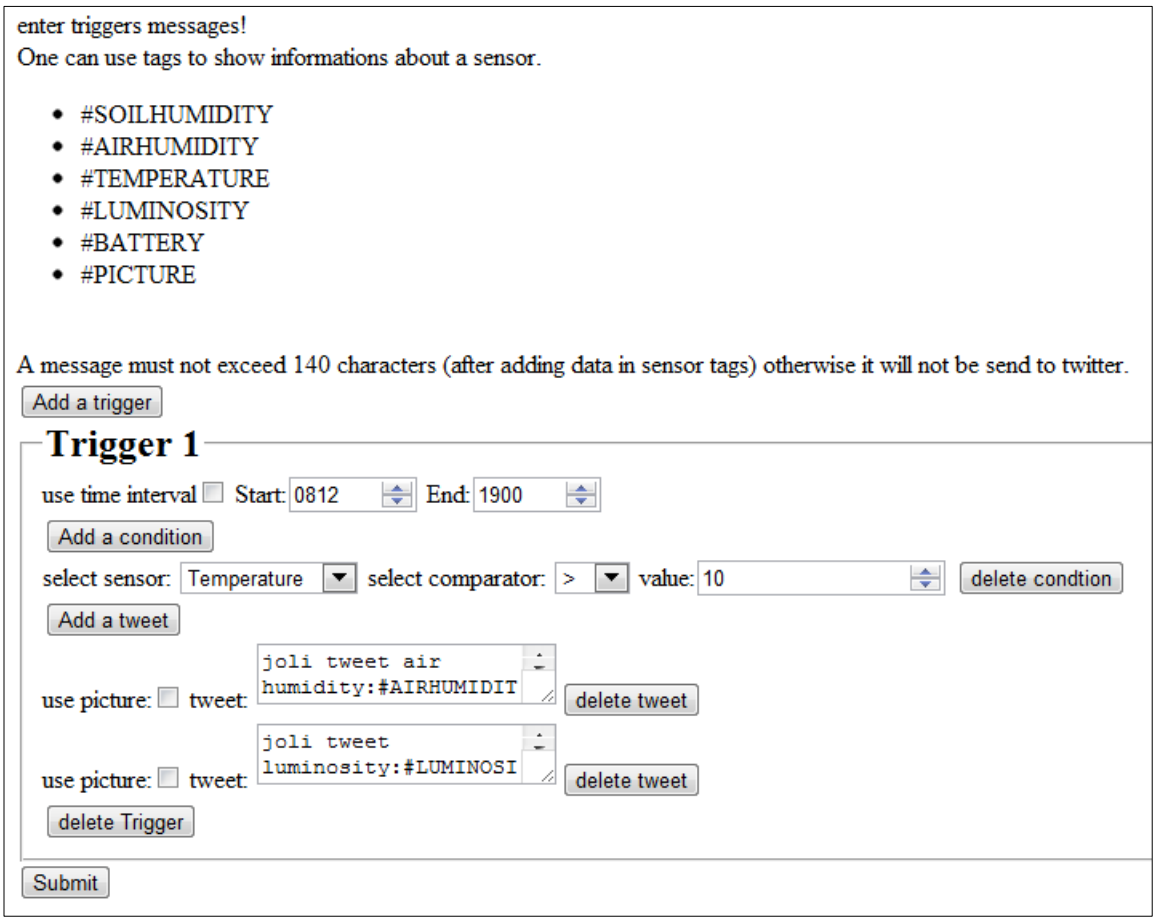

Figure 25: formulaire des déclencheurs

Chaque déclencheur peut contenir plusieurs messages ainsi que plusieurs conditions. Il est aussi possible de définir un intervalle de temps durant lequel le message doit être pris en compte. Les conditions sont des comparaisons par rapport à un capteur. Dans l'exemple de la Figure 25 on peut voir que les deux messages peuvent être émis à n'importe quelle heure et qu'il y a une condition basée sur le capteur de température. Cette condition fait que ces messages peuvent être émis que si la température dépasse les 10 degrés Celsius. Au niveau des messages on voit qu'aucun d'eux n'inclut de photo et que le premier va afficher l'humidité de l'air et que le deuxième va afficher la luminosité.

8 Protection du téléphone

Ce projet est fait pour fonctionner dans des conditions qui peuvent être jugées extrêmes, au niveau de l'environnement, pour un téléphone. En Equateur, le taux d'humidité est très élevé. Il atteint 80% d'humidité en moyenne.

Pour permettre au téléphone de supporter ces conditions ainsi que lui permettre un fonctionnement de longue durée, il a été décidé d'ajouter une boite de protection étanche autour de celui-ci ainsi qu'un système de recharge par panneaux solaires. Ces améliorations permettent de mieux supporter le climat local et d'augmenter grandement la durée de vie du téléphone.

Ce système est artisanal et est composé d'un seau stockant l'électronique et la batterie ainsi qu'un panneau solaire et un boitier transparent avec le téléphone, comme visible sur la Figure 26.

Figure 26 : dispositif de protection et recharge du téléphone

9 Suite possible du projet

Une suite à ce projet a été proposée en Master. Le but étant d'augmenter l'interaction de la plante en utilisant de l'analyse d'image pour détecter les fruits et autres changements physique de la plante.

Il est aussi possible d'ajouter d'autres capteurs pour augmenter cette interaction, ainsi que stocker ces informations pour s'y référer. Cela peut ajouter un facteur temps aux messages émit. Il serait alors possible d'émettre des messages tel que « cela fait trois semaines qu'il pleut, je me réjouis de revoir le soleil ! ».

10 Conclusion

Le projet fonctionne avec le capteur de Koubachi ainsi que le site web de Yahoo.

Il est possible de récupérer toutes les données des capteurs et d'envoyer un tweet avec ces données ainsi qu'une photo. Il est aussi possible de recevoir les emails d'erreurs de la part de l'application.

Le serveur de l'application est fonctionnel et il permet de gérer entièrement l'application du téléphone, ce qui est utile lors ce que l'on se trouve à des milliers de kilomètres de celui-ci.

La protection pour le téléphone ainsi que son alimentation a été testée avec succès et est prête pour son déploiement.

Ce projet m'as permis de découvrir Android ainsi que « node js » et d'ajouter des cordes à mon arc.

J'ai eu beaucoup de plaisir de voir avancer le développement de ce projet.

11 Bibliographie

1. Live @ Maison Cailler. *Maison Cailler.* [En ligne] [Citation : 7 juin 2012.] http://www.maisoncailler.com/fr/live-en-ligne-directe.

2. Caractéristiques Xperia S | Écran tactile 4,3 - Sony Smartphones (Global French). *Sony Mobile.* [En ligne] Sony, 2012. [Citation : 4 juin 2012.] http://www.sonymobile.com/global-fr/products/phones/xperia-s/specifications/.

3. Koubachi. *Koubachi.* [En ligne] Koubachi, 2009. [Citation : 4 juin 2012.] http://www.koubachi.com/.

4. Koubachi - Wi-Fi Plant Sensor. *Koubachi.* [En ligne] Koubachi, 2009. [Citation : 4 juin 2012.] http://www.koubachi.com/features/sensor?locale=en.

5. Yahoo Weather. *Yahoo.* [En ligne] Yahoo, 2012. [Citation : 4 juin 2012.] http://weather.yahoo.com/.

6. Implementing Sign in with Twitter. *Twitter Developpers.* [En ligne] [Citation : 4 juin 2012.] https://dev.twitter.com/docs/auth/implementing-sign-twitter.

7. Installing the SDK. *Android Developpers.* [En ligne] Android. [Citation : 13 juin 2012.] http://developer.android.com/sdk/installing.html.

8. **Guignard, Damien, Chable, Julien et Robles, Emmanuel.** *Programmation Android de la conception au déploiement avec le SDK Google Android 2.* Paris : Eyrolles, 2011. 978-2-212-12587-0.

9. Android Developers. *Android Developers.* [En ligne] [Citation : 13 juin 2012.] http://developer.android.com/index.html.

10. Activities | Android Devlelopers. *Android Developpers.* [En ligne] Google, 8 juin 2012. [Citation : 11 juin 2012.] http://developer.android.com/guide/topics/fundamentals/activities.html.

11. Camera.Parameters. *Android Developers.* [En ligne] [Citation : 26 juillet 2012.] http://developer.android.com/reference/android/hardware/Camera.Parameters.html#setRotation(int).

12. C2DM Architecture. *the Mobility expert.* [En ligne] [Citation : 18 juillet 2012.] http://themobilityexpert.blogspot.ch/2012/01/android-c2dm-part-2-lifecycle.html.

13. SpeCT/node-c2dm. *github.* [En ligne] [Citation : 18 juillet 2012.] https://github.com/SpeCT/node-c2dm.

14. *nodejs.* [En ligne] [Citation : 18 juillet 2012.]

12 Table des illustrations

13 Table des Codes

14 Annexes

Annexe 1 : Cahier des charges

Annexe 2 : Workflow de l'activité principale

Annexe 3 : UML du projet Java

Annexe 4 : Mode d'emploi