

SOCIAL MEDIA INTERACTION

CONTACTER UN CITOYEN SUR FACEBOOK OU TWITTER

VANDY BERTEN



Résumé – Ce document décrit de quelle façon deux médias sociaux (Facebook et Twitter) pourraient servir d'alternative à une adresse e-mail comme canal de communication entre les autorités et les citoyens. Nous décrivons pour ces deux outils comment il est possible d'envoyer de façon automatisée un message privé à un utilisateur qui s'est au préalable authentifié et a lié de façon sécurisée son compte Facebook ou Twitter à une « e-box ».

Dans le cas de Facebook, nous utilisons une « application Facebook », permettant d'envoyer des notifications à l'utilisateur. Dans le cas de Twitter, nous utilisons une « application Twitter », permettant d'envoyer, au nom d'un compte officiel, un « direct message » à un citoyen.

Quelques détails techniques seront donnés, mais pas suffisamment pour pouvoir réaliser l'ensemble d'une telle application, dont nous avons développé un prototype fonctionnel. Il sera nécessaire de consulter la documentation technique spécifique au réseau social visé, afin de développer une application complète.

Après une introduction à portée générale, nous rentrerons, tant pour Facebook que pour Twitter, dans un niveau de détails s'adressant à des développeurs ou des lecteurs à profil technique.

Samenvatting – Dit document beschrijft op welke manier twee sociale media (Facebook en Twitter) een alternatief zouden kunnen vormen voor een e-mailadres als communicatiekanaal tussen de overheid en de burgers. We beschrijven voor deze twee tools hoe men automatisch een privébericht kan sturen naar een gebruiker die zich vooraf geauthenticeerd heeft en die zijn Facebook- en Twitteraccount veilig gekoppeld heeft aan een "e-box".

In het geval van Facebook gebruiken we een "Facebook-applicatie" waarmee we meldingen kunnen versturen naar de gebruiker. In het geval van Twitter gebruiken we een "Twitter-applicatie" waarmee we, in de naam van een officieel account, een "rechtstreeks bericht" sturen naar een burger.

Er zullen een aantal technische details gegeven worden, maar niet genoeg om zo'n applicatie, waarvan we een functioneel prototype ontwikkeld hebben, volledig te kunnen realiseren. U zal de technische documentatie voor elk sociaal netwerk moeten raadplegen om een volledige applicatie te kunnen ontwikkelen.

Na een algemene inleiding zullen we, zowel voor Facebook als voor Twitter, meer details geven waarbij we ons richten tot de ontwikkelaars of lezers met een technisch profiel.

Note : toutes les URL de ce document ont été contrôlées le 5 novembre 2014.

Table des matières

1.	Introduction	3
2.	Mécanisme d'autorisation « OAuth »	6
3.	Facebook	7
3.1.	Introduction.....	7
3.2.	Création/configuration de l'application	8
3.2.1.	Application intégrée ou externe	8
3.2.2.	PHP ou Javascript.....	9
3.2.3.	Combinaisons	9
3.2.4.	Configuration.....	10
3.3.	Autorisation.....	10
3.4.	Requêtes	12
3.5.	Notifications	12
3.5.1.	href	13
3.5.2.	template	13
3.6.	App Requests.....	13
4.	Twitter	15
4.1.	Introduction.....	15
4.2.	Authentification	15
4.3.	Envoi de Direct Message	16
5.	Autres réseaux	18
5.1.	LinkedIn	18
5.2.	Google+	18
5.3.	Snapchat	18
5.4.	Instagram, Pinterest	18
5.5.	WhatsApp.....	19
5.6.	Skype.....	19
5.7.	Netlog	19
6.	Conclusions et recommandations	20

1. Introduction

Il est devenu fréquent que les administrations demandent aux citoyens de leur fournir une adresse e-mail pour les contacter. C'est par exemple le cas sur l'Irisbox¹ pour les Bruxellois, sur Tax-on-web² pour tous les belges et résidents (ou dans leur « dossier fiscal »³) et même de certaines communes. La demande est souvent optionnelle et on ne sait pas toujours à quelle fin cette adresse est demandée. La récente législation sur le recommandé électronique en France⁴ ou en Belgique⁵ va vraisemblablement augmenter le nombre d'adresses e-mail stockées par les autorités.

L'e-mail existe depuis le milieu des années 70, et bien qu'on ait à plusieurs reprises prédit sa fin, force est de constater qu'il s'agit toujours d'un média largement utilisé et que celui-ci n'est probablement pas prêt de disparaître. Cependant, lors d'une étude que nous avons menée récemment à propos de la gestion des adresses e-mail dans les bases de données⁶, nous avons fait deux constats :

1. Il est difficile de gérer correctement une grande base de données d'adresses e-mail ; de nombreux facteurs d'incertitude, des standards multiples et non respectés ainsi que des caractéristiques intrinsèquement dynamiques rendent la tâche très ardue. Nous avons toutefois proposé un ensemble de techniques et de méthodes afin d'en faciliter la gestion dans le temps⁶ ;
2. De plus en plus de spécialistes s'accordent à dire qu'un nombre croissant de gens, en particulier parmi les jeunes, délaissent l'usage de leur adresse e-mail au profit des réseaux sociaux⁷. De nombreux adolescents se créent par exemple une adresse e-mail dans le seul but de pouvoir se créer un profil sur Facebook.

¹ <https://irisbox.irisnet.be/>

² <http://www.tax-on-web.be/>

³ <http://www.mymifin.be/>

⁴ <http://www.journaldunet.com/management/pratique/vie-de-l-entreprise/1694/lettre-recommandee-electronique.html>

⁵ <http://datanews.levif.be/ict/actualite/un-cadre-legal-pour-la-lettre-recommandee-electronique/article-400431534747.htm>, ou, au moniteur : http://www.ejustice.just.fgov.be/cgi/article_body.pl?numac=2013024113&caller=list&article_lang=F&row_id=1&numero=1&pub_date=2013-03-29&pdda=2013&dt=LOI&language=fr&fr=f&choix1=ET&choix2=ET&pdfa=2013&pdj=01&fromtab=+mofxt+UNION+montxt&nl=n&pddm=03&pdfj=31&sql=dt+%3D+%27LOI%27+and+pd+between+date%272013-03-01%27+and+date%272013-03-31%27+&pdfm=03&rech=32&tri=dd+AS+RANK+&trier=promulgation

⁶ « Email Address Reliability », Vandy Berten & Isabelle Boydens, Smals Research, (<http://www.smalsresearch.be/publications/document?docid=88>, <http://www.amazon.com/E-mail-Address-Reliability-Isabelle-Boydens/dp/1291883215/>)

⁷ <http://geeko.lesoir.be/2014/01/29/necrivez-plus-de-mails-cest-ringard/> ou <http://www.sharethis.com/blog/2014/01/16/pinterest-surpasses-email-sharing-online-beats-facebook-growth-2013>

Bien que la question d'une bonne gestion de contacts par réseaux sociaux se pose également, il est judicieux de se demander si, en complément d'une adresse e-mail, il est également possible pour un citoyen d'être contacté via le réseau social de son choix. Le ministère des finances en a fait une annonce récemment⁸, ainsi que, dans le privé, KLM⁹. Chez Smals, on y réfléchit depuis quelques mois.

Notons que ces mécanismes peuvent aller bien au-delà de la notification administrative. On peut également imaginer le même système pour être averti en cas de crue d'un cours d'eau ou en cas d'alerte près d'un site Seveso, en supplément des canaux déjà utilisés (presses, sms...)

Pour qu'un réseau social puisse être candidat alternatif à l'adresse e-mail dans le contexte de l'e-gouvernement, il doit remplir quelques conditions.

1. Il faut qu'il offre un mécanisme de messagerie privée, qui permette d'envoyer un message à un utilisateur qu'il sera seul à pouvoir lire. Il n'est bien évidemment pas question de poster sur le mur public d'un citoyen qu'il a oublié de payer ses impôts ou d'envoyer un tweet pour signaler qu'il a reçu un recommandé électronique.
2. Il faut que la plateforme offre une « API » (Application Programming Interface), permettant d'interagir automatiquement avec le réseau, sans qu'un fonctionnaire ait à se connecter explicitement sur le compte et doive envoyer les messages « à la main ».
3. Le réseau doit pouvoir permettre qu'un de ses utilisateurs, sur la base volontaire et explicite, autorise un serveur à le contacter via son profil. Il va de soi qu'une institution ne pourra jamais se servir d'un profil Facebook pour contacter son propriétaire sans son consentement explicite. En effet, rien ne permet à une administration de s'assurer que le « Albert Durant » qu'il a trouvé sur Facebook est bien celui qui n'a pas payé ses impôts. Par ailleurs, les réseaux sociaux sont – heureusement ! – suffisamment cloisonnés pour qu'on ne puisse pas interagir avec quelqu'un qui n'a pas marqué son accord, même si la demande émane d'un ministère.
4. Le mécanisme d'autorisation doit cibler précisément ce que l'accès à un profil permet de faire. En général, pouvoir envoyer un message ou une notification est suffisant. Aucun citoyen ne sera prêt à accepter un tel mécanisme s'il sait que les autorités ont soudainement accès à toutes ses photos ou les commentaires qu'il poste sur les statuts de ses amis¹⁰.

Nous supposons pour la suite qu'un citoyen possède un accès à une « e-box », sorte de boîte aux lettres électronique, typiquement sécurisée au moyen d'une carte d'identité électronique, dans laquelle les autorités peuvent déposer des documents. Cette e-box est amenée de temps à autre à lui notifier que, par exemple, un nouveau document officiel y est disponible.

Après avoir très brièvement présenté ce que permet le protocole « OAuth », indispensable à l'approche, nous présenterons une revue rapide et non exhaustive de quelques-uns des réseaux sociaux majeurs que nous avons étudiés ou testés, en développant un petit « proof-of-concept » (PoC). Pour deux

⁸ <http://datanews.levif.be/ict/actualite/le-fisc-envoie-des-avertissements-via-facebook/article-4000518869273.htm>

⁹ <http://geeko.lesoir.be/2014/02/12/klm-propose-de-payer-vos-tickets-davion-sur-facebook-et-twitter/>

¹⁰ Bien que nettement plus de données soient accessibles que ce que les gens pensent en général, comme nous le mentionnons dans <http://www.smalsresearch.be/la-vie-privee-selon-facebook/>

réseaux sociaux – Facebook et Twitter – nous donnerons des détails techniques avancés sur l'implémentation de la solution que nous proposons.

2. Mécanisme d'autorisation « oAuth »

Le protocole « oAuth¹¹ » est un mécanisme d'autorisation très populaire parmi les réseaux sociaux, qui permet principalement deux choses :



1. Optimiser la connexion à un portail quelconque (forum, e-commerce...), qui consiste en général en la création d'un « compte », avec un nom d'utilisateur et un mot de passe. L'inconvénient de cette façon classique de faire est qu'elle oblige les gens à se créer de multiples comptes sur un tas de services et à en retenir les identifiants. Grâce à oAuth, pour autant qu'il soit implémenté sur le portail en question, il est également possible de s'y connecter avec, par exemple, son compte Facebook. Ceci se fait en toute sécurité, sans que le mot de passe de Facebook ne soit transmis au portail.
2. Autoriser un portail ou une application (c'est-à-dire une page web) indépendante du réseau social, non seulement à accéder de façon contrôlée à certaines informations disponibles sur ce réseau social (par exemple ses photos, sa liste de contacts, sa date de naissance...), mais également à agir au nom de l'utilisateur, comme par exemple poster un message sur un mur ou recevoir une notification. Chaque utilisateur de Facebook voit régulièrement un de ses amis poster un message du type « J'ai obtenu un score de XXX au jeu YYY ; seriez-vous capable de me battre ? », message qu'il n'a pas écrit lui-même, mais qui l'a été par le jeu en question.

Notons que la plupart des systèmes présentés ci-dessous et qui implémentent oAuth permettent également, quand cela s'applique et que l'autorisation a été accordée, d'accéder à l'adresse e-mail renseignée sur le réseau social. On pourrait dès lors proposer de lier son e-box à un réseau social et de s'en servir uniquement pour récupérer l'adresse e-mail renseignée. Nous ne détaillerons pas davantage cette possibilité ci-dessous.

¹¹ <http://oauth.net/>

3. Facebook

3.1. Introduction



En commençant cette étude, nous avons trouvé deux façons de permettre sur Facebook ce que nous présentons plus haut, dont seulement une était réellement satisfaisante.

La première consistait à envoyer un e-mail classique à l'adresse associée à un compte Facebook (souvent du type prenom.nom@facebook.com). Cette solution avait pour avantage sa facilité, puisqu'il n'est pas nécessaire de mettre en place un mécanisme de contact différent de celui habituellement utilisé pour les e-mails : un citoyen peut choisir, à la place de son adresse e-mail « classique », de fournir celle de Facebook. Elle a par contre un inconvénient majeur : le message en question n'apparaît pas directement dans la boîte de réception, mais dans une catégorie « Autre », peu visible, que beaucoup d'utilisateurs de Facebook n'ont même jamais remarquée.

Cependant, ce lundi 24 février 2014, Facebook a annoncé la fin du service pour les adresses @facebook.com¹². Prochainement, les e-mails envoyés à ces adresses seront transférés à l'adresse e-mail renseignée dans le profil et n'apparaîtront plus dans les messages. La première solution présentée ici n'est donc plus d'actualité.

La seconde nécessite le développement d'une application utilisant l'API de Facebook, basée sur le protocole d'autorisation OAuth décrit plus haut. Le citoyen pourra d'une part se connecter avec son compte Facebook à une application sur son e-box, prouvant ainsi qu'il a bien accès à ce compte Facebook et d'autre part autoriser l'e-box à interagir avec lui, au travers de « notifications » (petite icône de globe terrestre en haut à droite, voir illustration en section 3.5) ou de « requêtes » (nombre apparaissant dans la colonne de gauche, à côté du nom d'une application, voir illustration en section 3.6).

Le citoyen contrôle parfaitement ce à quoi l'e-box aura accès et dans notre cas, seules les informations déjà publiques seront suffisantes, puisque tout ce qui nous intéresse, c'est de pouvoir recevoir des notifications. Aucun risque donc que l'État sache où le citoyen est parti en vacances, ni qu'il participe à une fête alors qu'un certificat médical lui interdit toute sortie ... Par ailleurs, il peut décider à tout moment de désactiver l'application et donc d'empêcher toute interaction avec son e-box, qui n'aura plus aucun moyen de le contacter via son compte Facebook, ni d'accéder aux informations limitées auxquelles l'application avait accès.

Une fois la connexion entre son e-box et son compte Facebook établie, l'organisme gérant cette e-box peut lui envoyer un message. Il en sera alors notifié dès qu'il se connecte sur Facebook. Notons que lorsqu'on parle d'application Facebook, il ne s'agit en rien d'un programme à installer sur son ordinateur ou son smartphone, mais simplement de fonctionnalités supplémentaires que l'on rajoute sur la page de Facebook.

Nous rentrons maintenant dans ces considérations beaucoup plus techniques sur la façon d'implémenter ce que nous présentons ci-dessus. Le reste de la section est donc destinée à des lecteurs ayant un profil plus technique.

¹² <https://www.facebook.com/help/703286543026907>

3.2. Création/configuration de l'application

Dans les sections suivantes, nous allons détailler les considérations techniques permettant d'arriver au résultat présenté ci-dessus. Il ne s'agit cependant pas d'un manuel du développeur (il sera nécessaire d'avoir déjà une connaissance de base de l'API de Facebook pour bien comprendre la suite), mais simplement d'une vue d'ensemble de la démarche à suivre.

Supposons, pour la suite, que nous développons une application nommée « BeTizenConnect », que nous hébergeons sur un serveur nommé www.betizenconnect.be.

Une application Facebook (*Facebook app*) est un site web, développé sur des serveurs externes à Facebook. Il existe plusieurs configurations possibles.

3.2.1. Application intégrée ou externe

L'application (donc le site web) peut être chargée soit au sein d'une page Facebook, soit tout à fait à l'extérieur de Facebook.

Dans le premier cas (Figure 1), elle apparaît au sein de Facebook, dans une « *iframe* » (avec un menu supérieur et un panneau latéral droit). Dans le second cas, elle est consultée tout à fait à l'extérieur, sans menu ou autre élément visible de Facebook. Dans les deux cas, le contenu de l'application n'est pas géré par Facebook et ne passe pas par ses serveurs : il est directement récupéré par le navigateur de l'utilisateur sur le serveur de l'application. Le fait de ne pas être intégré dans une application Facebook ne limite pas les possibilités de requêtes. Par contre, si l'application génère une notification, elle ne sera visible que lorsque l'utilisateur se connecte directement à Facebook.

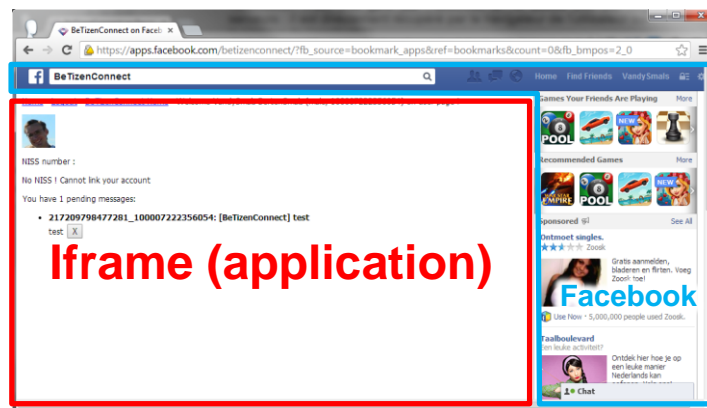


Figure 1 App intégrée dans une iframe (<https://apps.facebook.com/betizenconnect>)

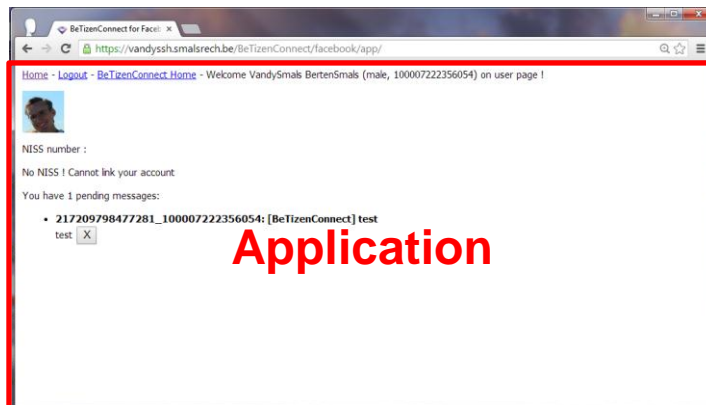


Figure 2 Page web indépendante. (<http://www.betizenconnect.be>)

3.2.2. PHP ou Javascript

Les interactions entre l'application et les serveurs de Facebook peuvent se faire (principalement) de deux façons :

1. Soit le serveur de l'application interagit avec le serveur de Facebook, avec une librairie en PHP fournie par Facebook et génère une page qui contient des informations extraites de Facebook.
2. Soit le serveur de l'application envoie une page « statique » (c'est-à-dire sans données émanant de Facebook), qui contient du Javascript. C'est alors le navigateur de l'utilisateur qui va, avec Javascript, interagir avec le serveur de Facebook, pour ajouter à la page statique des informations dynamiques.

3.2.3. Combinaisons

On peut alors envisager les quatre scénarios présentés dans le Tableau 1 (qui peuvent bien entendu être combinés) :

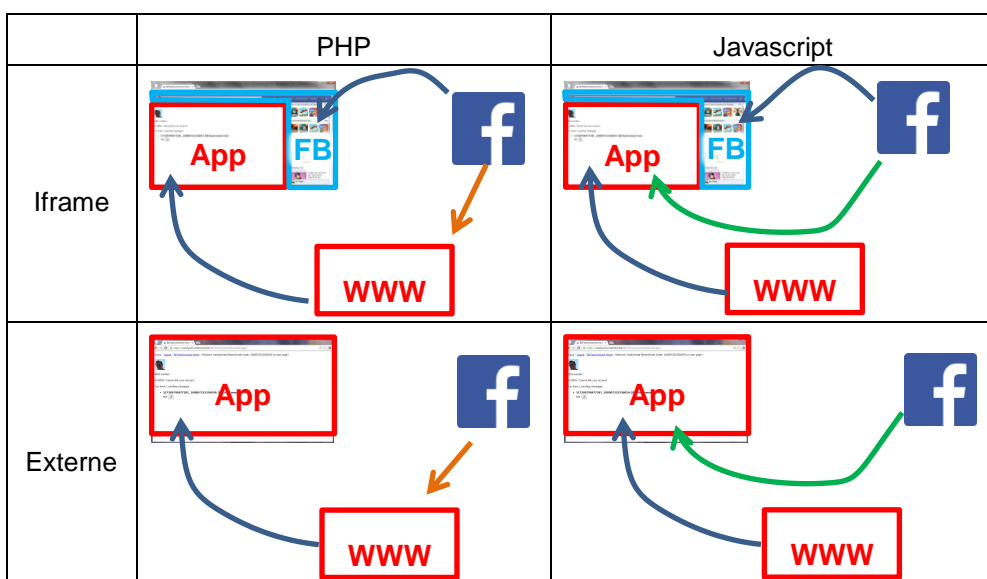


Tableau 1 Combinaisons possibles pour une application Facebook

Les **flèches bleues** correspondent à des requêtes http(s) « simples », initiées par le navigateur. Les **flèches vertes** correspondent à des requêtes OAuth, initiées par du javascript. Les **flèches orange** correspondent à des requêtes OAuth, initiées par le serveur de l'application, typiquement en PHP.

3.2.4. Configuration

Une application Facebook se crée sur <https://developers.facebook.com>. Il faut lui donner un nom (qui peut être « verbeux »), ainsi qu'un « namespace », qui ne peut contenir que des lettres minuscules (non accentuées), des « # » et des « _ », et doit contenir minimum 7 lettres.

Une fois l'application créée, elle reçoit un identifiant (App Id, nombre d'une quinzaine de chiffres), ainsi qu'une clé secrète (App Secret), longue chaîne hexadécimale, qui joue le rôle de mot de passe.

On précisera ensuite à Facebook un « Canvas URL », qui correspond à l'URL de l'application, sur un serveur externe à Facebook, par exemple <http://www.betizenconnect.com/index.php>.

3.3. Autorisation

Si l'obtention d'une App Id (et de son App Secret correspondant) est suffisante pour extraire des informations publiques de Facebook, telles que les messages postés sur une « page », cela ne suffit pas pour accéder à des informations privées, telles que le contenu d'un profil d'utilisateur. Pour ce faire, ce dernier devra donner son autorisation.

En utilisant l'API, on crée d'abord, dans le fichier index.php accessible à l'adresse <http://www.betizenconnect.be/index.php>, un objet « FacebookSession ». Aucune action n'est entreprise, les *credentials* sont simplement stockés dans l'objet créé.

```
FacebookSession::setDefaultApplication('[appid]', '[secret]');
$helper = new FacebookCanvasLoginHelper();
$session = $helper->getSession();
```

Si \$session n'est pas vide, l'utilisateur est connecté et l'on peut commencer à exécuter des requêtes. S'il s'agit d'une première connexion, le résultat sera vide.

On peut alors demander à l'API de générer une URL de connexion, grâce à un appel à :

```
$loginUrl = $helper->getLoginUrl($scope);
```

Le paramètre « scope » représente la liste des autorisations que Facebook accordera à l'application. Il peut s'agir de la consultation de l'adresse e-mail ou de la permission d'écrire sur le mur au nom de l'utilisateur. Si ce paramètre est vide, l'application peut uniquement demander l'accès en lecture aux informations déjà publiques, à savoir les informations de base (nom, prénom, photo de profil et liste d'amis).

L'URL générée sera par exemple de la forme suivante :

```
https://www.facebook.com/dialog/oauth
?client_id=624192117627745
&redirect_uri=http%3A%2F%2Fwww.betizenconnect.be%2Fcallback.php%2F
&state=2bc948edf1f09fe5ead35b18f178b553
&scope=email%2C+user_birthday%2C+user_about_me
```

Dans cette URL :

- `client_id` est l'identifiant de l'application (AppId)

- `redirect_url` est l'adresse de la page vers laquelle l'utilisateur sera dirigé après l'authentification
- `state` est un nombre aléatoire, qui sert à éviter le « *Cross-site request forgery*¹³ »
- `scope`, dans ce cas-ci, autorise l'accès à l'adresse e-mail de l'utilisateur, à sa date de naissance et au texte du « À propos ».

Il faudra maintenant rediriger l'utilisateur vers l'URL reçue, soit en affichant un lien à cliquer, soit en effectuant une redirection automatique, typiquement avec un script JavaScript du type « `top.location.href = '$loginUrl';` ».

L'utilisateur quitte alors « `myserver.com` » et est dirigé vers Facebook, où il devra, si ce n'est pas encore fait, se connecter. Remarquons que le nom d'utilisateur et le mot de passe sont communiqués à Facebook uniquement et jamais à l'application.

Un exemple d'application avec un « `scope` » vide ferait ensuite apparaître une fenêtre du type :

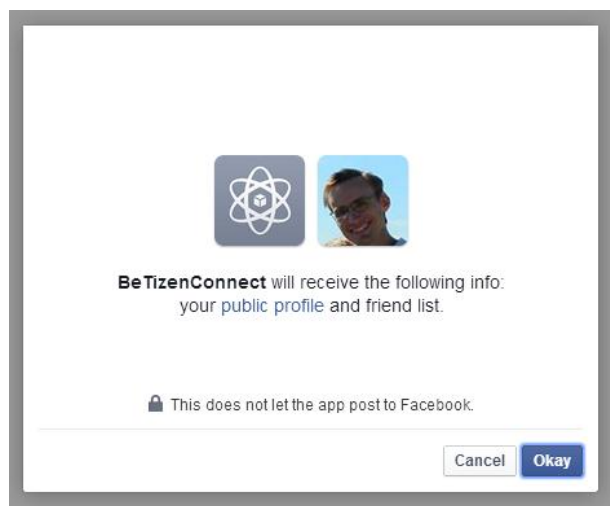


Figure 3 - Message d'avertissement de Facebook à l'installation d'une application.

Si l'utilisateur accepte, Facebook génère une page qui effectue une redirection vers l'adresse de « `callback` » mentionnée dans le paramètre « `redirect_uri` » présenté ci-dessus, avec, en plus, deux paramètres :

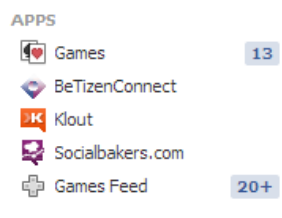
- un « `authorization code` », longue chaîne de caractères qui servira lors des requêtes ultérieures durant la session.
- un « `state` », identique à celui envoyé juste avant, permettant de s'assurer que l'on n'essaie pas de réinjecter un « `callback` » précédent.

[https://www.betizenconnect.be/callback.php?code=AQDTZYILOq04rT4V4Zv4D4tujTwdAY5EA82SBzDn\[...\]DEIzInVqVcR16m5&state=2bc948edf1f09fe5ead35b18f178b553#_=_](https://www.betizenconnect.be/callback.php?code=AQDTZYILOq04rT4V4Zv4D4tujTwdAY5EA82SBzDn[...]DEIzInVqVcR16m5&state=2bc948edf1f09fe5ead35b18f178b553#_=_)

Le développeur ne doit rien faire explicitement de ces « `code` » et « `state` » : ils sont automatiquement récupérés et stockés par l'API de Facebook. Mais c'est grâce à ce code que les requêtes ultérieures pourront être effectuées.

¹³ Plus de détails sur http://fr.wikipedia.org/wiki/Cross-Site_Request_Forgery

Une fois que l'utilisateur a accepté l'application en cliquant sur le bouton « Okay » de la figure ci-dessus, il verra apparaître, sur sa page Facebook, une mention de l'application sur la gauche de la page :



En cliquant sur « BeTizenConnect », on se retrouve sur <https://apps.facebook.com/betizenconnect>, tel que présenté en Figure 1.

3.4. Requêtes

Une fois la « connexion » établie, comme décrite ci-dessus, l'obtention d'informations s'effectue pas un simple appel à une fonction de l'API. Voici quelques exemples de requêtes et une partie du résultat de la première :

- ```

$user_profile = (new FacebookRequest($session, 'GET', " /$userid"))-
>execute()->getGraphObject()->asArray();
// Informations à propos de l'utilisateur courant
Array (
 [id] => 639839589
 [name] => Vandy Berten
 [first_name] => Vandy
 [last_name] => Berten
 [link] => https://www.facebook.com/vandy.berten
 [...]
 [locale] => fr_FR
 [verified] => 1
 [updated_time] => 2014-04-02T09:38:47+0000
 [username] => vandy.berten
)

```
- ```

$user_profile = (new FacebookRequest($session, 'GET',
"/639839589"))->execute()->getGraphObject()->asArray();
// Informations à propos d'un utilisateur ayant installé
l'application

```

3.5. Notifications

Il existe deux mécanismes permettant d'avertir l'utilisateur : le premier passe par les « notifications », et fait apparaître un nombre en rouge à côté du globe :

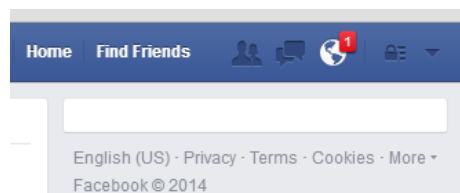


Figure 4 - Notification

Un clic sur ce globe ouvre une liste des dernières notifications. La création d'une telle notification s'effectue simplement par un appel à l'API :

```

$session = new FacebookSession('[appid]', '[secret]');
new FacebookRequest($session, 'POST', ' /[user_id]/notifications',
array(

```

```
'href' => '?param1=value1&param2=value2',
'template' => 'Texte de la notification'
));
```

Notons qu'il n'est pas nécessaire que l'utilisateur [user_id] soit connecté pour exécuter cette opération. La session Facebook n'est ici pas créée avec les informations d'un utilisateur, mais avec celles de l'application. La requête est donc effectuée au nom de l'application et non d'un utilisateur. Il faut par contre qu'il ait préalablement « installé » l'application pour qu'il puisse recevoir cette notification.

3.5.1. href

Le paramètre « href » permet de passer des paramètres « GET » à l'application, et donc d'avoir un comportement différent en fonction de la notification qui a été cliquée. On ne peut donc pas fournir d'URL externe à l'application.

3.5.2. template

Le paramètre « template » précisera le texte qui apparaîtra lorsque l'on clique sur le globe. Il n'y a pas de distinction entre un sujet et un corps de texte.

3.6. App Requests

Indépendamment des messages « classiques » (uniquement accessibles entre amis, les applications ne peuvent pas en envoyer), Facebook gère un mécanisme de « app requests », sorte de messages, qu'une application peut envoyer à l'utilisateur, lire ou supprimer. Une application ne peut évidemment pas consulter les requests des autres applications, mais peut obtenir, pour chaque utilisateur, même s'il n'est pas connecté, la liste de « requests » qu'il n'a pas supprimées.

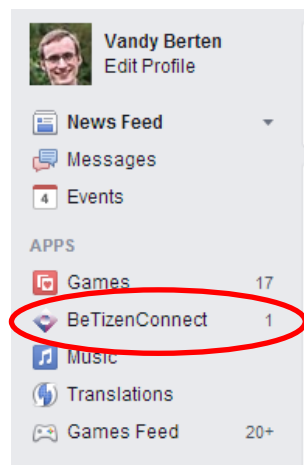


Figure 5 - App request

Lorsqu'une application envoie une « app request » à un utilisateur, celui-ci verra apparaître, dans le volet de gauche, le nombre de requests qu'il a reçues depuis la dernière fois où il a accédé à l'application (voir Figure 5).

Notons que le nombre ne disparaît que lorsque l'on clique sur le nom de l'application à gauche, et pas si on accède à l'application autrement (en tapant directement l'URL ou en cliquant sur la notification présentée ci-dessus). Par ailleurs, il disparaît même si l'application n'affiche pas la requête en question.

L'envoi d'un « app request » s'effectue grâce à la requête suivante :

```
new FacebookRequest($session, 'POST',
    '/[user_id]/apprequests',
    array(
        'message' => '[Subject]',
        'data' => '[Content]',
    ));
```

La récupération de la liste des requêtes s'effectue avec :

```
new FacebookRequest($session, 'GET', '/[user_id]/apprequests');
```

La suppression d'une requête « request_id » s'effectue grâce à :

```
new FacebookRequest($session, 'DELETE', '/[request_id]');
```

Remarquons que si le contenu généré par l'application n'est pas accessible à Facebook, le texte des notifications et des requêtes l'est bien. Il est donc prudent de ne rien mettre de confidentiel dans ces données.

4. Twitter

4.1. Introduction



Le cas de Twitter est un peu plus simple. Il existe deux types de messages sur Twitter : les « tweets », qui sont publics, et les « messages privés » (ou « direct messages »), accessibles via la petite icône d'enveloppe au-dessus de la page d'accueil. Ce sont ces derniers qui nous intéressent ; seul l'utilisateur y aura accès. Pour pouvoir envoyer un message privé à un autre utilisateur, il suffit d'être suivi par cet autre utilisateur (être donc un de ses « followers »).

On pourra donc créer un profil, disons « @eBox », et inviter chaque citoyen désirent être contacté via Twitter à devenir « follower » de ce compte. Une application peut ensuite être autorisée par ce profil « @eBox » à écrire des messages privés en son nom, et le tour est joué.

Il faut cependant lier le compte Twitter du citoyen à son compte sur l'e-box. Cela peut se faire soit en communiquant dans l'e-box son « screen name » de Twitter (nom d'utilisateur), soit, pour garantir que le citoyen en question a bien un accès à ce compte, en utilisant la fonctionnalité « Se connecter avec Twitter » de OAuth, décrite plus haut.

Notons qu'avec le mécanisme présenté ci-dessous, le citoyen recevra un message dans un système ressemblant à un « chat », mais qu'il ne pourra pas y répondre si le compte « @eBox » ne le suit pas.

À nouveau, la suite de cette section s'adresse à un public averti, avec des compétences techniques IT de base.

Remarque : Le présent document est basé sur la version 1.1 de l'API de Twitter et non sur le SDK « Fabric », nouvellement recommandé par Twitter. Une mise à jour est donc nécessaire.

4.2. Authentification

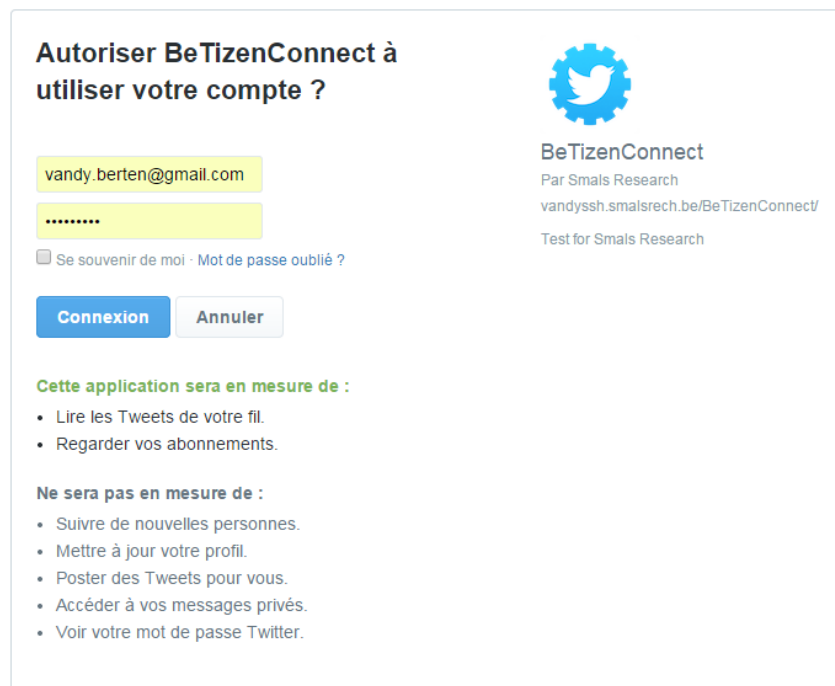
La première phase, optionnelle, consiste à s'assurer que la personne qui renseigne un « *screen name* » a bien accès à ce compte Twitter. Fondamentalement, cela n'est pas nécessaire à l'envoi d'un message à ce compte, puisqu'il suffit que le compte en question suive « @eBox » pour qu'on puisse lui envoyer un message. Mais si l'on veut limiter les erreurs et empêcher un citoyen d'introduire un mauvais compte Twitter (intentionnellement ou par erreur), il est préférable de passer par cette étape.

Pour ce faire, il est possible d'interagir directement avec l'API de Twitter, mais celle-ci est de bas niveau et laborieuse à manipuler. Il est préférable d'utiliser une des nombreuses bibliothèques disponibles¹⁴. Dans notre PoC, nous nous sommes servis de « *twitteroauth* »¹⁵. Le principe est le suivant :

¹⁴ <https://dev.twitter.com/overview/api/twitter-libraries>

¹⁵ <https://github.com/abraham/twitteroauth>

1. Sur le portail (hors de Twitter), on invite l'utilisateur à se rendre sur une page, par exemple <https://www.betizenconnect.be/twitter/login.php>
2. Sur cette page, on redirige automatiquement l'utilisateur vers une page de Twitter, sur une URL générée par la librairie twitteroauth, contenant les tokens d'authentification de l'application, ainsi qu'une url de « *callback* »
3. Twitter demande alors à l'utilisateur de s'authentifier (si ce n'est déjà fait), et s'assure qu'il est d'accord d'installer l'application, en lui précisant ce que l'application pourra faire (dans notre cas, uniquement accéder aux informations déjà publiques ; c'est le minimum qu'il est possible de demander) :



The screenshot shows a Twitter authorization window titled "Autoriser BeTizenConnect à utiliser votre compte ?". It features a Twitter logo and the application name "BeTizenConnect" with details: "Par Smals Research" and "vandyssh.smalsrech.be/BeTizenConnect/". Below this, there are input fields for the email "vandy.berten@gmail.com" and a password field. A checkbox for "Se souvenir de moi" and a link for "Mot de passe oublié ?" are present. Two buttons, "Connexion" and "Annuler", are at the bottom. A section titled "Cette application sera en mesure de :" lists permissions: "Lire les Tweets de votre fil" and "Regarder vos abonnements." Another section titled "Ne sera pas en mesure de :" lists actions not permitted: "Suivre de nouvelles personnes", "Mettre à jour votre profil", "Poster des Tweets pour vous", "Accéder à vos messages privés", and "Voir votre mot de passe Twitter."

Figure 6 - Message d'avertissement de Twitter à l'installation d'une application.

4. Une fois l'utilisateur authentifié, Twitter le redirige automatiquement vers l'URL de callback fournie plus haut, en lui transmettant des « tokens » permettant à notre page (plus précisément la librairie) de s'assurer que la redirection vient bien de Twitter. On peut donc enregistrer l'utilisateur et associer le screen name au citoyen.

Il reste maintenant à s'assurer que le citoyen suive (follow) le compte @eBox, ce qui peut se faire en effectuant une requête « GET » (accompagnée des tokens adéquats, grâce à une librairie adéquate) vers :

[https://api.twitter.com/1.1/friendships/show.json?source_screen_name=\[SOURCE\]&target_screen_name=\[TARGET\]](https://api.twitter.com/1.1/friendships/show.json?source_screen_name=[SOURCE]&target_screen_name=[TARGET])

Cette requête renverra une réponse de type « json ». Si le citoyen ne suit pas le compte nécessaire, on peut directement lui proposer un bouton « Follow », dont le code peut être obtenu sur <https://about.twitter.com/fr/resources/buttons>.

4.3. Envoi de Direct Message

Pour qu'un compte A puisse envoyer un « Direct Message » à un compte B, il faut, d'une part, qu'une application Twitter ait été créée par A et, d'autre part, que

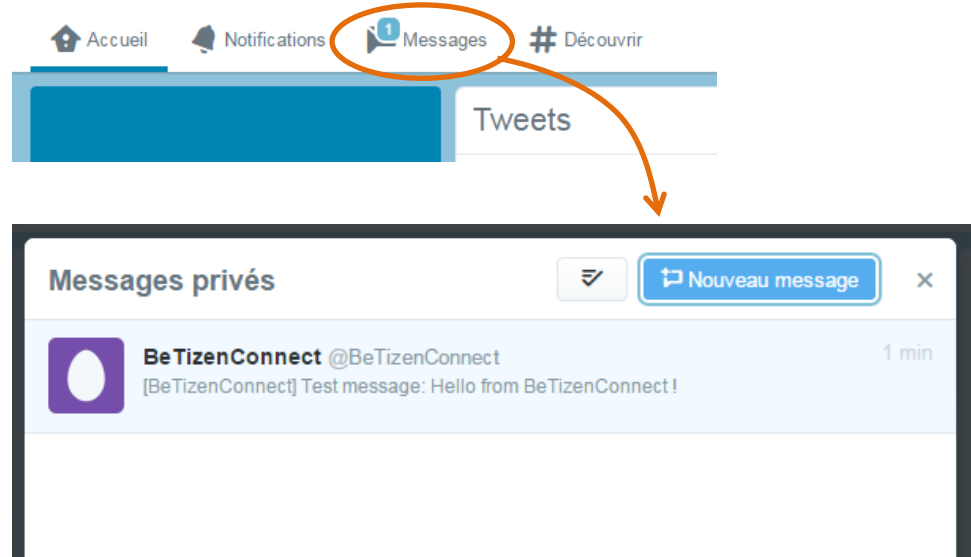
B suit A. Ensuite, il suffit d'envoyer une requête POST, accompagnée des informations adéquates.

À nouveau, il est préférable d'utiliser une librairie permettant de gérer les éléments de bas niveau tels que les tokens. Nous avons, dans notre PoC, utilisé Twitter-API-PHP¹⁶.

Le code d'envoi d'un message tient en quelques lignes (TwitterAPIExchange est la classe principale fournie par la librairie Twitter-API-PHP).

```
$twitter = new TwitterAPIExchange($twitter_settings);  
$url = 'https://api.twitter.com/1.1/direct_messages/new.json' ;  
$postfield = array('user_id' => [USERID], 'text' => [MESSAGE]);  
$result = $twitter->setPostfields($postfield)  
           ->buildOauth($url, 'POST')  
           ->performRequest();
```

Le message apparaît alors directement dans Twitter :



¹⁶ <https://github.com/J7mbo/twitter-api-php>

5. Autres réseaux

5.1. LinkedIn



D'après les informations dont nous disposons jusqu'ici, LinkedIn ne permet pas de mettre en place une procédure similaire à celle décrite ci-dessus. En effet, s'il est également possible de se connecter à un site web ayant implémenté les bibliothèques de LinkedIn en se servant de son compte LinkedIn, ce dernier empêche à une application l'envoi d'un message à un utilisateur A de la part d'un utilisateur B sans qu'il n'y ait d'action manuelle effectuée par l'utilisateur B. Il ne semble donc pas possible à un serveur d'envoyer automatiquement des messages à des utilisateurs, sans l'intervention manuelle d'un fonctionnaire, ce qui n'est bien évidemment pas envisageable.

5.2. Google+



Le réseau social de Google offre une série d'API permettant d'écrire des applications et de faire en sorte que plusieurs utilisateurs d'une même application puissent "chatter" entre eux (en utilisant "Hangout"), un peu comme cela se fait dans les "Google Docs" (drive.google.com). Mais rien qui, à notre connaissance, ne permette d'envoyer un message à un utilisateur qu'il verrait dès qu'il se connecte sur la page d'accueil de Google+. Il ne nous semble donc pas qu'à ce stade, ce réseau soit un candidat pour notre objectif.

5.3. Snapchat



Snapchat est un réseau social dans lesquels les messages envoyés sont « autodétruits » après un temps maximum de 10 secondes... et n'est donc de toute évidence pas adapté au problème décrit ci-dessus.



5.4. Instagram, Pinterest

Ces deux réseaux sociaux n'offrent, à notre connaissance, pas de messagerie privée et ne sont donc pas candidats.



5.5. WhatsApp



Cet outil de messagerie, qui vient d'être racheté par Facebook, ne propose pas officiellement d'API permettant d'interagir avec d'autres utilisateurs. Les quelques librairies et outils que nous avons pu trouver jusqu'ici relèvent vraiment du « hacking », en permettant à un serveur de se faire passer pour un téléphone ; il ne serait donc pas raisonnable de les utiliser dans un contexte officiel.

5.6. Skype



Skype n'est pas à proprement parler un réseau social, mais pourrait très bien faire l'affaire pour ce qui nous concerne. Cependant, si une API était disponible par le passé pour envoyer des messages, elle a été supprimée lors du rachat de Skype par Microsoft. Il semblerait que certains outils soient cependant encore disponibles, mais étant donné qu'ils ne sont pas officiellement supportés, nous ne les avons pas étudiés plus en détails.

5.7. Netlog



Réseau social similaire à Facebook mais largement moins répandu, Netlog, développé en Belgique, propose une API principalement basée sur OpenSocial¹⁷ de Google. Ce réseau semble cependant en fort déclin¹⁸ et la documentation de son API est très incomplète, pleine d'erreurs et de liens morts vers des exemples, raisons pour lesquelles nous ne l'avons que partiellement testé.

¹⁷ <http://opensocial.org/>

¹⁸ <http://www.alexa.com/siteinfo/netlog.com>

6. Conclusions et recommandations

Il est parfaitement possible d'offrir aux citoyens une façon alternative d'être contacté par les administrations, en passant par les réseaux sociaux. Les deux réseaux sociaux les plus répandus, Facebook et Twitter, offrent tous les deux des mécanismes nécessaires, mais très différents.

Les compétences techniques nécessaires sont à la portée de n'importe quel développeur de qualité, les mécanismes de bases étant simples et relativement standards.

Contacté un citoyen par les réseaux sociaux peut être très pertinent dans le cadre d'une dématérialisation des échanges avec les administrations, lui permettant de choisir le canal le plus adapté à ses habitudes.

Il ne faut cependant pas négliger les problèmes de qualité des données, différents de ceux liés à la gestion des adresses e-mail, mais bien présents : un citoyen peut désinstaller l'application nécessaire (ce dont on peut se rendre compte facilement) ou ne plus utiliser son compte Twitter (difficile à détecter) ou Facebook (on peut voir depuis combien de temps il n'a plus ouvert l'application, mais pas s'il accède encore à son compte Facebook ou non).

Par ailleurs, si les standards des protocoles e-mail sont relativement stables, ce n'est pas le cas des API proposées par Facebook ou Twitter, dont des nouvelles versions, pas toujours compatibles avec les précédentes, sont régulièrement proposées. Il est donc nécessaire de suivre de près l'évolution des librairies.

De plus, si aujourd'hui Facebook et Twitter sont les réseaux les plus adaptés à ce que nous présentons ici, cela pourrait ne plus être le cas d'ici dans le futur. Une application de type « eBox » comme nous la proposons ici devra donc être conçue de façon suffisamment modulaire pour que l'ajout d'un nouveau réseau social ne nécessite pas une réécriture complète.

Pour autant que les moyens nécessaires, mais raisonnables, soient mis en place, il est donc possible de donner une image résolument moderne des administrations, tout en améliorant la qualité des interactions avec les citoyens.

La section Recherche de Smals produit régulièrement des publications couvrant de nombreux domaines du marché IT actuel. Vous pouvez obtenir ces publications via le site web de la section Recherche :

<http://www.smalsresearch.be>