

Les APIs Twitter demandent leur REST

Presented by: **Bérengère LAGRANGE - POWER**



Les réseaux sociaux tels que Facebook, Twitter et Instagram sont devenus incontournables pour la communication des entreprises, des marques et de leurs produits. On a même inventé un métier pour gérer les publications sur les réseaux sociaux : le Community Manager.

Seulement, voilà ! Toutes les entreprises n'ont pas, forcément, les ressources nécessaires pour créer ce genre de poste. En-deçà de 50 salariés, et, dépendamment de l'activité de la société, il est difficile d'embaucher quelqu'un qui serait spécifiquement en charge de la communication via ces réseaux sociaux.

Or, il est, néanmoins, assez facile de mettre en œuvre dans nos applications, des fonctionnalités particulières pour que nos clients puissent utiliser ces nouveaux moyens de communication avec leur clientèle et donner de la visibilité à son activité.

Dans ce document, nous allons appréhender toutes les notions nécessaires pour pouvoir communiquer sur une plateforme telle que Twitter. Lorsque vous aurez assimilé les principes appliqués à Twitter, vous serez en mesure de développer des passerelles vers la plupart des plateformes utilisant des technologies similaires.

TWITTER : QUÉSACO ?

Twitter est une plateforme de partage de contenus créée en 2006, aussi appelée outil de "microblogage". Pourquoi "micro" ? Parce que chaque message diffusé sur ce réseau social se doit d'être bref et concis. Les messages, communément appelés "tweets", sont limités à 140 caractères, le principe étant, initialement, de pouvoir diffuser de l'information par SMS. Cette limite de 140 caractères qui peut paraître incongrue de nos jours est l'ADN propre de Twitter qui en fait son charme et son succès.

Les utilisateurs peuvent suivre ou consulter les publications de n'importe quel autre utilisateur, devenant alors des "followers", autrement dit, des abonnés.

Lorsque vous créez votre compte sur Twitter, vous choisissez un pseudonyme et obtenez une page spécifique pour votre profil, que vous pouvez rendre privé ou public (par défaut c'est public). Comme pour tous les réseaux sociaux, vous pouvez définir une photo de profil, qui s'affichera à chacune de vos publications. Si ce n'est pas une photo vous représentant on parle alors d'avatar.

Voici un exemple de page sur le site twitter.com :

Accueil Notifications Messages Recherchez sur Twitter



Bérangère Lagrange
@BerenLagrange
Software & web developer #4D
#JavaScript
Lyon, France
Inscrit en septembre 2011
125 Photos et vidéos

TWEETS 1 797 ABONNEMENTS 285 ABONNÉS 195 AIMÉS 59 LISTES 10 Éditer le profil

Tweets Tweets & réponses Photos & vidéos

Bérangère Lagrange @BerenLagrange · 7 mars
140 caractères ne seront pas suffisants pour aborder les #APIs #Twitter et les #objets #4D lors du @4dSummit



Sessions - summit.4d.com
Inscription Au cœur du multithreading et de l'architecture 64 bits 4D tire déjà parti de l'adaptabilité des machines multi-cœurs, via notamment son serveur de base de d...
summit.4d.com

Bérangère Lagrange @BerenLagrange · 16 mars
Développeur, développeuse, viens rejoindre notre équipe ! Nous recherchons 2 développeurs web / applications
power.fr/recrutement.as...

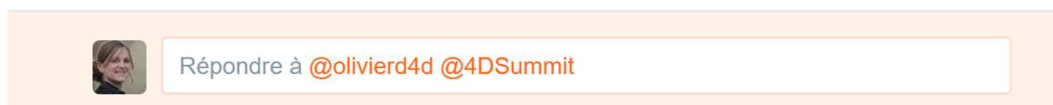
Suggestions · Actualiser · Tout afficher
Asmae Benkirane @abenkira

Nous y voyons les dernières publications, le nombre d'abonnés, le nombre d'utilisateurs auxquels vous êtes vous-même abonné, ...


Vous pourrez accéder à votre "**timeline**", page sur laquelle s'affichent les publications des comptes que vous suivez, triés de manière antéchronologique, principe réutilisé par Instagram par exemple.

Contrairement à Facebook, les tweets ne peuvent pas être commentés directement. Si une publication vous paraît pertinente, vous pouvez la "**re-tweeter**" pour la diffuser à vos propres followers.

A défaut de pouvoir commenter les publications, on peut y répondre. Un utilisateur peut également s'adresser à un autre utilisateur en le mentionnant dans son tweet avec un @ et son pseudonyme :



Autre élément très important qui fait partie des fondamentaux de l'utilisation de la plateforme : les "**hashtags**", traduits par "mots-dièse" en français. Il s'agit de mots-clefs, précédés du caractère #, permettant d'effectuer des recherches concernant un terme particulier et de regrouper les tweets traitant d'un même sujet. Initialement, on retrouve l'utilisation du dièse dans certains langages de programmation, ainsi que dans l'univers des IRC (canaux de discussion sur Internet) pour lesquels les « salons » de discussion étaient plutôt thématiques : le nom de ces canaux débutant alors par le # suivi du thème. Ce concept de hashtag, qui contribué au succès de Twitter, a largement été repris sur d'autres plateformes sociales, telles que Facebook ou Instagram. Par ailleurs, la plupart des émissions télévisées font référence à un hashtag pour inciter les "Twittos" à réagir en direct sur le déroulé de l'émission.

Certains comptes utilisateur portent un badge "certifié" , notamment pour les personnalités politiques, les célébrités, les grandes marques ou les personnes dites influentes sur la twittosphère. Cela permet d'authentifier l'identité de ces comptes, surtout présents dans les domaines de la musique, du spectacle, de la mode, du gouvernement, de la politique, de la religion, du journalisme, des médias, du sport, des affaires et d'autres centres d'intérêt. Un utilisateur ne peut demander de lui-même à ce que son compte soit certifié, c'est Twitter qui décerne le badge selon plusieurs critères. Rendez-vous sur cet article <https://support.twitter.com/articles/269158?lang=fr> pour vérifier si vous êtes éligible à la certification et si vous réunissez tous les critères.

A présent, vous connaissez les principes de base de Twitter. L'utilisation de la plateforme peut même aller très loin, jusqu'à la mise en avant des publications, moyennant finances, bien entendu. Certaines APIs ont été conçues dans cette optique, vous pouvez vous renseigner sur cette page <https://dev.twitter.com/ads/overview>.

Dans la suite de ce document, nous allons voir comment une application 4D peut interagir avec une plateforme telle que Twitter et quelles sont les notions essentielles à connaître.

API, REST, JSON, OAUTH, ...

Il convient de préciser une définition basique de quelques termes techniques pour mieux appréhender l'utilisation des services web proposés par Twitter.

API

Acronyme pour *Application Programming Interface*, traduire par Interface applicative de programmation.

Il s'agit d'un ensemble de fonctionnalités "partagées" par un programme tiers. En faisant appel à ces méthodes, vous pouvez bénéficier de services existants sans avoir besoin de les redévelopper vous-même. Par exemple, si vous souhaitez intégrer une cartographie interactive sur votre site web, vous pourrez utiliser les APIs de Google Maps.

Nul besoin de savoir comment fonctionnent ces méthodes en interne, le tout est d'obtenir le résultat escompté. 4D comporte lui-même une interface de programmation, n'est-ce pas ? En réalité, tout n'est qu'API, aujourd'hui ! Les systèmes d'exploitation, les SGBD, les langages de programmation, etc.

Twitter a mis en place des APIs web pour permettre aux développeurs de faire interagir leur application ou leur site web avec ce réseau social. L'avantage principal étant de pouvoir déléguer et automatiser les contenus à publier, d'autant plus si ces informations sont dans votre base de données!

Twitter met à disposition des développeurs deux types d'APIs : REST vs Streaming. Votre choix se portera sur les APIs les plus adaptées à vos besoins. En effet, pour avoir des mises à jour en temps réel, préférez les APIs de streaming. Dans ce document, nous nous intéresserons aux APIs REST qui sont, dans la majorité des situations, tout à fait adaptées aux besoins de nos clients.

Même si l'utilisation d'APIs de certains grands du web est gratuite, elle est souvent limitée. En effet, pour Twitter, par exemple, les requêtes seront limitées à 350 par heure en mode authentifié. Les APIs Google sont également limitées. Par exemple, pour Google Maps, leur utilisation est limitée à 25 000 chargements de cartes sur 24 heures, pendant 90 jours. Au-delà, il faudra passer sur une version payante du service.

REST

Acronyme pour *Representational State Transfer*.

Il s'agit d'un système d'architecture reposant sur le protocole HTTP pour procéder à divers traitements, tels que la lecture, l'écriture, la modification et la suppression. D'ailleurs, c'est le protocole d'échange entre 4D et Wakanda dans le cadre de 4D Mobile.

Bien que n'étant pas un standard, il utilise lui-même des standards du web, en particulier les suivants :

- URI pour identifier la ressource de manière très structurée.
- HTTP pour fournir les opérations nécessaires au traitement de la ressource. On utilisera ainsi les verbes suivants :
 - GET pour la récupération ;
 - POST pour la création ;
 - UPDATE pour la modification ;
 - DELETE pour la suppression.

Ex. GET `http://monserveur/rest/myresource/resource_id` nous retournera les informations sur une ressource donnée.

Le service REST nous renvoie une représentation des données dans un format choisi : XML, CSV, JSON, etc ... Si le service propose différents formats de retour, le client doit spécifier celui qu'il choisit dans l'entête de sa requête :

Ex. Accept : text/html

Ou encore Accept : application/json

Sur Twitter, par exemple, pour envoyer un tweet, on utilisera le verbe *POST* avec l'URI *status/update* :

```
POST https://api.twitter.com/1.1/statuses/update.json
```

Pour récupérer les tweets d'un utilisateur, on utilisera le verbe *GET* avec l'URI *statuses/user_timeline* :

```
GET https://api.twitter.com/1.1/statuses/user_timeline.json
```

Les ressources de l'API Twitter sont consultables ici : <https://dev.twitter.com/rest/public>

Enfin, il est généralement nécessaire d'utiliser un jeton d'authentification (token), qui permettra d'authentifier la requête et, éventuellement, l'utilisateur ou l'application.

D'ailleurs, pour les APIs Twitter, l'authentification se fait via OAuth.

OAuth

Il s'agit d'un protocole permettant d'autoriser un site web, un logiciel ou une application dite « consommateur » à utiliser une API sécurisée, le « fournisseur », pour le compte d'un utilisateur. L'avantage étant de ne pas avoir spécifiquement besoin de connaître les informations de connexion de l'utilisateur final. Ce n'est, donc, pas, en soi, un protocole d'authentification, mais une délégation d'autorisation.

Avec son framework 2.0, OAuth permet d'autoriser des applications tierces à utiliser, de manière restreinte, un service HTTP : comme une voiture de luxe pour laquelle vous disposeriez d'une clef spécifique à remettre au voiturier, ne lui permettant que de conduire sur une courte distance et ne lui permettant pas l'accès au coffre.

Sur Twitter, il y a plusieurs possibilités pour autoriser son application à utiliser les APIs :

- Si vous pouvez vous-même gérer le paramétrage du compte de l'utilisateur final (société ou marque) ou que vous souhaitez utiliser l'API à votre propre compte, vous pouvez utiliser les tokens mis à votre disposition sur Twitter pour autoriser votre application. C'est la méthode que nous allons utiliser pour notre démonstration.
En effet, l'avantage est de permettre l'accès aux utilisateurs à certaines fonctionnalités de Twitter, sans pour autant qu'ils ne connaissent les identifiants de connexion au compte leur ouvrant un contrôle total.
De plus, cela vous permet de mettre en place une automatisation des opérations en background (par exemple, pour récupérer chaque soir des données sur la journée écoulée, ou de poster des tweets de manière automatique). Vous pourrez également mettre en place une « liste d'attente » des opérations à effectuer. En effet, l'utilisation des APIs Twitter étant limitée à un certain nombre d'accès à l'API (rate limiting), il faut pouvoir soumettre une nouvelle fois des opérations lorsque l'utilisation est à nouveau disponible : pour cela, vous pourrez utiliser une procédure stockée sur votre serveur 4D qui viendra parcourir la file d'attente (principe largement inspiré d'un article sur le blog d'Olivier Deschanel : <http://www.4d.com/fr/blog/au-suivant.html>).
Vous pouvez également imaginer de pré-écrire des tweets qui seront diffusés à une heure précise pour être en phase avec un événement à venir. Vous restez concentré sur la gestion de votre événement et les tweets partent tous seuls au même moment !

- **Par identification de l'utilisateur** : en proposant une interface (web) de connexion où l'utilisateur pourra saisir des identifiants Twitter (login & mot de passe) à la suite de quoi, vous pourrez demander un token d'autorisation. Le token a une durée de vie limitée, vous obligeant ainsi soit à conserver les identifiants de connexion du compte, soit à redemander l'authentification de l'utilisateur lorsque le token est expiré.
Cette solution peut être nécessaire si votre outil est édité et destiné à être distribué « en masse ». Néanmoins, cela nécessite que l'utilisateur final connaisse les informations d'identification du compte : vous n'aurez, donc, pas d'intérêt particulier à restreindre certaines fonctionnalités de Twitter. De plus, la mise en place d'une automatisation et d'une liste d'attente sera plus complexe, du fait du *time to live* du token d'autorisation;
- « **Application-Only Authentication** » : accès restreint aux APIs, notamment pour les services dont le contexte utilisateur n'est pas requis. Ainsi, vous ne pourrez pas poster des tweets, ni rechercher des utilisateurs ;
- **Autorisation basée sur un code PIN** : si votre application ne peut accéder à un navigateur pour permettre la connexion de l'utilisateur, vous pourrez faire la demande à Twitter d'accéder à l'API : l'utilisateur devra, alors, saisir un code PIN pour autoriser l'accès ;

Selon le contexte, d'autres possibilités sont offertes pour l'autorisation, je vous invite à lire la documentation OAuth pour Twitter : <https://dev.twitter.com/oauth/overview>

Pour permettre à Twitter de vous identifier et de vous autoriser l'accès à ses APIs, il faut créer et déclarer votre application sur la page <https://apps.twitter.com/> (nécessite d'avoir un compte Twitter).

Spécifiez les champs obligatoires : nom, description, site web, et acceptez l'accord développer (que vous pouvez retrouver ici <https://dev.twitter.com/overview/terms/agreement-and-policy>) :

Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Vous pourrez alors obtenir les clefs et les tokens nécessaires pour utiliser les APIs :



Demo Summit

[Test OAuth](#)[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	YD8AN17RwqaCFGJ5w89HlfZVm
Consumer Secret (API Secret)	KFg4LivYh1wuw6IKRvbZfPvXcwwMFwujawsLBZ4wNRdhTIm9c0
Access Level	Read, write, and direct messages (modify app permissions)
Owner	BerenLagrange
Owner ID	377330954

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	377330954-cFmL6QBYygxZcqaPFVCx4IourWQIKfvJpVtmIf
Access Token Secret	hduAf7zvX57mHqWPBODzoeqJ05tfvJZ46hqrDbdCikDwP
Access Level	Read, write, and direct messages
Owner	BerenLagrange
Owner ID	377330954

Dans la section « Permissions », veuillez à bien cocher la permission de lecture écriture :

Demo Summit

[Test OAuth](#)[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Access

What type of access does your application need?

[Read more about our Application Permission Model.](#)

- ☐ Read only
- ☒ Read and Write
- ☐ Read, Write and Access direct messages

Note:

Changes to the application permission model will only reflect in access tokens obtained after the permission model change is saved. You will need to re-negotiate existing access tokens to alter the permission level associated with each of your application's users.

JSON

Acronyme pour *Javascript Object Notation*

Il s'agit d'un format de représentation textuelle de données, facilement lisible par un humain. Ce format est syntaxiquement identique au code JavaScript permettant de créer des objets. JSON est basé sur une structure très simple : une paire clef / valeur :

Exemple pour décrire une personne :

```
{ "firstname" : "Bérengère", "lastname" : "Lagrange", "company" : "Power" }
```

Les valeurs peuvent être de différents types : numérique, texte, booléen, tableau ou objet.

Ainsi, pour une liste de personnes, on pourra écrire :

```
{ "people" : [  
  { "firstname" : "Bérengère", "age" : 30 },  
  { "firstname" : "Arthur", "age" : 3 },  
  { "firstname" : "Augustine", "age" : 2 }  
]
```

Dans cet exemple, nous avons un tableau contenant 3 objets, à l'intérieur d'un seul objet.

Depuis la v14 de 4D, vous avez la possibilité de manipuler des objets JSON. Depuis la v15, 4D nous offre en plus la possibilité de créer des champs de type objet directement dans la base de données.

Les commandes du langage 4D relatives aux objets sont préfixées par **OB**. Voici comment créer les objets donnés en exemple ci-dessus :

```
C_OBJECT($person)  
OB SET($person;"firstname";"Bérengère")  
OB SET($person;"lastname";"Lagrange")  
OB SET($person;"company";"Power")
```

Dans notre second exemple, nous avons un tableau d'objets, imbriqué dans un autre objet. Pour ce faire, voici le code 4D que nous pourrions utiliser :

```
ARRAY OBJECT($_people;0)  
  
C_OBJECT($person1)  
OB SET($person1;"firstname";"Bérengère")  
OB SET($person1;"age";30)  
APPEND TO ARRAY($_people;$person1)  
  
C_OBJECT($person2)  
OB SET($person2;"firstname";"Arthur")  
OB SET($person2;"age";3)  
APPEND TO ARRAY($_people;$person2)  
  
C_OBJECT($person3)  
OB SET($person3;"firstname";"Augustine")  
OB SET($person3;"age";2)  
APPEND TO ARRAY($_people;$person3)  
  
C_OBJECT($myFamily)  
OB SET($myFamily;"people";$_people)
```


Aussi bien, si ces données sont en base de données, dans une table, nous pourrions créer notre objet de la sorte :

```
C_OBJECT($empty)
ARRAY OBJECT($_people;0)
ALL RECORDS([Person])
While (Not(LAST RECORD([Person])))
  C_OBJECT($person)
  $person:=OB COPY($empty)
  OB SET($person;"firstname";[Person]firstName)
  OB SET($person;"age";[Person]age)
  APPEND TO ARRAY($_people;$person)
  NEXT RECORD([Person])
End while

C_OBJECT($myFamily)
OB SET($myFamily;"people";$_people)
```

Toutes les réponses obtenues des APIs Twitter sont au format JSON. Ainsi, pour obtenir la valeur d'une clef d'un JSON, on utilisera les commandes **OB** inverses :

```
ARRAY OBJECT($_people;0)
$_people:=OB Get($myFamily;"people")
For ($i;1;Size of array($_people))
  CREATE RECORD([Person])
  [Person]lastName:=OB Get($_people{$i};"lastname")
  [Person]age:=OB Get($_people{$i};"age")
  SAVE RECORD([Person])
End for
```

MISE EN ŒUVRE SUR 4D

Nous allons découvrir comment envoyer, depuis votre application 4D, des tweets, des photos, partager des liens, récupérer des informations sur vos abonnés, le contenu de votre timeline, etc.

Authorization Header

Afin d'authentifier vos requêtes sur Twitter, chacun devra contenir une entête d'authentification contenant un certain nombre d'informations.

Pour obtenir votre entête, voici la liste des ingrédients :

- Un timestamp (aussi appelé Heure POSIX) ;
- Un « nonce » ;
- Vos clefs d'authentification :
 - o Consumer Key
 - o Consumer Secret
 - o Access Token
 - o Access Token Secret
- L'URI du service REST à utiliser ;
- La méthode HTTP utilisée ;
- La méthode de signature = HMAC-SHA1 ;
- La version OAuth utilisée : 1.0

Création de la signature

Ces informations vous permettront de créer une signature. Celle-ci dépendra, notamment, de la fonction que vous souhaitez utiliser. De ce fait, il faudra générer une nouvelle signature et un nouvel entête pour chacune de vos requêtes.

- *Le Timestamp*, ou heure POSIX : nombre de secondes écoulées depuis le 1^{er} Janvier 1970 à minuit UTC précise. Pour obtenir un timestamp, nous pouvons soit écrire une méthode qui fera le calcul, soit simplement utiliser la fonction PHP `time()` :

```
C_LONGINT($timeLong)
C_BOOLEAN($ok)
$ok:=PHP Execute("";"time";$timeLong)
```

- *Le Nonce* : il s'agit d'une chaîne aléatoire, générée par l'application cliente. Dans notre cas, pour être sûre qu'elle soit unique, j'utilise un UUID ;

```
C_TEXT($oAuthNonce)
$oAuthNonce:=Generate UUID
```

- *Parameter String* : cette chaîne de paramètres doit contenir les paramètres obligatoires relatifs à l'authentification, ainsi que ceux qui seront passés dans l'url pour votre requête. Ces paramètres devront être envoyés par ordre alphabétique :

Par exemple, pour appeler la fonction accessible via l'URL <https://api.twitter.com/1.1/statuses/update.json>, votre chaîne de paramètre sera la suivante :

```
POST&https%3A%2F%2Fapi.twitter.com%2F1.1%2Fstatuses%2Fupdate.json&oauth_consumer_key=YD8AN17RwqaCFGJ5w89H1fZVm&oauth_nonce=07680038309C794192EA791A07CDC29D&oauth_signature_method=HMAC-SHA1&oauth_timestamp=1460666424&oauth_token=377330954-cFmL6QBYygxZcqaPFVCx4IourWQIKfvJpVtmlff&oauth_version=1.0&status=Bienvenue%20au%204D%20Summit%20Europe%202016%20%21
```

Détaillons un peu les éléments qui composent cette superbe chaîne :

POST	Méthode HTTP utilisée pour l'appel à la fonction
https%3A%2F%2Fapi.twitter.com%2F1.1%2Fstatuses%2Fupdate.json	URL de la ressource appelée encodée en URL (<i>Percent Encoding</i> *)
paramètre	Vous devez faire figurer toutes les paramètres qui seront envoyés avec votre requête, ainsi que les informations d'authentification
valeur	Les valeurs doivent également être encodées via le <i>Percent Encoding</i>
=	Les paramètres et les valeurs sont séparés par le signe = (égal) ;
&	Les paires <i>paramètre/valeur</i> sont séparées par l'opérateur logique AND, représenté par le caractère & (esperluette) ;

Pour encoder les valeurs des paramètres, vous pouvez utiliser simplement la fonction PHP `rawurlencode()` :

```
C_TEXT($source;$result)
C_BOOLEAN($ok)
$ok:=PHP Execute("";"rawurlencode";$result;$source)
```

Cette chaîne va ensuite être cryptée avec la méthode HMAC-SHA1, à l'aide d'une clef établie sur le *Consumer Secret* et le *OAuth Token Secret* (ces deux clefs ne sont connues que par vous et par Twitter, c'est pourquoi on les dit « secrètes »), puis encodée à nouveau en base 64 pour en résulter une *signature*.

```

$signinKey:=percentEncode ($consumerSecret)+"&"+percentEncode ($oAuthTokenSecret)
C_BLOB($signatureEncoded)
C_BOOLEAN($ok)
$ok:=PHP Execute("";"hash_hmac";\
    $signatureEncoded;"sha1";$parameterString;$signinKey;True)
C_TEXT($signature)
BASE64_ENCODE($signatureEncoded;$signature)
$signature:=percentEncode ($signature)

```

Entête d'authentification

Lorsque vous souhaitez accéder à une ressource, Twitter demande à recevoir, dans l'entête de votre requête, les informations nécessaires à votre authentification. Pour ce faire, l'entête devra contenir les informations relatives à l'authentification OAuth, ainsi que la signature précédemment générée :

```

OAuth                                oauth_consumer_key="YD8AN17RwqaCFGJ5w89HlfZVm",
oauth_nonce="4B67A7BD4C2EB0489D6EB5321FBE00E7",
oauth_signature="nOFPw21vWcy6Waf097Ku3Wca%2BDs%3D",
oauth_signature_method="HMAC-SHA1",
oauth_timestamp="1460669733",
oauth_token="377330954-cFmL6QBYygxZcgeaPFVCx4IourWQIKfvJpVtmlff",
oauth_version="1.0"

```

Vous noterez que les jetons secrets ne sont pas envoyés « en clair » dans l'entête de votre requête. Néanmoins, Twitter, connaissant ces tokens, pourra décrypter la signature et cela permettra votre authentification sur la plateforme.

L'authentification fait souvent partie des sujets les plus délicats de l'utilisation d'une API, d'autant plus qu'il est indispensable d'intégrer ces mécanismes pour éliminer les éventuelles erreurs liées à l'authentification. Une fois ce sujet maîtrisé, vous pouvez vous concentrer sur les fonctionnalités de votre application.

À présent, vous avez tous les éléments pour pouvoir tweeter et récupérer des données de manière authentifiée, nous allons voir quelques exemples de mise en œuvre des APIs dans une application 4D.

Tweeter du texte

La fonctionnalité première de Twitter, est de pouvoir poster des messages (limités à 140 caractères, je vous le rappelle). Nous allons donc aborder l'utilisation des APIs par cette fonction.

Dans un tweet simple, l'utilisateur peut poster du texte, des urls, des hashtags ou mentionner d'autres twittos.

Exemple de tweet simple, avec du texte uniquement :

Bienvenue au 4D Summit 2016 !



Bérangère Lagrange
@BerenLagrange

Bienvenue au 4D Summit 2016 !

Voir la traduction

11:13 - 15 avr. 2016

Exemple de tweet plus complexe, intégrant une url, un hashtag et une mention :

140 caractères ne seront pas suffisants pour aborder les #APIs #Twitter et les #objets #4D lors du @4dSummit <http://bit.ly/1VrXY6Q>

Ce tweet sera représenté sur Twitter de la manière suivante :



- Les *hashtags* sont représentés sous forme de liens qui vous permettront d'afficher toutes les données de la Twittosphère traitant du sujet posé en mot-clef (comptes, tweets, photos, etc.) : <https://twitter.com/hashtag/apis?src=hash> ;
- La mention à un autre compte est également cliquable et renvoie sur le compte dudit utilisateur ;
- Enfin, le lien inséré dans le tweet (qui est, en réalité, l'url http://summit.4d.com/europe_fr/sessions-fr/ minifiée grâce au service <https://bitly.com>) n'apparaît pas directement sur le tweet, mais a été interprété de manière à présenter une image et une description de la page. D'ailleurs, dans la page d'un site web, pour définir quelles informations seront affichées dans un tweet, référez-vous aux « cards » <https://dev.twitter.com/cards/types>, mais ceci est un autre sujet...

Tout cela pour vous signifier que de votre côté, vous n'aurez absolument rien à développer pour que la représentation de votre message sur Twitter se fasse de la sorte.

Voyons dans 4D, comment envoyer un tweet avec hashtag : « *Bienvenue au #4DSummit 2016 !* » (cf. méthode ***twitter_statusUpdate*** dans la base de démonstration attachée).

1. Récoltez les paramètres :

Dans notre exemple, le texte à passer dans le paramètre *status*. Dans 4D, j'ai opté pour une solution adaptable : passer les paramètres dans un objet 4D (nous verrons que la méthode ***twitter_statusUpdate*** sera utilisée dans d'autres cas, avec d'autres paramètres) :

```
C_OBJECT($tweet)
C_OBJECT ($jsonStatus)
OB SET($jsonStatus;"status";percentEncode(tweetText))
$tweet:=twitter_statusUpdate ($jsonStatus)
```

2. Générez votre entête d'authentification :

```
$EndPoint:="https://api.twitter.com/1.1/statuses/update.json"
$authorizationHeader:=getAuthorizationHeader($EndPoint;\
    "POST";$jsonParameters)
```

3. Fixez les entêtes de votre requête HTTP :

```
ARRAY TEXT($_headers;0)
ARRAY TEXT($_headerValues;0)
APPEND TO ARRAY($_headers;"Content-Type")
APPEND TO ARRAY($_headerValues;\
    "application/x-www-form-urlencoded;charset=UTF-8")
APPEND TO ARRAY($_headers;"Authorization")
APPEND TO ARRAY($_headerValues;$authorizationHeader)
```

Vous remarquerez que nous utilisons un entête avec deux champs :

- Le *Content-Type* qui nous permet de spécifier le type MIME et l'encodage des caractères ;
- L'*Authorization Header*.

4. Etablissez une chaîne concaténant l'URL et les paramètres :

```
$paramString:=getParamString ($jsonParameters)
```

\$paramString vaut alors :

```
https://api.twitter.com/1.1/statuses/update.json?status=Bienvenue%20au%20%234DSummit%202016%20%21
```

5. Postez votre tweet !

```
C_TEXT($body)
C_OBJECT($jsonResponse)
C_LONGINT($httpStatus)
$httpStatus:=HTTP Request(HTTP POST method;$paramString ;\
    $body;$jsonResponse;$_headers;$_headerValues)
```

- \$body est une variable vide : obligatoire pour la commande **HTTP Request**;
- La méthode http utilisée est le POST, car nous soumettons une création de statut ;
- Les deux derniers paramètres correspondent aux champs et aux valeurs contenus dans notre entête. Après l'exécution de la commande **HTTP Request**, ils contiendront les champs de l'entête de la requête de réponse du serveur web de Twitter.

Dans le cas où notre requête a abouti (\$httpStatus>0), Twitter nous renvoie un objet JSON, contenant :

- Soit un message d'erreur si le tweet n'a pu être posté (\$httpStatus#200) :

```
{"errors": [
    {"message": "Rate limit exceeded", "code": 88}
]}
```

- Soit le détail de notre tweet au format JSON que nous pourrions manipuler pour conserver certaines informations, notamment l’ID ou la date de création (\$httpStatus=200) :

```
{
  "created_at": "Fri Apr 15 13:28:58 +0000 2016",
  "id": 7209671502368E+017, "id_str": "720967150236798976",
  "text": "Bienvenue au #4DSummit 2016 !",
  "entities": {
    "hashtags": [
      {
        "text": "4DSummit",
        "indices": [13, 22]
      }
    ],
    "symbols": [],
    "user_mentions": [],
    "urls": []
  },
  "truncated": false,
  "source": "<a href='\"http://www.power.fr\"' rel='\"nofollow\"'>Demo Summit</a>",
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 377330954, "id_str": "377330954", "name": "B  reng  reLagrange",
    "screen_name": "BerenLagrange", "location": "Lyon, France",
    "description": "Software & web developer #4D #JavaScript",
    "url": null, "entities": {
      "description": {
        "urls": []
      }
    },
    "protected": false, "followers_count": 197, "friends_count": 285,
    "listed_count": 10, "created_at": "Wed Sep 21 12:09:40 +0000 2011",
    "favourites_count": 61, "utc_offset": 7200, "time_zone": "Paris",
    "geo_enabled": true, "verified": false, "statuses_count": 1805,
    "lang": "fr", "contributors_enabled": false, "is_translator": false,
    "is_translation_enabled": false, "profile_background_color": "C0DEED",
    "profile_background_image_url": "http://pbs.twimg.com/profile_background_images/374299539/IMG_7553.JPG",
    "profile_background_image_url_https": "https://pbs.twimg.com/profile_background_images/374299539/IMG_7553.JPG",
    "profile_background_tile": true,
    "profile_image_url": "http://pbs.twimg.com/profile_images/1827675431/324315_10150336729436531_614381530_8663262_653205179_o_normal.jpg",
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/1827675431/324315_10150336729436531_614381530_8663262_653205179_o_normal.jpg",
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/377330954/1348899395",
    "profile_link_color": "FF691F", "profile_sidebar_border_color": "C0DEED",
    "profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333",
    "profile_use_background_image": true, "has_extended_profile": false,
    "default_profile": false, "default_profile_image": false,
    "following": false, "follow_request_sent": false, "notifications": false
  },
  "geo": null, "coordinates": null, "place": null, "contributors": null,
  "is_quote_status": false, "retweet_count": 0, "favorite_count": 0,
  "favorited": false, "retweeted": false, "lang": "fr"
}
```

Nous distinguons, dans ce JSON, deux grandes parties descriptives :

- une description relative au tweet (sur fond orange): date de cr  ation, ID, texte, hashtags utilis  s, etc. ;
- une description quasi-compl  te de l’utilisateur, auteur du tweet (sur fond bleu): ID, nom, photos de profile et de couverture, etc.

Dans l’objet « entities », nous retrouvons le hashtag #4DSummit qui appara  t dans notre message. Si nous avions mentionn   un utilisateur ou ins  r   une url, nous aurions pu retrouver ces informations ici. Nous pouvons   galement savoir si certains utilisateurs ont ajout   un tweet    leurs favoris, par exemple, ou combien de fois il aurait   t   retweet  . Dans notre exemple, nous venons tout juste de publier le message. De fait, ces informations sont encore inutiles, mais cela vous permet d’imaginer tout ce que vous pourriez faire avec de telles donn  es : statistiques, analyse des r  actions des twittos par rapport    vos publications, etc.

N.B. Vous remarquerez, que figurent, dans la r  ponse, le nom et l’url que nous avons configur  s lors de la cr  ation de l’application (sur fond vert).

Dans 4D, pour récupérer l'ID du tweet, utilisez la commande suivante :

```
"id":7.209671502368E+017,"id_str":"720967150236798976",
```

```
C_TEXT($ID)
$ID:=OB Get($jsonResponse;"id_str";Is_Text)
```

Il est, d'ailleurs, conseillé de récupérer l'ID sous forme de chaîne **id_str**, plutôt que l'identifiant **id** car ce dernier est un entier long 64 bits, format qui n'est pas encore pris en charge sur toutes les plateformes.

Vous pourrez, si vous le souhaitez, stocker ce JSON dans un champ de type objet d'une table prévue à cet effet.

Tweeter une image avec du texte

L'envoi d'un tweet avec une image ne varie pas vraiment, si ce n'est qu'en paramètre de notre méthode, nous devons spécifier l'identifiant d'une image que l'on aurait au préalable uploadée. La taille de l'image envoyée doit être au maximum de 5Mo, mais pour l'utiliser dans un tweet, elle ne devra pas dépasser 3Mo.

Uploadez votre image

URL : <https://upload.twitter.com/1.1/media/upload.json>

Vous pouvez envoyer votre photo de deux manières :

- En utilisant le paramètre **media** contenant le fichier binaire ;
- Soit le paramètre **media_data** contenant le fichier encodé en base64. C'est la solution que j'ai choisie pour notre exemple.

```
PICTURE TO BLOB($picture;$pictureBlob)
C_TEXT($base64Picture)
BASE64 ENCODE($pictureBlob;$base64Picture)
```

Entête : vous devez spécifier le type "multipart/form-data" dans le champ "Content-Type" de l'entête de votre requête, cela permet de spécifier que vous souhaitez envoyer un fichier. Il faudra également spécifier un « *boundary* », un délimiteur qui permet de définir à quel endroit commence une partie de la requête, et à quel endroit elle finit :

```
$boundary:="4DSummit2016Boundary"
APPEND TO ARRAY($_headers;"Content-Type")
APPEND TO ARRAY($_values;"multipart/form-data;boundary=\""+\
    $boundary+"\"")
```

N'oubliez pas d'ajouter la partie authentification dans votre entête :

```
$authorizationHeader:=getAuthorizationHeader ($EndPointUrl;"POST")
APPEND TO ARRAY($_headers;"Authorization")
APPEND TO ARRAY($_values;$authorizationHeader)
```

N.B. Vous remarquerez que nous ne passons pas le paramètre **media_data** pour constituer l'entête d'authentification : ce paramètre sera contenu dans le *body* de la requête et non pas dans l'URL.

Le contenu de votre requête devra être structuré d'une manière bien précise, c'est-à-dire qu'il devra faire apparaître les différentes parties de la requête avec l'image encodée en base64:

```
C_TEXT($body)
$CrLf:="\r\n"

$headerData:="Content-Disposition:                                form-
data;name=\"media_data\";filename=\""+$pictureName+"\""+$CrLf+\"
\"Content-Type: application/octet-stream"+$CrLf+\"
\"Content-Transfer-Encoding: base64"+$CrLf

$body:="--"+$boundary+$CrLf
$body:=$body+$headerData
$body:=$body+$CrLf
$body:=$body+$base64Picture
$body:=$body+$CrLf
$body:=$body+"--"+$boundary+"--"+$CrLf
```

Voici à quoi ressemblera le *body* de votre requête :

```
--4DSummit2016Boundary

Content-Disposition:form-
data;name="media_data";filename="20160414_122510945_iOS.jpg"

Content-Type: application/octet-stream

Content-Transfer-Encoding: base64


/9j/4AAQSkZJRgABAQEASABIAAD/4QgMRXhpZgAATU0AKgAAAAgADgAfAAUAAAAB.....4oor
0lsWMfUvskxyGNJfMQHddqKKKgD/9k=

--4DSummit2016Boundary--
```

Une fois votre requête constituée, soumettez-la à l'aide de la commande **HTTP Request** :

```
C_OBJECT($jsonResponse)
C_LONGINT($httpStatus)
$httpStatus:=HTTP Request(HTTP POST method;$EndPointUrl;\
    $body;$jsonResponse;$_headers;$_values)
```

Twitter vous retournera, au format JSON, les informations sur votre image :

```
{
  "media_id": 553639437322563584,
  "media_id_string": "553639437322563584",
  "size": 998865,
  "image": {
    "w": 2234,
    "h": 1873,
    "image_type": "image/jpeg"
  }
}
```

Pour récupérer l'identifiant de votre image, utilisez la commande suivante :

```
C_TEXT($mediaId)
$mediaId:=OB Get($jsonResponse;"media_id_string")
```

1. Poster un tweet avec une référence à l'image :

Nous utiliserons la même méthode que pour l'envoi d'un tweet « simple », mais en y ajoutant quelques paramètres :

- **media_ids** : liste des médias préalablement uploadés : vous pouvez envoyer jusqu'à 3 références de photos, ou 1 vidéo, ou 1 gif ;
- **lat** : latitude de l'endroit où a été prise la photo. Ex : 45.77522
- **long** : la longitude de l'endroit où a été prise la photo. Ex : 4.85606

Pour récupérer les coordonnées GPS d'une photo, utilisez les métadonnées de votre image. En effet, lorsque vous prenez une photo depuis un smartphone avec le GPS activé, les coordonnées du lieu de la prise de photo sont stockées dans les métadonnées. Pour les retrouver, utilisez la commande 4D :

```
C_REAL($latitude_degree;$latitude_minute;$latitude_seconde)
C_REAL($latitude_sign;$latitude)
C_REAL($longitude_degree;$longitude_minute;$longitude_seconde)
C_REAL($longitude_sign;$longitude)
C_TEXT($latitude_dir;$longitude_dir)

GET PICTURE METADATA(tweetPicture;GPS latitude deg;$latitude_degree)
GET PICTURE METADATA(tweetPicture;GPS latitude min;$latitude_minute)
GET PICTURE METADATA(tweetPicture;\
    GPS latitude sec;$latitude_seconde)

GET PICTURE METADATA(tweetPicture;\GPS latitude dir;$latitude_dir)

GET PICTURE METADATA(tweetPicture;\
    GPS longitude deg;$longitude_degree)

GET PICTURE METADATA(tweetPicture;\
    GPS longitude min;$longitude_minute)

GET PICTURE METADATA(tweetPicture;\
    GPS longitude sec;$longitude_seconde)
GET PICTURE METADATA(tweetPicture;GPS longitude dir;$longitude_dir)

$latitude_sign:=Choose(( $latitude_dir="N");1;-1)
$longitude_sign:=Choose(( $longitude_dir="E");1;-1)

$latitude:=Round(( $latitude_degree+($latitude_minute/60)+\
    ($latitude_seconde/3600))*$latitude_sign;5)

$pictureLatitude:=Replace string(String($latitude);",",".")

$longitude:=Round(( $longitude_degree+($longitude_minute/60)+\
    ($longitude_seconde/3600))*$longitude_sign;5)

$pictureLongitude:=Replace string(String($longitude);",",".")
```

Constituez votre objet contenant les paramètres à transmettre pour l'upload de l'image, puis réutilisez la méthode qui nous a permis d'envoyer le tweet de notre premier exemple :

```

C_OBJECT($jsonParameters;$tweet)
OB SET($jsonParameters;"status";percentEncode (tweetText))
OB SET($jsonParameters;"media_ids";$mediaId)
C_OBJECT($tweet)
$tweet:=twitter_statusUpdate ($jsonParameters)

```

Et voilà ! Les tweets n'ont, maintenant, plus de secret pour vous !

Je souhaiterais terminer avec un dernier exemple qui consiste à récupérer la liste des comptes qui vous suivent, vos « followers », pour vous expliquer une spécificité importante à connaître : les curseurs.

Récupérer la liste de vos followers

Nous avons vu plusieurs exemples nous permettant de soumettre des données. Nous allons, maintenant, utiliser une fonction nous permettant d'en récupérer.

Twitter utilise un système de « curseur » pour la récupération de listes de données importantes. Prenons l'exemple d'une requête qui permettrait d'obtenir la liste de vos followers : GET followers/list

Le compte [@4DSummit](#) est suivi par 298 abonnés, à l'heure où j'écris ce document. Or, comme le stipule la documentation de la fonction <https://dev.twitter.com/rest/reference/get/followers/list> celle-ci retourne une collection de 200 utilisateurs maximum, une sorte de pagination, en somme. De fait, nous devons exécuter a minima deux fois la requête pour obtenir la liste complète, trois fois si nous décidons de récupérer une liste de 100 followers par requête (paramètre **count** à spécifier, qui vaut 20 par défaut).

La navigation par curseur permet de récupérer la valeur du prochain curseur **next_cursor_str**, équivalent au numéro de la prochaine page à récupérer, si l'on peut dire. Lorsque le curseur vaut 0, cela signifie que vous avez parcouru la totalité de la collection. Par défaut, si vous omettez le paramètre **cursor**, Twitter considère qu'il vaut -1.

Dans la méthode **twitter_followersList** de la base de démonstration, j'utilise ce concept : je soumetts ma requête, jusqu'à obtenir un curseur à 0.

```

$EndPoint:="https://api.twitter.com/1.1/followers/list.json"
ARRAY OBJECT($_followers;0)
C_POINTER($1)
C_TEXT($2;$screenName)
$screenName:=$2
C_TEXT($cursor)
$cursor:="-1"

C_OBJECT($jsonParameters)
OB SET($jsonParameters;"cursor";$cursor)
OB SET($jsonParameters;"screen_name";$screenName)
OB SET($jsonParameters;"count";100) // Max 200
OB SET($jsonParameters;"skip_status";False)
OB SET($jsonParameters;"include_user_entities";False)

```

Repeat

```

$authorizationHeader:=getAuthorizationHeader ($EndPoint;\
"GET";$jsonParameters)

ARRAY TEXT($_headers;0)
ARRAY TEXT($_headerValues;0)

C_TEXT($body)
C_OBJECT($jsonResponse)
C_LONGINT($httpStatus)

$paramsString:=getParamString ($jsonParameters)

```

```

APPEND TO ARRAY($_headers;"Content-Type")
APPEND TO ARRAY($_headerValues;\
    "application/x-www-form-urlencoded;charset=UTF-8")

APPEND TO ARRAY($_headers;"Authorization")
APPEND TO ARRAY($_headerValues;$authorizationHeader)

$httpStatus:=HTTP Request(HTTP GET method;\
    $sendPoint+$paramString;$body;$jsonResponse;\
    $_headers;$_headerValues)

If ($httpStatus=200)
    ARRAY OBJECT($_users;0)
    $cursor:=OB Get($jsonResponse;"next_cursor_str")
    OB SET($jsonParameters;"cursor";$cursor)
    OB GET ARRAY($jsonResponse;"users";$_users)
    For ($i;1;Size of array($_users);1)
        APPEND TO ARRAY($_followers;$_users{$i})
    End for
Else
    errorHandler ($jsonResponse)
End if
Until ($cursor="0")

```

N.B. vous remarquerez que nous exécutons une requête pour récupérer des données. Nous utilisons donc la méthode HTTP **GET** lors de l'appel à la commande **HTTP Request**.

Pour cette requête, vous pouvez récupérer la liste des followers de n'importe quel compte, il suffit de spécifier en paramètre soit son **user_id**, soit son **screen_name**. Il s'agit du nom d'utilisateur.

Par exemple : *"berenlagrange"*

Le JSON résultant de cette requête contient le un tableau d'objets « users » :

```

{"users":[
    {
        "id":"632829252","id_str":"632829252","name":"4D
        Develop","screen_name":"4Ddevelop","location":"France","url":"http://summit.4d.com/europe_en/","descriptio
        n":"Develop and deploy desktop, client-server and Web
        applications","protected":false,"followers_count":171,"friends_count":138,"listed_count":4,"created_at":"Wed
        Jul 11 09:30:50 +0000
        2012","favourites_count":29,"utc_offset":7200,"time_zone":"Amsterdam","geo_enabled":true,"verified":false,"
        statuses_count":96,"lang":"fr","contributors_enabled":false,"is_translator":false,"is_translation_enabled":false,"
        profile_background_color":"EEEEEE","profile_background_image_url":"http://pbs.twimg.com/profile_backgro
        und_images/378800000071928509/667cf1338900b2eeb9f4fbc62b9d84c0.png","profile_background_image_url
        _https":"https://pbs.twimg.com/profile_background_images/378800000071928509/667cf1338900b2eeb9f4fbc6
        2b9d84c0.png","profile_background_tile":true,"profile_image_url":"http://pbs.twimg.com/profile_images/3788
        00000442988837/ddffdf05d167d82e327dd43e14b34ff_normal.jpeg","profile_image_url_https":"https://pbs.twi
        mg.com/profile_images/378800000442988837/ddffdf05d167d82e327dd43e14b34ff_normal.jpeg","profile_ban
        ner_url":"https://pbs.twimg.com/profile_banners/632829252/1459929860","profile_link_color":"0084B4","prof
        ile_sidebar_border_color":"FFFFFF","profile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","pr
        ofile_use_background_image":true,"has_extended_profile":false,"default_profile":false,"default_profile_image"
        :false,"following":true,"follow_request_sent":false,"notifications":false,"muting":false,"blocking":false,"blocke
        d_by":false},
    {
        "id":"74516489","id_str":"74516489","name":"Olivier
        DESCHANELS","screen_name":"olivierd4d","location":"","url":"http://www.4d.com/fr/company/blogs/Olivier
        %20Deschanel","description":"4D / Wakanda / Web /
        JS","protected":false,"followers_count":199,"friends_count":128,"listed_count":10,"created_at":"Tue Sep 15
    }
}

```

```

18:28:23                                                                 +0000
2009", "favorites_count":13, "utc_offset":7200, "time_zone": "Paris", "geo_enabled":true, "verified":false, "status
s_count":325, "lang": "en", "contributors_enabled":false, "is_translator":false, "is_translation_enabled":false, "profil
e_background_color": "C0DEED", "profile_background_image_url": "http://abs.twimg.com/images/themes/them
e1/bg.png", "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png", "prof
ile_background_tile":false, "profile_image_url": "http://pbs.twimg.com/profile_images/420009817/Olivier_30_n
ormal.jpg", "profile_image_url_https": "https://pbs.twimg.com/profile_images/420009817/Olivier_30_normal.jp
g", "profile_link_color": "0084B4", "profile_sidebar_border_color": "C0DEED", "profile_sidebar_fill_color": "DD
EEF6", "profile_text_color": "333333", "profile_use_background_image":true, "has_extended_profile":false, "defa
ult_profile":true, "default_profile_image":false, "following":true, "follow_request_sent":false, "notifications":false
, "muting":false, "blocking":false, "blocked_by":false},
.....
], "next_cursor":1.4185081938124E+018, "next_cursor_str":"1418508193812427210", "previous_cursor":0, "prev
ious_cursor_str":"0"}

```

Dans 4D, utilisez la commande **OB GET ARRAY** pour obtenir un tableau d'objets contenant la collection des followers ainsi obtenue et les manipuler comme bon vous semble :






```

ARRAY OBJECT($_users;0)
OB GET ARRAY($_jsonResponse;"users";$_users)

```

Dans la base de démonstration, j'affiche la liste de mes followers dans une listbox avec l'image de profile, le nom, le nombre de followers, le nombre de comptes auxquels il est lui-même abonné :



Profile ...	Screen Name	Name	Followers	Friends
	_tpenner	Tim Penner	48	65
	tiranbe	Tiran Behrouz	323	680
	huntertrek	Joshua Hunter	77	106
	4DSummit	4D Summit	298	26
	waynestewart	Wayne Stewart	219	249
	terrykolee	Terry Lee	18	66
	madamov	madamov	412	780

A présent, vous avez toutes les clefs pour implémenter les fonctionnalités mises à disposition via les APIs Twitter dans votre application 4D. Sachez que les concepts que nous avons vus ici peuvent s'appliquer à la majorité des APIs, comme celles de Facebook, par exemple.

Je vous laisse gazouiller avec le petit oiseau bleu, c'est à vous de jouer !

Vous pouvez me suivre sur mon compte [@BerenLagrange](https://twitter.com/BerenLagrange).

A bientôt sur la Twittosphère !

